IBM

Rational Development Studio for i

ILE COBOLプログラマーの手引き

7.1

SD88-5045-07
(英文原典：SC09-2540-07)

IBM

Rational Development Studio for i

ILE COBOLプログラマーの手引き

7.1

SD88-5045-07
(英文原典：SC09-2540-07)

ご注意

本書および本書で紹介する製品をご使用になる前に、 739 ページの『特記事項』に記載されている情報をお読みください。

| 本書は、IBM Rational Development Studio for i のバージョン 7 リリース 1、モディフィケーション・レベル 0
| (5770-WDS) に適用されます。また、改訂版で特に断りのない限り、これ以降のすべてのリリースおよびモディフィ
| ケーションにも適用されます。本書は、RISC システムにのみ適用されます。

| 本書は SD88-5045-06 の改訂版です。

資料のご注文方法については、http://www.ibm.com/jp/manuals の「ご注文について」をご覧ください。 (URL は、変更になる場合があります)

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： SC09–2540–07
IBM Rational Development Studio for i
ILE COBOL Programmer's Guide Programming
7.1

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2010.4

目次

本書について

本書は、IBM i で、Integrated Language Environment (ILE) COBOL コンパイラー・プログラムの作成、コンパイル、バインド、実行、デバッグ、および保守を行う方法を説明しています。この資料は、他の ILE COBOL プログラムや ILE COBOL 以外のプログラムを呼び出す方法、他のプログラムとデータを共用する方法、ポインターを使用する方法、および例外を処理する方法に関するプログラミング情報を提供します。また、外部接続装置、データベース・ファイル、表示装置ファイル、および ICF ファイルに対する入出力操作の実行方法が説明されています。

本書を使用して、次のことができます。
- ILE COBOL プログラムの設計およびコーディング
- ILE COBOL プログラムの入力、コンパイル、およびバインド
- ILE COBOL プログラムの実行およびデバッグ
- コーディングされた ILE COBOL 例の学習

注: 他の章に進む前に、本書の 1 ～ 6 章の内容を十分に理解しておく必要があります。

本書では、他の IBM® 資料に言及しています。これらの正式な資料名および資料番号については 743 ページの『参考文献』にその一覧があります。本文で参照するときは、資料名の略称が使われます。

本書の対象読者

本書は、COBOL プログラミング言語について、ある程度の知識を有しているアプリケーション・プログラマー、およびプログラムを稼働させるオペレーターを対象としています。本書は、ILE COBOL 言語でプログラミングを行う i5/OS のユーザーのための手引きです。

本書を使用するにあたって、以下についての基本的な知識が必要です。
- データ処理の概念
- COBOL プログラミング言語
- IBM IBM i (以前は OS/400) オペレーティング・システム
- Integrated Language Environment (ILE) の概念
- アプリケーション・プログラミング・インターフェース (API)
- *Application Development ToolSet* (ADTS) などの、非プログラマブル端末 (NPT) ベース用開発ツール

  注: WebSphere Development Studio Client for System i を使用してください。これが推奨される方法であり、ワークステーションのツールに関する資料は、その製品のオンライン・ヘルプにあります。
- ディスプレイ装置上で制御および標識を使用する方法、およびキーボードで次のようなキーを使用する方法。
  - カーソル移動キー

- ファンクション・キー
- フィールド終了キー
- 挿入キーおよび削除キー
- エラー・リセット・キー

- IBM System i にリンクされ、i5/OS ソフトウェアを実行している場合にディスプレイ装置を操作する方法。これには、IBM i オペレーティング・システム、およびその制御言語 (CL) を使用して、次のようなことを行う方法を知っていることも含まれます。
  - ディスプレイ装置のサインオンおよびサインオフ
  - 表示画面との対話
  - ヘルプの使用
  - CL コマンドの入力
  - 適用業務開発ツールの使用
  - メッセージへの応答
  - ファイル管理の実行
- IBM i CL 機能の基本概念
- データ管理サポートを使用して、アプリケーションがファイルを処理できるようにする方法。
- 以下の*Application Development ToolSet* ツールを使用する方法。
  - 画面の設計およびコーディングに使用できる画面設計機能 (SDA)、またはクライアント製品の一部である DDS 設計ユーティリティー。
  - ソース・メンバーの入力および更新に使用できる原始ステートメント入力ユーティリティー (SEU) または クライアント製品の一部である言語依存のエディター。

    注: WebSphere Development Studio Client for System i を使用してください。これが推奨される方法であり、ワークステーションのツールに関する資料は、その製品のオンライン・ヘルプにあります。

    クライアント・ツール入門については、『クライアント製品のアプリケーション開発ツールの使用』を参照してください。
- ILE COBOL プログラムに SQL ステートメントを挿入するのに使用される構造化照会言語 (SQL)。

前提条件および関連情報

IBM i、System i および AS/400e のテクニカル情報を調べる開始点として Information Center を使用してください。 Information Center には、次の 2 通りの方法でアクセスすることができます。

- 以下の Web サイトから。

  `http://www.ibm.com/systems/i/infocenter/`
- 注文された IBM i と一緒に出荷される CD-ROM から。

  iSeries Information Center および PDF ライブラリー CD パッケージ, SK88-8055-00.

i5/OS Information Center には、CL コマンド、システム・アプリケーション・プログラミング・インターフェース (API)、論理区画、クラスター化、Java™、TCP/IP、Web サーブ、およびセキュア・ネットワークなど、アドバイザー情報および重要なトピックが含まれています。また、関連する IBM Redbooks へのリンク、および Technical Studio など他の IBM Web サイトや IBM ホーム・ページへのインターネット・リンクも含まれています。

これらの資料は、ILE COBOL コンパイラーに最も関係のあるものであり 743 ページの『参考文献』にその一覧が示されています。

このリリースにおける新しい機能

以下のリストで、V7R1 において ILE COBOL に対して行われた機能強化について説明します。

- COMPUTATIONAL-5 (ネイティブ 2 進数) データ・タイプ

  COMPUTATIONAL-5 (COMP-5) は、USAGE 節でサポートされるようになったネイティブ 2 進数データ・タイプです。 COMP-5 データ項目は、2 進データとしてストレージで表され、最大でネイティブ 2 進数表記の容量 (2、4、または 8 バイト) の値を入れることができます。数値データが COMP-5 項目に移動または保管されると、COBOL ピクチャー・サイズ限界ではなく、2 進数フィールド・サイズで切り捨てが行われます。 COMP-5 項目が参照される場合、その操作でフル 2 進数フィールド・サイズが使用されます。このサポートにより、他の IBM プラットフォームおよびオペレーティング・システムでの COBOL との移植性が強化されます。
- 国別データ項目の VALUE 節に非数値リテラルを指定する機能。
- XML GENERATE のパフォーマンスの改善および PROCESS オプション

  APPEND オプションを指定した場合に、 XML GENERATE のパフォーマンスが改善されるようになりました。ユーザーは、データ構造またはストリーム・ファイルに追加するデータ・レコードが多数存在する場合に、今回の変更によるメリットを受けることができます。この改善では、新しい PROCESS ステートメント・パラメーター XMLGEN が追加され、そのオプション値は以下のとおりです。

  – NOKEEPFILEOPEN / KEEPFILEOPEN

    XML GENERATE ステートメントの完了時に XML ストリーム・ファイルを開いたままにして閉じないように指示する場合には、KEEPFILEOPEN を指定します。これにより、後続の XML GENERATE FILE-STREAM APPEND ステートメントでデータをストリーム・ファイルに速やかに追加できます。

  – NOASSUMEVALIDCHARS / ASSUMEVALIDCHARS

    特殊文字 (小なり記号「<」、大なり記号「>」、アンパーサンド「&」、および単一/二重引用符)、および XML でサポートされない、16 進として生成する必要がある文字の検査を XML GENERATE でバイパスする場合には、ASSUMEVALIDCHARS を指定します。これを指定しないと、デフォルトの NOASSUMEVALIDCHARS で通常の検査が実行されます。
- リスト・デバッグ・ビューを暗号化する機能

新しい CRTBNDCBL / CRTCBLMOD パラメーターが追加され、リスト・デバッグ・ビューの暗号化がサポートされるようになりました。 DBGENCKEY で、デバッグ・ビューに埋め込まれるプログラム・ソースを暗号化するために使用する暗号鍵を指定します。

- 大きなプログラムのサポート

  CRTBNDCBL / CRTCBLMOD OPTIMIZE パラメーターで、新しい *NEVER オプション値がサポートされるようになりました。 *NEVER 値により、プログラムの最適化コードを生成せずに、大きなプログラムをコンパイルできます。 PROCESS ステートメント・オプション NEVEROPTIMIZE も追加されました。

- テラスペース・ストレージ・モデルのサポート

  新しい CRTBNDCBL / CRTCBLMOD パラメーター STGMDL を以下のオプション値とともに使用して、プログラム/モジュールのストレージ・モデルを指定できるようになりました。

  – *SNGLVL は、単一レベル・ストレージ・モデルを使用してプログラム/モジュールを作成することを指定します。
  – *TERASPACE は、テラスペース・ストレージ・モデルを使用してプログラム/モジュールを作成することを指定します。
  – *INHERIT は、プログラム/モジュールが呼び出し元のストレージ・モデルを継承することを指定します。

  また、CRTBNDCBL コマンドの活動化グループ・パラメーター ACTGRP に、以下の新しいデフォルト・オプション値が追加されました。

  – *STGMDL: STGMDL(*TERASPACE) を指定した場合には、プログラムは、QILETS 活動化グループに活動化されます。その他のすべてのストレージ・モデルでは、プログラムは呼び出し時に、QILE 活動化グループに活動化されます。

- 新しい PROCESS ステートメント・オプション
  – PROCESS ステートメント・パラメーターとして、ACTGRP が、以下のオプション値を指定して使用できるようになりました。
    - STGMDL
    - NEW
    - CALLER
  – NEVEROPTIMIZE が PROCESS ステートメント・オプションとして使用できるようになりました。
  – PROCESS ステートメント・パラメーターとして、STGMDL が、以下のオプション値を指定して使用できるようになりました。
    - INHERIT
    - SNGLVL
    - TERASPACE
  – PROCESS ステートメント・パラメーターとして、XMLGEN が、以下のオプション値を指定して使用できるようになりました。
    - NOKEEPFILEOPEN / KEEPFILEOPEN
    - NOASSUMEVALIDCHARS / ASSUMEVALIDCHARS

V7R1 用の本書「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」(SD88-5045-07) は、V6R1 用の資料 (SD88-5045-06) とは多くの点で異なります。ほとんどの変更は機能強化に関連するものであり、それ以外は、小規模な技術的な訂正を反映したものです。

V6R1 での変更点

以下に、V6R1 において ILE COBOL に対して行われた機能強化について説明します。

- National UCS-2 CCSID の機能強化

  NTLCCSID パラメーターが、CRTCBLMOD コマンド、CRTBNDCBL コマンド、および PROCESS ステートメントに追加され、UCS-2 CCSID を指定して、National データ項目用に使用できるようになりました。このパラメーターを使用すると、デフォルトの 13488 ではなく、CCSID 1200 などの CCSID を指定して、National の項目用に使用できます。
- モジュール・サポートにおける PCML

  CRTCBLMOD コマンド、および CRTBNDCBL コマンドの PGMINFO パラメーターが強化され、生成された PCML を置くロケーションを指定できるようになりました。ユーザーが PGMINFO キーワードの最初のパラメーターとして *PCML を指定すると、*STMF、*MODULE、または *ALL のロケーションを指定する 2 番目のパラメーターも指定できます。

  *STMF を指定すると、PCML は INFOSTMF パラメーター上に指定したストリーム・ファイル内に置かれます。

  *MODULE を指定すると、PCML は生成されたモジュール内に置かれます。

  *ALL を指定すると、PCML は これらのロケーションすべてに置かれます。
- 複合 OCCURS DEPENDING ON (ODO) デバッガー・サポート

  サポートが追加され、システム・デバッガーおよびクライアント・デバッガーが複合 OCCURS DEPENDING ON 配列をデバッグできるようになりました。
- ラージ・プログラム・サポート

  コンパイラーが拡張され、ラージ・プログラムおよび多量のデータ項目を含むプログラムをコンパイルできるようになりました (システム制限によって異なります)。

V5R4 での変更点

以下に、V5R4 において ILE COBOL に対して行われた機能強化について説明します。

- XML のサポートが強化されました。新規のステートメント XML GENERATE は、COBOL データ・レコードの内容を XML 形式に変換します。 XML GENERATE は、Unicode UCS-2 またはいくつかの 1 バイト EBCDIC または ASCII CCSID の 1 つでエンコードされた XML 文書を作成します。 335 ページの『第 12 章 XML 出力の作成』を参照してください。

- ヌル終了非数値リテラル

  非数値リテラルをヌル終了にすることができます。ヌル終了非数値リテラルは、ヌル終了リテラルが「すべてリテラル」の表意定数でサポートされていない場合を除き、非数値リテラルを指定できるところであればどこでも使用することができます。
- 新規の CRTBNDCBL / CRTCBLMOD オプション

  *NOCOMPRESSDBG/*COMPRESSDBG は、DBGVIEW オプションの *LIST または *ALL が指定されている場合、コンパイラーがリスト・ビュー圧縮を実行する必要があるかどうかを指定します。
- 新規の組み込み関数は、以下のとおりです。
  - DISPLAY-OF
  - NATIONAL-OF
  - TRIM
  - TRIML
  - TRIMR

V5R3 での変更点

以下に、V5R3 において ILE COBOL に対して行われた機能強化についてリストします。

- ラージ VALUE 文節のサポート

  コンパイラー・オプション *NOSTDTRUNC が有効であると、使用法が BINARY または COMP-4 と記述されたデータ項目で、PICTURE 文節にピクチャー記号の P が含まれないものは、ネイティブの 2 進表記の容量内の値を持つことができます。
- CONSTANT データ・タイプ

  CONSTANT データ・タイプは、リテラルを示す CONSTANT 文節を含む、レベル 01 の項目を指定することで定義します。それ以降は、CONSTANT データ項目をリテラルの代わりに使用することができます。
- XML サポート

  XML PARSE ステートメントにより、COBOL ランタイムの一部である高速 XML パーサーへのインターフェースが提供されます。 XML PARSE ステートメントは、構文解析して XML 文書を個々の部分に分け、各部分を 1 つずつユーザー作成の処理プロシージャーに渡します。

  以下の XML 特殊レジスターが、XML パーサーとユーザー作成の処理プロシージャーとの間の情報の通信に使用されます。
  - XML-CODE
  - XML-EVENT
  - XML-NTEXT
  - XML-TEXT
- 代替レコード・キーのサポート

ALTERNATE RECORD KEY 文節は、索引付きファイルに関連する代替レコード・キーの定義を可能にします。これらの代替キーを使用すると、ファイル・レコードに異なる論理的順序付けを用いてファイルにアクセスすることができます。

- DBCS データ項目名 (DBCS ワードのサポート)
- 63 桁サポート
  - パック 10 進数、ゾーン 10 進数、および数字編集項目の最大長は、31 桁から 63 桁の数字に拡張されました。
  - CRTCBLMOD コマンド、CRTBNDCBL コマンド、および PROCESS ステートメントの中の ARITHMETIC パラメーターには、新しい EXTEND63 オプションがあります。
- 以下の 7 つの新しい ANSI 組み込み関数があります。
  - INTEGER
  - REM
  - ANNUITY
  - INTEGER-PART
  - MOD
  - FACTORIAL
  - RANDOM
- 以下の新しい CRTBNDCBL/CRTCBLMOD オプションがあります。
  - *NOCRTARKIDX/*CRTARKIDX は、永続的な索引が検出できない場合に、代替レコード・キーによる一時的な索引を作成するかどうかを指定します。
  - *STDINZHEX00 は、値の文節を持たないデータ項目は、16 進数の 0 で初期化されることを指定します。
  - ARITHMETIC パラメーター用の *EXTEND63 オプションは、63 桁までの固定小数点演算の中間結果の精度を増やします。
- 以下の新しい PROCESS ステートメントのオプションがあります。
  - PROCESS ステートメントの NOCOMPRESSDBG/COMPRESSDBG オプションは、 DBGVIEW オプションで *LIST または *ALL が指定されたときに、コンパイラーがリスト・ビューを圧縮するべきかどうかを指示します。
  - NOCRTARKIDX/CRTARKIDX
  - STDINZHEX00
  - ARITHMETIC パラメーターの EXTEND63 オプション
- プログラム状況構造体

  プログラム状況構造体は、COBOL プログラムがエラーを受け取ったときにエラー情報を格納する、事前定義の構造体です。 PROGRAM STATUS 文節は、受け取るエラー情報を指定します。

V5R2 での変更点

以下に、V5R2 において ILE COBOL に対して行われた機能強化について説明します。

- 再帰的プログラムのサポート

  再帰的プログラムをサポートするために、オプションの RECURSIVE 文節が追加されました。これらの再帰的プログラムは、再帰再入が可能な COBOL プログラムです。

- ローカル・ストレージ・セクションのサポート

  呼び出しごとに割り振りと解放が行われるストレージを定義する、新規データ・セクションが追加されました。再帰的プログラムおよび非再帰的プログラムの両方で、ローカル・ストレージ・セクションを指定できます。

- Java の相互運用性

  Java の相互運用性を高めるために、2 つの新規機能が追加されました。以下の機能です。

  – UTF8String 組み込み関数

    この関数を使用すると、ストリングを UTF-8 形式に変換することができます。

  – PCML のサポート

    ユーザーが、コンパイラーに、COBOL プログラムの PCML ソースを生成するよう指示できるように、CRTCBLMOD コマンドおよび CRTBNDCBL コマンドに新規パラメーターが追加されました。ユーザーが、INFOSTMF パラメーターに PGMINFO(*PCML) およびストリーム・ファイルの名前を指定した場合、コンパイラーは、指定されたストリーム・ファイル内に PCML を生成します。PCML が生成されると、より少ない Java コードで Java プログラムが、この COBOL プログラムを簡単に呼び出せるようになります。

- 追加の組み込み関数

  このリリースには、新規の組み込み関数がいくつか追加されています。以下の関数です。

  – Max
  – Median
  – Midrange
  – Min
  – ORD-Max
  – ORD-Min
  – Present Value
  – Range
  – Standard Deviation
  – Sum
  – Variance

- IFS

  IFS ストリーム・ファイルに保管されている ILE COBOL ソースはコンパイルすることができます。ユーザーが、コンパイラーに、IFS ストリーム・ファイルに保管されているソースからコンパイルするよう指示できるように、CRTCBLMOD コマンドおよび CRTBNDCBL コマンドに SRCSTMF と INCDIR のパラメーターが追加されました。

V5R1 での変更点

以下に、V5R1 において ILE COBOL に対して行われた機能強化について説明します。

- UCS-2 (Unicode) サポート

  新しいデータ項目のタイプである「国別データ」が追加され、ISO/IEC 10646-1 で UCS-2 として指定されたコード化文字セットに対するサポートを提供します。このコード・セットは、Unicode 規格で定義された基本セットです。

  – UCS-2 文字セット

    このコード化文字セットは、世界中で使用されている基本のスクリプトで表示される各文字ごとに、固有なコードを提供します。それぞれの文字は、16 ビット (2 バイト) コードで表されます。

  – 国別データ

    この新しいデータ項目のタイプには、UCS-2 コード・セットを使用してコード化されたデータが含まれます。その記述に USAGE NATIONAL 文節が含まれる基本データ項目、またはその記述に USAGE NATIONAL 文節が含まれるグループ項目に従属する基本データ項目は、国別データ項目です。

  – NTLPADCHAR コンパイラー・オプションおよび PROCESS ステートメント・オプション

    このオプションにより、SBCS 埋め込み文字、DBCS 埋め込み文字、および国別埋め込み文字の 3 つの値を指定することができます。ある値が国別データ・タイプ項目に移動され、その国別データ・タイプ項目が完全に埋め込まれない場合は、該当する埋め込み文字が使用されます。

  – ALL 国別リテラル

    国別 16 進リテラルが許されるところで ALL という語を使用することが可能であり、それによって、たとえば、すべての UCS-2 ブランクを国別データ項目に移動することができます。

  – PROCESS ステートメントのオプション NATIONAL

    このオプションが指定されると、PICTURE 記号 N を使用して定義された基本データ項目は、暗黙の USAGE NATIONAL 文節を持つようになります。このコンパイラー・オプションが使用されないと、これらの項目に対して USAGE DISPLAY-1 文節が暗黙に指定されます。

  – 国別 16 進リテラル

国別データ値が含まれるリテラルは、以下の構文を使用して指定することができます。

```
NX"hexadecimal-character-sequence..."
```

– 形象定数

形象定数 SPACE/SPACES は、国別データ項目と一緒に使用された場合、1 つ以上の UCS-2 1 バイト・スペース文字 (U+0020) を表します。

- JAVA 相互運用性サポート
  – QCBLLESRC.JNI ファイル

    このファイルは JNI.h ファイルで提供されるのと同じ定義とプロトタイプを提供しますが、C ではなく COBOL で書かれています。
  – Java と COBOL データ・タイプの間のデータ・マッピング
- メインフレーム移植性サポート
  – NOCOMPASBIN/COMPASBIN PROCESS ステートメント・オプションは、USAGE COMPUTATIONAL または COMP が、USAGE COMP-3 または USAGE COMP-4 と同じ意味を持つかどうかを指示します。
  – NOLSPTRALIGN/LSPTRALIGN PROCESS ステートメント・オプションは、USAGE POINTER または PROCEDURE-POINTER を指定したデータ項目が、リンケージ・セクションのレコードの先頭から相対的に、16 バイトの倍数で位置合わせされるかどうかを指示します。
  – NOADJFILLER/ADJFILLER PROCESS ステートメント・オプションは、ポインター・データ項目の位置合わせのためにコンパイラーによって挿入される暗黙の充てん文字を、グループの最初のメンバーとしてポインター・データ項目を持つグループの前に挿入するか後に挿入するかを指示します。
  – 複合 OCCURS DEPENDING ON (ODO) サポート

    複合 ODO は、以下のように構成されます。
    - 1 つの OCCURS または 1 つの ODO 文節のサブジェクトに従属する記入項目は、ODO 文節を含むことができる (可変長エレメントを持つテーブル)。
    - 1 つの ODO によって記述されたデータ項目の後に、ODO 文節によって記述された非従属データ項目を続けることができる (可変位置テーブル)。
    - ODO 文節を含む記入項目の後に、非従属項目を続けることができる (可変位置フィールド)。ただし、これらの非従属項目は、ODO 文節のオブジェクトにすることはできません。
    - ODO 文節を含む項目に続く、従属項目または非従属項目の位置は、ODO オブジェクトの値による影響を受ける。
    - ODO 文節を含む従属項目を持つテーブルに対して、INDEXED BY 句を指定することができる。
- CRTCBLMOD および CRTBNDCBL コマンドに LICOPT パラメーターが追加され、上級ユーザーは、ライセンス内部コード・オプションを指定することができます。

- OPTVALUE PROCESS ステートメント・オプションは、作業用ストレージ・セクションの VALUE 文節を含んでいるデータ項目を初期設定するコードの生成を最適化するかどうかを指示します。

V4R4 での変更点

以下に、V4R4 において ILE COBOL に対して行われた機能強化について説明します。

- スレッド・セーフティー・サポート

  Lotus® Domino® または Java など、スレッド化されたアプリケーションから ILE COBOL プロシージャーを呼び出すためのサポート。 PROCESS ステートメントに THREAD パラメーターが追加され、マルチスレッド環境で ILE COBOL モジュールが使用可能になりました。そのモジュール内のプロシージャーのアクセスは、逐次化されている必要があります。

- 31 桁サポート
  - パック 10 進数、ゾーン 10 進数、および数字編集項目の最大長は、18 桁から 31 桁の数字に拡張されました。
  - ARITHMETIC パラメーターが、CRTCBLMOD コマンド、CRTBNDCBL コマンド、および PROCESS ステートメントに追加され、数値データに対して算術モードを設定できるようになりました。これにより、数値データの計算動作を指定することができます。
- ユーロ通貨サポート
  - 参加国の間で 1999 年 1 月から 3 年間有効になる二重通貨システムをサポートするため、COBOL プログラム内で複数の通貨記号を指定する機能。
  - 複数文字で通貨記号を表すことにより、単一文字通貨記号 (たとえば、「¥」) だけでなく国際通貨記号 (たとえば、USD、FRF、DEM、EUR) を COBOL 編集フィールドに指定できる機能。
  - OPTION パラメーター値 *MONOPIC/*NOMONOPIC が CRTCBLMOD および CRTBNDCBL コマンドに追加され、MONOPIC/NOMONOPIC が PROCESS ステートメントに追加されました。これにより、PICTURE 文字ストリングの中で、大文字の通貨記号か大文字小文字を区別する通貨記号かのいずれかを選択することができます。

V4R2 での変更点

以下に、V4R2 において ILE COBOL に対して行われた機能強化について説明します。

- ユーザー定義データ・タイプ

  ユーザー定義データ・タイプは、TYPEDEF 文節が含まれるレベル 01 記入項目を指定することにより定義されます。このレベル 01 記入項目に従属するすべての記入項目が、ユーザー定義データ・タイプの一部として考えられます。ユーザー定義データ・タイプは、そのユーザー定義データ・タイプを参照する新しいデータ項目に対して TYPE 文節を指定することにより、レベル 01、77、または 02 ～ 49 の新しいデータ項目を定義するのに使用できます。

- プログラム・プロファイル・サポート

PRFDTA パラメーターが、CRTCBLMOD と CRTBNDCBL の両方のコマンドおよび PROCESS ステートメントに追加され、最適化のためにプログラムでプロファイルを作成することができます。

- ヌル値サポート

  以下のステートメントおよび文節にヌル値サポート (NULL-MAP および NULL-KEY-MAP キーワードを使用する) が追加され、データベース・レコード内でヌル値を操作することができるようになりました。
  - ASSIGN 文節
  - COPY-DDS ステートメント
  - DELETE ステートメント
  - READ ステートメント
  - REWRITE ステートメント
  - START ステートメント
  - WRITE ステートメント

- ロケール・サポート

  i5/OS ロケール・オブジェクト (*LOCALE) は、日付形式または時刻形式のような、特定の文化圏固有のエレメントを指定します。この文化圏固有の情報は、ILE COBOL の日付項目、時刻項目、および数字編集項目に関連付けることができます。以下の新しい文字、文節、句、およびステートメントがこれをサポートするために追加されました。
  - SPECIAL-NAMES 段落の LOCALE 文節
    - i5/OS ロケール・オブジェクトと COBOL 簡略名を関連付けます
  - 日付項目、時刻項目、または数字編集項目の LOCALE 句
    - データ項目を i5/OS ロケール・オブジェクトと関連付けるようにするために、ロケール簡略名を指定することができます
  - SPECIAL-NAMES 段落の LOCALE 文節に定義された特定のロケールと一緒に、現行ロケールとデフォルト・ロケールが定義されています。現行ロケールは、新しい SET LOCALE ステートメント (形式 8) を使用して変更することができます。
    - 1 つのロケール・オブジェクトは複数のロケール・カテゴリーから構成され、各ロケール・カテゴリーは SET LOCALE ステートメントを使用して変更できます。
  - ロケール・カテゴリーは、LC_TIME および LC_MONETARY のような名前を持っています。これらの名前には、下線文字が組み込まれています。この下線文字が COBOL 文字セットに追加されています。
    - COPY DDS ステートメントの SUBSTITUTE 句は、下線文字を取り込めるよう拡張されています。

  以下の新しい組み込み関数によって、特定の文化圏固有の日付と時刻を文字ストリングとして戻すことができます。
  - LOCALE-DATE
  - LOCALE-TIME

- 世紀サポートに対する追加

  ILE COBOL の世紀サポートに対して、以下の機能強化が行われています。

– データ項目の新しいクラスである、日時クラスが追加されています。日時クラスには、日付、時刻、およびタイム・スタンプのカテゴリーが含まれています。日時データ項目は、データ記述記入項目の新しい FORMAT 文節を使用して宣言されます。
– COPY-DDS と、CVTOPT コンパイラー・パラメーターの以下の値を使用して、i5/OS DDS データ・タイプの日付、時刻、およびタイム・スタンプを、COBOL の日付、時刻、およびタイム・スタンプの項目として、COBOL プログラムに取り込むことができます。
  - *DATE
  - *TIME
  - *TIMESTAMP
– CVTOPT パラメーター値 *CVTTODATE を使用して、DATFMT キーワードを指定したパック、ゾーン、および文字の i5/OS DDS データ・タイプを、データ項目として COBOL に取り込むことができます。
– 以下の新しい組み込み関数により、日時クラスの項目の演算を行い、項目を日時クラスに変換し、日時項目が有効かどうかを確認するためのテストを行い、日時項目の部分を抽出することができます。
  - ADD-DURATION
  - CONVERT-DATE-TIME
  - EXTRACT-DATE-TIME
  - FIND-DURATION
  - SUBTRACT-DURATION
  - TEST-DATE-TIME

V3R7 での変更点

以下に、V3R7 において ILE COBOL に対して行われた機能強化について説明します。

• 世紀サポート

  ユーザーが 4 桁の年を使用して作業するための機能が、以下のステートメントおよび関数に追加されています。
  – YYYYDDD および YYYYMMDD 句を指定した ACCEPT ステートメント
  – 2 桁の年を 4 桁の年に変換する、以下の組み込み関数
    - DATE-TO-YYYYMMDD
    - DAY-TO-YYYYDDD
    - YEAR-TO-YYYY
  – 4 桁の年を戻す、以下の組み込み関数
    - CURRENT-DATE
    - DAY-OF-INTEGER
    - DATE-OF-INTEGER
    - WHEN-COMPILED

• 浮動小数点サポート

  CRTCBLMOD および CRTBNDCBL コマンドの CVTOPT パラメーターの *FLOAT 値により、ILE COBOL プログラム内で浮動小数点データ項目を使用することができます。さらに、影響を受けるステートメント (ACCEPT、

DISPLAY、MOVE、COMPUTE、ADD、SUBTRACT、MULTIPLY、および DIVIDE など) は、浮動小数点をサポートします。

- データ域サポート

  ACCEPT および DISPLAY ステートメントの新しい形式が追加され、i5/OS データ域の内容の検索と更新の機能を提供するようになりました。

- 組み込み関数

  以下の組み込み関数が追加されています。

  ACOS
  ASIN
  ATAN
  CHAR
  COS
  CURRENT-DATE
  DATE-OF-INTEGER
  DAY-OF-INTEGER
  DATE-TO-YYYYMMDD
  DAY-TO-YYYYDDD
  INTEGER-OF-DATE
  INTEGER-OF-DAY
  LENGTH
  LOG
  LOG10
  LOWER-CASE
  MEAN
  NUMVAL
  NUMVAL-C
  ORD
  REVERSE
  SIN
  SQRT
  TAN
  UPPER-CASE
  WHEN-COMPILED
  YEAR-TO-YYYY

- バインド・ディレクトリー・パラメーター -- BNDDIR

  CRTBNDCBL コマンドに BNDDIR パラメーターが追加され、記号の解決に使用されるバインド・ディレクトリーのリストを指定することが可能になりました。

- 活動化グループ・パラメーター -- ACTGRP

  CRTBNDCBL コマンドに ACTGRP パラメーターが追加され、プログラムが呼び出されるときに、そのプログラムが関連付けられる活動化グループを指定することが可能になりました。

- ライブラリー修飾されたプログラム・オブジェクトおよびデータ域

  以下の ILE COBOL ステートメントに LIBRARY 句が追加され、OS/400® のプログラム・オブジェクトとデータ域を OS/400 ライブラリー名を使用して修飾することが可能になりました。
  - CALL
  - CANCEL
  - SET
  - ACCEPT
  - DISPLAY

- パフォーマンス収集データ

  ENBPFRCOL パラメーターが、CRTCBLMOD コマンド、CRTBNDCBL コマンド、および PROCESS ステートメントに追加され、モジュールまたはプログラム内で、パフォーマンス測定コードを生成することが可能になりました。収集されたデータは、システム・パフォーマンス測定ツールで使用して、アプリケーションのパフォーマンスのプロファイルを作成することができます。

- 新しい ILE デバッガー・サポート

  ILE デバッガーでは、以下のことが可能になりました。
  - ほとんどの OPM プログラムをデバッグすること
  - 監視条件を設定すること。これは、変数 (または保管場所のアドレスを決定する式) の値が変更されたときに、停止点を設定する要求となります。

V3R6/V3R2 での変更点

以下に、V3R6 および V3R2 において ILE COBOL に対して行われた機能強化について説明します。

- 新しい EXIT PROGRAM 句

  EXIT PROGRAM ステートメントに AND CONTINUE RUN UNIT 句が追加され、実行単位を停止せずに、呼び出し側プログラムを終了させることが可能になりました。
- 新しい SET ステートメントのポインター形式

  SET ステートメントの新しい形式が追加され、ポインター参照の更新が可能になりました。
- DBCS データ・サポート

  2 バイト文字セット (DBCS) データを ILE COBOL で処理できるようになりました。 ILE COBOL コンパイラーは DBCS (それぞれの論理文字が 2 バイトで表される) をサポートします。 DBCS は、IBM 日本語図形文字セット (漢字) などの表意文字言語に対するサポートを提供します。
- CALL...BY VALUE および CALL...RETURNING のサポート

  CALL...BY VALUE および CALL...RETURNING により、BY REFERENCE の代わりに BY VALUE で引き数を渡し、RETURN 値を受け取る機能が提供されます。これにより、ILE C for i5/OS と ILE RPG for i5/OS の両方が CALL... BY VALUE と CALL...RETURNING をサポートすることになり、マイグレーションがより容易になり、言語相互間のサポートが向上します。
- PROCEDURE DIVISION ヘッダーの BY VALUE 句および RETURNING 句のサポート

  PROCEDURE DIVISION ヘッダーの BY VALUE 句により、COBOL は、呼び出し側 COBOL プログラムまたは RPG、C、または C++ などの他の ILE 言語から、BY VALUE 引き数を受け取ることができます。 PROCEDURE DIVISION ヘッダーの RETURNING 句により、COBOL は、呼び出し側 ILE プロシージャーに VALUE を戻すことができます。

V3R1 での変更点

以下に、V3R1 において ILE COBOL に対して行われた機能強化について説明します。

- EXTERNAL データ項目

EXTERNAL 文節を使用することにより、ILE COBOL 実行単位内のすべてのプログラムで使用可能なデータ項目を定義することができます。プログラム間で共用されるすべての変数を、CALL ステートメントの引き数として渡す必要はなくなりました。このサポートは、CALL ステートメントの引き数とパラメーターを使用せずにデータを共用できるようにすることで、アプリケーションのより大きなモジュール性を促進します。

- EXTERNAL ファイル

  実行単位内のすべてのプログラムで使用可能なファイルを定義することができます。ファイルを EXTERNAL として宣言している、実行単位内のどの ILE COBOL プログラムからでも、同じファイルに対する入出力要求をシームレスに行うことができます。外部ファイルの場合、そのファイルを使用しているプログラムの数には関係なく、ファイル・カーソルは 1 つだけしかありません。ファイルを複数のプログラムで共用することができ、それによって、より小さな、保守性に優れたプログラムを開発することができます。 EXTERNAL ファイルを使用することは、そのファイルを使用するすべての関連プログラムについて、1 つの OPEN および CLOSE 操作しか必要がないため、共用オープン・ファイルを使用することの利点が得られます。ただし、EXTERNAL ファイルは、異なる活動化グループ内では共用できず、また他のプログラム言語で書かれたプログラムとも共用できません。

- ネストされたソース・プログラム

  1 つの ILE COBOL ソース・プログラムに、他の ILE COBOL ソース・プログラムを含めることができます。これらの含められたプログラムでは、それらが含められている元のプログラムのデータ項目やファイルなどの一部のリソースを参照するか、または定義しているプログラムだけが見ることができるリソースをローカルで定義することができます。 ILE COBOL プログラムはそれ自体がリソースであるため、その有効範囲は、そのプログラムに付加されたネスト構造および有効範囲属性によっても制御されます。これによって、1 つの ILE COBOL プログラムによって呼び出すことができる、ILE COBOL プログラムのセットの制御の柔軟性が大幅に増します。ネストされた ILE COBOL プログラムでは、リソースを隠蔽して、見ることができないようにするメカニズムを提供します。

- INITIAL 文節

  1 つの ILE COBOL プログラムとその中に含まれるすべてのプログラムが、呼び出されるたびに初期状態に置かれるようなメカニズムがあります。これは、PROGRAM-ID 段落に INITIAL を指定することによって可能になります。これによって、COBOL 実行単位の制御の柔軟性がさらに増します。

- REPLACE ステートメント

  REPLACE ステートメントは、コンパイルの処理中にソース・プログラムのテキストを置き換えるのに便利です。このステートメントは、REPLACING 句を指定した COPY ディレクティブとは異なり、ファイル全体か、または別の REPLACE ステートメントに出会うまで操作されます。 REPLACE ステートメントは、すべての COPY ステートメントが処理された後で処理されます。これにより、コンパイルすべき ILE COBOL テキストの変更の柔軟性がさらに増します。

- DISPLAY WITH NO ADVANCING ステートメント

DISPLAY ステートメントで NO ADVANCING 句を使用することによって、カーソルを、表示された最後の文字の後にそのまま置いておく機能が得られます。これにより、単一行に表示すべき項目を、ILE COBOL プログラム内のさまざまな点から集めて 1 つのストリングにすることができます。

- ACCEPT FROM DAY-OF-WEEK ステートメント

  ILE COBOL では、曜日 (月曜日 = 1、火曜日 = 2 ...) を受け入れ、それに ID を割り当てることができるようになりました。このサポートは、既存の ACCEPT FROM DAY/DATE/TIME サポートを補うものです。

- 相対ファイルに対する SELECT OPTIONAL 文節

  これにより、ファイルが I-O でオープンされた場合であっても、相対ファイルの自動作成が可能になります。これは、順次ファイルの場合にすでに使用可能であるサポートを拡張するものです。

- ネストされた COPY ステートメントのサポート

  COPY メンバーに COPY ステートメントを含めることができ、それによって COPY ステートメントの能力が拡張されます。 COPY メンバーに COPY ディレクティブが含まれている場合は、COPY ディレクティブを含めた側にも、含められた COPY ディレクティブにも、REPLACING 句を指定することができません。

- 拡張 ACCEPT および DISPLAY ステートメントに対する機能強化

  拡張 ACCEPT ステートメントで、テーブルの作業を行うことができます。これにより、テーブルのエレメントを、容易かつ選択的に更新することができます。

  拡張 ACCEPT および DISPLAY ステートメントでは、可変長テーブルも許されます。

  また、拡張 ACCEPT では、SIZE 文節もサポートされます。

- プロシージャー・ポインターのサポート

  プロシージャー・ポインターは、ILE COBOL プログラムまたは非 ILE COBOL プログラムのアドレスを入れることができる新しいデータ・タイプです。プロシージャー・ポインターは、データ項目に USAGE IS PROCEDURE-POINTER 文節を指定することによって定義されます。この新しいデータ・タイプは、パラメーターとしてこのタイプのデータ項目を期待している、呼び出し側プログラムまた ILE プロシージャーにとって便利です。また、プロシージャー・ポインター・データ項目は、別のプログラムを呼び出すための CALL ステートメントのターゲットとしても使用することができます。

- 新しい特殊レジスター
  - RETURN-CODE 特殊レジスター

    戻り情報を ILE COBOL プログラム間で渡すことができます。通常、このレジスターは、呼び出し先プログラムの成功または失敗に関する情報を渡すのに使用されます。

  - SORT-RETURN 特殊レジスター

SORT または MERGE ステートメントの成功に関する情報を戻します。またこれによって、エラー宣言または入出力プロシージャーから、SORT/MERGE の処理を終了させることもできます。

- 新しいコンパイラー・オプション
  - *PICGGRAPHIC/*NOPICGGRAPHIC

    *PICGGRAPHIC は CVTOPT オプションの新しいパラメーターであり、それによって、ユーザーは DBCS データを ILE COBOL プログラムに取り込むことができます。
  - *IMBEDERR/*NOIMBEDERR オプション

    *IMBEDERR は新しいコンパイラー・オプションであり、それによって、コンパイラー・リストの終わりだけでなく、発生時点でコンパイル時エラーをコンパイラー・リストに組み込みます。
  - *FLOAT/*NOFLOAT

    *FLOAT は CVTOPT オプションの新しいパラメーターであり、それによって、DDS 名と COMP-1 (単精度) または COMP-2 (倍精度) の USAGE を使用して、浮動小数点データ項目を ILE COBOL プログラムに取り込むことができます。
  - *NOSTDTRUNC/*STDTRUNC オプション

    *NOSTDTRUNC は新しいコンパイラー・オプションであり、BINARY データ項目の切り捨てを抑止します。このオプションは、IBM System/390® からアプリケーションをマイグレーションするのに便利です。
  - *CHGPOSSGN/*NOCHGPOSSGN オプション

    このオプションは、OS/400 と IBM S/390® の間でデータを共用するのに便利です。このオプションは、IBM System/390 との互換性のために提供されています。このオプションは、符号付きのパックおよびゾーンのデータ項目が算術ステートメントまたは MOVE ステートメントで使用され、それらのデータ項目の値が正である場合に、それらのビット表記を変更します。
- 引用符付きシステム名のサポート

  システム名が許されるところで、リテラルが許されるサポートが追加されています。システムがサポートする名前であればどのような名前でも使用することができ、有効 COBOL 名に対する制限は無くなりました。
- 以下の機能については、COBOL での制限が無くなり、システムの制約によって決定されるようになりました。
  - 宣言済みファイルの数。
  - CALL ステートメントおよび手続き部 USING 句のパラメーターの数。ここでは、ILE プロシージャーの場合の 400 というシステム制限と、プログラム・オブジェクトの場合の 255 というシステム制限が適用されます。
  - SORT-MERGE 入力ファイルの数および SORT-MERGE キーの数。
    SORT-MERGE 入力ファイルの最大数は 32 であり、SORT-MERGE キーの最大長は 2000 バイトです。
- NO LOCK 付きの START ステートメント。

START ステートメントで NO LOCK 句を使用することにより、レコード上にロックを置かずに、読み取るべき最初のレコードにファイル・カーソルが位置付けられます。このサポートは、索引付きファイルおよび相対ファイルに対して提供され、すでに使用可能になっている、NO LOCK 付きの READ 機能を補うものです。

注: NO LOCK 付きの START は、ILE COBOL と OPM COBOL/400 の両方における新しいステートメントです。

- 静的プロシージャー呼び出しサポート

  より小さく、保守性に優れたモジュール・オブジェクトでアプリケーションを開発して、それらを一緒にリンクして、動的プログラム呼び出しのオーバーヘッドによる不利益を被ることなく、1 つのプログラム・オブジェクトにすることができます。またこの機能は、システムによって提供される共通の実行時環境と相まって、アプリケーションを混合言語で作成する能力も向上させます。 ILE プログラム言語では、ソース言語が混合しているかどうかに関係なく、C、RPG、COBOL、および CL を単一のプログラム・オブジェクトにバインドすることが許されます。

  CALL リテラル・ステートメントの新しい構文と新しいコンパイラー・オプションが ILE COBOL に追加され、静的プロシージャー呼び出しと動的プログラム呼び出しを区別するようになりました。

- 可変長レコード・サポート (RECORD IS VARYING 文節)

  標準の ANSI COBOL 構文を使用して、同じファイル上で長さが異なるレコードを定義し、それらを容易に使用することができます。これは、ストレージを大幅に削減するだけでなく、他のシステムから複雑なアプリケーションをマイグレーションする作業も容易にします。

- 拡張されたコンパイラー限界値

  ILE COBOL は、以下の拡張されたコンパイラー限界値を提供します。
  - グループ・データ項目および基本データ項目のサイズ
  - 固定長および可変長のテーブルのサイズ
  - 条件ステートメントのネスト・レベルの数
  - さまざまな手続き部ステートメントのオペランドの数

工業規格

本書全体を通して、標準 COBOL とは、次の文書で定義されている COBOL プログラミング言語のことを言います。

- American National Standard for Information Systems - Programming Language - COBOL, ANSI X3.23-1985, ISO 1989:1985。これについては、以下の文書の内容に従って、下記の順序で更新されています。
  - ANSI X3.23a-1989, American National Standard for Information Systems - Programming Language - Intrinsic Function Module for COBOL and ISO 1989:1985/Amd.1:1992, Programming Languages - COBOL, Amendment 1: Intrinsic function module

– ANSI X3.23b-1993, American National Standard for Information Systems - Programming Language - Correction Amendment for COBOL and ISO/IEC 1989 DAM2 Programming Languages - COBOL, Amendment 2: Correction and clarification amendment for COBOL

ILE COBOL コンパイラーは、標準 COBOL (上記で定義されているもの) および以下をサポートできるように設計されています。

- FIPS Publication 21-4, Federal Information Processing Standard 21-4, COBOL

これは、中間のサブセット・レベルにあるものとして IBM では解釈しています (1995 年 1 月現在)。

本書における標準 COBOL とは、上記で述べた ANSI 規格に従ったものとします。

このマニュアルの一部は、標準 COBOL の資料から転載、もしくは許可を受けて複製したものです (著作権 1985 年、米国規格協会)。その資料の入手先は、American National Standard Institute at 1430 Broadway, New York, New York, 10018 です。

COBOL 言語は、ANSI Technical Committee X3J4 によって維持管理されています。

ILE COBOL コンパイラーがサポートしている業界標準の詳細については 665 ページの『付録 A. 言語サポートのレベル』を参照してください。

使用承諾について

以下は、ユーザーのための情報および指針とするために、U.S. Government Printing Office Form Number 1965-0795689 からの抜粋を和訳したものです。

どのような組織であっても、この全体または一部の COBOL レポートおよび仕様を複製しようとする場合、指示書の基礎としてまたは他の目的でこのレポートのアイデアを無料で使用できます。しかし、そのような組織はすべて、文書の序文としてこの部分を複製しなければなりません。書評などで短い引用を行う場合、出所の承認では COBOL に言及しなければなりませんが、そのすべての部分を引用する必要はありません。

COBOL は産業用言語であり、単一または複数の会社が所有しているものでも、単一または複数の組織が所有しているものでもありません。

プログラミング・システムおよび言語の正確さや機能について、寄稿者または COBOL 委員会は明示的にも暗示的にも何の保証もしません。さらに、関係する事柄について寄稿者や委員会が責任を負うこともありません。

COBOL の保守に関するプロシージャーは確立されています。変更をするためのプロシージャーについての照会は、データ・システム言語に関する会議の実行委員会に対して行う必要があります。

著作権が設定されている資料の著者および著作権保持者は次のとおりです。

- Programming for the UNIVAC® I and II, Data Automation Systems copyrighted 1958, 1959, by Unisys Corporation;
- IBM Commercial Translator, Form No. F28-8013, copyrighted 1959 by IBM;
- FACT, DSI 27A5260-2760, copyrighted 1960 by Minneapolis-Honeywell

これらの企業は、COBOL の仕様において、この資料の全体または一部を使用する権限が特別に付与されています。この権限には、プログラミング資料または同様の出版物での COBOL 仕様の再編成および使用までが含まれます。

ILE COBOL 構文表記法

ILE COBOL での基本形式は、一定の構文表記法に従って示されます。この表記法は、COBOL ソース・ステートメントを書く際に役立つように設計されています。

- COBOL キーワードおよびオプションの語は大文字で示してあります。次に例をあげます。

  ```
  MOVE
  ```

  これは示されたとおり正確に入力しなければなりません。キーワードが脱落していると、コンパイラーはそれをエラーと見なします。
- ユーザーが指定する名前または値を表記する変数は、すべて小文字 (または日本語) で示されます。次に例をあげます。

  ```
  parmx
  ```
- 本文中で参照しやすいように、次のように、後にハイフンおよび数字または英字を付けた語があります。

  ```
  identifier-1
  ```

  この接尾部が付いていても、その語の構文上の定義は変わりません。
- 構文の形式の中にある算術演算子および論理演算子 (+、-、*、/、**、>、<、=、>=、および <=) は、必ず指定しなければなりません。 ILE COBOL の予約語の完全なリストについては、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。
- 図に記載されているすべての句読点および他の特殊文字は、それが表示されている場合には、その形式の構文で必要です。それらを含めない場合、プログラムでエラーになります。
- 必要およびオプショナルの文節が使用されている場合は、図中に示されている順序でそれをコーディングする必要があります。ただし、関連する規則が明示的に別の方法について言及している場合を除きます。

構文図の読み方

本書では、構文を下記に定義した構造で記述しています。

- 構文図は、左から右へ、上から下へ次のような線の経路に従って読みます。

  ►►── これはステートメントの開始を示します。文節、句、および段落のようなステートメント以外の構文単位の図も、この記号で始まります。

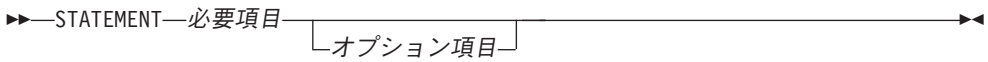
  ──► これはステートメントの構文が、次の行に継続することを示します。

  ►── これはステートメントの構文が、前の行から継続していることを示します。

  ──►◄ これはステートメントの終わりを示します。文節、句、および段落のようなステートメント以外の構文単位の図も、この記号で終わります。

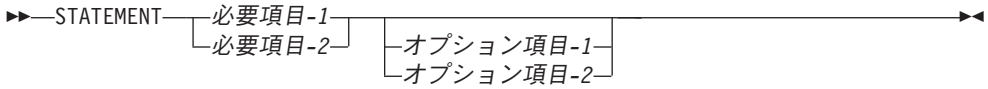
注: 段落全体の図の中にあるステートメントが ▶▶— で始まり —▶◀ で終わるのは、その始まりか終わりが段落の始まりか終わりと一致する場合だけです。

- 必要項目は、水平線 (主パス) 上に示してあります。オプション項目は、主経路より下に示されます。



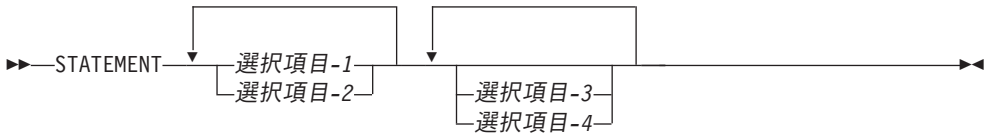
- 複数の項目の中から選択できる場合には、縦に重ねて示されます。

  それらの項目のどれか 1 つを選択しなければならない場合、重ねられた項目の 1 つが主経路上に示されます。 1 つの項目の選択がオプショナルである場合は、重ねられた項目全体が主経路より下に示されます。



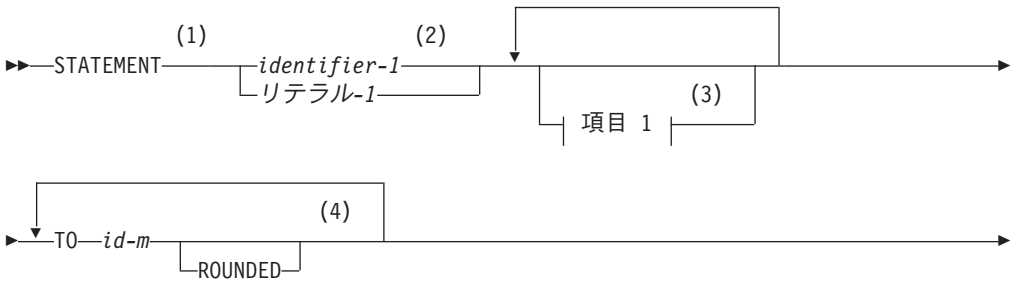
- 項目の上側に左向きの矢印がある場合には、その項目を繰り返して指定できることを表しています。

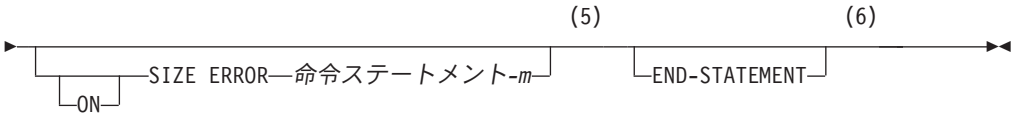
- 必要項目またはオプション項目の上の繰り返しの矢印は、積み重なった項目の中から複数選択することができるか、または 1 つの選択項目を繰り返すことができることを示します。



次の例で、構文の使い方を示します。

形式





項目 1:

注:

1 STATEMENT キーワードを指定し、図のようにコーディングしなければなりません。

2 このオペランドは必要です。「id-1」または「リテラル-1」のいずれかをコーディングしなければなりません。

3 「項目 1」の断片はオプションです。これは、アプリケーションの要求に応じてコーディングしてもしなくてもかまいません。項目 1 をコーディングする場合、各項目を 1 つまたは複数の COBOL 区切り文字で区切って繰り返すことができます。この断片に許される項目の選択範囲は、図の下の方に示されています。

4 オペランド「id-m」とそれに関連した「TO」キーワードは必要であり、各項目を 1 つまたは複数の COBOL 区切り文字で区切って繰り返すことができます。各記入項目には、キーワード ROUNDED を割り当てることができます。

5 「命令ステートメント-m」に関連した ON SIZE ERROR 句はオプションです。 ON SIZE ERROR 句をコーディングする場合、キーワード ON はオプションです。

6 END-STATEMENT キーワードをステートメントの終わりにコーディングできます。これは、必要な区切り文字ではありません。

文書化のための構文の識別

構文図中に示される COBOL 文節およびステートメントのうち、ILE COBOL コンパイラーによって構文検査は行われるが、文書化のための記述として取り扱われるものは、脚注を付けて示されます。

制御言語 (CL) 入力コードの解釈

各 CL 構文図の右上の部分に表示されているコードには、コマンドが入力できる環境を指定する入力コードが含まれています。このコードは、以下のことがコマンドで可能かどうかを示します。

- バッチまたは対話式のジョブ (コンパイル済みプログラムの外側) で使用される (ジョブ:B または I)
- バッチまたは対話式のコンパイル済みプログラムで使用される (プログラム:B または I)
- バッチまたは対話式の REXX プロシージャーで使用される (REXX:B または I)

- CALL CL コマンドのパラメーターとして使用されるか、または文字ストリングとしてシステム・プログラム QCMDEXC (Exec) に渡される。

クライアント製品のアプリケーション開発ツールの使用

「Remote Systems」ビュー、System i® テーブル・ビュー、および Remote Systems LPEX Editor を使用して、ご使用の System i で最もよく実行される開発タスクを行うことができます。これらのビューおよびその関連機能は、クライアント製品の Remote System Explorer パースペクティブから使用可能です。

PDM での作業になれている場合、テーブル・ビューで同様のサポートを利用できます。 SEU での作業になれている場合、Remote Systems LPEX Editor により同様の操作を行うことができます。

注: これらのトピック全体に関する *チュートリアル* については、クライアント製品の Tutorials Gallery にある次のチュートリアルを参照してください。

- *Remote System Explorer を使用した ILE COBOL の管理 (英語)*
- *Remote System Explorer を使用した ILE RPG の管理 (英語)*

詳細は、次のトピックを参照してください。

Remote System Explorer パースペクティブご使用に際して
Remote System Explorer パースペクティブにより、ご使用のシステムですべてのアイテムにアクセス、編集、実行、コンパイル、およびデバッグできます。

「Remote Systems」ビュー
「Remote Systems」ビューを使用して、ご使用のアプリケーションを作成するのにアクセスする必要があるオブジェクトをナビゲートおよびリストします。

System i テーブル・ビュー
System i テーブル・ビューは、「Remote Systems」ビューと同じ情報を表示しますが、追加された機能として、項目のソート、記述の表示、および PDM に類似したより多くのアクションの実行があります。

Remote Systems LPEX Editor
Remote Systems LPEX Editor は、基本 LPEX Editor を基としており、System i 固有の機能を含んでいます。

Remote System Explorer パースペクティブご使用に際して

Remote System Explorer パースペクティブにより、ご使用のシステムですべてのアイテムにアクセス、編集、実行、コンパイル、およびデバッグできます。

最初に Remote System Explorer を開いたときは、ご使用のローカル・ワークステーション以外のシステムには接続しません。リモートのSystem i に接続するには、プロファイルおよび接続を定義する必要があります。

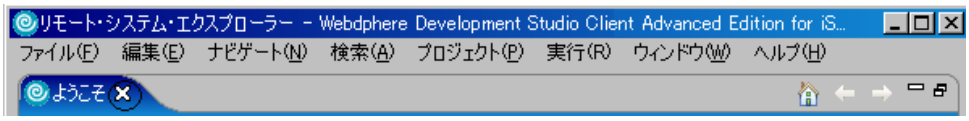
- プロファイルは、接続をグループ化したり、共用したり、専用にするために使用します。

- 接続は、ご使用の System i への TCP/IP ネットワーク接続で、この接続により、ご使用のシステムですべてのアイテムにアクセス、編集、実行、コンパイル、およびデバッグできます。接続を定義する場合、リモート・システムの名前または IP アドレスを指定します。また、接続自体にはご使用のワークスペースでラベルとして機能する固有の名前を指定します。これにより、容易に接続および切断ができます。 System i へ接続する場合、ワークベンチにより、そのシステムに対するユーザー ID およびパスワードを要求するプロンプトが表示されます。

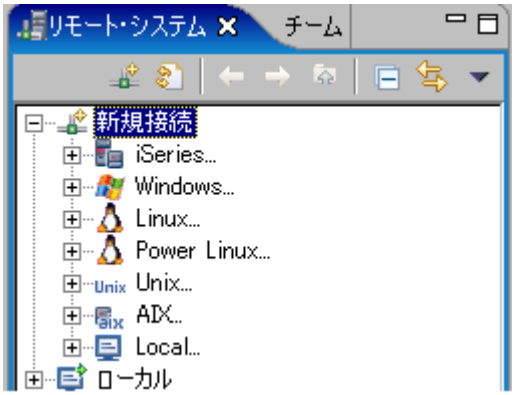
Remote System Explorer (RSE) で作業を開始するには:

1. ワークベンチの開始
2. プロンプトが表示されたら、ワークスペースを指定してください
3. ワークベンチを開いたら、Remote System Explorer パースペクティブが開いているか確認します。パースペクティブが開いていない場合、「**ウィンドウ**」 > 「**Open Perspective**」 > 「**Remote System Explorer**」を選択して開くことができます。

   X をクリックして、「Welcome」ビューを閉じます。



4. 「Remote Systems」ビューの「**New Connection**」は、「Remote Systems」ビューを介して接続できるリモート・システムの様々なタイプを示しています。



5. 接続の作成:
   a. ビューの「**New Connection**」以下にある、 **iSeries®** を拡張して、「NAME」個人用プロファイルページを開きます。デフォルト・プロファイル値を受け入れて、接続ページを開きます。
   b. 「**Parent profile**」デフォルト値はそのままにします。
   c. ホスト・システム名を「**Host name**」フィールドに入力します。「**Connection name**」フィールドには自動的にホスト名が入ります。
   d. 「**Verify host name**」チェック・ボックスは選択したままにします。
   e. 「**Finish**」をクリックして接続を作成します。

同じ System i に複数の接続を定義できますが、それに加えて、接続の開始に、例えば異なるユーザー ID およびパスワードの保管、初期ライブラリー・リストの保管といった異なる構成を組み込むことができます。 System i への接続を作成すると、容易に接続および切断を行うことができます。

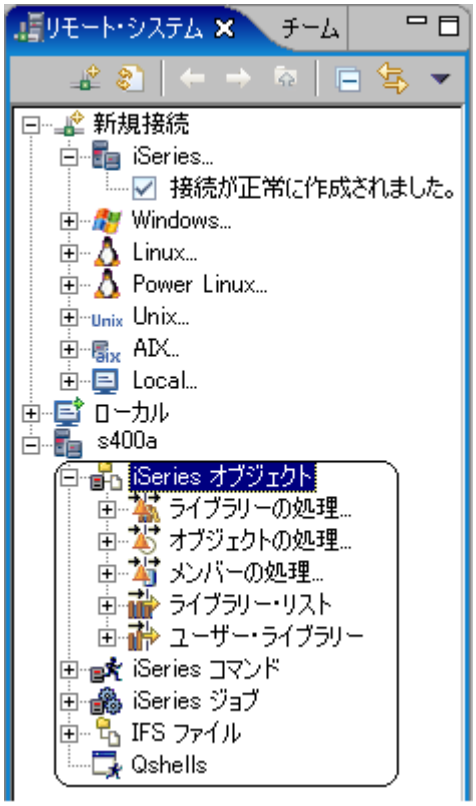
詳細については、クライアント製品のオンライン・ヘルプにあるトピック『リモート・システムへの*接続の構成 (英語)*』を参照してください。チュートリアルの『*System i への接続の構成および i5/OS への接続 (英語)*』も参照してください。

ヒント:

- 接続を作成するときは、デフォルト・プロファイル名を使用してください。デフォルト・プロファイルを使用すると、接続を他の人と共有したり、フィルター・プールを使用したりできます。 (フィルター・プールの詳細については、クライアント製品のオンライン・ヘルプにあるトピック『*Remote System Explorer フィルター、フィルター・プール、およびフィルター・プール解説書 (英語)*』を参照してください。)
- 「**Host name**」を指定するときは、IP アドレス、または完全修飾名 (必要な場合) のどちらかを指定できます。
- 「**Verify host name**」チェック・ボックスにより、指定した System i に実際に接続できるかを確認します。ご使用の System i に必要なすべての PTF がインストールされていることを確認するには、(接続を作成したあとで) その接続を右クリックし、「**Verify connection**」を選択します。これにより、必要なすべてのポートが開いており、コールバックを実行することができ、必要なホスト PTF が適用されることを確認します。
- 接続の開始プロパティーを定義するには、接続を右クリックして「**Properties**」を選択します。

サブシステム

System i への接続を構成すると、容易に接続および新しい接続を拡張して、サブシステムを表示できます。サブシステムは、ご使用のリモート・システム上で、ライブラリー、コマンド・セット、およびジョブを示すコンテナーで表されます。このコンテキストのサブシステムは、System i のサブシステムとは関連していません。



System i 接続には 5 つの異なるサブシステムがあります。

1. System i オブジェクト: これを使用してライブラリー、オブジェクトおよびメンバーにアクセスできます。
2. System i コマンド: デフォルトでは、このサブシステムにはリモート・オブジェクトに対して実行するために使用できる定義済みコマンドのセットが準備されています。また、コマンド・セット、および独自のコマンドを定義できます。結果はコマンド・ログ・ビューに記録されます。(コマンド・ログ・ビューの詳細については、クライアント製品のオンライン・ヘルプにあるトピック『*プログラムおよびコマンドの実行 (英語)*』を参照してください。)
3. System i ジョブ: このサブシステムを使用してジョブをリストします。ジョブ属性でサブセットを定義することができ、保留、再開、停止などのジョブ操作を実行できます。
4. IFS ファイル: Integrated File System でファイルおよびフォルダー構造を探索し、操作を実行します。
5. Qshell: 接続のために実行中のアクティブ Qshell のリストにアクセスし、このサブシステムを使用して Qshell を開始します。(詳細については、クライアント製品のオンライン・ヘルプにあるトピック『*リモート・シェル・ビューを使用したコマンドおよびシェルの実行および表示 (英語)*』を参照してください。)

接続されているビューは、「Remote Systems」ビューと呼ばれます。このビューは Windows® File Explorer とほぼ同じ働きをします。“プラス” (+) をクリックして展開し、目的の項目にアクセスします。例えば、***LIBL** フィルターを展開してライブラリー・リストにあるすべてのライブラリーを表示してから、ファイルを拡張してすべてのファイル・メンバーを表示します (PDM のオプション 12 とほぼ同じ)。

フィルター

サブシステムを展開すると、そのサブシステムのフィルター・リストが表示されます。フィルターとは、指定、再利用、および共用ができる項目のリスト名です。フィルターにより、現在は表示する必要のない項目が *取り除かれます*。フィルターを作成するときは、汎用値を使用することができ、必要な数だけフィルターを作成することができます。各サブシステムごとにフィルターを作成できるので、例えば、IFS ファイル、ローカル・ファイル、i5/OS オブジェクトにフィルターを持つことができます。

ヒント:

- 項目がコンテナー (コンテナーの例としては、ライブラリーおよびファイルがあります) である場合、常にフィルター内で展開できます。
- 汎用値の複数レベルを指定できます。例えば、フィルターを作成する場合、ライブラリーに `BOB`、ファイルに `QRPG*`、メンバーに `A*` を指定できます。
- フィルター名を指定するページに注意してください。このページでは、フィルターが指定した接続専用であるかどうか、またはすべての接続に表示するかを選択します。また、他の人とフィルターを共有する場合は、プロファイルを指定することもできます。

フィルターは RSE 内で接続と共に保管される名前であるため、すべてのフィルターはセッションを切断しても存続します。

フィルター・ストリング

フィルターを最初に作成したとき、フィルターにはフィルター・ストリングが 1 つだけ含まれています。フィルターのプロパティーを変更することで、フィルター・ストリングを追加できます。フィルター・ストリングにより、より複雑なリストを生成できます。フィルターにある複数のフィルター・ストリングを使用すると、単一の名前付きフィルターにある異なるファイル、および異なるライブラリーにもメンバーをリストすることができます。

ヒント:

- フィルターには、同じタイプの項目が含まれている必要があります。例えば、同じフィルターにリストおよびメンバーをリストすることはできません。
- プロジェクトまたはアプリケーションで、項目をフィルターにグループ化します。例えば、フィルター・ストリングを追加すると、RPG および COBOL ファイルと同じリストに DDS メンバーを表示できます。
- フィルターの詳細については、クライアント製品のオンライン・ヘルプにあるトピック『メンバーのフィルター *(英語)*』を参照してください。また、チュートリアルの『フィルターの*紹介 (英語)*』も参照してください。

検索

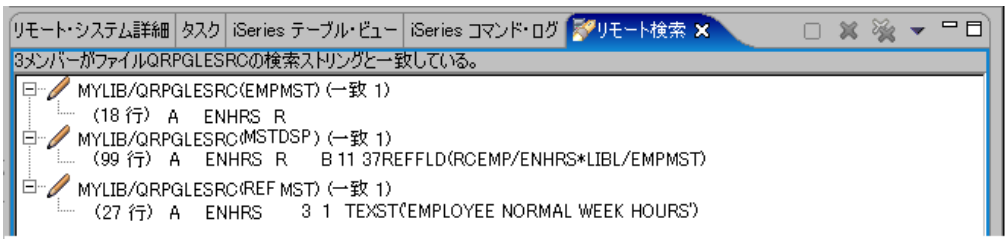
RSE を検索するには、次の 2 つの方法があります。

1. 「**Search**」メニュー・オプションから (次に「System i」を選択)

2. 「Remote Systems」ビューおよび System i テーブル・ビューの「Find String」アクションから

RSE では、ライブラリー、ファイル、およびメンバーだけでなく、フィルターも検索できます。つまり、非常に柔軟な検索パターンを使用して検索できるということです。例えば、ライブラリー MYLIB のファイル QRPGLESRC にあるすべてのメンバー、およびライブラリー PROJECT のファイル PRJA* にあるメンバー A* を、探索できます。これらの検索対象のフィルター・ストリングを含むフィルターに、検索ストリング・アクションを呼び出すことで探索が可能です。

検索結果は「Remote Search」ビューに表示され、ビューには検索履歴があります。すべての検索結果は 1 箇所に表示され、目的とするどのメンバーでも最初に開くことができ、決定したメンバーのどの一致も使用できます。「Remote Search」ビューにより、ポップアップ・メニューを介してリストからメンバーおよび一致を削除することで、結果リストを管理できます。



ヒント:

- 「Remote Search」ビューのメンバー名をダブルクリックして、編集のため、および選択された一致に配置するために xlviii ページの『Remote Systems LPEX Editor』 にあるメンバーを開きます。
- 「Remote Search」ビューのポップアップには、System i テーブル・ビューと同様のオプションのリストがあります。
- 「Remote Search」タブをダブルクリックしてビューを最大化し、フルウィンドウのワークベンチにします。これにより、1 度により多くの一致を表示できます。
- 一致したメンバーを拡張または縮小して、重要な一致に素早く焦点を合わせることができます。
- クライアント製品のオンライン・ヘルプにあるトピック『*System i にあるテキスト・ストリングの検索 (英語)*』を参照してください。また、チュートリアルの『*複数ファイルの検索 (英語)*』も参照してください。

RSE と PDM の比較

次の表では、このトピックで説明された RSE の機能と、PDM の同等もしくは類似の機能を比較します。

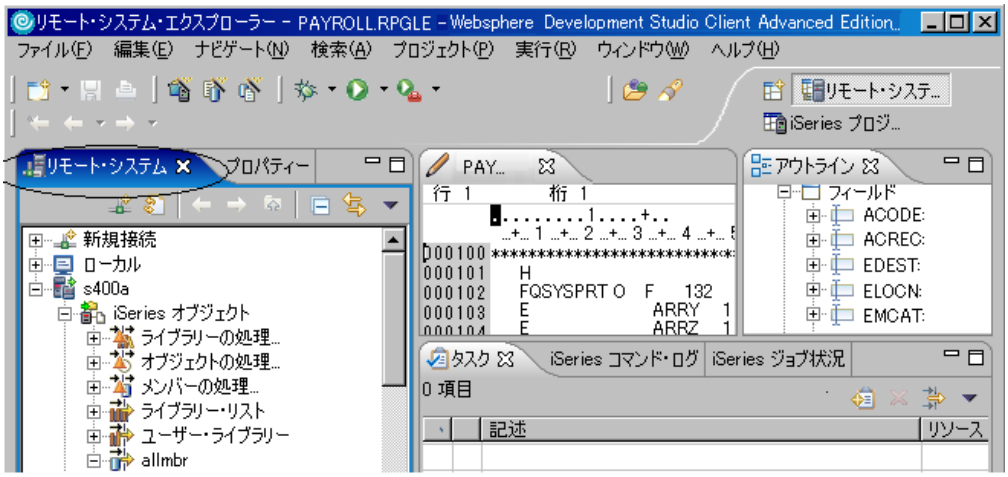
表 1.

RSE	PDM
接続を作成	エミュレーター・セッションを開始
汎用値でフィルターを作成	汎用値でフィルターを作成
コンテナーを展開して追加の項目を表示	オプション 12
汎用項目の複数レベルを指定	利用不可
フィルターはセッションの間存続	WRKxxxPDM コマンドの以前のパラメーターを保存
複数のフィルター・ストリングをフィルターに定義して、メンバーを異なるファイルにリストすることで、複合リストを作成	単一ライブラリーの 1 つのソース物理ファイルにメンバーをリスト
柔軟な検索パターンによりフィルターを検索	オプション 25 または FNDSTRPDM を使用した単一検索パターン
すべての検索結果は、「Remote Search」ビューで使用可能	検索結果およびメンバーは、一致が検索された順序で 1 度に 1 つ使用可能

「Remote Systems」ビュー

「Remote Systems」ビューを使用して、ご使用のアプリケーションを作成するのにアクセスする必要があるオブジェクトをナビゲートおよびリストします。

項目を開く、または展開してその項目の子を表示します。右クリックしてポップアップ・メニューで使用可能なアクションにアクセスします。ドラッグ・アンド・ドロップ、コピー、貼り付け、削除、および名前変更などの標準のアクションはすべて、ポップアップ・メニューで使用可能です。これらのオプションは、PDM と比べて非常に強力です。コピー・アンド・ペースト、またはドラッグ・アンド・ドロップを使用して、ある System i から別の System i にメンバーおよびオブジェクトをコピーまたは移動できます (もはや SAVOBJ および FTP を使用する必要はありません)。ポップアップ・メニューには、項目上で実行できる多くのアクションが他にもあり、独自の追加アクションを作成することもできます。



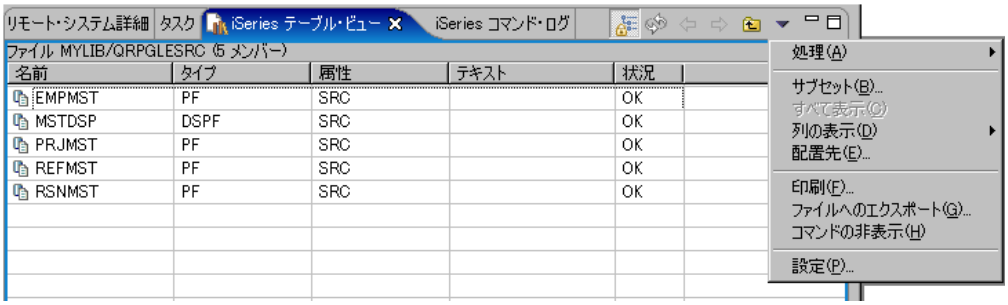
クライアント製品のオンライン・ヘルプにある『ユーザー・アクション (英語)』、およびトピック『*Remote System Explorer にある項目の取り扱い (英語)*』を参照してください。また、チュートリアルの『*Remote System Explorer のオブジェクトの表示およびアクセス (英語)*』も参照してください。

System i テーブル・ビュー

System i テーブル・ビューは、「Remote Systems」ビューと同じ情報を表示しますが、追加された機能として、項目のソート、記述の表示、および PDM に類似したより多くのアクションの実行があります。

System i テーブル・ビューを使用すると、すべての項目のプロパティーを同時に表示できます。プロパティーは表の行として表示されます。ビューは、現在選択しているファイル、ライブラリー、またはフィルターを System i オブジェクト・サブシステムに入力として取り込み、ポップアップ・メニューから「Show in Table」オプションを選択した場合、内容をテーブルに表示します。また、System i Table ビュー自身から「Work with」アクションを使用して、ライブラリー、オブジェクト、またはメンバーのデータをビューに追加します。

Remote System Explorer パースペクティブの下部にある System i テーブル・ビュー・タブを選択するか、「Remote Systems」ビューのポップアップにある「**Show in Table**」アクション項目を選択すると、ビューを直接開くことができます。「**Work with**」メニューを使用して、リストを生成できます。「**Work with**」メニューには、System i テーブル・ビューに以前表示されたリストの小さなリスト (10個) があります。コマンド行は System i テーブル・ビューの下部に表示され、コマンドやパラメーターを入力してアクションを実行できます。



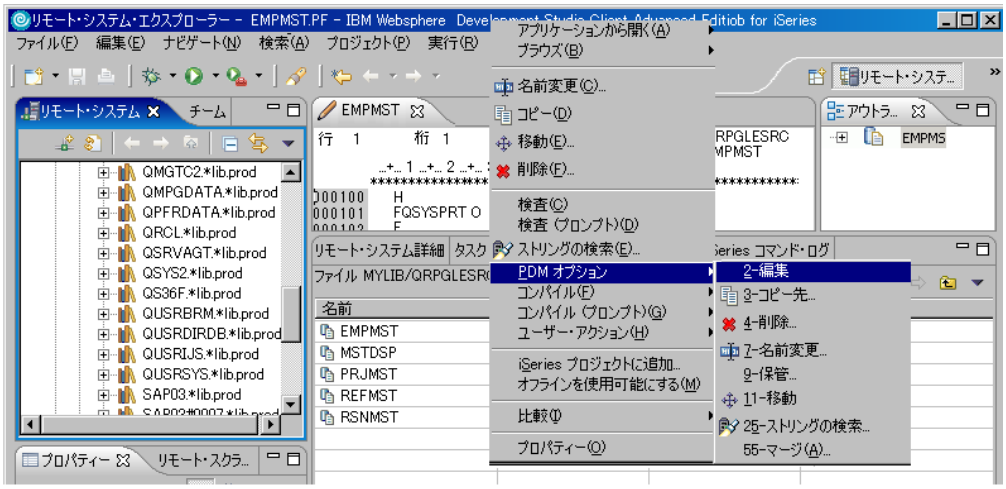
System i テーブル・ビューに表示される列を変更できます。任意のそれぞれの列の非表示、または表示を選択できます。文字を入力すると、「Position To」ダイアログを立ち上げることができます。このダイアログにより、素早く目的の項目にスクロールできます。

ヒント:

- 列見出しをクリックすると、その列でソートします。
- テーブルに表示ビューを使用すると、System i テーブル・ビューのフィルターの内容が表示されます。
- コマンド行を縮小表示すると、画面がすっきりしてより多くの項目を表示できます。
- System i テーブル・ビュー・タブをダブルクリックすると、ビューが最大化し、フル・ワークベンチになります。これにより、1 つの画面でより多くの項目を表示できます。
- フィルターを使用すると、複雑なリストが生成されますが、「Work with」サブメニューを使用すると、頻繁には使用しないリスト、または単純リストにアクセスできます。

System i テーブル・ビュー・アクション

「Remote Systems」ビューと同じように、System i テーブル・ビューには、その項目で呼び出すことができるアクションがあります。「Remote Systems」ビューと同じように、ポップアップ・メニューを右クリックしてアクションにアクセスします。 System i テーブル・ビューからポップアップで、アクションのリストを PDM オプション番号と共に表示して、そのメニューをわかりやすいものにします。「**User actions**」メニューを使用して、独自のアクションを作成および追加します。テーブル・ビューまたはリモート・システムのどちらかに追加されたユーザー・アクションは、両方のビューの「**User Actions**」メニューに表示されます。



ヒント:

- System i テーブル・ビュー内から「Show in Table」アクションを使用すると、ライブラリーのリストからオブジェクトのリストに移動します。
- メンバーをダブルクリックすると、編集モードの xlviii ページの『Remote Systems LPEX Editor』 にメンバーが開きます。
- 項目名をクリックすると、テーブル・ビューの編集セルが直接開き、素早く項目名を変更できます。記述列、およびメンバー・タイプ列も編集可能で、これらの列にある値を素早く変更できます。
- System i テーブル・ビュー、または「Remote Systems」ビューのポップアップ・メニューにある「**Properties**」メニュー項目を選択して、オブジェクトのプロパティーにアクセスします。
- クライアント製品のオンライン・ヘルプにある『*System i*テーブル・ビューのオブジェクトの管理 *(英語)*』を参照してください。また、チュートリアルの『*Remote Systems Explorer* のオブジェクトの表示およびアクセス *(英語)*』も参照してください。

ユーザー・アクション

ユーザーのアクションにより、使用するアクションで System i テーブル・ビューおよび「Remote Systems」ビューを拡張できます。独自のアクションを作成し、そのアクションで実行するコマンドのプロンプトを出すことができ、コマンドを実行する方法を定義できます。

注: RSE には次の 3 つのコマンド・モードがあります。

1. 通常: RSE ジョブはバッチで実行するので、通常モード (この場合は即時)でも、STRPDM のような対話式コマンドを実行できません
2. バッチ: コマンドは新しいバッチ・ジョブにサブミットされます
3. 対話式: コマンドは STRRSESVR ジョブで対話式に実行されます

アクションを作成する場合、「**Insert variable**」ボタンを使用すると、使用可能な変数のリストが表示されます。アクションはカスタマイズ可能で、以下のことができます。

- 実行後に「Remote Systems」ビューまたは System i テーブル・ビューを最新表示にするかどうかを指定します。
- コマンドがそれぞれ選択したオブジェクトを 1 回ずつ呼び出すか、もしくはすべてのオブジェクトを一括で呼び出すかどうかを指定します。これにより、PDM よりも高い柔軟性を実現します。例えば、RSE で次のような「Save Object」アクションの定義が可能です。このアクションは、複数のオブジェクトを選択し、コマンドの実行時には選択したオブジェクトすべてに対して単一コマンドを生成し、1 つの保管ファイルに保管します。
- アクションを詳細化することができ、適切なタイプだけを表示できます。事前定義された複数のタイプがありますが、ユーザー・タイプは容易にリストに追加できます。例えば、これにより、*PGM オブジェクトだけ、または CBLLE メンバーだけに対してアクションを持つことができます。
- アクションに対して実行する CL コマンドを定義する場合、選択プロンプトを使用できます。

RSE ユーザー・アクションの利点の 1 つは、アクションに名前を付けることができ、使用したり記憶しておくのが容易になります。 (詳細については、クライアント製品のオンライン・ヘルプにあるトピック『*ユーザー・アクションの管理 (ユーザー・オプション) (英語)*』を参照してください。また、チュートリアルの『*ユーザー・アクションの作成 (英語)*』も参照してください。)

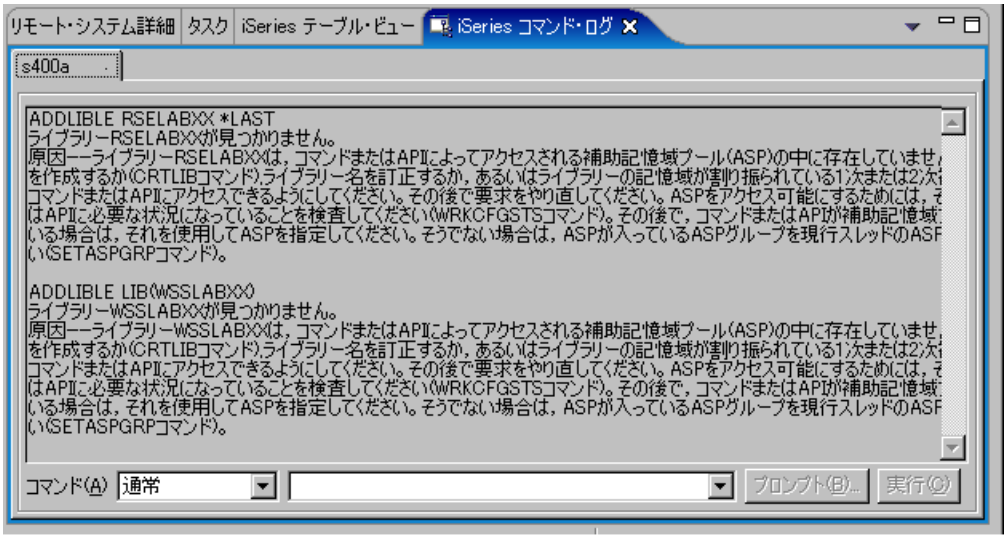
コマンド行

System i テーブル・ビューにはコマンド行があります。

コマンド行を使用して任意のコマンドを実行することができ、System i テーブル・ビューで、PDM オプションの追加パラメーターを指定できます。任意のコマンドの結果は、メッセージ・フィールドに表示されます。次の使いやすい PDM キーを使用できます。

- **F9** 最後の行をリトリーブします
- **F4** コマンドのプロンプトを出します

「**Show Log**」ボタンを使用して、System i コマンド・ ログ・ビューを表示できます。



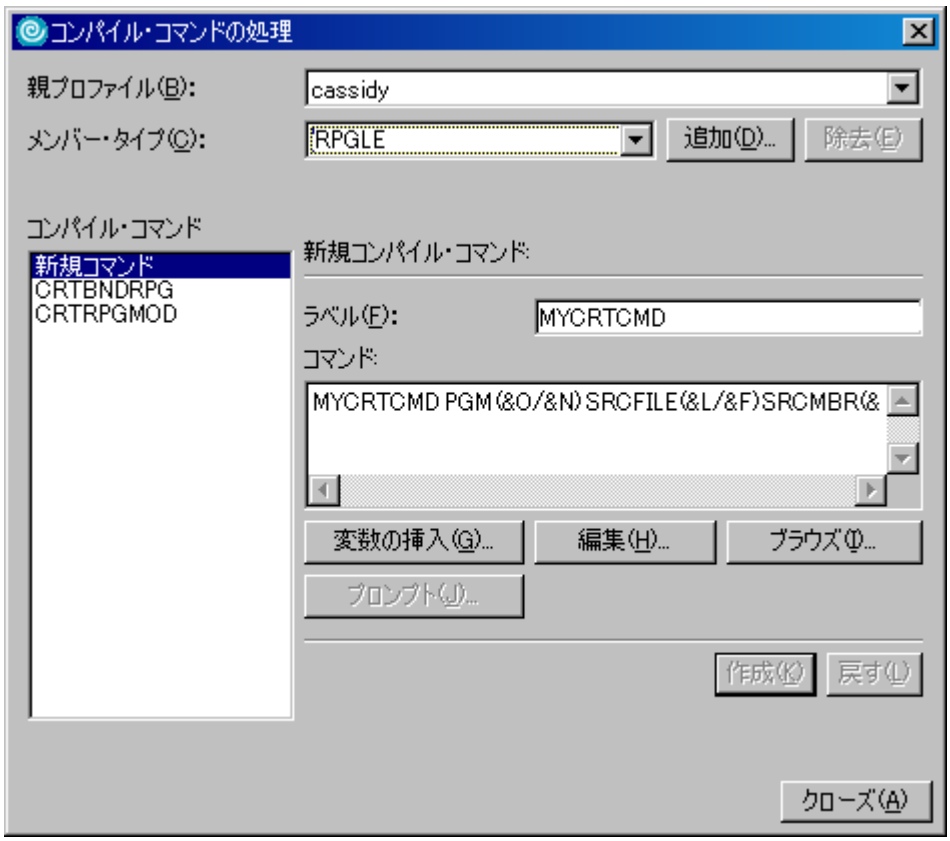
コマンド実行モード も選択できます。

ヒント:

- System i テーブル・ビュー・コマンド行は、クイック・コマンドに適切です。
- System i コマンド・ログ・ビューは、結果がより重大で、メッセージの第 2 レベルのヘルプを表示したいコマンドにより適切です。
- System i コマンド・ログには、コンパイルの結果、および「Remote Systems」ビューまたは System i テーブル・ビューのどちらかから実行されたユーザー・アクションが表示されます。
- クライアント製品のオンライン・ヘルプにあるトピック『*System i テーブル・ビューからのプログラムおよびコマンドの実行 (英語)*』を参照してください。また、チュートリアルの『*System i テーブル・ビューのコマンドのサブミット (英語)*』も参照してください。

コンパイル

コンパイル・アクションは、プロンプトありとプロンプトなしの 2 つのメニューに分けられます。独自のコンパイル・コマンドをコンパイル・メニューに追加できます。これはユーザー・アクションの追加とほとんど同じです。



コンパイル・アクションは他のアクションと異なります。コマンドの結果自体は System i コマンド・ログに表示されますが、イベント・ファイルをサポートするコマンドについては、コンパイラーで生成されたエラーがエラー・リスト・ビューに表示されます。イベント・ファイルの詳細については、クライアント製品のオンライン・ヘルプにあるトピック『イベント・ファイルのフォーマット *(英語)*』を参照してください。

ヒント:

- 追加のコマンド実行設定は、「**ウィンドウ**」>「**Preferences**」>「**Remote Systems**」>「System i」>「**Command Execution**」設定ページで使用可能です。
- コンパイル・コマンドは、ホストとして同じデフォルトを使用します。
- コンパイル・アクションは、メンバー・タイプにより最後に使用したコンパイル・コマンドを記憶しています。
- 独自のコンパイル・コマンドを追加し、適用先のソース・タイプを指定するか、プロパティーおよび CL コマンド・パラメーターを変更して、既存のコンパイル・コマンドを詳細化します。
- デフォルトでは、バッチにおいてコンパイルが実行されます。コマンド実行設定を使用して、追加の SBMJOB パラメーターを指定するか、通常の実行に切り替えます (前述したとおり、RSE ジョブはバッチで実行されるため、技術的には対話式ではないことに注意してください)。

- クライアント製品のオンライン・ヘルプにあるトピック『プログラムのコンパイル *(英語)*』を参照してください。また、チュートリアルの『ソースの検査およびコンパイル *(英語)*』も参照してください。

System i テーブル・ビューと PDM の比較

次の表では、このトピックで説明された System i テーブル・ビューの機能と、PDM の同等もしくは類似の機能を比較します。

表 2.

System i テーブル・ビュー	PDM
「Work with」メニューを使用してリストを生成。最大 10 個の以前の表示リストを保存します。	WRKxxxPDM コマンドを使用してリストを生成
アクションに対してコマンド行を提供	アクションに対してコマンド行を提供
どの列を表示するかを決定し、任意のそれぞれの列の非表示、または表示を選択	オプション 14 により同様の操作が可能だが、カスタマイズ機能が劣る
ポップアップ・メニューにより PDM オプション番号と共にアクションをリスト	該当せず
ビュー内から「Show in Table」アクションを使用して、ライブラリーのリストからオブジェクトのリストに移動	オプション 12
ポップアップ・メニューを使用して「**Properties**」メニューを選択し、オブジェクトのプロパティーにアクセス	オプション 8
ユーザー・アクションでテーブル・ビューを拡張	F16
アクション作成時の置換変数は PDM と同じ	該当せず
アクションを詳細化して、適切なタイプだけを表示	利用不可
名前別にユーザー・アクションを定義	ユーザー・アクション名は 2 文字まで
「**ログの表示 Show Log**」ボタンを使用して、System i コマンド・ログ・ビューをオープン	F20
コンパイル・アクションをプロンプトありとプロンプトなしの 2 つのメニューにグループ化	オプション 14 およびオプション 15
追加のコマンド実行設定が使用可能	F18

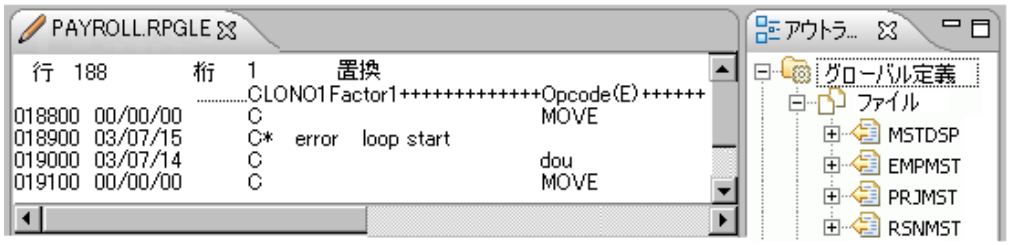
Remote Systems LPEX Editor

Remote Systems LPEX Editor は、基本 LPEX Editor を基としており、System i 固有の機能を含んでいます。

メンバーをダブルクリックすると、「Remote Systems」ビュー、System i テーブル・ビュー、および「Remote Search」ビューから、Remote Systems LPEX Editor を

編集モードで素早く起動できます。メンバー上のポップアップ・メニューを使用してエディターを起動し、編集モードまたはブラウズ・モードで開くこともできます。

エディターを開いて最初に気付くのは、ソースにおける色の使い方でしょう。これはトークン化と呼ばれるもので、言語トークンを色分けして区別しやすくします。

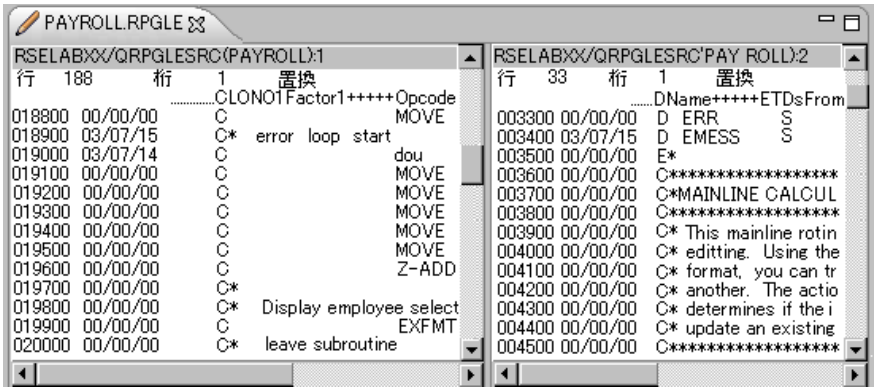


接頭部域にはメンバーのシーケンス番号が含まれることに注意してください。Remote Systems LPEX Editor の接頭部域は SEU コマンドをサポートします (例えば、`CC`、`B`、`A`、`LL` など)。

また、多くのソース・タイプに対して、一括表示ビューが表示されることにも注意してください。一括表示ビューにより、ご使用のソースが一括表示され、そのソース内をナビゲートするのに使用できます。

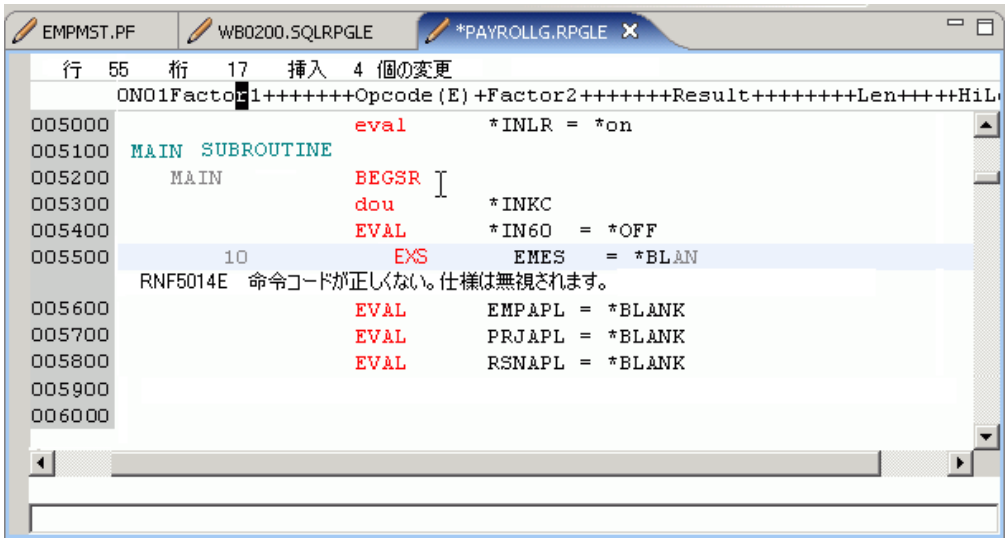
ヒント:

- デフォルトでは、日付領域はエディターに表示されません。常に日付を表示したい場合は、「**ウィンドウ**」>「**Preferences**」>「**Remote Systems**」>「**Remote Systems LPEX Editor**」で設定する必要があります。エディター・ビューのポップアップで、「**Source**」メニューを介して単一セッションの日付領域をオンにできます。
- `X`、`XX`、`Xn` を使用すると行を除外できますが、`+` を使用すると、容易に除外した行を表示したり非表示にしたりすることができます。
- 除外した行に加え、エディター・ビューのポップアップ・メニューにある「**Filter**」メニューを使用して、例えばコメントや制御ステートメントを表示します。各言語には、フィルター・メニューに表示できる独自の項目リストがあります。
- エディター・メニューのポップアップ・メニューには、編集されているソース・タイプによって異なる適切な項目、ソース内のカーソル位置、および選択されたテキストがあるかどうかが表示されるだけです。
- ポップアップ・メニューまたは **Ctrl+2** を使用して、同じメンバーの異なる部分を処理するために、現行のエディター・ビューを分割できます。最大 5 つ分割できます。



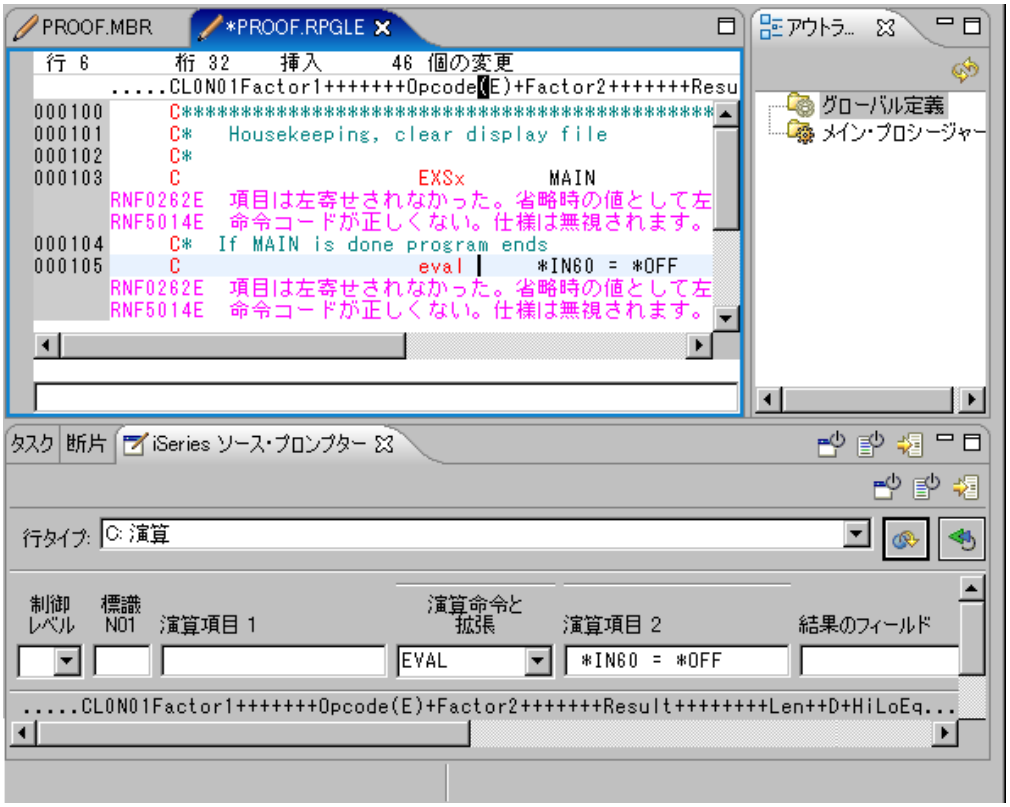
構文検査、プロンプト、およびヘルプ

Remote Systems LPEX Editor には、自動構文検査機能があります。エラーが検査されると、すぐに画面上に表示されます。すべての構文エラーがエディター・ビューに組み込まれており、エラーを表示するために画面を下までスクロールする必要はありません。 Remote Systems LPEX Editor は、最新の言語構文を使用して、DDS、RPG および COBOL の構文を検査します。SQL および CL の構文検査をするには、アクティブな接続が必要です。 CL は構文情報をキャッシュに入れるので、キャッシュ付き情報が存在する場合、構文検査は接続を切断するときに行うことができます。



ヘルプ (Remote Systems LPEX Editor の **F1**) は、エラーに対して使用できませんが、ソースに対して使用できます。コンテキスト・ヘルプの **F1** を押すと、編集中のコードのトピックについての解説書が開きます。例えば、ILE COBOL ステートメントで **F1** を押した場合、そのステートメントのヘルプがヘルプ・ブラウザーに開きます。解説書はまた、Remote Systems LPEX Editor の「**Source**」メニューによりアクセス可能です。こうして印刷物のマニュアルの必要を減らすことができます。

F4 により、Remote Systems LPEX Editor で編集中にプロンプトを表示することができます。CL 以外の言語に対して、プロンプト・ビューが開き、ご使用のソースを変更できます。 CL の場合、ウィンドウがご使用のプロンプトと一緒に開きます。コンテキスト・ヘルプ **F1** はすべてのプロンプトから使用可能です。



詳細については、クライアント製品のオンライン・ヘルプにあるトピック『*RPG、COBOL、CL、C、C++、および DDS メンバーの編集 (英語)*』を参照してください。また、チュートリアルの『ソースの編集 *(英語)*』も参照してください。

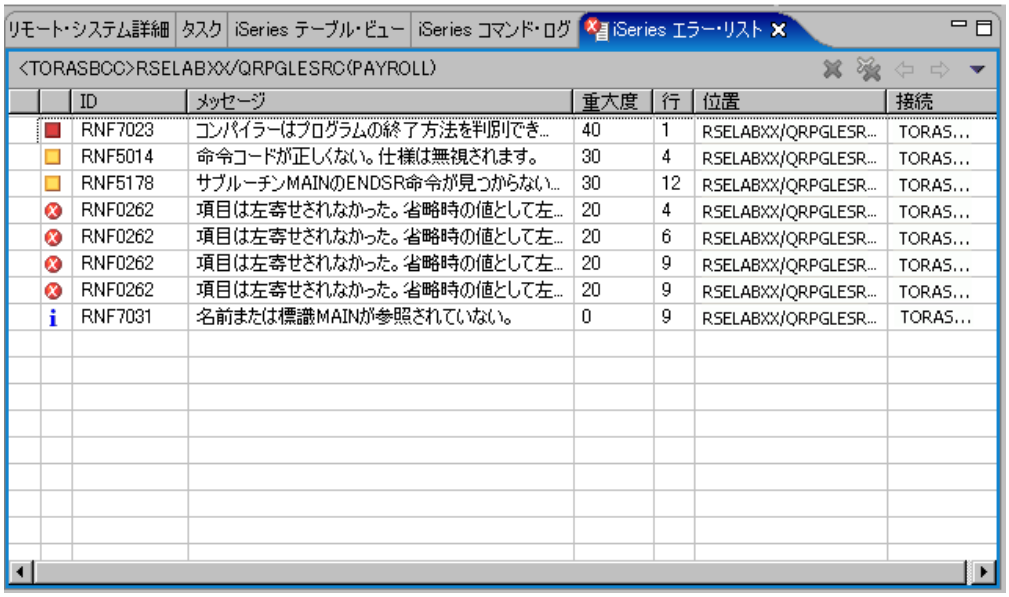
ベリファイヤーおよび System i エラー・リスト・ビュー

構文検査により、入力した行にエラーがないことを確認しますが、Remote Systems LPEX Editor には、ベリファイヤーと呼ばれる追加の検査があります。ベリファイ

ヤーは、オブジェクトを生成せずに、コンパイラーが行うのと同じ構文検査およびセマンティック検査を行います。つまり、宣言されていない変数を使用しようとすると、ベリファイヤーがそのことを通知します。

ベリファイヤー機能は、「Source」メニューから、または **Ctrl+Shift+V**を押すことで、COBOL、RPG および DDS に対して使用できます。「**Source**」メニューを使用すると、プロンプトで検査し、追加のオプションを指定できます。

検査で検出されたどのエラーも、コンパイルから出力されたエラーと全く同じように、System i エラー・リスト・ビューに表示されます。



System i エラー・リスト・ビューにより、ダブルクリックでエラーをエディター・ビューに挿入できます。 **F1** キーを使用すると、エラーに対するヘルプを利用できます。「**ビュー**」メニューを使用すると、表示したくないエラー (例えば、通知メッセージなど) をフィルターできます。メニューを使用して、エラー・メッセージをエディター・ビューに挿入するかどうか、およびその方法を指示することもできます。

任意の挿入済みエラーをクリーンアップするには、エディターから **Ctrl+F5** を使用して、リストをリフレッシュできます。リフレッシュ・アクションには次があります。

- 任意の構文、ベリファイヤー、およびコンパイル・エラーを除去する
- 任意の除外された行をクリアする

または、「**Source**」メニューから「**Remove messages**」メニュー・オプションを使用できます。

System i エラー・リスト・ビューを To Do リストとして使用できます。行を変更した場合、行には、その行が削除されていることを示すために X でマーク付けされているか、エラーが検出されていることを示すためにチェック・マークが付けられています。別の検査によってのみ、エラーが本当に修正されていることを確認できます。

ベリファイヤーを使用すると、次の利点があります。

- 実際にコンパイルする前に、コンパイルがクリーンであることを確認できます。このことは、コンパイルをオフピーク時のみで行う必要のあるマシンにとって重要です。
- オフラインで作業している場合、ご使用の System i に再接続する時にコンパイルするソースで作業していることを確認できます。

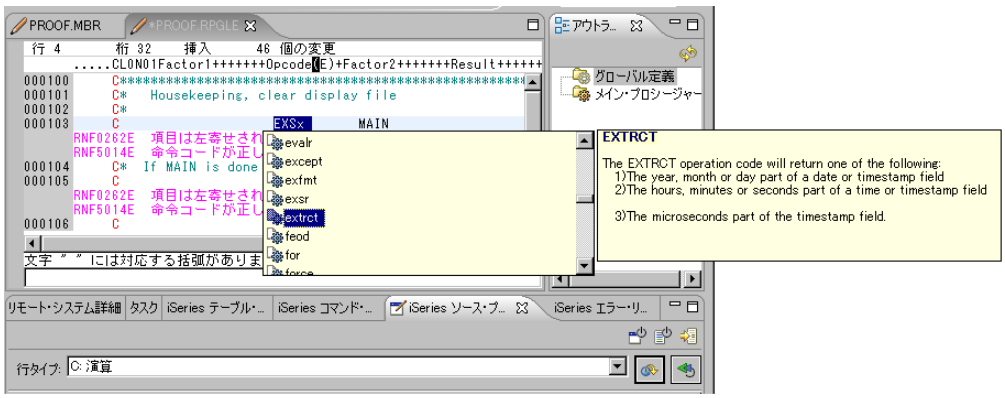
System i エラー・リスト・ビューにも、(コンパイル結果またはコンパイル検査のどちらに使用するかにかかわらず) いくつかの利点があります。

- スプール・ファイルおよびソース間でスイッチしたり、同時に表示するために 2 つのエミュレーターを開いたりする必要はありません。すべてのエラーをソースに挿入することができます。
- To Do リストとして使用する場合、すべてのエラーが対処されているのかを簡単に確認できます。 すべてのエラーに対処するまでの、エラーを修正し、再コンパイルし、修正を繰り返す作業は必要ありません。
- エラーが RPG の `/COPY` または `/INCLUDE` メンバーで発生した場合、または COBOL のサンプル集で発生した場合、エラーをダブルクリックしてそのメンバーを素早く開き、1 次ソース・メンバーと同様に、エラーを挿入します。
- エラーに対する F1 ヘルプにより、解説書を使用せずにエラーを修正することができます。
- (通知メッセージや警告メッセージなどの) 表示したくないメッセージを非表示にできる設定を使用すると、重要なエラーを素早く検出されていることを確認するのが容易になります。
- 詳細については、クライアント製品のオンライン・ヘルプにあるトピック『*検査 (英語)*』および『*エラー・リスト・ビュー (英語)*』を参照してください。また、チュートリアルの『*ソースの検査 (英語)*』も参照してください。

コンテンツ・アシスト、テンプレート、および RPG ウィザード

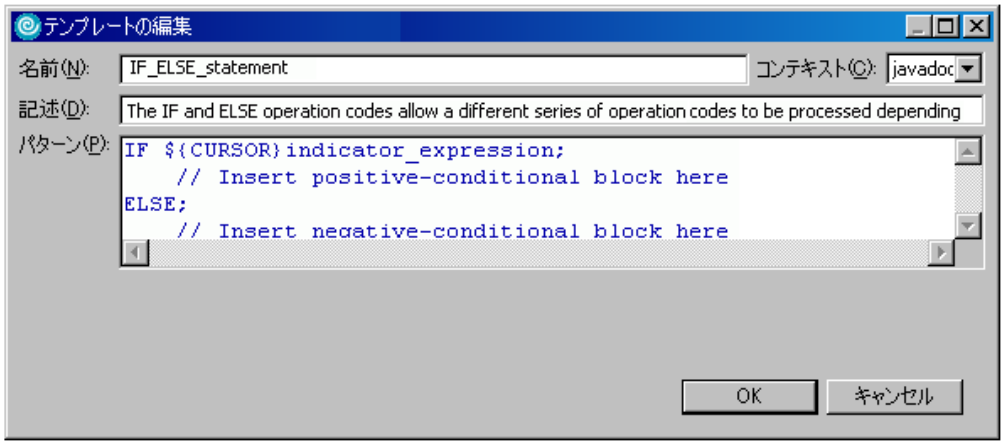
Remote Systems LPEX Editor には、素早くコードを入力できるようにするためのいくつかの機能があります。

コンテンツ・アシスト (**Ctrl+Space**) は、カーソル位置で完成したコードを提示、表示、および挿入します。コンテンツ・アシストが呼び出されると、現時点で入力された文字から判断して以前に入力したコードから、有効な完成したコードの候補が表示されます。RPG の場合は、判断材料は文字に加えて列の位置に基づきます。例えば、これは COBOL ステートメントに必要なパラメーターの数値が不確かな場合や、パラメーター・タイプが不確かなときに便利です。



詳細については、クライアント製品のオンライン・ヘルプにあるトピック『コンテンツ・アシスト *(英語)*』を参照してください。

テンプレートを使用して、頻繁に使用されるコードのブロックを生成できます。テンプレートはインポートおよびエクスポートできます。つまり、テンプレートを共有できます。例えば、各プログラムに追加する必要のある標準ヘッダー、または標準インターフェースがある場合、テンプレートの名前を入力して、**Ctrl+Space** を押すと、テンプレートを定義し、挿入できます。



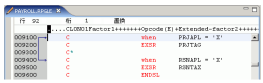
詳細については、クライアント製品のオンライン・ヘルプにあるトピック『コンテンツ・アシストを使用したコードの完成 *(英語)*』および『テンプレート *(英語)*』を参照してください。また、チュートリアルの『ソースの検査 *(英語)*』も参照してください。

ヒント: コンテンツ・アシスト機能および COBOL の一括表示ビューは、検査によって生成された情報により動作します。この検査により、COPY メンバーにあるプロシージャー、またはディスプレイ・ファイルからのフィールドおよびレコード等の外部情報が収集されます。一括表示ビューを少なくとも 1 度最新表示にしてからコンテンツ・アシストを呼び出すことは重要です。そうしない場合、限られたコンテンツ・アシストの機能しか使用できません。

追加 Remote Systems LPEX Editor のパーサー・アクションおよび設定

追加の設定およびアクションが System i の言語で使用可能です。

- 列依存編集: この機能は RPG および DDS などの列に依存する言語に便利です。通常ウィンドウ・アプリケーションで、テキストを挿入したり削除すると、残りのテキストが左右にプッシュされ、こうした言語ではこのために構文エラーが起こります。列依存編集機能を有効にすると、言語に対して指定されたコラムへの挿入および削除が制限されます。
- シグニチャー: RPG および DDS で使用可能です。この機能により、指定したシグニチャーで各行に自動的にフラグを立てることができます。 Remote Systems LPEX Editor の変更された行には、メンバー・タイプにかかわらず、SEU に変更された日付があることに注意してください。
- 自動大文字化: 変更された行を大文字にします。CL、DDS、RPG、COBOL メンバーで使用可能です。
- 自動インデント: 続く行で Enter (キー) が押されると、カーソルがインデントされ、ソースを見やすく表示することができます。CL、RPGLE で使用可能です。
- 自動編集: 指定した設定に従って、ソースの入力時にソースをフォーマットします。 CL およびフリー・フォームの SQLRPGLE で使用可能です。
- Open/Browse /COPY メンバーまたはサンプル集: RPG および COBOL 言語で、「Editor」メニューのポップアップを介して、ソースで参照されたメンバーを開いたりブラウズできます。
- ブロック・ネストの表示: **Ctrl+Shift+O** を使用して、またはポップアップ・メニューの「**Source**」メニューから、カーソル位置でネスト・レベルを示す矢印を表示できます。



- フィールドの表示: ファイルがプログラムで参照されている場合、ポップアップ・メニューからこのメニュー・オプションを使用して System i テーブル・ビューのファイルにフィールドを表示できます。RPG、COBOL、および CL で利用可能です。
- RPG アクション:
  - フリー・フォームへ変換 (RPGLE)
  - ILE へ変換 (RPG)
  - インデントの表示 (RPG、RPGLE)

コンテンツ・アシスト、テンプレート、および RPG ウィザードは、編集作業の速度を速め、改善できるように設計されています。

追加 LPEX キー

Remote Systems LPEX Editor では、ほとんどの機能はメニューおよびキー・ストロークで使用可能です。以下は LPEX で便利と思われる追加キーのリストです。

表 3.

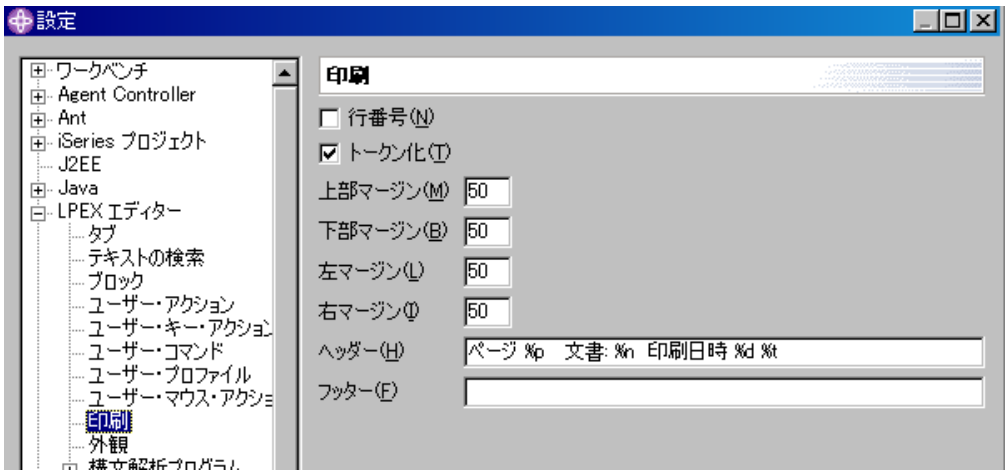
キーの組み合わせ	説明
Ctrl+Home	先頭に移動します (SEU では TOP)
Ctrl+End	文末に移動します (SEU では BOTTOM)
Ctrl+L	行番号に移動します (また、SEU の処理と同じように接頭部域に行番号を入力します)

表 3. (続き)

キーの組み合わせ	説明
Alt+S	行を分割します
Alt+J	行を結合します
Alt+L	行を選択します
Alt+R	長方形を選択します
Ctrl+W	すべての行を表示します (行がフィルターされている場合に便利です)
Ctrl+Z	取り消し
Ctrl+Y	再実行
Ctrl+S	保管
Ctrl+M	突き合わせ (マッチング・ブラケットを選択します。CL および RPG などの言語では、DO/ENDDO、IF/ENDIF などの制御ステートメントを選択します)
Ctrl+Shift+M	突き合わせを検索します

印刷

Windows アプリケーションと同じように、「**File**」>「**Print**」メニュー・オプションから、または **Ctrl+P** を押して印刷を行うことができます。これは編集中に行うことができます。「**Tokenized**」チェック・ボックスを選択している限り、印刷はまた、印刷済みのソースでトークン化します。Remote Systems LPEX Editorの印刷は、System i プリンターではなく、ご使用の Windows プリンターで行われます。印刷オプションは、「**ウィンドウ**」>「**設定 (Preferences)**」>「**LPEX Editor**」>「**印刷**」で使用可能です。



次の置換変数は、ヘッダーおよびフッターで使用可能です。

- %p: ページ番号
- %n: ソース名、基本ファイル名、または文書名
- %f: 絶対パス・ファイル名または文書名
- %d: 日付

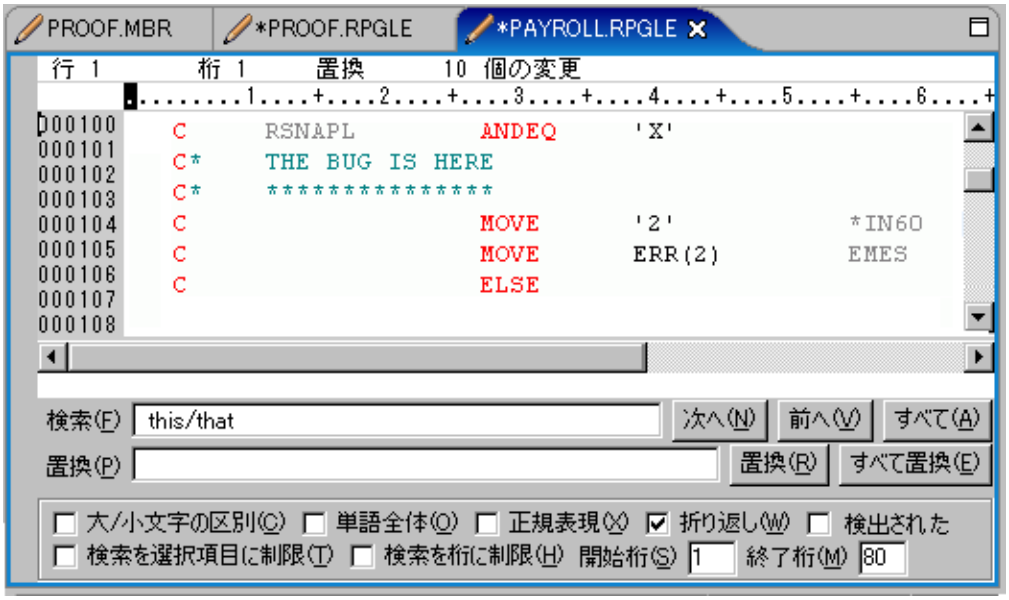
- %t: 時間

ヒント:

- ホスト・システムに印刷するには、「Remote Systems」ビュー、および印刷オプションで STRSEU コマンドを呼び出した System i テーブル・ビューにユーザー・アクションを追加します。
- カラー・プリンターには、トークン化を使用した印刷が最適です。

Remote Systems LPEX Editor での検索および置換

Remote Systems LPEX Editor で、**Ctrl+F** を使用して LPEX の検索機能を開くことができます。検索機能は大変柔軟な機能です。パターンを検索できる正規表現を指定できます。例えば、「**Regular expression**」チェック・ボックスを選択して、this|that を検索ストリングとして指定した場合、エディターは this または that を含む行を検索します。**Ctrl+N** または **Shift+F4** を使用して、次の一致を検索します。



ヒント:

- 列依存編集が有効で、置換によってテキストが不適切にシフトしていないことを確認します。
- 「**All**」ボタンをクリックして、検索ストリングを含まないすべての行をフィルターします。そうすると、一致している行だけが表示されます (接頭部域の前で **+** をクリックするか、または **Ctrl+W** を押すと、簡単に再表示できます)。

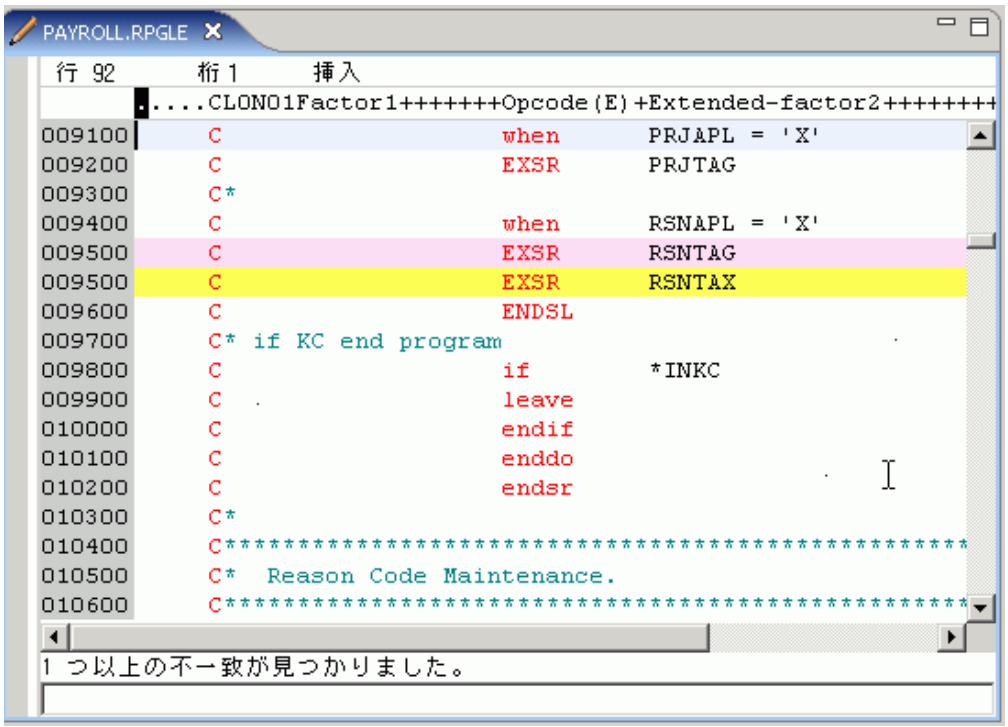
詳細については、クライアント製品のオンライン・ヘルプにあるトピック『テキストの*検索および置換 (英語)*』を参照してください。また、チュートリアルの『テキストの*検索および置換 (英語)*』も参照してください。

Remote Systems LPEX Editor でのファイルの比較

Remote Systems LPEX Editor でメンバーの比較をするには、エディターでメンバーを開く必要があります。 1 度開いたら、ツールバーの「**Compare**」ボタンを選択するか、または「**Edit**」>「**Compare**」>「**Compare**」ファイル・メニュー・オプションから、あるメンバーを他のメンバーと容易に比較できます。

1 度比較がトリガーされると、ソースが、色別にフラグが立てられた異なる行でマージされて表示されます。ピンクは、比較されているソースに使用する色で、黄色は開かれたソースの色です。

スプール・ファイルおよび SEU に開かれたソース間で反転する必要のある System i とは異なり、Remote Systems LPEX Editor において比較をすると、元々開かれていたメンバーの変更を続行できます。次の不一致にナビゲートするには **Ctrl+Shift+N** を、前の不一致にナビゲートするには **Ctrl+Shift+P** を使用します。ソースを変更する場合、**Ctrl+Shift+R** を使用して比較をリフレッシュし、最後に「**Edit**」>「**Compare**」>「**Clear**」で終了します。



ヒント:

- 「**ウィンドウ**」>「**Preferences**」>「**LPEX Editor**」>「**Compare**」で追加設定を指定します。
- Eclipse の他の比較ツールとは異なり、Remote Systems LPEX Editor はシーケンス番号を認識するため、シーケンス番号が変更された場合のミスマッチは発生しません。
- ファイルの比較の詳細については、クライアント製品のオンライン・ヘルプにあるトピック『*RPG、COBOL、CL、C、C++、および DDS メンバーの編集 (英語)*』を参照してください。また、チュートリアルの『*「Remote Systems」ビューからのファイルの違いの比較 (英語)*』を参照してください。

Remote Systems LPEX Editor からのコンパイル

Remote Systems LPEX Editor でソースを開く場合、「Remote Systems」ビューまたは System i テーブル・ビューに移動してコンパイルを実行するのは便利ではありません。かわりに、次の 1 つを使用できます。

- ツールバー・ボタン (メンバー・タイプに対して最後に使用されたコマンドを使用して、プロンプトなしでコンパイルを実行します)
- **Ctrl+Shift+C** (メンバー・タイプに対して最後に使用されたコマンドを使用して、プロンプトなしでコンパイルを実行します)
- コンパイル・メニュー (プロンプトあり、およびプロンプトなしでのコンパイルを選択でき、項目に対して実行したいどのコンパイル・コマンドも選択できます) コンパイル前に保管していない場合は、そうするようにプロンプトが出されます。

詳細については、クライアント製品のオンライン・ヘルプにあるトピック『コンパイル *(英語)*』を参照してください。また、チュートリアルの『*リモート側でのソースのコンパイル (英語)*』も参照してください。

Remote Systems LPEX Editor と SEU の比較

次の表では、このトピックで説明された Remote Systems LPEX Editor の機能と、SEU の同等もしくは類似の機能を比較します。

表 4.

Remote Systems LPEX Editor	SEU
編集またはブラウズ・モードでメンバーのポップアップ・メニューからエディターを起動	PDM のオプション 5 で SEU を起動
編集およびブラウズの両方でフルスクリーン・モード (エディター・タブをダブルクリック)。しかし、Remote Systems LPEX Editor では、SEU よりもより多くの行がフルスクリーンで表示されます。	ブラウズ時のみフルスクリーン・モード (F13)
編集およびブラウズ用に画面を分割: • エディター・タブをドラッグ・アンド・ドロップして、1 度により多くのメンバーを表示 • エディター・ビューのポップアップ、または **Ctrl+2** を使用して、同じメンバーの異なる部分を処理するために、現行のエディター・ビューを分割 (最大 5 個まで分割可能)	画面の分割およびブラウズ
言語トークンを色別に表示 (トークン化)	利用不可
シーケンス番号を含む接頭部域を提供し、SEU 編集コマンドをサポート	接頭部域は編集コマンドで使用可能
日付領域をシーケンス番号のとなりに表示。デフォルトではオフですが、設定またはポップアップ・メニューから使用可能にできます。	日付領域は右方にあって、常に表示
項目を展開 (+ をクリック)、または縮小 (- をクリック) して除外された行を表示、または非表示	除外された行は表示できない
自動構文検査 - すべてのエラーをすぐに表示	自動構文検査 - 最初のエラーを表示

表 4. (続き)

Remote Systems LPEX Editor	**SEU**
コンテンツ・アシスト、コード・テンプレート、および RPG ウィザードが、コード作成における支援のために使用可能	プロンプターが、コード作成における支援のために使用可能
ほとんどの編集機能が、メニューおよびキー・ストロークから使用可能	すべての編集機能が、キー・ストロークから使用可能
印刷が「**File**」メニュー、または **Ctrl+P** から使用可能	STRSEU、オプション 6 を使用して印刷
編集中に印刷可能	利用不可
比較のためにエディターに開かれたメンバーは変更可能	ソースを編集するには、SEU でスプール・ファイルからソースにスイッチ

第 1 部 ILE COBOL プログラムのコンパイル、実行、およびデバッグ

第 1 章 概要

COBOL (COmmon Business Oriented Language) は、英語に類似したプログラミング言語です。その名が示すとおり、COBOL はビジネス問題の処理に特に効果的です。 COBOL は、データ項目および入出力レコードの記述および処理に重きを置いています。したがって、COBOL は大きなデータ・ファイルの処理に適しています。

この章では、次のことを記載しています。

- Integrated Language Environment (ILE) の紹介
- ILE COBOL の紹介
- OPM COBOL/400 では使用できない、ILE COBOL に組み込まれた機能の概要
- 実行可能プログラム・オブジェクトの作成における主なステップの概要
- ILE COBOL アプリケーションの開発をさらに効果的に行うために利用できる、他の適用業務開発ツールの概要

Integrated Language Environment

Integrated Language Environment (ILE) は、IBM i プログラム・モデルの発展における最新のステージです。各ステージは、アプリケーション・プログラマーのニーズが変わるのに合わせて発展してきました。 ILE に関連した概念と用語の詳細については、「*ILE 概念*」を参照してください。

i5/OSが最初に発表されたときに提供されたプログラミング環境は、**オリジナル・プログラム・モデル (OPM)** と呼ばれます。 このモデルのもとで、COBOL、RPG、CL、BASIC、および PL/1 は作動します。バージョン 1 リリース 2 で、**拡張プログラム・モデル (EPM)** が導入されました。 EPM は C、Pascal、および FORTRAN などの言語をサポートするために作成されました。 OPM および EPM の基本特性の詳細については、「*ILE 概念*」を参照してください。

OPM COBOL/400 環境と ILE COBOL 環境との間の最大の相違点は、実行可能プログラム・オブジェクトの作成方法にあります。 ILE COBOL コンパイラーは実行可能プログラム・オブジェクトを作成しません。 1 つまたは複数の**モジュール・オブジェクト**を作成し、それらをさまざまな組み合わせで**バインド**して、**プログラム・オブジェクト**と呼ばれる 1 つまたは複数の実行可能単位を作成することができます。

ILE では、さまざまな言語で作成されたモジュール・オブジェクトをバインドすることができます。これにより、COBOL、RPG、C、C++、および CL で別々に作成されたモジュール・オブジェクトで構成される、実行可能プログラム・オブジェクトを作成することができます。

実行可能 ILE COBOL プログラム・オブジェクト作成の主要なステップ

図 1 に、ILE COBOL で作成される実行可能プログラム・オブジェクトの開発に含まれる一般的なステップを示します。

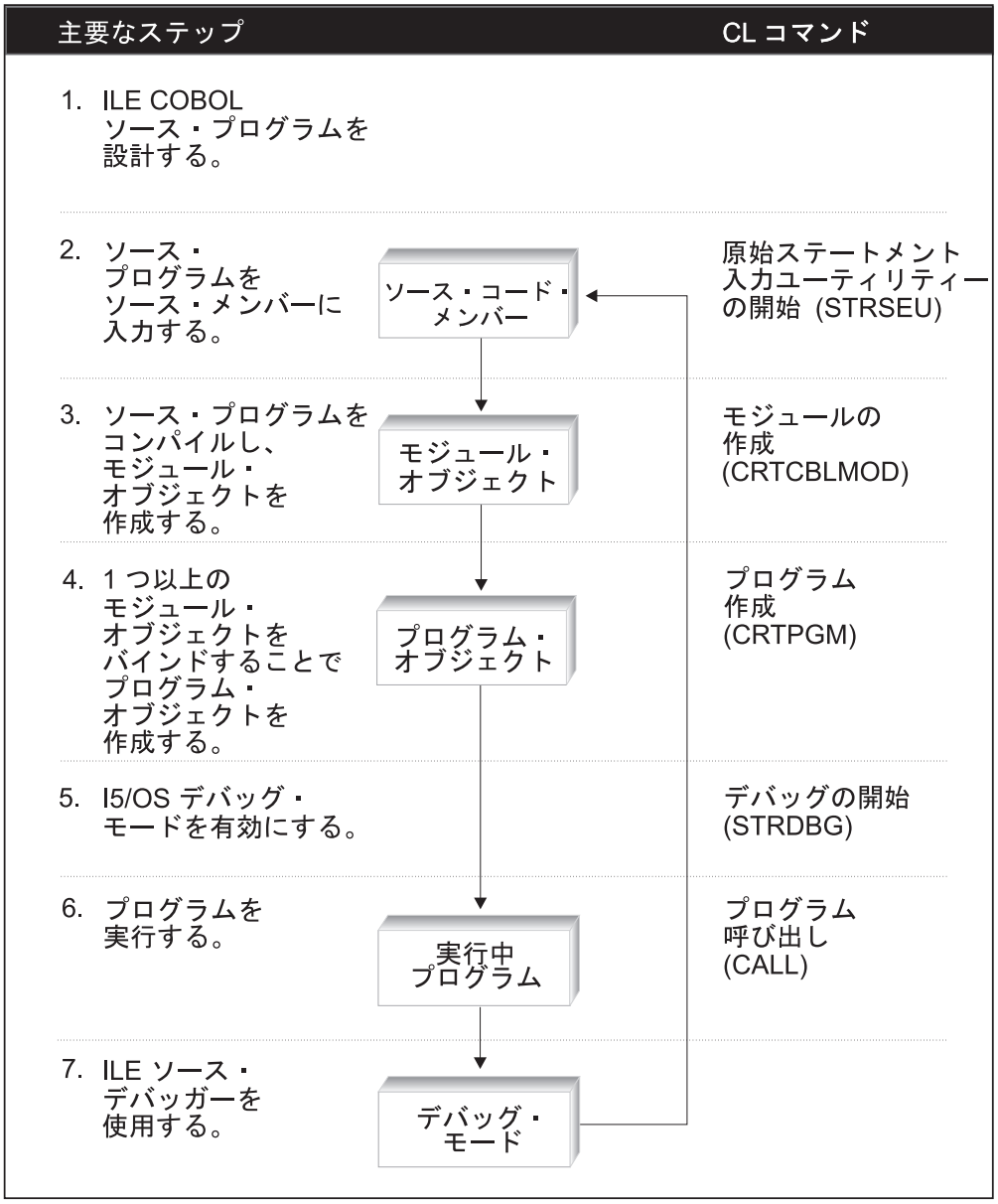


図 1. 実行可能 ILE COBOL プログラム・オブジェクト作成の主要なステップ

ステップ 3 および 4 は、単一のコマンド CRTBNDCBL で実行されます。このコマンドは ILE COBOL ソース・プログラムから一時モジュール・オブジェクトを作成して、それからプログラム・オブジェクトを作成します。プログラム・オブジェクトが作成されると、モジュール・オブジェクトは削除されます。

ILE COBOL ソース・プログラムの設計

実行可能 ILE COBOL プログラム・オブジェクトの作成での最初のステップは、ILE COBOL ソース・プログラムを設計することです。

ILE COBOL ソース・プログラムは、4 つの部から構成されます。 6 ページの図 2 に示すプログラムの骨組みは、ILE COBOL ソース・プログラムの構造を示します。これは、ILE COBOL ソース・プログラムの設計のサンプルとして使用されます。

ILE COBOL プログラムは、他の ILE COBOL プログラムの中に含めることができます。この概念はネストと呼ばれ、含められるプログラムは**ネストされたプログラム**と呼ばれます。 6 ページの図 2 に、ネストされた ILE COBOL プログラムが最も外側の ILE COBOL プログラムにどのように組み込まれるかを示します。プログラムの骨組みで提供される項目のすべてが必要であるわけではありません。そのほとんどは情報の目的でのみ提供されています。

```
 IDENTIFICATION DIVISION. 1
   PROGRAM-ID. outermost-program-name.
   AUTHOR. comment-entry.
   INSTALLATION. comment-entry.
   DATE-WRITTEN. comment-entry.
   DATE-COMPILED. comment-entry.
   SECURITY.
*   The SECURITY paragraph can be used to specify
*   copyright information pertaining to the
*   generated module object.  The first 8 lines
*   of the SECURITY paragraph generate the
*   copyright information that is displayed on
*   the Copyright Information panel when the
*   Display Module (DSPMOD) CL command is issued.
 ENVIRONMENT DIVISION. 2
  CONFIGURATION SECTION. 3
   SOURCE-COMPUTER. IBM-ISERIES.
   OBJECT-COMPUTER. IBM-ISERIES.
   SPECIAL-NAMES.   REQUESTOR IS CONSOLE.
 INPUT-OUTPUT SECTION. 4
   FILE-CONTROL.
     SELECT file-name ASSIGN TO DISK-file-name
       ORGANIZATION IS SEQUENTIAL
       ACCESS MODE IS SEQUENTIAL
       FILE STATUS IS data-name.
 DATA DIVISION. 5
  FILE SECTION.
  FD file-name.
  01 record-name  PIC X(132).
  WORKING-STORAGE SECTION.
  77 data-name PIC XX.
  LINKAGE SECTION.
 PROCEDURE DIVISION. 6
  DECLARATIVES
  END DECLARATIVES.
  main-processing SECTION.
   mainline-paragraph.
    ILE COBOL statements.
       STOP RUN.
 IDENTIFICATION DIVISION. 7
   PROGRAM-ID. nested-program-name.
 ENVIRONMENT DIVISION. 8
  INPUT-OUTPUT SECTION.
   FILE-CONTROL.
     SELECT file-name ASSIGN TO DISK-file-name
       ORGANIZATION IS SEQUENTIAL
       ACCESS MODE IS SEQUENTIAL
       FILE STATUS IS data-name.
 DATA DIVISION.
  FILE SECTION.
  FD file-name.
  01 record-name  PIC X(132).
  WORKING-STORAGE SECTION.
  77 data-name PIC XX.
  LINKAGE SECTION.
 PROCEDURE DIVISION.
  DECLARATIVES
  END DECLARATIVES.
  main-processing SECTION.
   mainline-paragraph.
    ILE COBOL statements.
          EXIT PROGRAM.

 END PROGRAM nested-program-name. 9
 END PROGRAM outermost-program-name.
```

図 2. ILE COBOL プログラム構造の例

- 見出し部 (IDENTIFICATION DIVISION) 1 は、組み込みが必須である唯一の部です。他のすべての部の組み込みはオプションです。
- 環境部 (ENVIRONMENT DIVISION) 2 は 2 つのセクションから構成されます。構成セクション 3 は、ソースおよびオブジェクト・コンピューターの全体的な仕様を説明し、入出力セクション 4 は、各ファイルを定義して、外部メディアと ILE COBOL プログラムとの間のデータの伝送に必要な情報を指定します。
- データ部 (DATA DIVISION) 5 は、ファイル内に含まれるプログラムとレコードで使用されるファイルを記述する部分です。また、必要となる内部作業記憶域またはローカル・ストレージ・データ項目も記述します。
- 手続き部 (PROCEDURE DIVISION) 6 はオプションの宣言で構成され、その処理には、セクションまたは段落 (あるいはその両方)、文、およびステートメントが含まれます。
- この 2 番目の見出し部 7 は、最も外側の ILE COBOL プログラムの中に含まれるネストされた ILE COBOL プログラムの始まりです。
- ネストされたプログラムには、環境部に構成セクション 8 がありません。構成セクションのオプションが必要なら、それは最も外側のプログラムで指定しなければなりません。
- ネストされたプログラムおよび最も外側のプログラムは、END PROGRAM 9 ヘッダーによって終了しなければなりません。

ILE COBOL プログラムは、IDENTIFICATION DIVISION 内の PROGRAM-ID によって識別されます。これには特定のタスクを実行する、自己完結型のステートメントの集合が含まれています。

ILE では、ILE COBOL ソース・プログラムは **ILE プロシージャー**と見なされています。 ILE COBOL プログラムに、ネストされた ILE COBOL プログラムが含まれている場合、ネストされた各 ILE COBOL プログラムは ILE プロシージャーです。ネストされるプログラムの名前は、それを含むプログラム内においてだけ認識されます。ネストされるプログラムに COMMON 属性が付いている場合、ネストされるプログラムの名前も同じコンパイル単位の他のプログラムに認識されます。ILE プロシージャーを、COBOL プロシージャーと混同しないようにしてください。後者は COBOL プログラムの手続き部 (PROCEDURE DIVISION) にあり、それにはセクション、段落、文、およびステートメントが含まれます。

ILE COBOL プログラムの作成の詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

ソース・メンバーへのソース・ステートメントの入力

ILE COBOL プログラムを設計したら、ソース・メンバーにそのプログラムを入力する必要があります。

WebSphere Development Studio Client for System i を使用してください。Remote Systems LPEX editor により、プログラム編集作業は単純化されます。エディターは、ご使用のワークステーションまたは System i のソース・ファイルにアクセスできます。コンパイルがエラーに終わると、コンパイラーのメッセージから直接、ソ

ースを含むエディターに飛ぶことができます。 エディターは、ユーザーが問題のソース・ステートメントを修正できるよう、カーソルが問題のソース・ステートメントの位置にある状態で開きます。

原始ステートメント入力ユーティリティー (SEU) の原始ステートメント入力ユーティリティーの開始 (STRSEU) コマンドを使用することによって、ILE COBOL ソース・ステートメントを入力および編集することもできます。正確な ILE COBOL ステートメントをシステムに入力するのに役立てるために、SEU の表示画面は標準 COBOL コーディング形式に対応しており、コードの行を入力または変更するときに、COBOL 構文検査機能がエラーの行を検査します。

コンパイル単位とは、最外部の ILE COBOL プログラムと、その最外部プログラム内でネストされた何らかの ILE COBOL プログラムのことです。複数のコンパイル単位が単一のソース・メンバーの中に入力されている場合もあります。

ソース・ステートメントの入力については 13 ページの『第 2 章 ソース・メンバーへのソース・ステートメントの入力』を参照してください。

ソース・プログラムをコンパイルしてモジュール・オブジェクトにする

ソース・メンバーのソース・ステートメントの入力または編集が完了したなら、COBOL モジュールの作成 (CRTCBLMOD) コマンドを使用して、モジュール・オブジェクトを作成する必要があります。このコマンドは、ソース・メンバー内のソース・ステートメントをコンパイルして、1 つまたは複数のモジュール・オブジェクトにします。ソース・メンバー内の各コンパイル単位は、それぞれ別個のモジュール・オブジェクトを作成します。

モジュール・オブジェクトとは、ILE COBOL コンパイラーの出力のことです。それらは *MODULE と入力するとシステムに表示されます。モジュール・オブジェクトを実行するには、まずバインドしてプログラム・オブジェクトにしなければなりません。

CRTCBLMOD コマンドを使用したモジュール・オブジェクトの作成の詳細については 25 ページの『第 3 章 ソース・プログラムのモジュール・オブジェクトへのコンパイル』を参照してください。

プログラム・オブジェクトの作成

実行可能**プログラム・オブジェクト**を作成するためには、モジュール・オブジェクトをバインドしなければなりません。 1 つまたは複数のモジュール・オブジェクトをバインドすることにより、プログラム・オブジェクトが作成されます。実行できるのはプログラム・オブジェクトだけなので、モジュール・オブジェクトをバインドしてプログラム・オブジェクトにすることが必要です。さまざまなプログラム言語で作成されたモジュール・オブジェクトをバインドして、1 つのプログラム・オブジェクトを作成することができます。たとえば、報告書には COBOL または RPG モジュール・オブジェクトを使用し、計算には C モジュール・オブジェクトを使用して、1 つのプログラム・オブジェクトを構成することができます。プログラム・オブジェクトは、以下のコマンドのうちのいずれか 1 つを使って作成できます。

- プログラムの作成 (CRTPGM)
- バインド COBOL プログラムの作成 (CRTBNDCBL)

これらのコマンドについては 89 ページの『第 4 章 プログラム・オブジェクトの作成』を参照してください。

プログラム・オブジェクトの実行

プログラム・オブジェクトは、それを呼び出すことによって実行します。プログラム・オブジェクトを呼び出すには、次のいずれかの方法を使用できます。

- 任意のコマンド入力行で CALL CL コマンドを出す
- 高水準言語 CALL ステートメント
- アプリケーション指向のメニュー
- ユーザー作成コマンド
- WebSphere Development Studio Client for System i ワークベンチの、メニュー・アクションの「実行」またはツールバー・アイコンの「実行」

プログラムの実行の詳細については 127 ページの『第 6 章 ILE COBOL プログラムの実行』を参照してください。

プログラムのデバッグ

ILE ソース・デバッガーを使用して、プログラム・オブジェクトおよびサービス・プログラムのエラーを検出および除去できます。 ILE ソース・デバッガーは、以下のことを実行するために使用します。

- プログラム・ソースの表示
- 条件付き停止点および無条件停止点の設定と除去
- プログラムのステップスルー
- 変数、構造、レコード、および配列の値の表示
- 変数の値の変更
- 参照有効範囲の変更
- 変数、式、またはデバッグ・コマンドに短縮名を付ける

デバッガーの詳細については 133 ページの『第 7 章 プログラムのデバッグ』を参照してください。

その他の適用業務開発ツール

System i は、プログラミングに役立つツールをフルセットで提供します。

IBM Rational Development Studio for System i

IBM Rational Development Studio for System i は、ユーザーが System i 用 e-ビジネス・アプリケーションの数を急速に、そしてコスト効率良く増やすことを支援するアプリケーション開発パッケージです。このパッケージは、すべての主要な System i 開発ツール、ホストおよびワークステーションの両方を総合し、 1 つの System i5 として提供します。

ホスト開発ツールには大幅な改善が行われました。最近のコンパイラーの機能強化には既存のバージョンに代わり、最新の AIX コンパイラーによって完全に更新された C および C++ コンパイラーが含まれています。これは、お客様およびソリュー

ション・プロバイダーが他のプラットフォームから e-ビジネス・ソリューションを移植する際に役立ちます。 ILE RPG にも大幅な強化が行われました。 Java の相互運用性の向上とフリー・フォーム C 仕様が最大の機能拡張点です。 COBOL には、COBOL/Java 相互運用性機能の導入と共に、z/OS マイグレーション機能が追加されました。言語に関する新しい機能の完全なリストについては、『言語解説書』または『プログラマーの手引き』の「新機能」をご覧ください。

WebSphere Development Studio for System i には、以下のコンポーネントが含まれています。

ホスト・コンポーネントには、以下のものがあります。

- ILE RPG
- ILE COBOL
- ILE C/C++
- Application Development ToolSet (ADTS)

ワークステーション・コンポーネントには、以下のものがあります。

- IBM WebFacing Tool
- System i 開発ツール: Remote System Explorer および System i プロジェクト
- Java 開発ツール (System i 機能拡張付き)
- Web 開発ツール (System i 機能拡張付き)
- Struts 環境のサポート
- データベース開発ツール
- Web サービス開発ツール
- システム開発ツール
- XML 開発ツール
- CODE
- VisualAge RPG
- 統合 i5/OS デバッガー

WebSphere Developoment Studio for System i

WebSphere Development Studio for System i (Development Studio Client) は、ユーザーが System i 用 e-ビジネス・アプリケーションの数を急速に、そして高いコスト効率で増やすことを支援するアプリケーション開発用のワークステーション・ツールのパッケージです。

このパッケージは、ワークステーション・ベースのすべての主要な System i 開発ツールを統合し、 1 つの System i5 として提供します。

WebSphere Development Studio for System i の機能リスト

ワークベンチ・ベースの統合開発環境

IBM WebSphere Development Studio Client for System i は、WebSphere Studio Workbench (WSWB) バージョン 2.1 を使用しています。

IBM WebFacing Tool

IBM WebFacing Tool は、ユーザーの DDS 表示のソース・ファイルを、ブラウザー上で実行できるアプリケーションに変換します。

Remote System Explorer および System i 開発ツール

System i 開発ツールの一部として含まれた Remote System Explorer は、System i 機能のフレームワーク、ユーザー・インターフェース、編集機能、およびファイル/コマンド/ジョブのアクションを包括します。

System i Java 開発ツール

Java 開発ツールと System i Java 開発ツールを使用すると、Java アプリケーションを開発し、また、Java アプリケーション開発用にプログラム言語 Java で作成されたプログラムを作成、コンパイル、テスト、デバッグ、および編集することができます。

System i Web 開発ツール

Web 開発ツールを使用すると、 Web ベースのフロントエンドを使用し、i5/OS ホストに常駐する ILE 言語および非 ILE 言語のプログラムに含まれるビジネス・ロジックと通信する e-ビジネス・アプリケーションを新しく作成することができます。

Struts 環境のサポート

Development Studio Client は Struts および Web ダイアグラム・エディターに対するサポートを提供します。

データベース開発ツール

データベース開発ツールは、Java Database Connectivity (JDBC) ドライバーを持つローカルまたはリモートのあらゆるデータベースをサポートします。

Web サービス開発ツール

Web サービス開発ツールにより、開発者はワールド・ワイド・ウェブ上で呼び出すことのできるモジュラー・アプリケーションを作成することができます。

システム開発ツール

システム開発ツールは、ローカル側またはリモート側でインストールされた実行時環境でアプリケーションをテストするために使用します。

XML 開発ツール

XML 開発ツールは、XML ベースの開発をサポートします。

CODE (CoOperative Development Environment)

CODE は System i 開発のための Windows ツールの標準セットです。 ソースと DDS ファイルの作成、およびプロジェクトの管理のための、一組のユーティリティーを提供します。

VisualAge RPG

VisualAge RPG は、ワークステーション上でクライアント/システム・アプリケーションを作成および保守することができる、視覚的な開発環境です。

統合 i5/OS デバッガー

統合 i5/OS デバッガーでは、 i5/OS または Windows システム上で実行しているコードを、ユーザーのワークステーション上のグラフィカル・ユーザー・インターフェースを使用してデバッグすることを支援します。

WebSphere Development Studio for System i についてより詳しく知るには、http://www.ibm.com/systems/i/infocenter/ にて、ワールド・ワイド・ウェブ上で得られる現行の情報を参照してください。

第 2 章 ソース・メンバーへのソース・ステートメントの入力

この章では、ILE COBOL ソース・ステートメントを入力するために必要な情報について説明します。そのステップを完了するために必要なツールや方法論についても簡潔に説明します。

クライアント・ツール入門については、『クライアント製品のアプリケーション開発ツールの使用』を参照してください。

システムに ILE COBOL ソース・ステートメントを入力するには、次のどちらかの方法を使用してください。

1. 原始ステートメント入力ユーティリティー (SEU) を使用して、ソース・ステートメントを入力する。この方法については、この章で詳しく説明します。
2. IBM i CL コマンドの CPYFRMTAP および CPYFRMDKT を使用して、ディスケットまたはテープからソース・ステートメントを入力する。

   CL コマンドを使用してソース・ステートメントを入力する方法に関する情報を入手するには、 Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** のカテゴリー「*プログラミング*」の中の『*CL および API*』セクションを参照してください。
3. EDTF (ファイルの編集) CL コマンドを使用して、ストリーム・ファイルにソース・ステートメントを入力する。このコマンドは、ILE COBOL ソース・ステートメントの入力にも使用できる、汎用のストリーム・ファイル・エディターです。ただし、このエディターは、ソース入力に役に立つ、SEU のような、構文検査機能や特殊フォーマット行を備えていません。ストリーム・ファイルに ILE COBOL ソース・ステートメントを保管したいが、SEU の構文検査機能も使用したい場合は、以下のステップを実行します。
   - SEU を使用して、ソース・ステートメントをファイル・メンバーに入力する
   - CPYTOSTMF (ストリーム・ファイルへコピー) コマンドを使用して、ファイル・メンバーの内容をストリーム・ファイルにコピーする

ライブラリーおよびソース物理ファイルの作成

ソース・ステートメントは物理ファイルの 1 つのメンバーに入力されます。ソースを入力する前に、ライブラリーとソース物理ファイルをまず作成しなければなりません。

ライブラリーとは、他のオブジェクトのディレクトリーとなるシステム・オブジェクトのことです。ライブラリーは関連したオブジェクトをグループ化して、名前によってオブジェクトを検索することができます。ライブラリーのオブジェクト・タイプは *LIB です。

ソース物理ファイルとは、メンバーを保管するファイルのことです。 それらのメンバーには、ILE COBOL ソース・ステートメントなどのソース・ステートメントが入れられます。

MYLIB というライブラリーを作成するには、以下のライブラリー作成 (CRTLIB) コマンドを使用します。

```
CRTLIB LIB(MYLIB)
```

ライブラリー MYLIB 内に QCBLLESRC というソース物理ファイルを作成するには、以下のソース物理ファイル作成 (CRTSRCPF) コマンドを使用します。

```
CRTSRCPF FILE(MYLIB/QCBLLESRC)
```

注: 上記の例では、ソース物理ファイルを作成する前に MYLIB ライブラリーが存在していなければなりません。

ライブラリーおよびソース物理ファイルの詳細については、「*AS/400 プログラム開発管理機能 (PDM)*」を参照してください。

ライブラリーとソース物理ファイルを作成したなら、編集セッションを開始することができます。編集セッションを開始してソース・ステートメントを入力するには、クライアントの製品エディター、または原始ステートメント入力ユーティリティーの開始コマンドを使用できます。

注: IBM i コピー機能を使って、ディスケットまたはテープからソース・プログラムを入力することもできます。 IBM i コピー機能の詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** のカテゴリー「プログラミング」の中の『*CL および API*』セクションを参照してください。

原始ステートメント入力ユーティリティーを用いてソース・ステートメントを入力する

原始ステートメント入力ユーティリティーには、COBOL の特別な表示形式が用意されています。それは COBOL コーディング用紙に対応したものであり、COBOL ソース・ステートメントの入力に役立つように設計されています。 図 3 に、SEU で提供される COBOL の表示形式の例を示します。 SEU では、位置ごとにソース・コードに入力したり変更を加えたりするのに役立つ仕様表示行を表示させることができます (**1** を参照)。

クライアント・ツール入門については、『クライアント製品のアプリケーション開発ツールの使用』を参照してください。

```
 桁 . . . . . :     1  71          編集                              MYLIB/QCBLLESRC
 SEU==> _____________________________________________________________    XMPLE1
FMT CB ......-A+++B++++++++++++++++++++++++++++++++++++++++++++++++++++ 1
       ***************  データの始め  ****************************************
0001.00        IDENTIFICATION DIVISION.
0002.00          PROGRAM-ID. XMPLE1.
0003.00
0004.00        ENVIRONMENT DIVISION.
0005.00        CONFIGURATION SECTION.
0006.00          SOURCE-COMPUTER.  IBM-ISERIES.
0007.00        INPUT-OUTPUT SECTION.
0008.00        FILE-CONTROL.
0009.00            SELECT FILE-1 ASSIGN TO DATABASE-MASTER.
       ***************  データの終わり  ************************************

 プロンプト・タイプ. . .  CB      順序番号  . . . . .  0008.00

   継続

  区域-A      区域-B
  FILE       -CONTROL.

F3= 終了     F4= プロンプト     F5= 最新表示            F11= 前のレコード
F12= 取り消し     F23= 選択プロンプト     F24= キーの続き
```

図 3. SEU 表示形式

SEU を使用したソース・ステートメントの入力方法については、「*AS/400 適用業務開発ツールセット AS/400 用 原始ステートメント入力ユーティリティー 使用者の手引きと参照*」を参照してください。

コンパイル単位とは、最外部の ILE COBOL プログラムと、その最外部プログラム内でネストされた何らかの ILE COBOL プログラムのことです。複数のコンパイル単位が単一のソース・メンバーの中に入力されている場合もあります。

COBOL ソース・ファイルの形式

ソース・ファイルの標準レコード長は 92 文字です。これらの 92 文字は 6 文字のシーケンス番号、80 文字のデータ・フィールド、および 6 文字の最終変更日の区域で構成されています。

ILE COBOL コンパイラーは 102 という追加レコード長をサポートします。補足情報を含む 10 文字のフィールドはレコードの最後になります (93 桁目 ～ 102 桁目の位置)。ここの情報は ILE COBOL コンパイラーによっては使用されませんが、コンパイラー・リストの右端に入れられます。このフィールドに情報を記入するのは、プログラマー自身の責任です。この追加フィールドを使用したい場合は、レコード長が 102 のソース・ファイルを作成してください。

ユーザー独自のファイルを作成する必要がなければ、ソース・レコードを保管できる場所にソース・ファイルが提供されます。このファイルは QCBLLESRC という名前で、ライブラリー QGPL 内にあり、レコード長は 92 文字です。

SEU の開始

SEU を使用して ILE COBOL ソース・プログラムを入力するには、SEU 開始 (STRSEU) コマンドを入力し、TYPE パラメーターに CBLLE を指定します。ソース・プログラムに組み込み SQL が含まれている場合、TYPE パラメーターに SQLCBLLE を指定します。

TYPE パラメーターを指定しない場合、SEU はデフォルト値としてメンバーが最後に編集されたときに使用されたのと同じタイプを使用します。 TYPE パラメーターを指定しないで新しいメンバーを作成している場合、SEU はソース物理ファイルの名前と関連したデフォルトのメンバー・タイプを割り当てます。 ILE COBOL の場合、このデフォルトのメンバー・タイプは CBLLE です。 SEU を開始する他の方法については、「*AS/400 適用業務開発ツールセット AS/400 用 原始ステートメント入力ユーティリティー 使用者の手引きと参照*」を参照してください。

クライアント・ツール入門については、『クライアント製品のアプリケーション開発ツールの使用』を参照してください。

SEU で COBOL 構文検査機能を使用する

COBOL 構文検査機能を SEU で使用するには、STRSEU コマンドの TYPE (CBLLE) パラメーターを指定します。 COBOL 構文検査機能は、新しい行を入力したり既存の行を変更する際に、エラーがないかどうか各行を検査します。誤ったソース・ステートメントが識別され、エラー・メッセージが表示されるので、プログラムをコンパイルする前にそのエラーを訂正できます。

クライアント・ツール入門については、『クライアント製品のアプリケーション開発ツールの使用』を参照してください。

ソース行を入力したり変更を加えるときはいつでも、ソース・コードの他の行は構文検査単位の一部として構文検査されます。単一の構文検査の単位の長さは、次のように、入力または変更された行から拡張することによって判別されます。

- 構文検査の単位は、ソース・メンバーの先頭の方に向かって最初のソース行の始まりまで、または行の最後になっているピリオドが見つかるまでをその範囲とします。
- 構文検査の単位は、ソース・メンバーの末尾の方に向かって最後のソース行の終わりまで、または行の最後になっているピリオドが見つかるまでをその範囲とします。

COBOL 構文検査機能は、入力または変更を加えた 1 文だけを、先行または後続する文から独立して検査するので、各ソース・ステートメント内の構文エラーだけが検出されます。未定義名および名前の参照間違いなどの、相互関係のエラーは検出されません。それらのエラーは、プログラムがコンパイルされるときに ILE COBOL コンパイラーによって検出されます。

反対に、変更されたのが見出し部のオプショナルの段落であるコメント記入項目の一部となっている文である場合、構文検査機能は、その文脈では任意の組み合わせの文字が入力可能であることを認識できません。文の内容を有効な COBOL ステートメントとして識別しようとするため、エラーを何度も生成してしまう可能性があります。これは、段落名と同じ行で始まる単一文としてコメント記入項目を入力することにより、または一連の複数のコメント行に置き換えることにより、回避することができます。

構文検査の単位にエラーがある場合、エラーがあると識別された単位の部分は反転表示されます。画面下部のメッセージは、単位内の最初のエラーについてのものです。

構文検査は、ソース・コードが入力されるたびに行われます。入力行の内容が不完全なステートメントであるなら、そのたびにエラー・メッセージが生成されます。これらのエラー・メッセージは、例にあるように、ステートメントが完了すると消えます。

```
 桁 . . . . . :     1  71         編集                      TESTLIB/QCBLLESRC
 SEU==> _____________________________________________________________  ADDATOB
 FMT CB ......-A+++B++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        ***************  データの始め  ****************************************
0000.10        IDENTIFICATION DIVISION.
0000.20        PROGRAM-ID. ADDATOB.
0000.30        ENVIRONMENT DIVISION.
0000.40        CONFIGURATION SECTION.
0000.50          SOURCE-COMPUTER. IBM-ISERIES.
0000.60          OBJECT-COMPUTER. IBM-ISERIES.
0000.70        DATA DIVISION.
0000.80        WORKING-STORAGE SECTION.
0000.90        01  A     PIC S9(8) VALUE 5.
0001.00        01  B     PIC S9(8) VALUE 10.
0001.10        PROCEDURE DIVISION.
0001.20        MAINLINE.
0001.30           MOVE A
'''''''
        ***************  データの終わり  ***************************************


F3= 終了  F4= プロンプト  F5= 最新表示  F9= コマンドの複写  F10= カーソル
F11= 切り替え    F16= 検索の反復    F17= 変更の反復      F24= キーの続き
COBOL 予約語または特殊文字 'TO' が必要である。'TO' と見なされます。          +
```

図 4. 不完全なステートメントに対して生成される COBOL 構文検査機能のエラー・メッセージ

```
 桁 . . . . . :     1  71         編集                      TESTLIB/QCBLLESRC
 SEU==> _____________________________________________________________  ADDATOB
 FMT CB ......-A+++B++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        ***************  データの始め  ****************************************
0000.40        IDENTIFICATION DIVISION.
0000.50        PROGRAM-ID. ADDATOB.
0000.60        ENVIRONMENT DIVISION.
0000.70        CONFIGURATION SECTION.
0000.80          SOURCE-COMPUTER. IBM-ISERIES.
0000.90          OBJECT-COMPUTER. IBM-ISERIES.
0000.91        DATA DIVISION.
0000.92        WORKING-STORAGE SECTION.
0000.93        01  A     PIC S9(8) VALUE 5.
0000.94        01  B     PIC S9(8) VALUE 10.
0001.00        PROCEDURE DIVISION.
0001.10        MAINLINE.
0002.00            MOVE A
0003.00            TO B.
'''''''
        ***************  データの終わり  ***************************************

F3= 終了  F4= プロンプト  F5= 最新表示  F9= コマンドの複写  F10= カーソル
F11= 切り替え    F16= 検索の反復    F17= 変更の反復      F24= キーの続き
```

図 5. COBOL 構文検査機能のエラー・メッセージはステートメントの完了後消える

最初の行が入力された段階でエラー・メッセージが生成されますが、2 番目の行が入力されてステートメントが完了すると消えます。

ILE COBOL ソース・コードの構文検査には、次の規則が適用されます。

- 7 桁目にアスタリスク (*) またはスラッシュ (/) がある行のソース・コードは構文検査されません。アスタリスクはコメント行を示します。スラッシュはコメント行とページ替えを示します。
- 構文検査の間、コンパイラー・オプションは考慮されません。

  たとえば、構文検査における 1 単位内で混用されていない限り、引用符またはアポストロフィはいずれも非数値区切りとして構文検査機能に受け入れられます。構文検査機能では、区切り文字が CRTCBLMOD または CRTBNDCBL コマンド内で、あるいは PROCESS ステートメント内で指定されているものかどうかは検査しません。
- SPECIAL-NAMES 段落の CURRENCY および DECIMAL-POINT 文節によって指定される文字置換は、対話式構文検査では考慮されません。
- COPY ステートメントの REPLACING *id-1* BY *id-2* 文節を使用する場合、それら ID のどちらかに参照変更が含まれるなら、SEU の COBOL 構文検査機能は括弧の対応しか検査しません。
- COPY ステートメントおよび REPLACE ステートメントは、構文構造が検査されます。
- 組み込み SQL ステートメントは構文検査されます。

ソース・メンバーへのソース・ステートメントの入力の例

クライアント・ツール入門については、『クライアント製品のアプリケーション開発ツールの使用』を参照してください。

この例は、ライブラリーおよびソース物理ファイルの作成方法、編集セッションの開始方法、およびライブラリー作成 (CRTLIB)、ソース物理ファイル作成 (CRTSRCPF) および SEU 開始 (STRSEU) コマンドを使用したソース・ステートメントの入力方法を示すものです。

注: それらのコマンドを使用してタスクを実行するには、それらのコマンドを使用するための権限がまず必要です。

1. MYLIB というライブラリーを作成するには、次のように入力します。

   ```
   CRTLIB LIB(MYLIB)
   ```

   そして、実行キーを押します。

   CRTLIB コマンドによって MYLIB というライブラリーが作成されます。
2. QCBLLESRC というソース物理ファイルを作成するには、次のように入力します。

   ```
   CRTSRCPF FILE(MYLIB/QCBLLESRC)
   TEXT ('Source physical file for an ILE COBOL for iSeries program')
   ```

   そして、実行キーを押します。

   CRTSRCPF コマンドによって、ライブラリー MYLIB 内に QCBLLESRC というソース物理ファイルが作成されます。
3. 編集セッションを開始し、ソース物理ファイル・メンバー XMPLE1 を作成するには、次のように入力します。

```
STRSEU SRCFILE(MYLIB/QCBLLESRC) SRCMBR(XMPLE1)
TYPE(CBLLE) OPTION(2)
```

そして、実行キーを押します。

STRSEU コマンドによって、ライブラリー MYLIB のファイル QCBLLESRC に新しいメンバー XMPLE1 が作成されます。

SEU の「編集」画面が、図 6 で示すように表示されます。

```
 桁 . . . . . :   1  71          編集                        MYLIB/QCBLLESRC
SEU==> _____________________________________________________________      XMPLE1
FMT CB ......-A+++B++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       ***************  データの始め  ****************************************
''''''''
''''''''
''''''''
''''''''
''''''''
''''''''
''''''''
''''''''
''''''''
''''''''
''''''''
''''''''
''''''''
''''''''
''''''''
       ***************  データの終わり  **************************************

F3= 終了  F4= プロンプト  F5= 最新表示  F9= コマンドの複写  F10= カーソル
F11= 切り替え     F16= 検索の反復     F17= 変更の反復     F24= キーの続き
メンバー XMPLE1 がファイル MYLIB/QCBLLESRC に追加された。                    +
```

図 6. 新しいメンバーの「編集」画面

4. SEU の「編集」画面でソースを入力します。
5. F3 (終了) を押して「終了」画面に移ります。 Y (はい) を入力して XMPLE1 を保管し、実行キーを押します。

   メンバー XMPLE1 が保管されます。

コード化文字セット ID の使用

コード化文字セット ID (CCSID) は、エンコード・スキーム ID、文字セット ID、コード・ページ ID、および使用されるコード化図形文字を固有に識別する追加コード関連情報の特定のセットを識別する番号です。

$、@、および # を除くすべての有効な ILE COBOL 文字は、Syntactic/Invariant 文字セット 640 に含まれます。このセットの文字のコード・ポイントは、すべての 1 バイト文字 EBCDIC コード・ページのコード・ポイントと同じです。ただし、コード・ページ 290 (ここでは、他のコード・ページで小文字の英字に使用されるコード・ポイントが、カタカナ文字に割り当てられます)、および " (引用符) 文字として異なるコード・ポイントを使用する特定のコード・ページを除きます。

注: @ および # 文字は、IBM 拡張および規則をサポートします。 @ 文字は、FORMAT 文節リテラル内の変換指定子として現れる可能性があります。 @ お

よび # 文字は、CALL ステートメント内でプログラム名を定義するリテラルを検査する場合に、有効として受け入れられます。

ILE COBOL コンパイラーは、コード・ページ 290 に基づくもの (たとえば、CCSID 290 または CCSID 930) を除いて、すべての 1 バイト文字または混合バイト文字の EBCDIC CCSID で書かれたソース・コードを受け入れます。ソース・コードがストリーム・ファイル中に保管されている場合は、 EBCDIC でない CCSID を持っている可能性があります。その場合は、コンパイラーは、ストリーム・ファイルをストリーム・ファイルの CCSID に関連した EBCDIC CCSID に変換してからソース・コードをコンパイルします。

CCSID はシステム全体で文字データの整合性を維持するために役立ちます。

Character Data Representation Architecture (CDRA) は、文字を表すのに使用されるコード・ポイントを識別するため、およびその文字の意味を保持するのに必要なコードを変換するための CCSID 値を定義します。

拡張 ACCEPT および DISPLAY ステートメントは CCSID 変換をサポートしていません。

CCSID をソース物理ファイルに割り当てる

CCSID は、各ソース・ファイルがシステムで作成される時点で、ソース・ファイルに割り当てられます。ソース物理ファイル作成時に CRTSRCPF コマンドの CCSID パラメーターを指定して使用したい文字セットを明示的に指定したり、デフォルトである *DFTCCSID をそのまま受け入れたりすることができます。たとえば、CCSID 273 でソース物理ファイルを作成するには、次のように入力してください。

```
CRTSRCPF FILE(MYLIB/QCBLLESRC) CCSID(273)
```

デフォルトを受け入れた場合、ジョブの CCSID がソース物理ファイルに割り当てられます。割り振られる CCSID は、ソース・ファイルが作成される System i が使用しているコード・ページによって異なります。

デフォルトの i5/OS 用 CCSID は CCSID 65535 です。システムの CCSID が 65535 の場合、ソース物理ファイルに割り当てられる CCSID は、ジョブの言語 ID によって判別されます。

CCSID の異なる複数のコピー・メンバーをソース・ファイルに含める

ILE COBOL ソース・プログラムは、複数のソース・ファイルから構成されます。コピー・ブックや DDS ファイルなどの 1 つの 1 次ソース・ファイルと複数の 2 次ソース・ファイルという構成にすることができます。

2 次ソース・ファイルの CCSID は、1 次ソース・ファイルの CCSID と異なるものにすることができます。この場合、2 次ファイルの内容は 1 次ソース・ファイルが ILE COBOL コンパイラーによって処理されるときに 1 次ソース・ファイルの CCSID に変換されます。

CCSID 65535 は、ソース・ファイルの変換が行われないということを暗黙のうちに指定するものです。 1 次ソース・ファイル、2 次ソース・ファイルのどちらか、ま

たはそれら両方に CCSID 65535 が割り当てられる場合、変換は行われません。 2 次ソース・ファイルに 1 次ソース・ファイルの CCSID によって指定される文字セットで認識されない文字が含まれていると、ILE COBOL コンパイラーから構文エラーが報告されることがあります。

形式 2 の COPY ステートメントを使用してソース・プログラムに DDS ファイル記述が組み込まれた場合、CCSID 変換は行われません。 DDS ソースの CCSID がコピー先ソース・メンバーの CCSID と違っている場合には、コピー元 DDS ソースに無効な文字がいくつか含まれていることがあります。これらの文字は構文エラーとしてフラグが付けられます。

1 次ソース・ファイルと 2 次ソース・ファイルで CCSID が違っており、そのどちらも CCSID 65535 ではない場合、コンパイル時のパフォーマンスに影響を与えることがあります。 ILE COBOL コンパイラーは、2 次ソース・ファイルの CCSID を 1 次ソース・ファイルの CCSID に変換するために時間を費やさなければなりません。この時間はソース・ファイルのサイズに応じてかなり長時間になる場合もあります。これは、以下の図のように示されます。

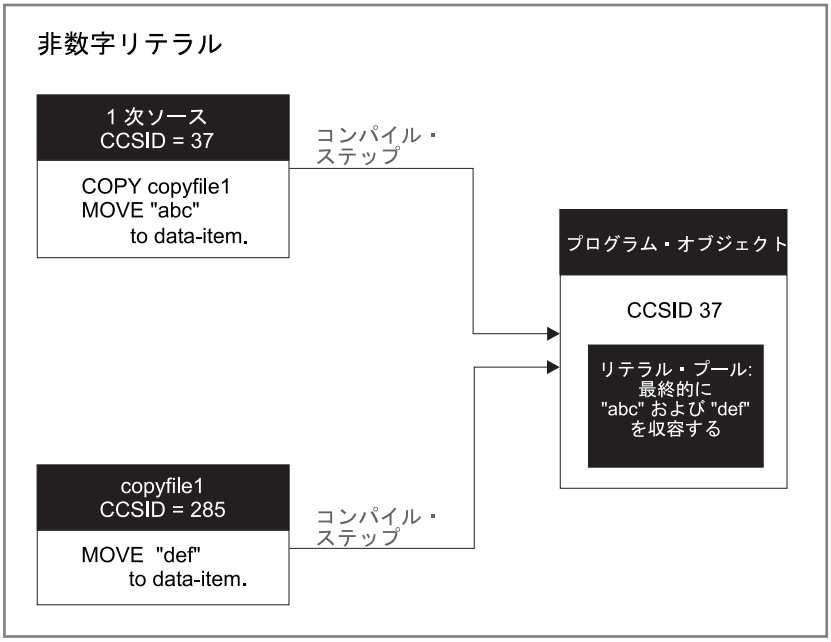


図 7. CCSID に基づくソース・ファイルの変換

SEU での COBOL 構文検査機能用 CCSID の設定

SEU の COBOL 構文検査機能の動作を ILE COBOL コンパイラーと同じにするためには、SEU ジョブの CCSID を編集中の 1 次ソース・ファイルの CCSID と同じになるように設定する必要があります。ほとんどの場合には、すでにそれらの設定は同じになっています。しかし、それらが異なっている場合には、CHGJOB コマンドの CCSID パラメーターに新しい CCSID 番号を指定することによって、ジョブの CCSID を変更できます。たとえば、現行ジョブの CCSID を 280 に変更するには、次のように入力します。

```
CHGJOB CCSID(280)
```

ジョブの属性の変更の詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** のカテゴリー「プログラミング」の中の『*CL および API*』セクションの CHGJOB コマンドを参照してください。

CCSID をロケールに割り当てる

CCSID は、各ロケールがシステムで作成される時点でそのロケールに割り当てられます。ファイルと異なり、ロケールを作成するときに CCSID を指定する必要があります。これは、CRTLOCALE (ロケール作成) コマンドに CCSID パラメーターを指定することによって行います。たとえば、CCSID 273 でロケールを作成するには、次のように入力してください。

```
CRTLOCALE LOCALE('/qsys.lib/testlib.lib/en_us.locale')
SRCFILE('/qsys.lib/qsyslocale.lib/qlocalesrc.file/en_us.mbr')
CCSID(273)
```

実行時 CCSID に関する考慮事項

ここでは、次のものについての実行時 CCSID に関する考慮事項について説明します。

- i5/OS ファイル、およびそれぞれの関連付けられた COBOL ファイル。
- i5/OS ロケールおよびそれぞれの関連付けられた COBOL データ項目。これには、数字編集データ項目、日付データ項目、および時刻データ項目が含まれます。

ロケールとファイルの場合

i5/OS オブジェクト (たとえば、ファイルまたはロケール) に CCSID を割り当てると、ILE COBOL は、実行時に使用された CRTCBLMOD (ILE COBOL モジュールの作成) コマンドの指定された CCSID パラメーターを調べて、変換が必要かどうかを決めます。 CCSID パラメーターに指定できる値は、次のものです。

***JOBRUN**
: 実行時ジョブの CCSID が使用されます。

***JOB** コンパイル・ジョブの CCSID が使用されます。

***HEX** CCSID 65535 が使用されます。

コード化文字セット ID
: 指定した CCSID が使用されます。

上記の CCSID のどれでも 65535 に等しい場合には、変換は行われません。

ロケール・オブジェクトを作成するときに、それに CCSID を割り当てることができます。たとえば、前述の『CCSID をロケールに割り当てる』で作成されるロケール・オブジェクトは 273 という CCSID で作成されます。プログラムをコンパイルするときに、CCSID を割り当てることもできます。コンパイル時に指定した CCSID がそのロケール・オブジェクトの作成時に指定された CCSID と異なる場合には、実行時に、コンパイル時に指定された CCSID に変換されます。

日時データ項目と数字編集項目の場合

ロケール・オブジェクトの場合、ILE COBOL のロケールは、数字編集項目および日付と時刻のカテゴリーの日時項目に関連付けられます。以下に、ロケールと日時項目および数字編集項目をどのように関連付けるかについての例を示します。

```
SPECIAL-NAMES.
    LOCALE "EN_US" IN LIBRARY "QSYSLOCALE" IS usa. 1
:
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DATE-WITH-LOCALE FORMAT DATE SIZE 10 LOCALE USA. 2
01 DATE-NO-LOCALE FORMAT DATE "@Y-%m-%d"    VALUE "1997-08-09". 3
01 NUMERIC-EDITED-WITH-LOCALE  PIC +$9(6).99 SIZE 15  LOCALE USA.  4
01 NUMERIC-EDITED-NO-LOCALE    PIC +9(6).99   VALUE "+123456.78".
:
PROCEDURE DIVISION.
    MOVE DATE-NO-LOCALE TO DATE-WITH-LOCALE.
    MOVE NUMERIC-EDITED-NO-LOCALE TO NUMERIC-EDITED-WITH-LOCALE.
    DISPLAY "date-with-locale = " date-with-locale.
    DISPLAY "numeric-edited-with-locale = "
             numeric-edited-with-locale.
    STOP RUN.
```

プログラムの出力は、次のようになります。

```
date-with-locale = 08/09/97
numeric-edited-with-locale =    $123,456.78
```

上記の例で、行 1 は、ロケール簡略名 usa を定義し、そのロケール簡略名 usa をライブラリー QSYSLOCALE の EN_US と関連付けます。この行はロケール・オブジェクトを定義しますが、コンパイル時に存在する必要はありません。ただし、ロケール・オブジェクトは実行時に存在する必要があります。ロケール・オブジェクトの作成の詳細については 221 ページの『i5/OS でのロケールの作成』または 22 ページの『CCSID をロケールに割り当てる』を参照してください。

行 2 は、行 1 に定義されたロケール簡略名を日付データ項目 DATE-WITH-LOCALE と関連付けます。行 4 は、行 1 に定義されたロケール簡略名を数字編集データ項目 NUMERIC-EDITED-WITH-LOCALE と関連付けます。

実行時に、DATE-NO-LOCALE 内のデータが DATE-WITH-LOCALE に移動されると、行 1 (EN_US) に定義されたロケール・オブジェクトの CCSID はコンパイル時に指定された CCSID と比較されます。 2 つの CCSID が異なるものであると、DATE-NO-LOCALE (行 3) に定義されたデータは、コンパイル時 CCSID に変換され、MOVE ステートメントの結果として生じる定様式データは新しい CCSID に基づくものになります。

ILE COBOL のステートメントのほとんどは、データが CCSID 37 (すなわち、1 バイト文字 EBCDIC) になっているものとします。しかし、ステートメントのなかには、1 つまたは複数の CCSID をサポートするものもあります。

- ロケールと関連付けられた受け取り側をもつ MOVE ステートメントは、送り出すデータをコンパイル時 CCSID に変換します。
- ロケールと関連付けられた送信側をもつ MOVE ステートメント、すなわち、暗黙の移動を伴うステートメントは、次の場合に、送信側を CCSID 37 に変換します。
  - 数字編集項目を編集解除する場合

- 日時項目を編集解除する場合
- 日時比較、非数値比較、または数値比較で生じる比較条件。

ILE ソース・デバッガーで異なる複数の CCSID を処理する

ILE ソース・デバッガーで異なる複数の CCSID を処理する方法については 171 ページの『ILE ソース・デバッガーの各国語サポート』を参照してください。

第 3 章 ソース・プログラムのモジュール・オブジェクトへのコンパイル

WebSphere Development Studio Client for i5/OS を使用してください。 これが推奨される方法であり、ソースのコンパイルに関する資料は、その製品のオンライン・ヘルプにあります。

クライアント・ツール入門については、『クライアント製品のアプリケーション開発ツールの使用』を参照してください。

ILE COBOL コンパイラーは実行可能プログラム・オブジェクトを作成しません。 1 つまたは複数のモジュール・オブジェクトを作成して、それらをさまざまな組み合わせでバインドし、プログラム・オブジェクトと呼ばれる 1 つまたは複数の実行可能単位を作成することができます。実行可能プログラム・オブジェクトの作成の詳細については 89 ページの『第 4 章 プログラム・オブジェクトの作成』を参照してください。

この章では次のことについて説明します。

- モジュール・オブジェクトの作成方法
- CRTCBLMOD コマンドおよびそのパラメーター
- PROCESS ステートメントを使用してコンパイラー・オプションを指定する方法
- ILE COBOL コンパイラーが作成する出力を理解する方法

モジュール・オブジェクトの定義

モジュール・オブジェクトは、ILE COBOL コンパイラーを含め、すべての ILE コンパイラーの出力です。それらはタイプ *MODULE のシステム・オブジェクトです。 ILE COBOL の場合、永続的に作成されたどのモジュール・オブジェクトの名前も、最外部の ILE COBOL ソース・プログラムの CRTCBLMOD コマンドまたは PROGRAM-ID 段落によって決定されます。ソース・メンバー内の各コンパイル単位は、それぞれ別個のモジュール・オブジェクトを作成します。あるモジュール・オブジェクトの最外部の ILE COBOL プログラムは、バインド・プロシージャー呼び出しによって、異なるモジュール・オブジェクトの別の ILE COBOL プログラムにより呼び出すことができます。このプログラムはまた、モジュール・オブジェクトがプログラム・オブジェクトにバインドされた後に、動的プログラム呼び出しを使用して呼び出すこともできます。バインド・プロシージャー呼び出しと動的プログラム呼び出しの説明については 239 ページの『ILE COBOL プログラムの呼び出し』を参照してください。

モジュール・オブジェクトは単独で実行することはできません。モジュール・オブジェクトは、まずバインドしてプログラム・オブジェクトにしなければなりません。 1 つまたは複数のモジュール・オブジェクトをバインドして、1 つのプログラム・オブジェクト (タイプ *PGM) またはサービス・プログラム (タイプ *SRVPGM) を作成することができます。このようにモジュール・オブジェクトを結合できることによって、次のことが可能になります。

- 部分ごとのコードを再使用することにより、一般に小さいプログラムにすることができる。
- 複数のプログラム間でコードを共用するので、共用の部分の更新時には、全体のプログラムの他の部分にエラーが入り込む可能性をなくすことができる。
- 言語を混在させて使用できるので、実行が必要なタスクを最善の方法で行う言語を選択できる。

1 つのモジュール・オブジェクトは、1 つまたは複数の ILE プロシージャーで構成できます。

COBOL モジュールの作成 (CRTCBLMOD) コマンドにより、ILE COBOL ソース・ステートメントから 1 つまたは複数のオブジェクトを作成できます。これらのモジュール・オブジェクトは、明示的に削除または置換されるまで、指定されたライブラリーの中に保管されます。その後モジュール・オブジェクトは、プログラム作成 (CRTPGM) コマンドを使用してバインドして実行可能プログラム・オブジェクトにするか、またはサービス・プログラム作成 (CRTSRVPGM) コマンドを使用してバインドしてサービス・プログラムにすることができます。モジュール・オブジェクトは、プログラム・オブジェクトまたはサービス・プログラムが作成された後もライブラリーに存在しています。 1 つまたは複数のモジュール・オブジェクトからプログラム・オブジェクトを作成する詳細については 92 ページの『プログラム作成 (CRTPGM) コマンドの使用』を参照してください。 1 つまたは複数のモジュール・オブジェクトからサービス・プログラムを作成する詳細については 121 ページの『第 5 章 サービス・プログラムの作成』を参照してください。

バインド COBOL PGM の作成 (CRTBNDCBL) コマンドにより、ILE COBOL ソース・ステートメントから 1 ステップでプログラム・オブジェクトを作成できます。CRTBNDCBL コマンドはモジュール・オブジェクトも作成します。しかし、これらのモジュール・オブジェクトは一時的に作成されるもので、再使用はできません。CRTBNDCBL コマンドでのプログラム・オブジェクトの作成が完了すると、一時的なモジュール・オブジェクトは削除されます。

1 ステップのプログラム・オブジェクト作成の詳細については 95 ページの『バインド COBOL PGM の作成 (CRTBNDCBL) コマンドの使用』を参照してください。

モジュール・オブジェクトの作成時には、次のものを含めることもできます。

- デバッグ・データ

  デバッグ・データとは、ILE ソース・デバッガーを使用してプログラム・オブジェクトをデバッグするために必要なデータです。このデータは、CRTCBLMOD または CRTBNDCBL コマンドの DBGVIEW パラメーターで指定されるオプションに基づいて生成されます。
- プログラム入り口プロシージャー (PEP)

  プログラム入り口プロシージャーとは、動的プログラム呼び出しにおいてプログラム・オブジェクトの入り口点となるコンパイラー生成コードのことです。プログラム・オブジェクトが動的プログラム呼び出しを使用して呼び出されると、その PEP に制御が渡されます。これは OPM プログラムの入り口点に提供されるコードと類似しています。 PEP は、そのプログラム・オブジェクトが動的プログラム呼び出しを使用して呼び出されると最初に実行されることになっているモ

ジュール・オブジェクト内の ILE プロシージャーを識別します。モジュール・オブジェクトが ILE COBOL コンパイラーによって作成された時点で、PEP が生成されます。この PEP は、コンパイル単位に含まれる最外部の ILE COBOL プログラムを呼び出します。

複数部分からなるモジュール・オブジェクトをバインドしてプログラム・オブジェクトを作成する場合、どのモジュール・オブジェクトに作成されるプログラム・オブジェクトの PEP が入るかを指定しなければなりません。これは CRTPGM コマンドの ENTMOD パラメーターでモジュール・オブジェクトを識別することによって実行します。これによりモジュール・オブジェクトの PEP はプログラム・オブジェクトの PEP になります。他のすべてのモジュール・オブジェクトの PEP は、プログラム・オブジェクトから論理的に削除されます。

- ユーザー入り口プロシージャー (UEP)

  ユーザー入り口プロシージャーとは、モジュール・オブジェクトが ILE COBOL コンパイラーによって作成されるときに、コンパイル単位に含まれる最外部の ILE COBOL プログラムです。動的プログラム呼び出しにおいて、UEP は PEP から制御を獲得する ILE プロシージャーです。複数の異なるモジュール・オブジェクトに含まれる ILE プロシージャーの間での静的プロシージャー呼び出しでは、UEP に直接制御が与えられます。

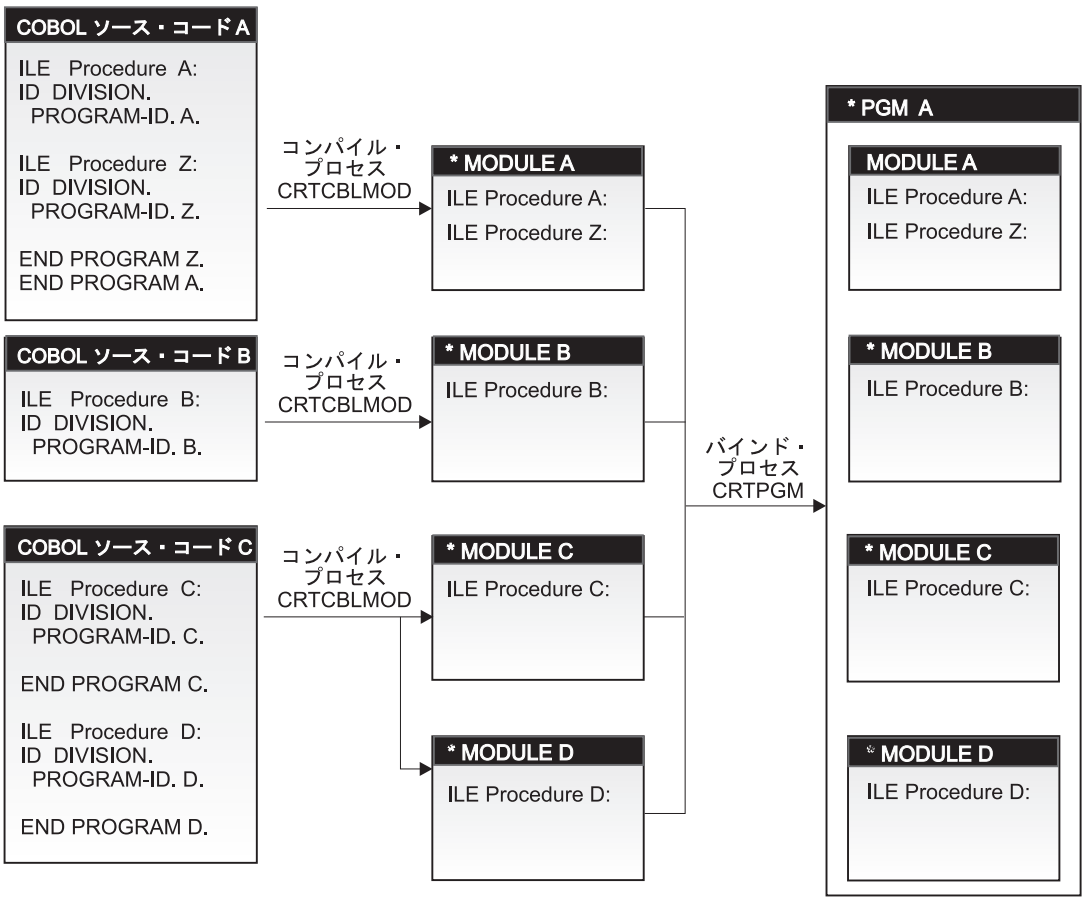


図 8. CRTCBLMOD コマンドを使用してモジュール・オブジェクトを作成する

27 ページの図 8 では、*PGM A は、プログラム・オブジェクトの入り口点を含むモジュール・オブジェクトとして *MODULE A が指定されて作成されます。*MODULE A の PEP は ILE プロシージャー A を呼び出します。 *MODULE A の PEP は *PGM A の PEP にもなるため、*PGM A の PEP は ILE プロシージャー A を呼び出します。 *PGM A の UEP も ILE プロシージャー A になります。 *MODULE B、*MODULE C、および *MODULE D にも PEP はありますが、それらは *PGM A では無視されます。さらに、ILE プロシージャー Z は ILE プロシージャー A からしか呼び出せません。 ILE プロシージャー B、C、および D はそれぞれ独立したモジュール・オブジェクトに入っており、ILE プロシージャー Z は *MODULE A の最外部の COBOL プログラムではないため、ILE プロシージャー Z は ILE プロシージャー B、C、および D からは見えません。 ILE プロシージャー A、B、C および D は相互に呼び出すことができます。それらすべてのプロシージャーは、再帰プロシージャーでないため、再帰を行うことはできません。

ILE COBOL ソース・プログラム内の各宣言型プロシージャーは、それぞれ独立した ILE プロシージャーを生成します。

ネストした COBOL プログラムは、それぞれ独立した ILE プロシージャーを生成します。

モジュール・オブジェクトには、モジュール・エクスポートとモジュール・インポートを関連付けることができます。

モジュール・エクスポートとは、バインディング処理によって他の ILE オブジェクトが使用できるプロシージャーまたはデータ項目の名前です。モジュール・エクスポートは、その名前とそれに関連したタイプによって、それがプロシージャーであるかデータであるかが識別されます。モジュール・エクスポートの有効範囲は、プログラム・オブジェクトと活動化グループとの 2 通りがあります。プログラム・オブジェクトにエクスポートされるすべての名前が活動化グループにエクスポートされるわけではありません。 ILE COBOL コンパイラーは、次の COBOL プログラミング言語構成要素ごとにモジュール・エクスポートを作成します。

- コンパイル単位で最外部の ILE COBOL プログラムに対応しているプロシージャー名。
- コンパイル単位で最外部の ILE COBOL プログラムに対応している取り消しプロシージャー名。
- EXTERNAL ファイルまたは EXTERNAL データの弱エクスポート。

モジュール・インポートとは、参照元モジュール・オブジェクトでは定義されていないプロシージャーやデータ項目の名前を使用したり参照したりすることです。モジュール・インポートは、その名前と関連タイプによって、プロシージャーまたはデータのどちらであるかを識別します。 ILE COBOL コンパイラーは、次の COBOL プログラミング言語構成要素ごとにモジュール・インポートを作成します。

- 静的プロシージャー呼び出しを使用して呼び出される ILE COBOL プログラムに対応しているプロシージャー名。
- 静的プロシージャー呼び出しを使用して呼び出される ILE COBOL プログラムに対応している取り消しプロシージャー名。

- EXTERNAL ファイルまたは EXTERNAL データの弱インポート。
- SET *プロシージャー・ポインター項目* TO ENTRY *プロシージャー名* ステートメントによって設定される、ILE COBOL プログラムに対応しているプロシージャー名 (*プロシージャー名* の名前は ILE プロシージャーであると解釈されます)。

ターゲット・プロシージャーが参照元モジュール・オブジェクトで定義されていない場合に、モジュール・インポートが生成されます。データ項目への弱インポートは、ILE COBOL プログラム内でデータ項目が参照される場合に生成されます。

COBOL モジュールの作成 (CRTCBLMOD) コマンドの使用

ILE COBOL ソース・ステートメントをコンパイルして 1 つまたは複数のモジュール・オブジェクトにするには、CRTCBLMOD コマンドを使用します。このコマンドは、ソース・メンバーにある ILE COBOL ステートメントに基づいてモジュール・オブジェクトを作成する ILE COBOL コンパイラーを開始します。 CRTCBLMOD コマンドは対話式で、またはバッチ・モードで、あるいは i5/OS の CL プログラムから使用できます。

注: CRTCBLMOD コマンドでモジュール・オブジェクトを作成するには、そのコマンドを使用する権限がなければなりません。

プログラム中で形式 2 の COPY ステートメントを使用して外部記述ファイルにアクセスすると、外部記述ファイルに関する情報がオペレーティング・システムからコンパイル済みプログラムに提供されます。

ILE COBOL コンパイラーが停止した場合には、以下のメッセージ LNC9001 が表示されます。

`Compile failed. module-name not created.`

メッセージ・モニター (MONMSG) CL コマンドを使用することにより、この例外をモニターできる制御言語プログラムを使用することができます。

プロンプト画面からの CRTCBLMOD コマンドの使用

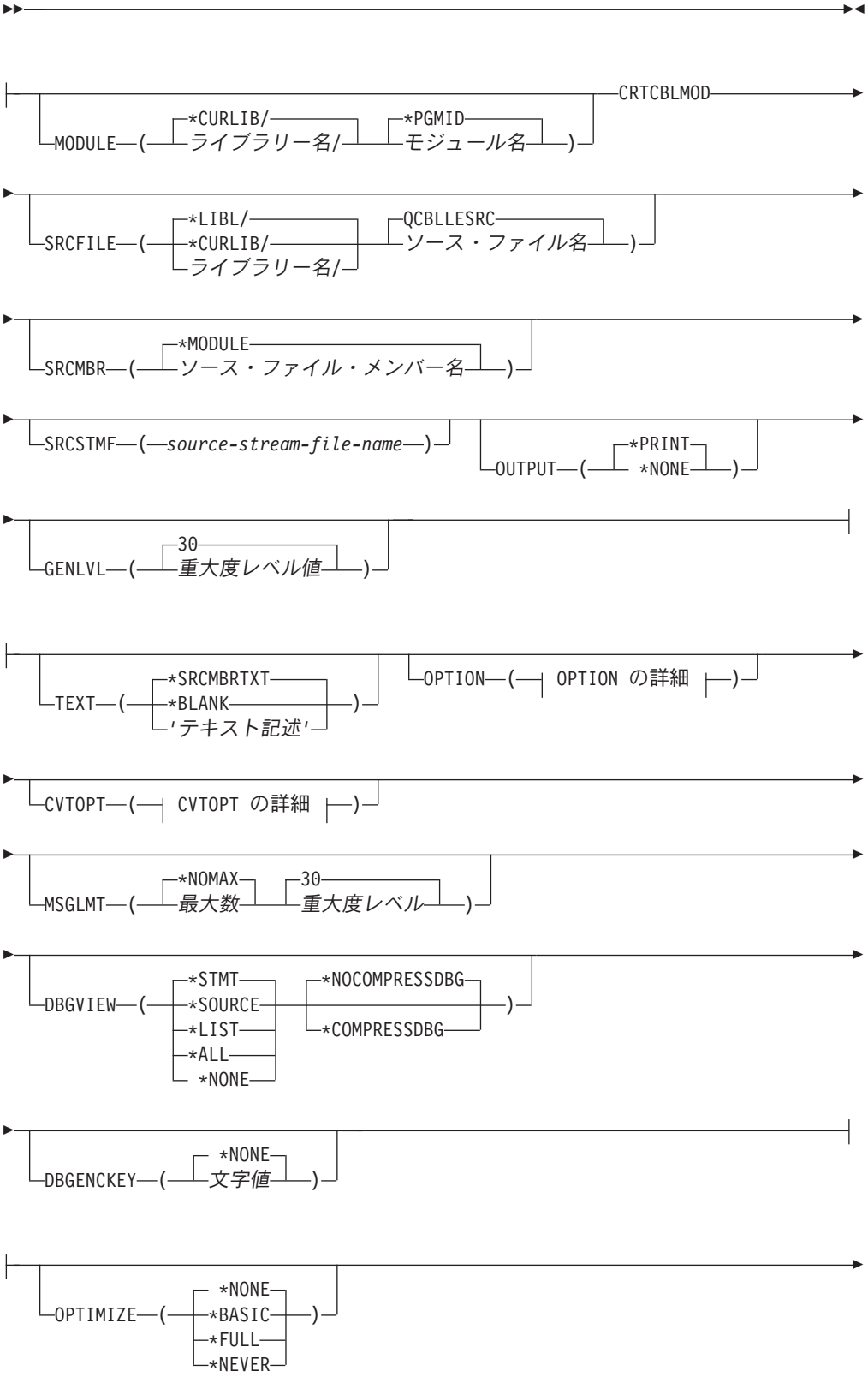
CRTCBLMOD コマンドは、プロンプト画面を使用して入力することができます。この方法でコマンド・パラメーターを入力するには、CRTCBLMOD と入力して、F4 キーを押します。

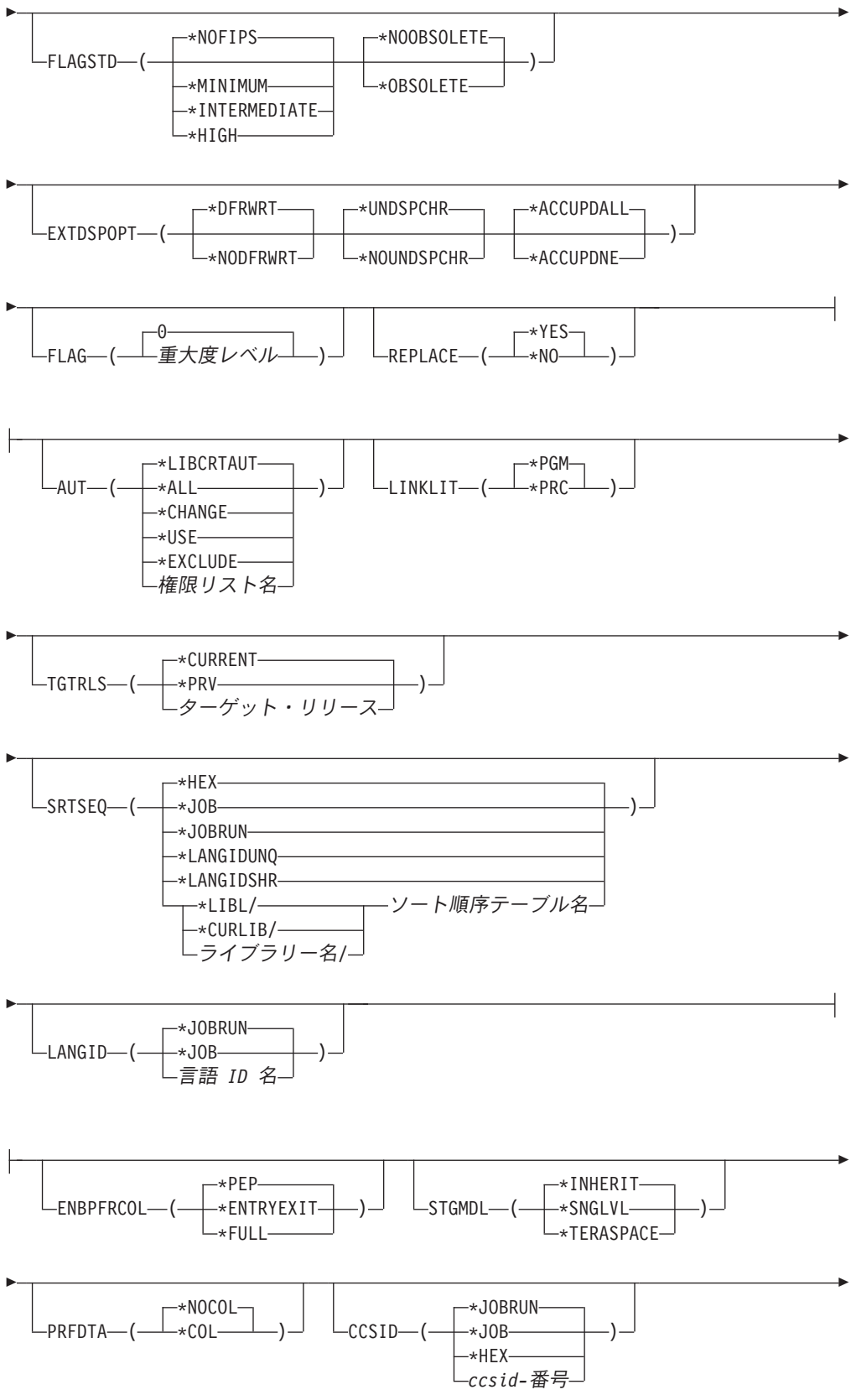
この画面の各パラメーターにはデフォルト値が示されます。任意の項目を上書きして、別の値またはオプションを設定してください。パラメーター値の設定がよくわからない場合には、フィールドの先頭の位置に疑問符 (?) を入力してから実行キーまたは F4 (プロンプト) を押すと、詳細な情報が表示されます。疑問符の後に 1 つのブランクが必要です。プロンプト画面を要求する前に一部のパラメーターを入力した場合、そのパラメーターにはその値が表示されます。

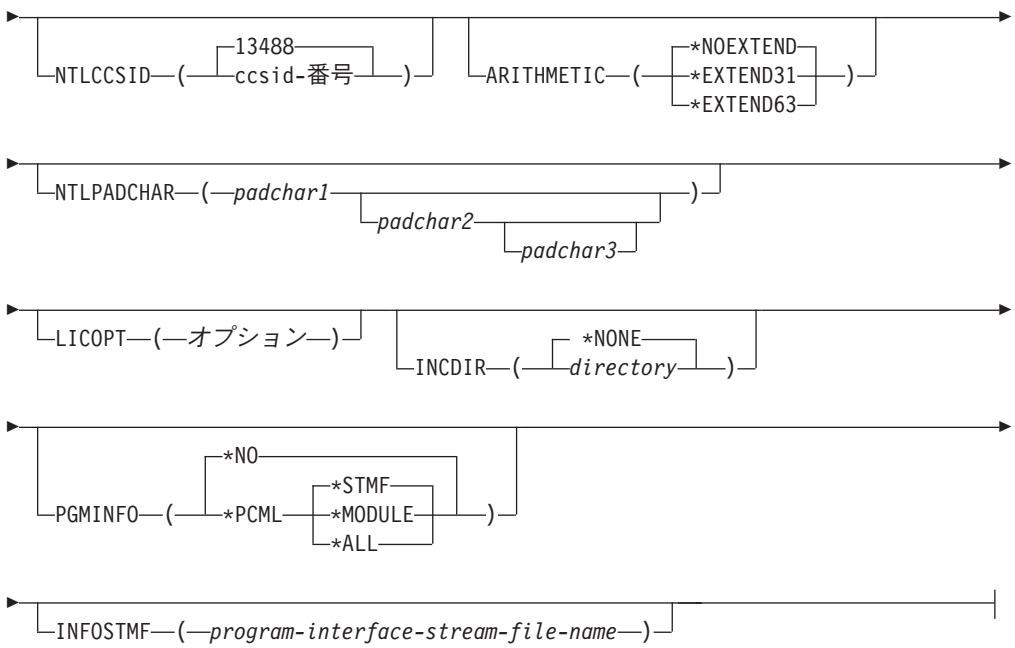
CRTCBLMOD コマンドのパラメーターの説明については 33 ページの『CRTCBLMOD コマンドのパラメーター』を参照してください。

CRTCBLMOD コマンドの構文

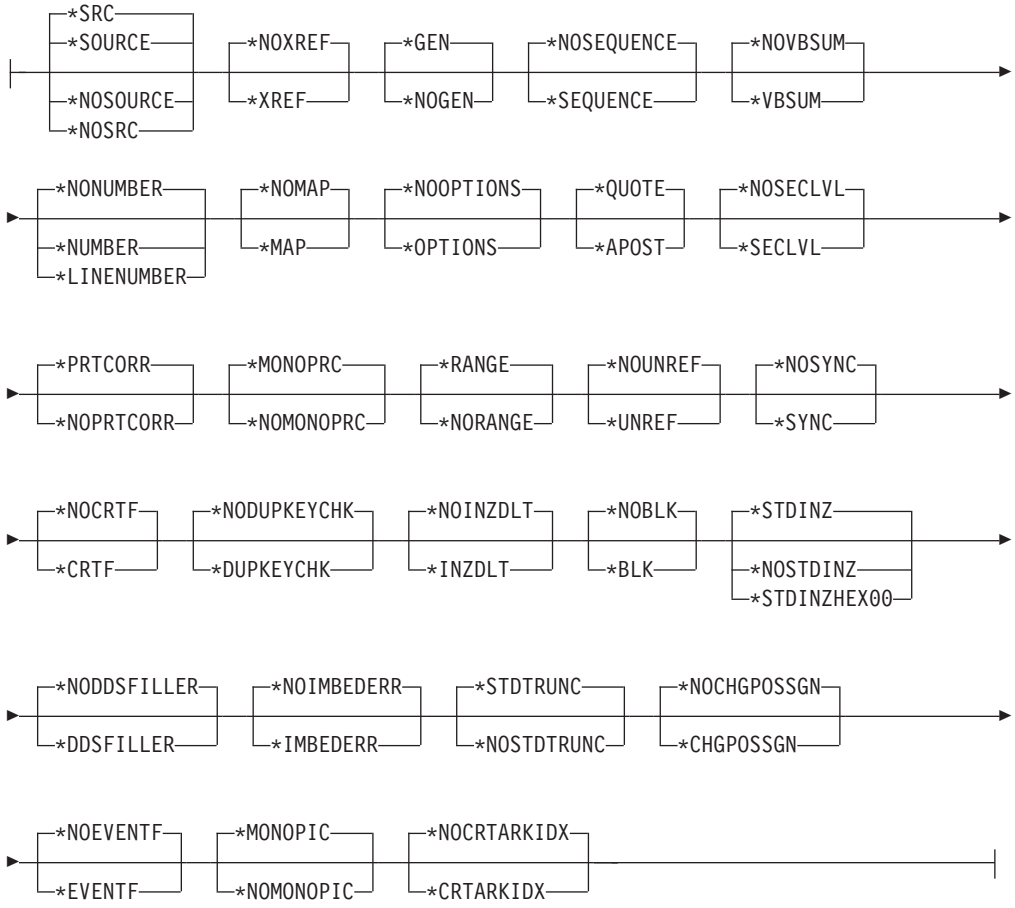
CRTCBLMOD コマンド - 形式



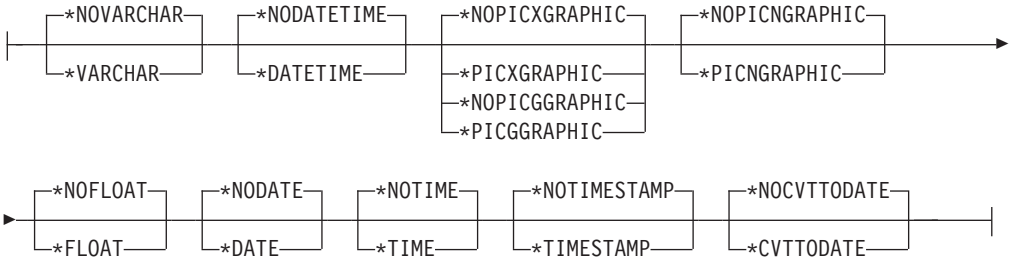

FLAGSTD
*NOFIPS
*MINIMUM
*INTERMEDIATE
*HIGH
*NOOBSOLETE
*OBSOLETE
EXTDSPOPT
*DFRWRT
*NODFRWRT
*UNDSPCHR
*NOUNDSPCHR
*ACCUPDALL
*ACCUPDNE
FLAG
0
重大度レベル
REPLACE
*YES
*NO
AUT
*LIBCRTAUT
*ALL
*CHANGE
*USE
*EXCLUDE
権限リスト名
LINKLIT
*PGM
*PRC
TGTRLS
*CURRENT
*PRV
ターゲット・リリース
SRTSEQ
*HEX
*JOB
*JOBRUN
*LANGIDUNQ
*LANGIDSHR
*LIBL/
*CURLIB/
ライブラリー名/
ソート順序テーブル名
LANGID
*JOBRUN
*JOB
言語 ID 名
ENBPFRCOL
*PEP
*ENTRYEXIT
*FULL
STGMDL
*INHERIT
*SNGLVL
*TERASPACE
PRFDTA
*NOCOL
*COL
CCSID
*JOBRUN
*JOB
*HEX
ccsid-番号


NTLCCSID
(
13488
ccsid-番号
)
ARITHMETIC
(
*NOEXTEND
*EXTEND31
*EXTEND63
)
NTLPADCHAR
(
padchar1
padchar2
padchar3
)
LICOPT
(
オプション
)
INCDIR
(
*NONE
directory
)
PGMINFO
(
*NO
*PCML
*STMF
*MODULE
*ALL
)
INFOSTMF
(
program-interface-stream-file-name
)

OPTION の詳細:


*SRC
*SOURCE
*NOSOURCE
*NOSRC
*NOXREF
*XREF
*GEN
*NOGEN
*NOSEQUENCE
*SEQUENCE
*NOVBSUM
*VBSUM
*NONUMBER
*NUMBER
*LINENUMBER
*NOMAP
*MAP
*NOOPTIONS
*OPTIONS
*QUOTE
*APOST
*NOSECLVL
*SECLVL
*PRTCORR
*NOPRTCORR
*MONOPRC
*NOMONOPRC
*RANGE
*NORANGE
*NOUNREF
*UNREF
*NOSYNC
*SYNC
*NOCRTF
*CRTF
*NODUPKEYCHK
*DUPKEYCHK
*NOINZDLT
*INZDLT
*NOBLK
*BLK
*STDINZ
*NOSTDINZ
*STDINZHEX00
*NODDSFILLER
*DDSFILLER
*NOIMBEDERR
*IMBEDERR
*STDTRUNC
*NOSTDTRUNC
*NOCHGPOSSGN
*CHGPOSSGN
*NOEVENTF
*EVENTF
*MONOPIC
*NOMONOPIC
*NOCRTARKIDX
*CRTARKIDX

CVTOPT の詳細:



CRTCBLMOD コマンドのパラメーター

ここでは、CRTCBLMOD コマンドのパラメーターについて説明します。パラメーターとオプションは、プロンプト画面に表示される順序で説明します。

デフォルト値がまず最初に表示され、見分けるために下線が付けられています。

CRTCBLMOD コマンドに指定するすべてのオブジェクト名は、 i5/OS 命名規則に
従ったものでなければなりません。名前が基本名の場合、長さ 10 文字の英数字で
構成できます。名前の先頭は英字でなければなりません。名前が引用符付き名の場
合、長さは 8 文字で、二重引用符で囲まれたものになります。

CRTCBLMOD コマンドの OPTION パラメーターを使用して、またはソース・プログラム内から PROCESS ステートメントを使用して、さまざまなコンパイラー・オプションを指定できます。 PROCESS ステートメントで指定したオプションは、CRTCBLMOD コマンドのそれに対応するオプションを指定変更します。

MODULE パラメーター:

作成するモジュール・オブジェクトのモジュール名とライブラリー名を指定します。モジュール名とライブラリー名は i5/OS の命名規則に適合していなければなりません。 指定できる値は次のとおりです。

***PGMID**

モジュールの名前は、コンパイル単位の最外部の ILE COBOL ソース・プログラムにある PROGRAM-ID 段落から取られます。

モジュール名

コンパイルされた ILE COBOL モジュールを識別するための名前を入力します。このパラメーターのモジュール名を指定する場合で、一連のソース・プログラム をコンパイルする場合 (単一ソース・ファイル・メンバーに複数のコンパイル単位が入っている場合)、一連のモジュールのうちの先頭のものがこの名前を使用します。他のすべてのモジュールは、コンパイル単位の対応する最外部の ILE COBOL ソース・プログラムにある PROGRAM-ID 段落に指定された名前を使用します。

指定できるライブラリー値は次のとおりです。

***CURLIB**

作成されるモジュール・オブジェクトは、現行ライブラリーに保管されます。特定のライブラリーを現行ライブラリーとして割り当てていない場合、QGPL が使用されます。

ライブラリー名
作成されるモジュール・オブジェクトが保管されるライブラリーの名前を入力します。

SRCFILE パラメーター:
コンパイルする ILE COBOL ソース・コードの入っているソース・ファイルおよびライブラリーの名前を指定します。このソース・ファイルのレコード長は 92 でなければなりません。指定できる値は次のとおりです。

QCBLLESRC
ソース・ファイル QCBLLESRC の中に、コンパイルする ILE COBOL ソース・コードが含まれていることを指定します。

ソース・ファイル名
コンパイルする ILE COBOL ソース・コードの入っているソース・ファイルの名前を入力します。

指定できるライブラリー値は次のとおりです。

***LIBL**
ライブラリー・リストを検索して、ソース・ファイルが含まれているライブラリーを見つけます。

***CURLIB**
現行ライブラリーが使用されます。特定のライブラリーを現行ライブラリーとして割り当てていない場合、QGPL が使用されます。

ライブラリー名
ソース・ファイルが含まれているライブラリーの名前を入力します。

SRCMBR パラメーター:
コンパイルする ILE COBOL ソース・コードの入っているメンバーの名前を指定します。このパラメーターを指定できるのは、SRCFILE パラメーターで参照されるソース・ファイルがデータベース・ファイルの場合だけです。指定できる値は次のとおりです。

***MODULE**
MODULE パラメーターで指定されるモジュール名と同じ名前のソース・ファイル・メンバーを使用します。

MODULE パラメーターのモジュール名が指定されていない場合、データベース・ソース・ファイルの最初のメンバーが使用されます。

ソース・ファイル・メンバー名
ILE COBOL ソース・コードの含まれているメンバーの名前を入力します。

SRCSTMF パラメーター:
コンパイルする ILE COBOL ソース・コードを含むストリーム・ファイルのパス名を指定します。パス名は、絶対修飾名と相対修飾名のどちらを指定することもできます。絶対パス名は「/」で始まり、相対パス名は「/」以外の文字で始まります。絶対修飾の場合、パス名は完全なものです。相対修飾の場合、パス名にジョブの現行作業ディレクトリーを付け加えることによって、パス名が完全なものになります。 SRCMBR と SRCFILE のパラメーターは、SRCSTMF パラメーターと一緒に指定することはできません。

OUTPUT パラメーター:

コンパイラー・リストを生成するかどうかを指定します。指定できる値は次のとおりです。

***PRINT**

コンパイラー・リストが生成されます。メンバーをコンパイルすると、出力ファイルの名前はメンバーと同じ名前になります。PGM パラメーターに *PGMID を指定して、ストリーム・ファイルをコンパイルすると、出力ファイルの名前は COBOLPGM00 になります。そうでない場合は、出力ファイルの名前はプログラムと同じ名前になります。

***NONE**

コンパイラー・リストを生成しません。

GENLVL パラメーター:

モジュール・オブジェクトを作成するかどうかを決める重大度レベルを指定します。重大度レベルは、コンパイル時に作成されるメッセージの重大度レベルに対応します。このパラメーターは、ソース・ファイル・メンバーの各コンパイル単位ごとに個別に適用されます。ソース・ファイル・メンバー内の他のコンパイル単位は、それ以前のコンパイル単位が失敗してもコンパイルされます。

指定できる値は次のとおりです。

30 重大度レベルが 30 以上のエラーが発生する場合、モジュール・オブジェクトは作成されません。

重大度レベル

0 ～ 30 の数値を 1 桁または 2 桁で指定します。これはモジュール・オブジェクトを作成するかどうかを決めるために使用する重大度レベルです。この重大度レベル以上のエラーが発生する場合、モジュール・オブジェクトは作成されません。

TEXT パラメーター:

モジュールとその機能について簡潔に説明するテキストを入力することができます。

***SRCMBRTXT**

ILE COBOL ソース・コードを含むデータベース・ファイル・メンバーの記述と同じテキストを使用して、モジュール・オブジェクトを記述します。ソースが装置またはインライン・ファイルから来ている場合、*SRCMBRTXT の指定の効果は *BLANK の指定と同じです。

***BLANK**

テキストは指定しません。

テキスト記述

モジュールとその機能について簡潔に説明するテキストを入力します。そのテキストは、長さが SBCS 文字で最大 50 文字までであり、単一引用符で囲む必要があります。単一引用符は 50 文字の文字ストリングの一部とは見なされません。

OPTION パラメーター:

ILE COBOL ソース・コードのコンパイル時に使用するオプションを指定します。

ILE COBOL ソース・プログラムの PROCESS ステートメントで指定したオプションは、OPTION パラメーターの、対応するオプションを指定変更します。

OPTION パラメーターに指定できる値は次のとおりです。

***SOURCE または *SRC**
コンパイラーは ILE COBOL ソース・プログラムおよびすべてのコンパイル時エラー・メッセージで構成される、ソース・リストを作成します。

***NOSOURCE または *NOSRC**
コンパイラーはリストのソース部分を作成しません。ソース・リストが必要でない場合、コンパイルの時間を短くするため、このオプションを使用するようにしてください。

***NOXREF**
コンパイラーは、ILE COBOL ソース・プログラムの相互参照リストを作成しません。

***XREF**
コンパイラーはソース・プログラムの相互参照リストを作成します。

***GEN**
コンパイラーは、ILE COBOL ソースのコンパイル後にモジュール・オブジェクトを作成します。

***NOGEN**
コンパイラーは、ILE COBOL ソース・プログラムのコンパイル後にモジュール・オブジェクトを作成しません。エラー・メッセージまたはリストだけが必要な場合、このオプションを指定してください。

***NOSEQUENCE**
参照番号のシーケンス・エラーはチェックされません。

***SEQUENCE**
参照番号のシーケンス・エラーをチェックします。 *LINENUMBER オプションを指定した場合、シーケンス・エラーは発生しません。

***NOVBSUM**
verb の使用カウントを印刷しません。

***VBSUM**
verb の使用カウントを印刷します。

***NONUMBER**
ソース・ファイルのシーケンス番号を参照番号として使用します。

***NUMBER**
ユーザー指定のシーケンス番号 (桁 1 ～ 6) を参照番号として使用します。

***LINENUMBER**
コンパイラーによって作成されるシーケンス番号が参照番号として使用されます。このオプションは、ILE COBOL プログラム・ソース・コードと、COPY ステートメントによって導かれたソース・コードとを結合して、1 つの連続した番号の順序列にします。 FIPS (連邦情報処理標準) フラグを指定する場合は、このオプションを使用してください。

***NOMAP**
コンパイラーはデータ部マップのリストを作成しません。

***MAP**
コンパイラーはデータ部マップのリストを作成します。

***NOOPTIONS**
このコンパイルでは、有効なオプションのリストを作成しません。

***OPTIONS**
このコンパイルで、有効なオプションのリストを作成します。

***QUOTE**
非数字リテラル、16 進数リテラル、およびブール・リテラルに、区切り引用符 (") を使用することを指定します。このオプションはまた、形象定数 QUOTE の値が引用符の EBCDIC 値になるように指定します。

***APOST**
非数字リテラル、16 進数リテラル、およびブール・リテラルに、区切りアポストロフィ (') を使用することを指定します。このオプションはまた、形象定数 QUOTE の値がアポストロフィの EBCDIC 値になるように指定します。

***NOSECLVL**
このコンパイルでは、2 次レベルのメッセージ・テキストのリストを作成しません。

***SECLVL**
このコンパイル・リストでは、2 次レベル・メッセージ・テキストは、1 次レベル・エラー・テキストとともに、コンパイラー・リストのメッセージ・セクションに示されます。

***PRTCORR**
CORRESPONDING 句を使用した結果として含められた基本項目を示すコメント行がコンパイラー・リストに挿入されます。

***NOPRTCORR**
CORRESPONDING 句が使用されても、コメント行をコンパイラー・リストに挿入しません。

***MONOPRC**
PROGRAM-ID 段落、CALL、CANCEL、または SET ENTRY ステートメント、および END PROGRAM ヘッダーで見つかったプログラム名 (リテラルまたはワード) は、すべて大文字に変換され、プログラム名形式の規則を強制的に適用します。

***NOMONOPRC**
PROGRAM-ID 段落、CALL、CANCEL、または SET ENTRY ステートメント、および END PROGRAM ヘッダーで見つかったプログラム名 (リテラルまたはワード) は、大文字には変換されず、プログラム名形式の規則の強制適用はされません。このオプションを指定すると、標準の COBOL では許可されていない特殊文字を CALL ターゲットで使用できるようになります。

***RANGE**
実行時に、添え字が正しい範囲内にあるかどうかが検査されますが、指標の範囲は検査されません。参照変更およびコンパイラー生成サブストリング操作も検査されます。

日付/時刻項目の形式が正しいか、また、それらが有効な日付、時刻、またはタイム・スタンプを表しているか確認するために、日付/時刻項目の内容が検査されます。

***NORANGE**
範囲は実行時に検査されません。

注: *RANGE オプションは添え字範囲を検査するコードを生成します。たとえば、エレメントが 20 個しかない配列で、エレメント 21 にアクセスしないようにします。

*NORANGE オプションでは、添え字範囲を検査するコードを生成しません。結果として、*NORANGE オプションのほうが、より速く実行されるコードが作成されます。

***NOUNREF**
非参照データ項目は、コンパイルされたモジュールに含められません。これによって使用ストレージの量が少なくなり、もっと大きなプログラムがコンパイルできるようになります。 *NOUNREF オプションを選択した場合は、デバッグ中に非参照データ項目を見たり、それに代入したりすることはできません。非参照データ項目は OPTION (*XREF) を指定して作成された相互参照表には表示されます。

***UNREF**
非参照データ項目はコンパイルされたモジュールに含められます。

***NOSYNC**
SYNCHRONIZED 文節は構文検査だけを行います。

***SYNC**
SYNCHRONIZED 文節はコンパイラーによってコンパイルされます。
SYNCHRONIZED 文節により、データ項目の位置は、右端 (最低位) がストレージの自然な境界になるように位置合わせされます。ストレージの自然な境界は、保管されるデータの長さとタイプに応じて、ストレージの中で次に最も近い 4 バイト、8 バイト、または 16 バイトのどれかの境界になります。この位置合わせを行うために、同期された項目に隣接する余分なストレージが取られます。 SYNCHRONIZED と記述される各基本データ項目は、そのデータ・ストレージの割り当てに対応するストレージの自然な境界に位置合わせされます。

***NOCRTF**
OPEN 操作時に利用できないディスク・ファイルは動的には作成されません。

***CRTF**
OPEN 操作時に利用できないディスク・ファイルを動的に作成します。

注: 動的に作成されるファイルの最大レコード長は 32,766 です。 *CRTF オプションを指定しても、索引ファイルは動的には作成されません。

***NODUPKEYCHK**
INDEXED ファイルの基本キーの重複を検査しません。

***DUPKEYCHK**
INDEXED ファイルの基本キーの重複を検査します。

***NOINZDLT**
順次アクセスの相対ファイルを OUTPUT 用にオープンした場合、CLOSE 操作中に削除されたレコードで初期化されることはありません。レコード境界は、OPEN OUTPUT 時に書き込まれるレコード数によって決まります。それ以降の OPEN 操作はレコード境界までのアクセスしか認められません。

***INZDLT**
順次アクセスの相対ファイルを OUTPUT 用にオープンした場合、CLOSE 操作中に削除されたレコードで初期化されます。ファイル内の活動レコードには影響はありません。それ以降の OPEN 操作において、レコード境界はファイル・サイズとして定義されます。

***NOBLK**
コンパイラーは、START ステートメントがない SEQUENTIAL アクセス・ファイルのブロック化だけを許します。これが指定される場合、BLOCK CONTAINS 文節は無視されます (テープ・ファイルを除く)。

***BLK**
*BLK が使用されると、コンパイラーは DYNAMIC アクセス・ファイルと SEQUENTIAL アクセス・ファイルをブロック化します。ブロック化は、出力操作用にオープンされる RELATIVE ファイルには許可されません。BLOCK CONTAINS 文節を指定した場合、ブロックされるレコードの数はこの文節によって決まります。指定しなかった場合、ブロックされるレコードの数は、オペレーティング・システムによって決定されます。

***STDINZ**
VALUE 文節がない項目の場合は、コンパイラーはデータ項目をデフォルト値に初期設定します。エリアを占有する最初のレベル 01 またはレベル 77 データ項目のストレージの各エリアに割り当てられる値。

***NOSTDINZ**
VALUE 文節がない項目の場合、コンパイラーはデータ項目をシステム・デフォルトに初期設定することはしません。

***STDINZHEX00**
VALUE 文節がない項目の場合は、コンパイラーはデータ項目を 16 進数の 0 に初期設定します。

***NODDSFILLER**
COPY DDS ステートメントで一致するフィールドが見つからない場合、フィールド記述は生成されません。

***DDSFILLER**
COPY DDS ステートメントで一致するフィールドが見つからない場合、単一文字の FILLER フィールド記述である ″07 FILLER PIC X″ が常に作成されます。

***NOIMBEDERR**
エラー・メッセージはコンパイラー・リストのソース・リスト・セクションに含められません。エラー・メッセージはコンパイラー・リストのエラー・メッセージ・セクションだけに表示されます。

***IMBEDERR**
コンパイラー・リストのソース・リスト・セクションに 1 次レベル・エラー・メッセージが含められ、エラーが発生した行のすぐ後に表示されます。エラー・メッセージは、コンパイラー・リストのエラー・メッセージ・セクションにも表示されます。

***STDTRUNC**
このオプションは USAGE BINARY データだけに適用されます。
*STDTRUNC が選択された場合、USAGE BINARY データは BINARY 受信フィールドの PICTURE 文節にある桁数に切り捨てられます。

***NOSTDTRUNC**
このオプションは USAGE BINARY データだけに適用されます。
*NOSTDTRUNC が選択された場合、BINARY 受信フィールドはハーフワード、フルワード、またはダブルワードの境界でのみ切り捨てられます。BINARY 送信フィールドは、ハーフワード、フルワード、またはダブルワードとしても処理されます。こうして、フィールドの 2 進数の内容全体が有効になります。さらに、DISPLAY ステートメントでは、BINARY フィールドの内容全部が、切り捨てなしで変換されます。

注: *NOSTDTRUNC は VALUE 文節に影響を及ぼしません。

***NOCHGPOSSGN**
ゾーンおよびパックの数値データのデフォルトの正の符号として、16 進数の F が使用されます。 16 進数 F は、オペレーティング・システムのシステム・デフォルトです。

***CHGPOSSGN**
ゾーンおよびパックの数値データのデフォルトの正の符号として、16 進数の C が使用されます。これは、VALUE 文節の結果だけでなく、MOVE、ADD、SUBTRACT、MULTIPLY、DIVIDE、COMPUTE、および INITIALIZE ステートメントのすべての結果に適用されます。

***NOEVENTF**
CoOperative Development Environment/400 (クライアント製品) で使用するイベント・ファイルを作成しません。クライアント製品はこのファイルを使用することにより、クライアント製品エディターと統合されたエラー・フィードバックを提供します。通常、イベント・ファイルが作成されるのは、モジュールまたはプログラムをクライアント製品内で作成した場合です。

***EVENTF**
クライアント製品で使用するイベント・ファイルを作成します。イベント・ファイルは、作成されたモジュールまたはプログラム・オブジェクトが保管されるライブラリー中のファイル EVFEVENT のメンバーとして作成されます。 EVFEVENT ファイルがない場合は、自動的に作成されます。イベント・ファイル・メンバー名は、作成されるオブジェクトの名前と同じです。

クライアント製品はこのファイルを使用することにより、クライアント製品エディターと統合されたエラー・フィードバックを提供します。通常、イベント・ファイルが作成されるのは、モジュールまたはプログラムをクライアント製品内で作成した場合です。

***MONOPIC**
PICTURE 文字ストリング内の英字は、すべて英大文字に変換されます。

***NOMONOPIC**
PICTURE 文字ストリングで使用される通貨記号は、大文字小文字の区別があります。すなわち、PICTURE 記号 A、B、E、G、N、P、S、V、X、Z、CR、および DB の英大文字に相当する英小文字は、PICTURE 文字ストリングの英大文字表記と同等です。しかし、その他のすべての英小文字は、それらに相当する英大文字表記と同等ではありません。

***NOCRTARKIDX**
永続的な代替レコード・キー (ARK) 索引が検出されないときに、一時的な代替レコード・キー索引は作成されません。

***CRTARKIDX**
永続的な代替レコード・キー (ARK) 索引が検出されないときは、一時的な代替レコード・キー索引が作成されます。

CVTOPT パラメーター:
外部記述ファイルから COPY DDS によってプログラムに渡された日付、時刻、およびタイム・スタンプのフィールド・タイプ、DBCS グラフィック・フィールド・タイプ、可変長フィールド・タイプ、および浮動小数点フィールド・タイプをコンパイラーが処理する方法を指定します。指定できる値は次のとおりです。

***NOVARCHAR**
可変長フィールドは FILLER フィールドとして宣言されます。

***VARCHAR**
可変長フィールドはグループ項目として宣言され、ILE COBOL ソース・プログラムからアクセスできます。

***NODATETIME**
日付、時刻、およびタイム・スタンプのデータ項目は FILLER フィールドとして宣言されます。

***DATETIME**
日付、時刻、およびタイム・スタンプの DDS データ項目に、それぞれの DDS 名に基づく COBOL データ項目名が与えられます。 COBOL データ項目のカテゴリーは、CVTOPT パラメーター値 *DATE、*TIME、または *TIMESTAMP のいずれかが指定された場合を除き、英数字です。この場合、COBOL データ項目のカテゴリーは、それぞれ、日付、時刻、またはタイム・スタンプです。

***NOPICXGRAPHIC**
DBCS グラフィック・データ項目は FILLER フィールドとして宣言されます。

***PICXGRAPHIC**
固定長の DBCS グラフィック・データ項目は固定長の英数字フィールドとして宣言され、ILE COBOL ソース・プログラムからアクセスすることができます。

*VARCHAR オプションも使用している場合、可変長 DBCS グラフィック・データ項目は固定長グループ項目として宣言され、ILE COBOL ソース・プログラムからアクセスすることができます。

***PICGGRAPHIC**
固定長の DBCS グラフィック・データ項目は固定長 G フィールドとして宣言され、ILE COBOL ソース・プログラムからアクセスすることができます。

*VARCHAR オプションも使用している場合、可変長 DBCS グラフィック・データ項目は固定長グループ項目 (数値フィールドの後に G タイプ・フィールドが続いたもの) として宣言され、ILE COBOL ソース・プログラムからアクセスできます。

***NOPICGGRAPHIC**
DBCS グラフィック・データ項目は FILLER フィールドとして宣言されます。 *NOPICGGRAPHIC は、リストには *NOPICXGRAPHIC として印刷されます。

***PICNGRAPHIC**
National CCSID コンパイラー・オプションまたは NTLCCSID PROCESS オプションで指定された CCSID に関連付けられた固定長グラフィック・データ項目は、固定長 N フィールドとして宣言され、ILE COBOL のソース・プログラムにアクセス可能です。

*VARCHAR オプションが併用される場合、National CCSID コンパイラー・オプションまたは NTLCCSID PROCESS オプションで指定された CCSID を伴う可変長のグラフィック・データ項目は、固定長のグループ項目 (N タイプ・フィールドが後に続く数字フィールドでできている) として宣言され、ILE COBOLのソース・プログラムにアクセス可能です。

***NOPICNGRAPHIC**
グラフィック・フィールドの処理は、PICXGRAPHIC/NOPICXGRAPHIC および PICGGRAPHIC/NOPICGGRAPHIC オプションに指定された値によって異なります。

***NOFLOAT**
浮動小数点データ項目は、2 進数の USAGE をもつ FILLER フィールドとして宣言されます。

***FLOAT**
浮動小数点データ項目は、それらの DDS 名、および COMP-1 (単精度) または COMP-2 (倍精度) の USAGE によってプログラムに取り込まれます。フィールドは ILE COBOL ソース・プログラムからアクセス可能になります。

***NODATE**
日付データ項目は、英数字カテゴリーの COBOL データ項目として宣言されます。たとえば、次のようになります。

```
06 FILLER PIC X(10).
```

COBOL データ項目名は、*NODATETIME/*DATETIME CVTOPT パラメーターによって決められます。

***DATE**
DDS 日付データ項目は、日付カテゴリーの COBOL データ項目として宣言されます。たとえば、次のようになります。

```
06 FILLER FORMAT DATE '@Y-%m-%d'.
```

COBOL データ項目名は、*NODATETIME/*DATETIME CVTOPT パラメーターによって決められます。

***NOTIME**
DDS 時刻データ項目は、英数字カテゴリーの COBOL データ項目として宣言されます。たとえば、次のようになります。

```
06 FILLER PIC X(8).
```

COBOL データ項目名は、*NODATETIME/*DATETIME CVTOPT パラメーターによって決められます。

***TIME**
DDS 時刻データ項目は、時刻カテゴリーの COBOL データ項目として宣言されます。たとえば、次のようになります。

```
06 FILLER FORMAT TIME '%H:%M:%S'.
```

COBOL データ項目名は、*NODATETIME/*DATETIME CVTOPT パラメーターによって決められます。

***NOTIMESTAMP**
DDS タイム・スタンプ・データ項目は、英数字カテゴリーの COBOL データ項目として宣言されます。たとえば、次のようになります。

```
06 FILLER PIC X(26).
```

COBOL データ項目名は、*NODATETIME/*DATETIME CVTOPT パラメーターによって決められます。

***TIMESTAMP**
DDS タイム・スタンプ・データ項目は、タイム・スタンプ・カテゴリーの COBOL データ項目として宣言されます。たとえば、次のようになります。

```
06 FILLER FORMAT TIMESTAMP.
```

COBOL データ項目名は、*NODATETIME/*DATETIME CVTOPT パラメーターによって決められます。

***NOCVTTODATE**
DATFMT キーワードをもつ DDS データ項目 (DDS 日付データ項目を除く) は、それぞれの元の DDS タイプに基づいて ILE COBOL で宣言されます。

***CVTTODATE**
DATFMT キーワードをもつ DDS データ項目 (DDS 日付データ項目を除く) は、日付データ・タイプとして ILE COBOL で宣言されます。
*CVTTODATE オプションの使用法については 59 ページの『日付、時刻、およびタイム・スタンプのデータ・タイプの指定』を参照してください。

MSGLMT パラメーター:
各コンパイル単位ごとに、発生する可能性がある特定のエラー重大度レベルのメ

ッセージの最大数を指定し、それに達した場合はコンパイルを停止するようにします。あるコンパイル単位がその最大数に達すると、すぐに、すべてのソース・メンバーのコンパイルが停止します。

たとえば、メッセージの最大数に 3 を、エラー重大度レベルに 20 を指定すると、重大度レベルが 20 以上のエラーが 3 つ以上発生した場合、コンパイルは停止します。指定したエラー重大度レベル以上のメッセージがない場合、見つかったエラーの数に関係なくコンパイルは続行されます。

メッセージ数
メッセージの最大数を指定します。指定できる値は次のとおりです。

***NOMAX**
見つかったエラーの数に関係なく、コンパイルは通常どおり完了するまで続行されます。

最大数
指定したエラー重大度レベル以上のメッセージの最大数を指定します。この数を超えるとコンパイルは停止します。有効な範囲は 0 ～ 9999 です。

メッセージ重大度制限
コンパイルを停止するかどうかを決めるために使用するエラー重大度レベルを指定します。指定できる値は次のとおりです。

30 重大度レベル 30 以上のエラーの数が指定されたメッセージの最大数を超えると、コンパイルは停止します。

エラー重大度レベル
1 桁または 2 桁の数値として 0～30 を入力します。これはコンパイルを停止するかどうかを決めるために使用するエラー重大度レベルです。この重大度レベル以上のエラーの数が指定されたメッセージの最大数を超えると、コンパイルは停止します。

DBGVIEW パラメーター:
ソース・プログラムのビューまたは生成されたリスト・ビューのどちらをコンパイル済みモジュールのデバッグに使用できるのか、およびデバッグ・リストを圧縮するのかしないのかを制御するオプションを指定します。

debug-view
デバッグに使用可能なビューを指定します。指定できる値は次のとおりです。

***STMT**
コンパイルされたモジュールは、シンボル名およびステートメント番号を使用してデバッグされます。

***SOURCE**
コンパイル済みモジュールをデバッグするために、1 次ソース・メンバーおよび COPY ステートメントによって含められたコピー・ソース・メンバーのソース・ビューを使用できるようにします。これらのビューを使用できるのは、1 次ソース・メンバーおよびコピー・ソース・メンバーがローカル・データベース・ソース・ファイルからのものである場合だけです。コンパイルおよびデバッグ時には、メンバーの変更および削除はしないでください。

***LIST**
コンパイル済みモジュールをデバッグするために、COPY および REPLACE ステートメントの処理後にソース・コードを表示する、リスト・ビューを使用できるようにします。このオプションを指定すると、コンパイル済みモジュールのサイズは大きくなりますが、コンパイル済みモジュールの実行時のパフォーマンスには影響を与えません。

リスト・ビューには、対応するコンパイラー・オプションが要求された場合、相互参照リスト、データ部マップ、および verb 使用カウントが含められます。たとえば、OPTION(*XREF) が指定された場合には、相互参照リストが含められます。

リスト・ビューは、1 次ソース・メンバーまたはコピー・ソース・メンバーがどこからのものであるかに関係なく生成できます。リスト・ビューは、コンパイルに続くソース・メンバーの変更または削除によって影響を受けることはありません。

***ALL**
*STMT、*SOURCE、および *LIST を結合して指定することと同じです。

***NONE**
コンパイル済みモジュールはデバッグできません。この場合、コンパイルされたプログラムのサイズは小さくなりますが、その実行時パフォーマンスには影響を与えません。このオプションを指定した場合、定様式ダンプをとることはできません。

compress-listing-view
デバッグ・ビューで *LIST または *ALL が指定されている場合に、リスト・ビューを圧縮するのかしないのかを指定します。指定できる値は次のとおりです。

***NOCOMPRESSDBG**
リスト・ビューは圧縮されません。

***COMPRESSDBG**
デバッグ・ビューで *LIST または *ALL が指定されていると、リスト・ビューは圧縮されます。このオプションを使用することで、すべてではありませんが一部の大規模な COBOL プログラムは、デバッグ・ビュー・オプション *LIST を使用してコンパイルすることができるようになります。

DBGENCKEY パラメーター:
デバッグ・ビューに埋め込まれるプログラム・ソースを暗号化するために使用する暗号鍵を指定します。

***NONE**
暗号鍵は指定しません。

文字値
モジュール・オブジェクトに保管される、デバッグ・ビューに埋め込まれるプログラム・ソースを暗号化するために使用する鍵を指定します。鍵の長さは、1 から 16 バイトにすることができます。鍵の長さが 1 から 15 バイ

トの場合には、ブランクを埋め込んで 16 バイトにしてから暗号化に使用されます。長さがゼロの鍵を指定すると、*NONE を指定した場合と同じになります。

すべてのコード・ページで不変ではない文字が鍵に含まれている場合には、ターゲット・システムでソース・システムと同じコード・ページが確実に使用されるようにユーザーが配慮する必要があります。そうしないと、鍵が一致せずに、暗号化解除が失敗することがあります。異なるコード・ページを使用するシステムで暗号鍵を入力する必要がある場合には、すべての EBCDIC コード・ページで不変な文字で鍵を構成することをお勧めします。

OPTIMIZE パラメーター:

モジュールの最適化レベルを指定します。指定できる値は次のとおりです。

***NONE**

コンパイル済みモジュールで最適化は実行されません。このオプションを使用すると、コンパイル時間は最小になります。このオプションにより、デバッグ中に変数を表示させたり変更したりできるようになります。

***BASIC**

コンパイル済みモジュールについて、ローカル・ブロック・レベルだけの一部の最適化が実行されます。このオプションにより、ユーザーは、デバッグ中に変数を表示できるようになりますが、変更することはできません。

***FULL**

コンパイルされたモジュールに対して、完全な最適化 (グローバル・レベルで) が実施されます。この最適化によってコンパイル時間は長くなりますが、最も効率的なコードを生成することができます。このオプションにより、ユーザーは、デバッグ中に変数を表示できるようになりますが、変更することはできません。表示された変数の値は、その現行値でない場合があります。一部の変数は表示できない場合があります。

***NEVER**

このオプションは、後から CHGMOD コマンドを使用してモジュールの最適化レベルを変更できないという点を除いて、*NONE と同じ効果を持ちます。このオプションは、最適化情報を一切生成しません。そのため、ずっと大きいプログラムでも、システムのストレージ限界を超過せずにコンパイルできます。

注: *NEVER オプション値を選択していなければ、ユーザーは、ソース・プログラムを再コンパイルすることなく、 CHGMOD、CHGPGM、または CHGSRVPGM コマンドを使用して、モジュール・オブジェクトの最適化レベルを変更できます。

FLAGSTD パラメーター:

FIPS フラグのオプションを指定します。 (*LINENUMBER オプションを選択して、FIPS メッセージで使用される参照番号が固有なものになるようにしてください。) 指定できる値は次のとおりです。

***NOFIPS**

ILE COBOL ソース・プログラムに FIPS フラグはありません。

***MINIMUM**

最小サブセット以上に対して、FIPS フラグを付けます。

***INTERMEDIATE**
中間サブセット以上に対して、FIPS フラグを付けます。

***HIGH**
高いサブセットに対して、FIPS フラグを付けます。

***NOOBSOLETE**
古くなった言語エレメントにフラグは付けません。

***OBSOLETE**
古くなった言語エレメントにフラグを付けます。

EXTDSPOPT パラメーター:
ワークステーション入出力の拡張 ACCEPT および拡張 DISPLAY ステートメントに使用するオプションを指定します。

***DFRWRT**
拡張 ACCEPT ステートメントが見つかるまで、またはバッファーがいっぱいになるまで、拡張 DISPLAY ステートメントはバッファーに保持されます。

バッファーの内容は、拡張 ACCEPT ステートメントが見つかるかバッファーがいっぱいになると、画面に表示されます。

***NODFRWRT**
それぞれの拡張 DISPLAY ステートメントは、それが見つかると実行されます。

***UNDSPCHR**
表示および非表示の文字は、拡張 ACCEPT および拡張 DISPLAY ステートメントによって処理されます。

***NOUNDSPCHR**
表示可能文字だけが拡張 ACCEPT および拡張 DISPLAY ステートメントによって処理されます。

このオプションは、リモート 3174 および 3274 制御装置に接続されているディスプレイ装置には必ず使用しなければなりませんが、ローカル・ワークステーションに使用することもできます。このオプションを使用する場合、データの内容は表示可能文字だけでなければなりません。データに 16 進数の 20 より小さい値が含まれる場合、結果は、予期しない表示形式のエラーから重大エラーまでの範囲で予測不能になります。

***ACCUPDALL**
UPDATE 句の存在にかかわらず、データのすべてのタイプが拡張 ACCEPT ステートメントに事前表示されます。

***ACCUPDNE**
数字編集データだけが、UPDATE 句を含まない拡張 ACCEPT ステートメントに事前表示されます。

FLAG パラメーター:
コンパイラー・リストに表示されるメッセージの最小重大度レベルを指定します。指定できる値は次のとおりです。

0 すべてのメッセージがコンパイラー・リストに表示されます。

重大度レベル
コンパイラー・リストに表示させる最小のメッセージ重大度レベルを指定する、1 桁または 2 桁の数を入力します。この指定された値またはそれより高い重大度レベルをもつメッセージは、コンパイル・リストに表示されます。

REPLACE パラメーター:
明示的または暗黙に指定されたライブラリーに同じ名前のモジュールがすでに存在するとき、新しいモジュールを作成するかどうかを指定します。指定できる値は次のとおりです。

***YES**
新しいモジュールが作成され、明示的または暗黙に指定されたライブラリーにある同じ名前の既存のモジュールと置き換えられます。明示的または暗黙のうちに指定されたライブラリーにある同じ名前の既存のモジュールは、ライブラリー QRPLOBJ に移されます。

***NO**
同じ名前のモジュールが、明示的または暗黙のうちに指定されたライブラリーにすでに存在する場合、新しいモジュールは作成されません。既存のモジュールは置換されず、メッセージが表示され、コンパイルは停止します。

AUT パラメーター:
モジュール・オブジェクトに対する特定の権限がないユーザー、権限リスト中にないユーザー、またはモジュール・オブジェクトに対する特定の権限がないユーザーのグループに付与される権限を指定します。モジュール・オブジェクトを GRTOBJAUT (オブジェクト権限付与) または RVKOBJAUT (オブジェクト権限取り消し) コマンドを使用すると、モジュール・オブジェクトの作成後に、すべてのユーザーまたは特定のユーザーの権限を変更することができます。

指定できる値は次のとおりです。

***LIBCRTAUT**
オブジェクトの共通認可はターゲット・ライブラリー (作成されたモジュール・オブジェクトを入れるライブラリー) の CRTAUT キーワードから取られます。この値は、モジュール・オブジェクトが作成されるときに決まります。モジュール・オブジェクトを作成した後でライブラリーの CRTAUT の値を変更する場合、新しい値が既存のオブジェクトに影響を与えることはありません。

***ALL**
所有者に限定されているものまたは権限リスト管理権限によって制御されているものを除き、モジュール・オブジェクトのすべての操作の権限を提供します。モジュール・オブジェクトの存在を制御したり、それのためのセキュリティーを指定したり、それを変更したり、その基本機能を実行したりすることはできますが、その所有権を転送することはできません。

***CHANGE**
すべてのデータ権限およびモジュール・オブジェクトに対するすべての操作を実行する権限を提供します。ただし、所有者に限定されたものまたはオブジェクト権限およびオブジェクト管理権限によって制御されているものを除きます。ユーザーはオブジェクトを変更してその基本機能を実行することができます。

***USE**
オブジェクト操作権と読み取り権限を提供します。すなわち、モジュール・オブジェクトの基本操作の権限です。ユーザーはオブジェクトの基本操作は実行できますが、オブジェクトの変更はできません。

***EXCLUDE**
ユーザーはモジュール・オブジェクトにアクセスできません。

権限リスト名
モジュールが追加される、ユーザーおよび権限の権限リストの名前。モジュール・オブジェクトはこの権限リストによって保護され、モジュール・オブジェクトの共通認可は *AUTL に設定されます。権限リストは、CRTCBLMOD コマンドが出されるときにシステムに存在していなければなりません。固有の権限リストを作成するには、権限リストの作成 (CRTAUTL) コマンドを使用します。

LINKLIT パラメーター:
外部 CALL/CANCEL 'リテラル' ターゲットおよび SET ENTRY ターゲットのリンケージ・タイプを指定します。 SPECIAL-NAMES 段落に次の文を指定することによって、特定の外部 CALL/CANCEL 'リテラル' ターゲットおよび SET ENTRY ターゲット・リスト用にこのオプションを指定変更できます。

```
LINKAGE TYPE IS implementer-name FOR target-list.
```

LINKLIT に指定できる値は次のとおりです。

***PGM**
CALL/CANCEL または SET ENTRY のターゲットはプログラム・オブジェクトです。

***PRC**
CALL/CANCEL または SET ENTRY のターゲットは ILE プロシージャーです。

TGTRLS パラメーター:
作成しているオブジェクトを使用する予定のオペレーティング・システムのリリースを指定します。下記の *CURRENT および *PRV 値の例では、ターゲット・リリース の値を指定する場合に、VxRxMx の形式で指定しています。ここで、Vx はバージョン、Rx はリリース、Mx はモディフィケーション・レベルを示します。たとえば、V2R3M0 はバージョン 2、リリース 3、モディフィケーション・レベル 0 になります。

このパラメーターに有効な値は、リリースごとに変わります。 指定できる値は次のとおりです。

***CURRENT**
このオブジェクトは、システムで現在実行中のオペレーティング・システムのリリースで使用します。例えば、システムで V2R3M5 が稼働中の場合、*CURRENT は V2R3M5 が導入されているシステムでオブジェクトを使用することを意味します。また、このオブジェクトは、オペレーティング・システムのそれより新しいリリースがインストールされているシステムでも使用できます。

注: V2R3M5 がシステムで稼働していて、V2R3M0 がインストールされているシステムでオブジェクトを使用する場合は、TGTRLS(*CURRENT) ではなく TGTRLS(V2R3M0) を指定してください。

***PRV**
このオブジェクトは、前のリリースのモディフィケーション・レベル 0 のオペレーティング・システムで使用されます。たとえば、システムで V2R3M5 が稼働している場合、*PRV は、V2R2M0 がインストールされているシステムでオブジェクトを使用する予定であるという意味になります。また、このオブジェクトは、オペレーティング・システムのそれより新しいリリースがインストールされているシステムでも使用できます。

ターゲット・リリース
VxRxMx の形式でリリースを指定します。このオブジェクトは、指定されたリリースまたはそれより新しいリリースのオペレーティング・システムがインストールされているシステムで使用できます。

有効な値は、現行バージョン、リリース、およびモディフィケーション・レベルによって異なり、新しいリリースごとに変わります。 指定するターゲット・リリース がこのコマンドでサポートしている最古のリリース・レベルよりも古い場合、サポートされる最古のリリースを示したエラー・メッセージが表示されます。

注: 現行バージョンのコマンドが、前のリリースのコマンドで利用できなかったオプションをサポートすることがあります。コマンドが前のリリースで使用される予定のオブジェクトを作成するために使用される場合、コンパイラーはそのリリースに適合するように処理され、サポートされていなかったオプションはどれも認識されません。コンパイラーは、処理できないオプションについて必ずしも警告を出すとは限りません。

SRTSEQ パラメーター:
ALPHABET 文節の英字名に NLSSORT が関連付けられている場合に使用されるソート・シーケンスを指定します。 SRTSEQ パラメーターは LANGID パラメーターと関連して使用され、モジュールが使用するシステム定義またはユーザー定義のソート・シーケンス・テーブルを判別します。指定できる値は次のとおりです。

***HEX**
ソート・シーケンス・テーブルは使用されず、文字の 16 進数の値を使用してソート・シーケンスが判別されます。

***JOB**
コンパイル・ジョブのソート・シーケンスを使用して、コンパイル時にソート・シーケンスが解決され、モジュールに関連付けられます。コンパイル・ジョブのソート・シーケンス・テーブルが、コンパイル時にシステムに存在していなければなりません。実行時に、実行時ジョブの CCSID がコンパイル時ジョブの CCSID と異なる場合、コンパイル時にロードされるソート・シーケンス・テーブルは、実行時ジョブの CCSID と一致するように変換されます。

***JOBRUN**
モジュールのソート・シーケンスは、実行時に解決され、モジュールに関連

付けられます。この値を指定すると、モジュールをいったんコンパイルした後で、実行時に異なるソート・シーケンスで使用できます。

***LANGIDUNQ**
使用するソート・シーケンス・テーブルに、コード・ページの各文字ごとに固有の重みを含めるように指定します。使用するソート・シーケンス・テーブルは、LANGID パラメーターで指定された言語と関連した固有の重みを持つテーブルになります。

***LANGIDSHR**
使用するソート・シーケンス・テーブルで、コード・ページ内の複数の文字に同じ重みを含めることができることを指定します。使用するソート・シーケンス・テーブルは、LANGID パラメーターで指定された言語と関連した共有の重みを持つテーブルになります。

テーブル名
使用されるソート・シーケンス・テーブルの名前を入力します。テーブルには、特定のコード・ページのすべての文字の重みが含まれます。コード・ポイントで定義される文字に重みが関連付けられます。ソート・シーケンス・テーブル名を使用するとき、オブジェクトの入っているライブラリーを指定することができます。ライブラリーに有効な値は次のとおりです。

***LIBL**
ライブラリー・リストを検索して、ソート・シーケンス・テーブルが含まれているライブラリーを見つけます。

***CURLIB**
現行ライブラリーが使用されます。特定のライブラリーを現行ライブラリーとして割り当てていない場合、QGPL が使用されます。

ライブラリー名
ソート・シーケンス・テーブルが含まれているライブラリーの名前を入力します。

LANGID パラメーター:
ソート・シーケンスと関連して使用される言語 ID を指定します。 LANGID パラメーターは、有効な SRTSEQ 値が *LANGIDUNQ または *LANGIDSHR のときにだけ使用されます。指定できる値は次のとおりです。

***JOBRUN**
モジュールの言語 ID は実行時に解決されます。この値を指定すると、モジュールをいったんコンパイルした後で、実行時に異なる言語 ID で使用できます。

***JOB**
モジュールの言語 ID は、コンパイル時にコンパイル・ジョブの言語 ID を使用することによって解決されます。

言語 ID 名
有効な 3 文字の言語 ID を入力します。

ENBPFRCOL パラメーター:
モジュールまたはプログラムのパフォーマンス測定コードを生成するかどうかを指定します。収集されたデータは、システム・パフォーマンス測定ツールで使用して、アプリケーションのパフォーマンスのプロファイルを作成することができ

ます。コンパイルされたモジュールやプログラムにパフォーマンス測定コードを追加すると、オブジェクトが若干大きくなり、パフォーマンスに影響が出ることがあります。

***PEP**
プログラム入力プロシージャーの入り口および出口でのみ、パフォーマンス統計情報が収集されます。 アプリケーションに関する全体的なパフォーマンス情報を収集したい場合には、この値を選択してください。このサポートは、以前 TPST ツールで提供されていたものと同じです。これはデフォルトです。

***ENTRYEXIT**
プログラムのすべてのプロシージャーの入り口と出口についてのパフォーマンス統計が収集されます。これには、プログラムの PEP ルーチンが含まれます。

この選択項目が役立つのは、全ルーチンに関する情報を把握したい場合です。アプリケーションで呼び出されるプログラムがすべて、*PEP、*ENTRYEXIT、または *FULL オプションを使用してコンパイルされていることが分かっている場合は、このオプションを使用してください。そうでない場合、アプリケーションが、パフォーマンス測定のできない他のプログラムを呼び出しているなら、パフォーマンス測定ツールは、リソースをその呼び出し側のアプリケーションが使用しているものとします。このため、リソースが現在どこで実際に使用されているのかを判別することは困難になります。

***FULL**
すべてのプロシージャーの入り口と出口のパフォーマンス統計が収集されます。また、外部プロシージャー呼び出しの前後でも統計が収集されます。

アプリケーションから呼び出されるプログラムが、*PEP、*ENTRYEXIT、または *FULL オプションを使用してコンパイルされていない場合には、このオプションを使用してください。このオプションによりパフォーマンス測定ツールは、アプリケーションが使用するリソースとアプリケーションで呼び出されるプログラム (パフォーマンス測定が可能でない場合であっても) が使用するリソースとを区別します。このオプションは最もコストがかかるものですが、これによって 1 つのアプリケーション内のさまざまなプログラムを選択して分析することができます。

STGMDL パラメーター:
モジュールが使用するストレージのタイプを指定します。

***INHERIT**
モジュールは、継承ストレージ・モデルを使用して作成されます。継承ストレージ・モデル・モジュールは、単一レベル、テラスペース、または継承のストレージ・モデルを使用したプログラムまたはサービス・プログラムにバインドできます。単一レベルおよびテラスペースのストレージ・モデルを使用したプログラムの自動および静的ストレージで使用されるストレージのタイプは、オブジェクトのストレージ・モデルに一致します。継承ストレージ・モデル・オブジェクトは、呼び出し元のストレージ・モデルを継承します。

| ***SNGLVL**
| モジュールは、単一レベル・ストレージ・モデルを使用して作成されます。単一レベル・ストレージ・モデル・モジュールは、単一レベル・ストレージを使用するプログラムおよびサービス・プログラムにのみバインドできます。そのようなプログラムおよびサービス・プログラムは、自動および静的ストレージ用に単一レベル・ストレージを使用します。

| ***TERASPACE**
| モジュールは、テラスペース・ストレージ・モデルを使用して作成されます。テラスペース・ストレージ・モデル・モジュールは、テラスペース・ストレージを使用するプログラムおよびサービス・プログラムにのみバインドできます。そのようなプログラムおよびサービス・プログラムは、自動および静的ストレージ用にテラスペース・ストレージを使用します。

PRFDTA パラメーター:

モジュールのプログラム・プロファイリング・データ属性を指定します。プログラム・プロファイリングは、統計データ (プロファイリング・データ) に基づいてプロシージャー内でプロシージャーおよびコードの順序を変更するのに使用される高度な最適化手法です。プロファイリング・データの収集については 58 ページの『プロファイリング・データの収集』を参照してください。

***NOCOL**

このモジュールは、プロファイリング・データの収集に使用できません。これはデフォルトです。

***COL**

このモジュールは、プロファイリング・データの収集に使用できます。

注: *COL を指定できるのは、モジュールの最適化レベルが *FULL の場合だけです。

CCSID パラメーター:

ファイル内のレコードと、LOCALE に関連付けられたデータが実行時に変換されるコード化文字セット ID (CCSID) を指定します。 組み込み関数で CCSID が指定されていない場合は、NATIONAL-OF および DISPLAY-OF 組み込み関数によってデフォルトの CCSID としても使用されます。また、National データ項目への MOVE 処理に使用されます。処理対象は単一バイトのデータ項目 (英字または英数字等)または DBCS データ項目です。 MOVE ステートメントの詳細は、『ILE COBOL 言語解説書』を参照してください。

***JOBRUN**
プログラムの CCSID は、実行時に解決されます。コンパイル済みプログラムが実行されるときには、現行ジョブの CCSID が使用されます。

***JOB**
コンパイル時の現行ジョブの CCSID が使用されます。

***HEX**
CCSID 65535 が使用されます。これは、フィールド内のデータがビット・データとして扱われ、変換されないことを指示します。

コード化文字セット-id
使用される CCSID を指定します。

NTLCCSID パラメーター:

National 項目で使用するコード化文字セット ID (CCSID) を指定します。

13488

CCSID 13488 は National 項目に使用します。

コード化文字セット-id

指定した CCSID は UCS-2、例えば UTF-16 CCSID 1200 と互換性のある必要があります。

ARITHMETIC パラメーター:

数値データの演算モードを指定します。 指定できる値は次のとおりです。

***NOEXTEND**

このオプションは、数値データのデフォルトの演算モードを指定します。固定小数点演算式の中間結果を 30 桁までにすることができますが、数値リテラルの最大長は 18 桁までです。

***EXTEND31**

固定小数点演算の中間結果の精度を増やすには、このオプションを使用します。固定小数点演算式の中間結果を 31 桁までにすることができ、数値リテラルの最大長を 31 桁にすることができます。

***EXTEND63**

固定小数点演算の中間結果の精度を増やすには、このオプションを使用します。固定小数点演算式の中間結果を 63 桁までにすることができ、数値リテラルの最大長を 63 桁にすることができます。

NTLPADCHAR パラメーター:

このオプションは、MOVE ステートメントで、国別データ項目が 1 バイト文字、2 バイト文字、または国別文字を受け入れる場合の埋め込み文字を指定します。埋め込み文字を以下の順序で指定してください。

1. 1 バイト文字から国別文字への移動

   送り出し項目は、英字または英数字などの 1 バイト項目です。国別 16 進文字を指定してください。デフォルトは `NX"0020"` です。

2. 2 バイト文字から国別文字への移動

   送り出し項目は 2 バイト項目です。国別 16 進文字を指定してください。デフォルトは `NX"3000"` です。

3. 国別文字から国別文字への移動

   送り出し項目は国別項目です。国別 16 進文字を指定してください。デフォルトは `NX"3000"` です。

LICOPT パラメーター:

1 つまたは複数のライセンス内部コード・コンパイル時オプションを指定します。このパラメーターによって、各コンパイル時オプションを選択することができます。このパラメーターは、選択した各タイプのコンパイラー・オプションの利点と欠点を理解している上級プログラマーを対象にしています。

INCDIR パラメーター:

コンパイラーがコピー・ファイルを検索するために使用する検索パスに追加す

る、 1 つ以上のディレクトリーを指定します。ソース・コードで指定されたコピー・ファイルを解決できないときには、コンパイラーは、ここで指定されたディレクトリーを検索します。

***NONE**
コピー・ファイルを探すために、ユーザー・ディレクトリーは検索されません。デフォルトでは、なお現行ディレクトリーが検索されます。

directory
コピー・ファイルの検索先として最大 32 のディレクトリーを指定します。指定したディレクトリー以外に、コピー・ファイルを探すために、現行ディレクトリーも検索されます。

PGMINFO パラメーター:
このオプションは、プログラム・インターフェース情報を生成するかどうか、どこに生成するかを指定します。次の順序でオプション値を指定してください。

生成
プログラム・インターフェース情報を生成するかどうかを指定します。指定できる値は次のとおりです。

***NO**
プログラム・インターフェース情報は生成されません。

***PCML**
PCML (プログラム呼び出しマークアップ言語) が、指定したロケーションの中に生成されることを指定します。PCML が生成されると、より少ない Java コードで Java プログラムが、この COBOL プログラムを簡単に呼び出せるようになります。生成された PCML が入るストリーム・ファイルの名前は、 INFOSTMF オプションで指定する必要があります。

ロケーション
生成パラメーターが *PCML の場合、生成されたプログラムのロケーションを指定します。指定できる値は次のとおりです。

***STMF**
ストリーム・ファイルの中にプログラム情報を生成することを指定します。生成された情報が入るストリーム・ファイルの名前は、 INFOSTMF オプションで指定する必要があります。

***MODULE**
COBOL モジュールにプログラム情報を保管することを指定します。

***ALL**
ストリーム・ファイルの中にプログラム情報を生成し、モジュールに格納することを指定します。生成された情報が入るストリーム・ファイルの名前は、 INFOSTMF オプションで指定する必要があります。

INFOSTMF パラメーター:
PGMINFO オプションで指定された、生成されたプログラム・インターフェース情報を入れるストリーム・ファイルのパス名を指定します。パス名は、絶対修飾名と相対修飾名のどちらを指定することもできます。絶対パス名は「/」で始まり、相対パス名は「/」以外の文字で始まります。絶対修飾の場合、パス名は完全なものです。相対修飾の場合、パス名にジョブの現行作業ディレクトリーを付

け加えることによって、パス名が完全なものになります。このパラメーターは、PGMINFO パラメーターの値が *NO 以外のときにだけ指定できます。

ソース・プログラムをコンパイルしてモジュール・オブジェクトにする例

この例では、CRTCBLMOD コマンドを使用して ILE COBOL モジュール・オブジェクトを作成する方法を示します。

1. モジュール・オブジェクトを作成するためには、次のように入力します。

```
CRTCBLMOD MODULE(MYLIB/XMPLE1)
SRCFILE(MYLIB/QCBLLESRC)  SRCMBR(MYLIB/XMPLE1)
OUTPUT(*PRINT)
TEXT('My ILE COBOL Program on iSeries')
CVTOPT(*FLOAT)
```

   CRTCBLMOD コマンドは、ソースが入れられているのと同じライブラリーである MYLIB に、モジュール XMPLE1 を作成します。出力オプション OUTPUT(*PRINT) はコンパイル・リストを指定します。 変換オプション CVTOPT(*FLOAT) は、浮動小数点データ・タイプが、それらの DDS 名により、COMP-1 (単精度) または COMP-2 (倍精度) の USAGE でプログラムに含められることを指定します。

2. コンパイル・リストを表示するには、次の CL コマンドのどれかを入力します。

   注: コンパイル・リストを表示するためには、次にリストされているコマンドを使用する権限がなければなりません。
   - DSPJOB、続いてオプション 4 (スプール・ファイルの表示) を選択する
   - WRKJOB
   - WRKOUTQ 待ち行列名
   - WRKSPLF

異なるターゲット・リリースの指定

ILE COBOL プログラムは、IBM i オペレーティング・システムの現行リリースを使用する System i 上でコンパイルできます。

注: ILE COBOL コンパイラーと OPM COBOL/400 コンパイラーは、それぞれ別個のプロダクト・オプションです。ここに示す情報は、ILE COBOL コンパイラーの現行リリースにしか適用されません。

CRTCBLMOD と CRTBNDCBL コマンドのターゲット・リリース (TGTRLS) パラメーターにより、モジュール・オブジェクトを使用する予定のリリース・レベルを指定することができます。 TGTRLS パラメーターに指定できるのは、*CURRENT、*PRV、およびターゲット・リリース の 3 種類の値です。

- モジュール・オブジェクトを現在システムで稼働しているオペレーティング・システムのリリースで使用する場合は、*CURRENT を指定します。たとえば、V4R4M0 がシステムで稼働している場合、*CURRENT は V4R4M0 がインストールされているシステムのプログラムを使用する予定であるという意味です。この値がデフォルトです。
- 前のリリースのモディフィケーション・レベル 0 のオペレーティング・システムで使用するオブジェクトの場合は、*PRV を指定します。たとえば、V4R4M0 が

システムで稼働している場合、*PRV は V4R3M0 がインストールされているシステムでオブジェクトを使用する予定であるという意味です。また、このオブジェクトは、オペレーティング・システムのそれより新しいリリースがインストールされているシステムでも使用できます。

- ターゲット・リリース により、モジュール・オブジェクトを使用する予定のリリース・レベルを指定することができます。このパラメーターに入力できる値は、現行のバージョン、リリース、およびモディフィケーション・レベルに応じて異なり、この値は新しいリリースごとに変更されます。

  ターゲット環境のリリース・レベルを、VxRxMx の形式で指定します。このオブジェクトは、指定されたリリースまたはそれより新しいリリースのオペレーティング・システムがインストールされているシステムで使用できます。

  たとえば、V4R2M0 を指定すると、オブジェクトは V4R2M0 システムで使用できます。

TGTRLS パラメーターの詳細については 49 ページの『TGTRLS パラメーター』を参照してください。

次の制約に注意してください。

- オブジェクト・プログラムは、現行リリース以降のリリースに復元できます。前のリリースのオブジェクト・プログラムのうち、TGTRLS ターゲット・リリースで許可されていないものについては復元できません。
- ライブラリー・リストのシステム部分にプロダクト・ライブラリーを入れることはできません。

CRTCBLMOD の国別言語ソート・シーケンスの指定

ILE COBOL ソース・プログラムをコンパイルする時点で、プログラムの実行時に使用する照合順序を明示的に指定したり、プログラムの実行時に照合順序が判別される方法を指定したりすることができます。

照合順序を指定するには、最初に ALPHABET 文節を使用して SPECIAL-NAMES 段落の*英字名* を定義し、その*英字名* を NLSSORT 実装者名と関連付けます。次に、ENVIRONMENT DIVISION の中の PROGRAM COLLATING SEQUENCE 文節、または SORT/MERGE ステートメントの COLLATING SEQUENCE 句の中でこの*英字名* を参照することにより、使用する照合順序が指定した*英字名* によって決まるものであることを指定します。

使用する実際の照合順序は、CRTCBLMOD および CRTBNDCBL コマンドの SRTSEQ および LANGID パラメーターのオプションによって指定します。たとえば、SRTSEQ(*JOBRUN) および LANGID(*JOBRUN) を指定する場合、プログラムの照合順序は実行時に解決されます。この値では、ソース・プログラムをいったんコンパイルして、実行時に異なった照合順序で使用できます。 SRTSEQ および LANGID と関連した PROCESS ステートメントのオプションを使用して照合順序を指定することもできます (60 ページの『PROCESS ステートメントを使用したコンパイラー・オプションの指定』を参照)。

ソース・プログラムにその ALPHABET 文節の英字名 に関連する NLSSORT がない場合、または NLSSORT を指定する ALPHABET 文節はあるが、関連した英字名が PROGRAM COLLATING SEQUENCE 文節または SORT/MERGE ステートメントの COLLATING SEQUENCE 句の中で参照されていない場合、SRTSEQ と LANGID パラメーターによって識別されるソート・シーケンスは使用されません。

NLSSORT と関連した英字名 は、ファイル記述 (FD) 記入項目の CODE-SET 文節の場合のようにして文字コード・セットを判別するために使用することはできません。文字コード・セットを判別するために使用される英字名 は、独立した ALPHABET 文節中で指定しなければなりません。

ALPHABET 文節、PROGRAM COLLATING SEQUENCE 文節、および SORT/MERGE ステートメントに関する詳しい説明については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。 SRTSEQ と LANGID パラメーターについては 33 ページの『CRTCBLMOD コマンドのパラメーター』を参照してください。

プロファイリング・データの収集

プロファイリング・コードがモジュールに追加されたならば、プロファイリング・コードは、プロファイリング・データが収集できるようにプログラム・オブジェクトまたはサービス・プログラム・オブジェクトに入れる必要があります。プロファイリング・データは、CHGPGM CL コマンドを使用してプログラム・オブジェクトに、また、CHGSRVPGM CL コマンドを使用してサービス・プログラムに適用することができます。すべてのプロファイリング・データをプログラム・オブジェクトまたはサービス・プログラムに適用するためには、全適用 (*APYALL) 値を用いて PRFDTA パラメーターを指定してください。プロシージャー内のコードの再順序付けを行うプロファイル作成データだけを適用するためには、値 *APYBLKORD を指定します。プロシージャーの再順序付けを行うプロファイル作成データだけを適用するためには、*APYPRCORD を指定してください。

プロファイリング・データは、プログラム・プロファイリングの開始 (STRPGMPRF) CL コマンドを指定すると収集されます。プロファイリング・データは、システム上で活動状態であり、プロファイル・コードを含んでいるすべてのプログラム・オブジェクトおよびサービス・プログラムから生成されます。

十分な量のプロファイリング・データが収集されたならば、プログラム・プロファイリングの終了 (ENDPGMPRF) CL コマンドを入力してください。

プログラム・プロファイリング・データは、CHGPGM および CHGSRVPGM CL コマンドに PRFDTA パラメーターの *CLR 値を指定すると、プログラム・オブジェクトまたはサービス・プログラム内のモジュールから除去できます。

モジュールを使用可能状態にしてプロファイリング・データを収集できるようにすると、モジュール・オブジェクト内で追加コードが生成されます。このコードは、プロシージャー内の基本ブロックが実行された回数のほか、プロシージャーが呼び出された回数を収集するのに使用されます。プロファイリング・データの収集を使用可能にするには、モジュールを 30 (*FULL) またはそれ以上の最適化レベルでコンパイルする必要があります。

プロシージャー内の基本ブロックについて収集されたデータを使用して、ILE 最適化変換プログラムは、キャッシュ使用率がさらに高くなるようにこれらのブロックの配置を変更します。ブロック情報は、モジュール内のプロシージャーに適用されますが、複数のモジュール境界にまたがることはありません。

バインダーがプロシージャー呼び出しデータを使用するのは、ページ使用率がさらに高くなるように相互に頻繁に呼び出すプロシージャーをパッケージするためです。すなわち、プロファイルが作成されたプログラム内で、モジュール A の中の PROCA をモジュール B の中の PROCB に並べてパッケージする (PROCA が PROCB に対して何回も呼び出しを行う場合) ことができます。プロシージャー呼び出しデータはプログラム・レベルで適用されるため、複数のモジュール境界にまたがります。

プロファイリング・データは、現行のターゲット・リリースを指定した場合のみ収集することができます。 ILE プログラムまたはサービス・プログラムのプロファイルが作成されるためには、プログラムが V4R2M0 またはそれ以上のターゲット・リリースである必要があります。これは、プロファイリング・データまたはプロファイリングされたプログラムを収集できるようになっているプログラムは、保管したり、V4R2M0 より前のリリースに復元したりすることができないということも意味しています。

PRFDTA パラメーターの詳細については 53 ページの『PRFDTA パラメーター』を参照してください。

注: 並列環境、たとえば、マルチスレッド・プロセスで実行しているプログラムについてプロファイリング・データが収集された場合には、データが不正確である可能性があります。

日付、時刻、およびタイム・スタンプのデータ・タイプの指定

COBOL 日時クラスの項目には、日付、時刻、およびタイム・スタンプ項目が含まれます。これらの項目は、データ記述項目の FORMAT 文節で宣言されます。たとえば、次のとおりです。

```
01  group-item.
    05 date1 FORMAT DATE "%m/%d/@Y".
    05 date2 FORMAT DATE.
```

日時クラスの項目の場合、FORMAT 文節は PICTURE 文節の代わりに使用されます。上の例では、キーワード FORMAT の後で、キーワード DATE が日付のカテゴリーの項目を宣言します。キーワード DATE の後で、フォーマット設定リテラルが日付データ項目の形式を記述します。データ項目 `date1` の場合、`%m` は月、`%d` は日、また `@Y` は年 (2 桁の世紀を含む) を表します。文字 `%` および `@` は指定子の先頭にきます。ここに示してある 3 つの指定子は、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」に記載されている指定子の集合の一部です。

もう 1 つの日付データ項目 `date2` では、フォーマット設定リテラルが明示的に指定されることはありませんが、SPECIAL-NAMES 段落にデフォルト日付形式を指定できます。以下に、その例を示します。

```
SPECIAL-NAMES.  FORMAT OF DATE IS "@C:%y:%j".
```

上の SPECIAL-NAMES 段落がデータ項目 date2 と同じプログラムに指定されていれば、その日付形式は @C:%y:%j になるはずです。また、SPECIAL-NAMES 段落が存在しない場合には、日付項目の形式はデフォルトである ISO がとられます。 ISO 日付の形式は @Y-%m-%d です。

デフォルトでは、COPY DDS は日時クラスの項目を宣言すると、英数字項目について PICTURE 文節を生成します。 PICTURE 文節を FORMAT 文節に変更するために、いくつかの新しい CVTOPT パラメーター値が定義されました。それらは、次のものです。

- *DATE
- *TIME
- *TIMESTAMP

*DATE が指定されていれば、DDS 日付データ・タイプはすべて COBOL 日付項目に変換されます。すなわち、PICTURE 文節ではなく、FORMAT 文節が生成されます。

DDS では、日付フィールドの形式を指定するために、DATFMT キーワードを指定することができます。 DATFMT キーワードは、ゾーン、パック、および文字の各フィールドに対しても指定できます。これらのタイプのフィールドでは、COPY DDS は、通常、数字ゾーン、数字パック、および英数字の各データ項目について、それぞれ PICTURE 文節を生成します。 CVTOPT パラメーターの *CVTTODATE 値を指定すると、これらの項目について COPY DDS に強制的に FORMAT 文節を生成させることができます。

ゾーン、パック、および文字の各 DDS フィールドに許されている DATFMT パラメーターと、CVTOPT(*CVTTODATE) 変換パラメーターが指定されたときに DDS から生成される、それぞれに相当する ILE COBOL 形式のリストについては 492 ページの『日時クラス』と 212 ページの『日時データ・タイプを処理する』を参照してください。

4 桁日付と 2 桁日付の混在を含む移動および比較の場合、ILE COBOL では、1900 という基本世紀と 40 という基本年でデフォルトのウィンドウ操作アルゴリズムを使用します。結果が不正確になる可能性があるため、デフォルトのウィンドウを指定変更する必要があります。 ILE COBOL ウィンドウ操作アルゴリズムおよびその指定変更方法については 217 ページの『2 桁年の 4 桁年または世紀への変換』を参照してください。

PROCESS ステートメントを使用したコンパイラー・オプションの指定

PROCESS ステートメントは、ILE COBOL ソース・プログラムのオプショナルの部分です。 PROCESS ステートメントを使用して、通常はコンパイル時に指定するオプションを指定することができます。

PROCESS ステートメントで指定されたオプションは、CRTCBLMOD または CRTBNDCBL CL コマンドで指定された対応するオプションを**指定変更**します。

次の規則が適用されます。

- このステートメントは、IDENTIFICATION DIVISION ヘッダーの直前にある、新しいコンパイル単位を開始する ILE COBOL ソース・プログラムの最初のソース・ステートメントの前に置く必要があります。
- ステートメントは PROCESS の語で始まります。オプションは複数行にわたって指定することができますが、PROCESS という語は 1 行目だけに入れることができます。
- 語 PROCESS およびすべてのオプションは、8 から 72 までの位置に記入する必要があります。位置 7 はブランクのままでなければなりません。 上記以外の位置は ILE COBOL ソース・ステートメントにおける場合と同じように使用できます。すなわち、位置 1 から 6 まではシーケンス番号、位置 73 から 80 までは識別の目的のために使用できます。
- オプションは、ブランクまたはコンマ、あるいはその両方で区切る必要があります。
- オプションは任意の順序で指定できます。対立するオプションが指定された場合 (たとえば、XREF と NOXREF)、最後に検出されたオプションが優先されます。
- オプション・キーワードは正しいが、サブオプションにエラーがある場合、デフォルトのサブオプションを指定したものと見なされます。

指定できる PROCESS ステートメントのオプション、およびそれに相当する CRTCBLMOD または CRTBNDCBL コマンド・パラメーターとそのオプションについては、次の表を参照してください。デフォルトには下線が付けられています。PROCESS ステートメント・オプションの説明は 33 ページの『CRTCBLMOD コマンドのパラメーター』にあるパラメーターとオプションの説明と同じです。

注: CRTCBLMOD および CRTBNDCBL コマンドのすべてのパラメーターに、それに対応する PROCESS ステートメントがあるとは限りません。さらに、いくつかのオプションは、PROCESS ステートメントでのみ使用可能です。 PROCESS ステートメントにのみ使用可能なオプションについては 68 ページの『PROCESS ステートメントのオプション』を参照してください。

PROCESS ステートメントのオプション	**CRTCBLMOD/CRTBNDCBL**
	OUTPUT パラメーターのオプション
OUTPUT NOOUTPUT	***PRINT** *NONE

PROCESS ステートメントのオプション	**CRTCBLMOD/CRTBNDCBL**
	GENLVL パラメーターのオプション
GENLVL(nn)	nn

PROCESS ステートメントのオプション	**CRTCBLMOD/CRTBNDCBL**
	OPTION パラメーターのオプション
SOURCE **SRC** NOSOURCE NOSRC	***SOURCE** ***SRC** *NOSOURCE *NOSRC

PROCESS ステートメントのオプション	CRTCBLMOD/CRTBNDCBL OPTION パラメーターのオプション
NOXREF XREF	***NOXREF** *XREF
GEN NOGEN	***GEN** *NOGEN
NOSEQUENCE SEQUENCE	***NOSEQUENCE** *SEQUENCE
NOVBSUM VBSUM	***NOVBSUM** *VBSUM
NONUMBER NUMBER LINENUMBER	***NONUMBER** *NUMBER *LINENUMBER
NOMAP MAP	***NOMAP** *MAP
NOOPTIONS OPTIONS	***NOOPTIONS** *OPTIONS
QUOTE APOST	***QUOTE** *APOST
NOSECLVL SECLVL	***NOSECLVL** *SECLVL
PRTCORR NOPRTCORR	***PRTCORR** *NOPRTCORR
MONOPRC NOMONOPRC	***MONOPRC** *NOMONOPRC
RANGE NORANGE	***RANGE** *NORANGE
NOUNREF UNREF	***NOUNREF** *UNREF
NOSYNC SYNC	***NOSYNC** *SYNC
NOCRTF CRTF	***NOCRTF** *CRTF
NODUPKEYCHK DUPKEYCHK	***NODUPKEYCHK** *DUPKEYCHK
NOINZDLT INZDLT	***NOINZDLT** *INZDLT
NOBLK BLK	***NOBLK** *BLK
STDINZ NOSTDINZ STDINZHEX00	***STDINZ** *NOSTDINZ *STDINZHEX00
NODDSFILLER DDSFILLER	***NODDSFILLER** *DDSFILLER
該当せず	***NOIMBEDERR** *IMBEDERR

PROCESS ステートメントのオプション	CRTCBLMOD/CRTBNDCBL
	OPTION パラメーターのオプション
STDTRUNC NOSTDTRUNC	***STDTRUNC** *NOSTDTRUNC
CHGPOSSGN NOCHGPOSSGN	***CHGPOSSGN** *NOCHGPOSSGN
該当せず	***NOEVENTF** *EVENTF
MONOPIC NOMONOPIC	***MONOPIC** *NOMONOPIC
NOCRTARKIDX CRTARKIDX	***NOCRTARKIDX** *CRTARKIDX

PROCESS ステートメントのオプション	CRTCBLMOD/CRTBNDCBL
	CVTOPT パラメーターのオプション
NOVARCHAR VARCHAR	***NOVARCHAR** *VARCHAR
NODATETIME DATETIME	***NODATETIME** *DATETIME
NOCVTPICXGRAPHIC CVTPICXGRAPHIC CVTPICGGRAPHIC NOCVTPICGGRAPHIC	***NOPICXGRAPHIC** *PICXGRAPHIC *PICGGRAPHIC *NOPICGGRAPHIC
NOCVTPICNGRAPHIC CVTPICNGRAPHIC	***NOPICNGRAPHIC** *PICNGRAPHIC
NOFLOAT FLOAT	***NOFLOAT** *FLOAT
NODATE DATE	***NODATE** *DATE
NOTIME TIME	***NOTIME** *TIME
NOTIMESTAMP TIMESTAMP	***NOTIMESTAMP** *TIMESTAMP
NOCVTTODATE CVTTODATE	***NOCVTTODATE** *CVTTODATE

PROCESS ステートメントのオプション	CRTCBLMOD/CRTBNDCBL
	OPTIMIZE パラメーターのオプション
NOOPTIMIZE BASICOPT FULLOPT NEVEROPTIMIZE	***NONE** *BASIC *FULL *NEVER

PROCESS ステートメントのオプション	CRTCBLMOD/CRTBNDCBL FLAGSTD パラメーターのオプション
NOFIPS MINIMUM INTERMEDIATE HIGH	***NOFIPS** *MINIMUM *INTERMEDIATE *HIGH
NOOBSOLETE OBSOLETE	***NOOBSOLETE** *OBSOLETE

PROCESS ステートメントのオプション **EXTDSPOPT**(*a b c*)	CRTCBLMOD/CRTBNDCBL EXTDSPOPT パラメーターのオプション
DFRWRT NODFRWRT	***DFRWRT** *NODFRWRT
UNDSPCHR NOUNDSPCHR	***UNDSPCHR** *NOUNDSPCHR
ACCUPDALL ACCUPDNE	***ACCUPDALL** *ACCUPDNE

PROCESS ステートメントのオプション	CRTCBLMOD/CRTBNDCBL FLAG パラメーターのオプション
FLAG(nn)	nn

PROCESS ステートメントのオプション	CRTCBLMOD/CRTBNDCBL LINKLIT パラメーターのオプション
LINKPGM LINKPRC	***PGM** *PRC

PROCESS ステートメントのオプション **SRTSEQ**(*a*)	CRTCBLMOD/CRTBNDCBL SRTSEQ パラメーターのオプション
HEX JOB JOBRUN LANGIDUNQ LANGIDSHR "LIBL/ソート順序テーブル名" "CURLIB/ソート順序テーブル名" "ライブラリー名/ソート順序テーブル名" "ソート順序テーブル名"	***HEX** *JOB *JOBRUN *LANGIDUNQ *LANGIDSHR *LIBL/ソート順序テーブル名 *CURLIB/ソート順序テーブル名 ライブラリー名/ソート順序テーブル名 ソート順序テーブル名

PROCESS ステートメントのオプション **LANGID**(*a*)	CRTCBLMOD/CRTBNDCBL LANGID パラメーターのオプション
JOBRUN JOB "言語 ID 名"	***JOBRUN** *JOB 言語 ID 名

PROCESS ステートメントのオプション ENBPFRCOL(*a*)	CRTCBLMOD/CRTBNDCBL ENBPFRCOL パラメーターのオプション
PEP ENTRYEXIT FULL	***PEP** *ENTRYEXIT *FULL

PROCESS ステートメントのオプション PRFDTA(*a*)	CRTCBLMOD/CRTBNDCBL PRFDTA パラメーターのオプション
NOCOL COL	***NOCOL** *COL

PROCESS ステートメントのオプション CCSID(*a b c d*)	CRTCBLMOD/CRTBNDCBL CCSID パラメーターのオプション
a = ロケールの 1 バイト・データ CCSID	
JOBRUN JOB HEX *コード化文字セット-id*	***JOBRUN** *JOB *HEX *コード化文字セット-id*
b = 非ロケールの 1 バイト・データ CCSID	
CCSID (上の “a” に指定されている CCSID を使用する) JOBRUN JOB HEX *コード化文字セット-id*	該当せず
c = 非ロケール 2 バイト・データ CCSID	
CCSID (上の “a” に指定されている CCSID を使用する) JOBRUN JOB HEX *コード化文字セット-id*	該当せず
d = XML GENERATE 1 バイトまたはユニコードのデータ出力 CCSID	
JOBRUN CCSID (上の “a” に指定されている CCSID を使用する) JOB HEX *コード化文字セット-id*	該当せず

PROCESS ステートメントのオプション NTLCCSID(a)	CRTCBLMOD/CRTBNDCBL NTLCCSID パラメーターのオプション
13488 コード化文字セット ID	**13488** コード化文字セット ID

PROCESS ステートメントのオプション DATTIM(*a b*)	CRTCBLMOD/CRTBNDCBL
4 桁の基本年 (デフォルトは 1900) *2 桁の基本年* (デフォルトは 40)	該当せず

PROCESS ステートメントのオプション THREAD(*a*)	CRTCBLMOD/CRTBNDCBL
NOTHREAD SERIALIZE	該当せず

PROCESS ステートメントのオプション ARITHMETIC(*a*)	CRTCBLMOD/CRTBNDCBL ARITHMETIC パラメーターのオプション
NOEXTEND EXTEND31 EXTEND63	***NOEXTEND** *EXTEND31 *EXTEND63

PROCESS ステートメントのオプション	CRTCBLMOD/CRTBNDCBL
NOGRAPHIC GRAPHIC	該当せず

PROCESS ステートメントのオプション	CRTCBLMOD/CRTBNDCBL
NONATIONAL NATIONAL	該当せず

PROCESS ステートメントのオプション	CRTCBLMOD/CRTBNDCBL
NOLSPTRALIGN LSPTRALIGN	該当せず

PROCESS ステートメントのオプション	CRTCBLMOD/CRTBNDCBL
NOCOMPASBIN COMPASBIN	該当せず

PROCESS ステートメントのオプション	CRTCBLMOD/CRTBNDCBL DBGVIEW パラメーターのオプション
NOCOMPRESSDBG COMPRESSDBG	***NOCOMPRESSDBG** *COMPRESSDBG

PROCESS ステートメントのオプション OPTVALUE(*a*)	CRTCBLMOD/CRTBNDCBL
NOOPT OPT	該当せず

PROCESS ステートメントのオプション	CRTCBLMOD/CRTBNDCBL
NOADJFILLER ADJFILLER	該当せず

PROCESS ステートメントのオプション NTLPADCHAR(a b c)	CRTCBLMOD/CRTBNDCBL NTLPADCHAR パラメーターのオプション
a = 1 バイト文字から国別文字へ移動する場合の埋め込み文字	
NX″0020″ 1 つの国別文字を示す 国別 16 進リテラル	**NX″0020″** 国別文字
b = 2 バイト文字から国別文字へ移動する場合の埋め込み文字	
NX″3000″ 1 つの国別文字を示す 国別 16 進リテラル	**NX″3000″** 国別文字
c = 国別文字から国別文字へ移動する場合の埋め込み文字	
NX″3000″ 1 つの国別文字を示す 国別 16 進リテラル	**NX″3000″** 国別文字

PROCESS ステートメントのオプション LICOPT(a)	CRTCBLMOD/CRTBNDCBL LICOPT パラメーターのオプション
ライセンス内部コード・オプション・ストリング	ライセンス内部コード・オプション・ストリング

PROCESS ステートメントのオプション PGMINFO(a b)	CRTCBLMOD/CRTBNDCBL PGMINFO パラメーターのオプション
a = 生成されるプログラム・インターフェース情報	
NOPGMINFO PCML	***NO***PCML
b = 生成されたプログラム情報のロケーション	
MODULE	***STMF** *MODULE*ALL

PROCESS ステートメントのオプション STGMDL(*a*)	CRTCBLMOD STGMDL パラメーターのオプション
INHERIT SNGLVL TERASPACE	***INHERIT** *SNGLVL *TERASPACE

PROCESS ステートメントのオプション STGMDL(*a*)	CRTBNDCBL STGMDL パラメーターのオプション
SNGLVL	***SNGLVL**
INHERIT	*INHERIT
TERASPACE	*TERASPACE

PROCESS ステートメントのオプション ACTGRP(*a*)	CRTBNDCBL ACTGRP パラメーターのオプション
STGMDL	***STGMDL**
NEW	*NEW*CALLER活動化グループ名
CALLER	
'活動化グループ名'	

PROCESS ステートメントの EXTDSPOPT、SRTSEQ、LANGID、ENBPFRCOL、PRFDTA、CCSID、DATTIM、ARITHMETIC、THREAD、NTLCCSID、STGMDL、ACTGRP、および PGMINFO オプションは、FLAG(nn) 構文と同様に、関連オプションを括弧に入れてコーディングする必要があります。

EXTDSPOPT オプションの括弧内には複数のオプションを指定できます。たとえば、DFRWRT と UNDSPCHR を指定するには、次のように入力します。

```
EXTDSPOPT(DFRWRT UNDSPCHR)
```

EXTDSPOPT または EXTDSPOPT( ) という指定も有効です。

PROCESS ステートメントで EXTDSPOPT だけが指定される場合、追加オプションのすべてのデフォルト値が有効になります。

EXTDSPOPT( ) を指定しても、プログラムに影響はありません。

DATTIM、CCSID、および PGMINFO 処理ステートメントのオプションでは、それぞれの括弧内に複数の値を入れることができます。これらのオプションでは、括弧内の値の順序は意味を持ちます。たとえば、DATTIM オプションには値が 2 つあります。最初の値は基本世紀で、2 つ目は基本年です。これは、基本年を指定するためには基本世紀を指定する必要があるということを意味しています。

PROCESS ステートメントのオプション

以下のオプションは、PROCESS ステートメントにのみ使用可能であり、CRTCBLMOD または CRTBNDCBL コマンドには同等のパラメーターはありません。

NOGRAPHIC オプション

NOGRAPHIC が明示的または暗黙のうちに指定されると、ILE COBOL コンパイラーは、16 進 0E および 16 進 0F を含む非数字リテラルを、SBCS 文字しか含まれていないものとして扱います。 16 進 0E および 16 進 0F はシフトイン文字およびシフトアウト文字として処理されず、SBCS 文字ストリングの一部であると見なされます。 DBCS サポートについては 677 ページの『付録 D. 2 バイト文字セットを使用する国別言語のサポート』を参照してください。

GRAPHIC オプション

PROCESS ステートメントの GRAPHIC オプションは、混合リテラルの中の DBCS 文字の処理に使用できます。**混合リテラル**とは、SBCS 文字と DBCS 文字の混じったリテラルのことです。 GRAPHIC オプションを指定する場合、混合リテラルは、16 進 0E および 16 進 0F が、それぞれシフトアウト文字およびシフトイン文字であると見なして処理され、それらで混合リテラルの DBCS 文字が囲まれます。シフトイン文字およびシフトアウト文字はそれぞれ 1 バイトを占めます。

DATTIM オプション

ILE COBOL がそのウィンドウ操作のアルゴリズムで使用する日付ウィンドウを指定します。 (218 ページの『DATTIM 処理ステートメントのオプションを使用して、デフォルト日付ウィンドウを指定変更する』を参照。)

4 桁の基本世紀

これは最初の引き数でなければなりません。 ILE COBOL がそのウィンドウ操作のアルゴリズムで使用する基本世紀を定義します。 DATTIM 処理ステートメント・オプションが指定されていない場合は、1900 が使用されます。

2 桁の基本年

これは 2 番目の引き数でなければなりません。 ILE COBOL がそのウィンドウ操作のアルゴリズムで使用する基本年を定義します。 DATTIM 処理ステートメント・オプションが指定されていない場合は、40 が使用されます。

THREAD オプション

作成されたモジュール・オブジェクトがマルチスレッド環境で実行可能かどうかを指定します。マルチスレッド化の ILE COBOL サポートについては 405 ページの『第 15 章 ILE COBOL プログラムをマルチスレッド化するための準備』を参照してください。指定できる値は次のとおりです。

NOTHREAD

作成されたモジュール・オブジェクトは、マルチスレッド環境で実行できません。 これはデフォルトです。

SERIALIZE

作成されたモジュール・オブジェクトは、1 つのジョブで複数のスレッドを使用して実行することができます。 1 つまたは複数のモジュール内のプロシージャーへのアクセスが、逐次化されます。すなわち、それぞれのスレッド・セーフ・モジュールは、プロシージャーに入ったときにロックされ、プロシージャーから出るときにアンロックされる、再帰的 mutex を持つことになります。実行単位内では、同じモジュールが一度に 1 つのスレッドのみをアクティブにすることができます。

NONATIONAL オプション

NONATIONAL が指定または暗黙指定されている場合、ピクチャー記号 N のみから構成され、明示的な USAGE 文節がないピクチャー文字ストリングを持つ項目には、USAGE DISPLAY-1 が暗黙指定されます。

NATIONAL オプション
NATIONAL が指定されている場合、ピクチャー記号 N のみから構成され、明示的な USAGE 文節がないピクチャー文字ストリングを持つ項目には、USAGE NATIONAL が暗黙指定されます。

NOLSPTRALIGN オプション
NOLSPTRALIGN が指定または暗黙指定されている場合、USAGE POINTER または PROCEDURE-POINTER が指定されたデータ項目は、充てんスペースなしで連続してリンケージ・セクションに置かれます。

LSPTRALIGN オプション
LSPTRALIGN が指定されている場合、USAGE POINTER または PROCEDURE-POINTER が指定されたデータ項目は、リンケージ・セクション内のレコードの先頭から相対的に 16 バイトの倍数の位置に置かれます。

NOCOMPASBIN オプション
NOCOMPASBIN が指定または暗黙指定されている場合、USAGE COMPUTATIONAL または COMP は USAGE COMP-3 と同じ意味を持ちます。

COMPASBIN オプション
COMPASBIN が指定されている場合、USAGE COMPUTATIONAL または COMP は USAGE COMP-4 と同じ意味を持ちます。

OPTVALUE オプション
指定できる値は次のとおりです。

NOOPT
作業用記憶域セクション内で VALUE 文節を含んでいるデータ項目を初期設定するコードの生成は最適化されません。これはデフォルトです。

OPT
作業用記憶域セクション内で VALUE 文節を含んでいるデータ項目を初期設定するコードの生成が最適化されます。

NOADJFILLER オプション
ポインター・データ項目がグループの最初のメンバーである場合、このポインター・データ項目を位置合わせするためにコンパイラーによって挿入される暗黙の充てん文字は、グループの直後に挿入されます。これはデフォルトです。

ADJFILLER オプション
ポインター・データ項目がグループの最初のメンバーである場合、このポインター・データ項目を位置合わせするためにコンパイラーによって挿入される暗黙の充てん文字は、グループの直前に挿入されます。

複数のソース・プログラムのコンパイル

PROCESS ステートメントは、入力ソース・メンバー内の一連の ILE COBOL ソース・プログラムの中の各コンパイル単位の先頭に置くことができます。複数の ILE COBOL ソース・プログラムをコンパイルするとき、CRTCBLMOD または CRTBNDCBL コマンド上で指定されるすべてのオプションをマージした結果に、すべてのデフォルト・オプションと、さらに ILE COBOL ソース・プログラムに先行する最後の PROCESS ステートメントで指定されるオプションを加えた結果が、そ

の ILE COBOL ソース・プログラムのコンパイルで有効になります。すべてのコンパイラー出力は、CRTCBLMOD または CRTBNDCBL コマンドによって指定された宛先に送られます。

すべてのモジュール・オブジェクトまたはプログラム・オブジェクトは、MODULE パラメーターまたは PGM パラメーターで指定されるライブラリーに保管されます。 MODULE パラメーターや PGM パラメーターにモジュール名 またはプログラム名 を指定する場合、ILE COBOL ソース・プログラムの順序列で最初の ILE COBOL ソース・プログラムに対応する最初のモジュール・オブジェクトまたはプログラム・オブジェクトにその名前が使用され、同じ入力ソース・メンバー中の他の ILE COBOL ソース・プログラムに対応するすべてのモジュール・オブジェクトまたはプログラム・オブジェクトには、ILE COBOL ソース・プログラム内の PROGRAM-ID 段落で指定された名前が使用されます。

PROCESS ステートメント内での COPY の使用

COPY ステートメントは、ソース・プログラムで文字ストリングや区切り文字が使用できるところではどこでも使用することができます。各 COPY ステートメントは、その前に 1 個のスペースと、その後にピリオドまたはスペースがなければなりません。 COPY ステートメントの詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の『COPY ステートメント』のセクションを参照してください。

PROCESS ステートメント内で形式 1 の COPY ステートメントを使用すると、それ以前にソース・ライブラリーに保管されたコンパイラー・オプションを検索し、PROCESS ステートメントにそれらを含めることができます。 COPY を使用することにより、コンパイラーによってデフォルトとして指定されたオプションを指定変更するオプションを含めることができます。どの PROCESS ステートメントのオプションも、COPY ステートメントで検索できます。

コンパイラー・オプションは、PROCESS ステートメント内の COPY ステートメントの前後のどちらにでも置くことができます。あるオプションが複数指定された場合、そのオプションの最後に見つかったものが、先行するそのオプションのすべての指定を指定変更します。

次の例は PROCESS ステートメント内の COPY ステートメントの使用法を示したものです。この例では、NOMAP がライブラリー・メンバーの対応するオプションを指定変更していることにも注意してください。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/COPYPROC       ISERIES1  06/02/15 11:39:37     ページ  2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
          000100 PROCESS XREF
          000200 COPY PROCDFLT.
         +000100         MAP, SOURCE, APOST                                          PROCDFLT
         +000200                                                                     PROCDFLT
          000300         NOMAP, FLAG(20)
    1     000400 IDENTIFICATION DIVISION.
    2     000500 PROGRAM-ID. COPYPROC.
    3     000600 ENVIRONMENT DIVISION.
    4     000700 CONFIGURATION SECTION.
    5     000800   SOURCE-COMPUTER. IBM-ISERIES.
    6     000900   OBJECT-COMPUTER. IBM-ISERIES.
    7     001000 PROCEDURE DIVISION.
          001100 MAINLINE.
    8     001200     DISPLAY "HELLO WORLD".
    9     001300     STOP RUN.
          001400
                         * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 9. PROCESS ステートメント内での COPY の使用

コンパイラー出力について

コンパイラーに、印刷された報告書の選択を作成するよう指示することができます。デフォルトでは、この出力はシステム・プリンター・ファイル QSYSPRT に送られます。

出力には次のものが含まれます。

- コマンド・オプションの要約
- **オプションのリスト**。コンパイルに有効なオプションのリストです。OPTION(*OPTIONS) を使用します。
- **ソース・リスト**。ソース・プログラムに含まれるステートメントのリストです。OPTION(*SOURCE) を使用します。
- **verb 使用状況のリスト**。 COBOL verb と、各 verb が使用される回数のリストです。 OPTION(*VBSUM) を使用します。
- **データ部マップ**。データに関するコンパイラー生成情報の用語集です。OPTION(*MAP) を使用します。
- **FIPS メッセージ**。 FIPS COBOL サブセット、すべてのオプション・モジュール、古くなった言語エレメント、または FIPS COBOL サブセットとオプション・モジュールとすべての古くなったエレメントの組み合わせに関するメッセージのリストです。 FIPS フラグに関して使用できる特定のオプションについては 46 ページの『FLAGSTD パラメーター』の『FLAGSTD パラメーター』を参照してください。
- **相互参照リスト**。OPTION(*XREF) を使用します。
- **組み込みエラー・リスト**。OPTION(*IMBEDERR) を使用します。
- **コンパイラー・メッセージ** (診断統計値を含む)。
- **コンパイル統計値**。
- **モジュール・オブジェクト**。CRTCBLMOD コマンドを使用します。
- **プログラム・オブジェクト**。CRTBNDCBL コマンドを使用します。

コンパイラー出力にこれらのタイプの一部を含めるかどうかは、PROCESS ステートメントで指定されたオプションによって、または CRTCBLMOD か CRTBNDCBL コマンドによって判別されます。印刷される診断メッセージのレベルは FLAG オプションによって異なります。 DBGVIEW オプションは、生成されたモジュール・オブジェクトまたはプログラム・オブジェクトに、どのような種類のデバッグ・データを含めるかを指示します。

リストの形式の指定

行の標識域 (7 桁目) に斜線 (/) があると、ソース・プログラム・リストのページ替えが行われます。この行の斜線 (/) の後にコメント・テキストを入力することもできます。斜線 (/) のコメント行は次のページの最初の行に出力されます。

プログラムに EJECT ステートメントを指定すると、コンパイラー・リストの次ページの最上部に次のソース・ステートメントが印刷されます。このステートメントは、区域 A または区域 B のどこにでも記述できますが、その行に他のステートメントを記述することはできません。

SKIP1/2/3 ステートメントは、ブランク行をコンパイラー・リストに挿入します。SKIP1/2/3 ステートメントは、区域 A または区域 B のどこにでも記述できますが、その行に他のステートメントを記述することはできません。

- SKIP1 は 1 つのブランク行を挿入します (2 行送り)。
- SKIP2 は 2 つのブランク行を挿入します (3 行送り)。
- SKIP3 は 3 つのブランク行を挿入します (4 行送り)。

上記の各 SKIP ステートメントごとに、1 行、2 行、または 3 行が一回ずつ挿入されます。

TITLE ステートメントは、指示されたそれぞれのページに表題を付けます。

*CONTROL、*CBL、または COPY ステートメントを使用することによって、ILE COBOL ソース・ステートメントを選択的にリストに含めたり、リスト出力を抑止したりすることができます。

- *CONTROL NOSOURCE および *CBL NOSOURCE は、ソース・ステートメントのリスト出力を抑止します。
- *CONTROL SOURCE および *CBL SOURCE は、ソース・ステートメントのリスト出力を続行します。
- SUPPRESS 句がある COPY ステートメントは、コピーされたステートメントのリストを抑止します。その間、このステートメントによって *CONTROL または *CBL ステートメントは指定変更されます。コピーされたメンバーに *CONTROL または *CBL ステートメントが含まれている場合、その COPY メンバーが処理されるとその最後のステートメントが実行されます。

EJECT、SKIP1/2/3、*CONTROL、*CBL、COPY、および TITLE ステートメントについては、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

時刻区切り文字

ジョブ関連のコマンド (CHGJOB など) の TIMSEP パラメーターは、コンパイラー・リストに表示されるタイム・スタンプで使用する時刻区切り文字を指定します。 TIMSEP 値が指定されていない場合、デフォルトで、システム値 QTIMSEP が使用されます。

SEU を使用したコンパイラー・リストのブラウズ

原始ステートメント入力ユーティリティー (SEU) により、出力待ち行列のコンパイラー・リストをブラウズすることができます。ソース・コードに必要な変更を加えながら、前のコンパイルの結果を見直すことができます。

クライアント・ツール入門については、『クライアント製品のアプリケーション開発ツールの使用』を参照してください。

コンパイラー・リストをブラウズする際には、エラーを走査して、エラーがあるソース・ステートメントを訂正することができます。エラーを走査するには、SEU コマンド行で `F *ERR` と入力します。

コンパイラー・リストのブラウズについては、「*AS/400 適用業務開発ツールセット AS/400 用 原始ステートメント入力ユーティリティー 使用者の手引きと参照*」を参照してください。

プログラムとリストのサンプル

次のリスト・サンプルは、プログラム・サンプルに対して作成されるコンパイラー・オプションとソース・リストを示しています。図については、それに続く本文で参照されています。これらの参照は、反転文字 (たとえば **Z**) で参照先が示されています。本文の中の反転文字は、図の中にある文字と対応しています。

コマンドの要約

この要約は、コンパイルの結果作成され、CRTCBLMOD または CRTBNDCBL コマンドで指定されたすべてのオプションのリストを示します。ユーザー定義オプションの詳細については 29 ページの『COBOL モジュールの作成 (CRTCBLMOD) コマンドの使用』を参照してください。

```
 5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/EXTLFL     ISERIES1   06/02/15 13:11:39      ページ  1
コマンド . . . . . . . . . . . . . . . :   CRTCBLMOD
実際の値 :
  モジュール . . . . . . . . . . . . . :   EXTLFL
  ライブラリー . . . . . . . . . . . . :   CBLGUIDE
  ソース・ファイル . . . . . . . . . . :   QCBLLESRC
  ライブラリー . . . . . . . . . . . . :   CBLGUIDE
    CCSID  . . . . . . . . . . . . . . :   37
  ソース・メンバー . . . . . . . . . . :   EXTLFL         02/03/05 10:50:50
  テキスト記述 . . . . . . . . . . . . :   *BLANK
コマンド・オプション :
  モジュール . . . . . . . . . . . . . :   EXTLFL
  ライブラリー . . . . . . . . . . . . :   CBLGUIDE
  ソース・ファイル . . . . . . . . . . :   QCBLLESRC
  ライブラリー . . . . . . . . . . . . :   CBLGUIDE
  ソース・メンバー . . . . . . . . . . :   EXTLFL
  出力 . . . . . . . . . . . . . . . . :   *PRINT
  生成重大度レベル . . . . . . . . . . :   30
  テキスト記述 . . . . . . . . . . . . :   *SRCMBRTXT
  コンパイラー・オプション . . . . . . :   *NONE
  変換オプション . . . . . . . . . . . :   *NONE
  メッセージ限界 :
    メッセージの数 . . . . . . . . . . :   *NOMAX
    メッセージ限界重大度 . . . . . . . :   30
  デバッグ・ビュー・オプション:
    デバッグ・ビュー . . . . . . . . . :   *STMT
    リスト・ビューの圧縮 . . . . . . . :   *NOCOMPRESSDBG
  最適化レベル . . . . . . . . . . . . :   *NONE
  FIPS フラグ付け. . . . . . . . . . . :   *NOFIPS *NOOBSOLETE
  拡張表示オプション . . . . . . . . . :   *NONE
  フラグ重大度 . . . . . . . . . . . . :   0
  モジュールの置き換え . . . . . . . . :   *NO
  権限 . . . . . . . . . . . . . . . . :   *LIBCRTAUT
  リンク・リテラル . . . . . . . . . . :   *PGM
  ターゲット・リリース . . . . . . . . :   *CURRENT
  ソート順序 . . . . . . . . . . . . . :   *HEX
    ライブラリー . . . . . . . . . . . :
  言語 ID . . . . . . . . . . . . . . . :   *JOBRUN
  パフォーマンス収集使用可能 :
    収集レベル . . . . . . . . . . . . :   *PEP
  プロファイル・データ . . . . . . . . :   *NOCOL
  コード化文字セット ID. . . . . . . . :   *JOBRUN
  演算モード . . . . . . . . . . . . . :   *NOEXTEND
  埋め込み文字 :
    単一バイト文字を国別文字へ . . . . :   NX"0020"
    2 バイト文字を国別文字へ . . . . . :   NX"3000"
    国別文字を国別文字へ . . . . . . . :   NX"3000"
  ディレクトリー組み込み . . . . . . . :   *NONE
  プログラム情報生成 . . . . . . . . . :   *NO
  コンパイラー . . . . . . . . . . . . :   IBM ILE COBOL
```

図 10. CRTCBLMOD コマンド要約のリスト

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/SAMPLE      ISERIES1   06/02/15 11:18:21        ページ    1
コマンド . . . . . . . . . . . . . . . :  CRTBNDCBL
実際の値 :
  プログラム . . . . . . . . . . . . . :  SAMPLE
  ライブラリー . . . . . . . . . . . . :  CBLGUIDE
  ソース・ファイル . . . . . . . . . . :  QCBLLESRC
  ライブラリー . . . . . . . . . . . . :  CBLGUIDE
    CCSID  . . . . . . . . . . . . . . :  37
  ソース・メンバー . . . . . . . . . . :  SAMPLE         02/03/05 14:13:55
  テキスト記述 . . . . . . . . . . . . :  *BLANK
コマンド・オプション :
  プログラム . . . . . . . . . . . . . :  SAMPLE
  ライブラリー . . . . . . . . . . . . :  CBLGUIDE
  ソース・ファイル . . . . . . . . . . :  QCBLLESRC
  ライブラリー . . . . . . . . . . . . :  CBLGUIDE
  ソース・メンバー . . . . . . . . . . :  SAMPLE
  出力 . . . . . . . . . . . . . . . . :  *PRINT
  生成重大度レベル . . . . . . . . . . :  30
  テキスト記述 . . . . . . . . . . . . :  *SRCMBRTXT
  コンパイラー・オプション . . . . . . :  *IMBEDERR
  変換オプション . . . . . . . . . . . :  *NONE
  メッセージ限界 :
    メッセージの数 . . . . . . . . . . :  *NOMAX
    メッセージ限界重大度 . . . . . . . :  30
    メッセージ限界重大度 . . . . . . . :  30
  デバッグ・ビュー・オプション:
    デバッグ・ビュー . . . . . . . . . :  *STMT
    リスト・ビューの圧縮 . . . . . . . :  *NOCOMPRESSDBG
  最適化レベル . . . . . . . . . . . . :  *NONE
  FIPS フラグ付け. . . . . . . . . . . :  *NOFIPS *NOOBSOLETE
  拡張表示オプション . . . . . . . . . :  *NONE
  フラグ重大度 . . . . . . . . . . . . :  0
  プログラムの置き換え . . . . . . . . :  *YES
  単純プログラム . . . . . . . . . . . :  *YES
  権限 . . . . . . . . . . . . . . . . :  *LIBCRTAUT
  リンク・リテラル . . . . . . . . . . :  *PGM
  ターゲット・リリース . . . . . . . . :  *CURRENT
  ユーザー・プロファイル . . . . . . . :  *USER
  ソート順序 . . . . . . . . . . . . . :  *HEX
    ライブラリー . . . . . . . . . . . :
  言語 ID . . . . . . . . . . . . . . . :  *JOBRUN
  パフォーマンス収集使用可能 :
    収集レベル . . . . . . . . . . . . :  *PEP
  バインディング・ディレクトリー . . . :  *NONE
    ライブラリー . . . . . . . . . . . :
  活動化グループ . . . . . . . . . . . :  QILE
  プロファイル・データ . . . . . . . . :  *NOCOL
  コード化文字セット ID. . . . . . . . :  *JOBRUN
  演算モード . . . . . . . . . . . . . :  *NOEXTEND
  埋め込み文字 :
    単一バイト文字を国別文字へ . . . . :  NX"0020"
    2 バイト文字を国別文字へ . . . . . :  NX"3000"
    国別文字を国別文字へ . . . . . . . :  NX"3000"
  ディレクトリー組み込み . . . . . . . :  *NONE
  プログラム情報生成 . . . . . . . . . :  *NO
  コンパイラー . . . . . . . . . . . . :  IBM ILE COBOL
```

図 11. CRTBNDCBL コマンド要約のリスト

有効なコンパイラー・オプションの識別

PROCESS ステートメントを指定すると、それが最初に印刷されます。 77 ページの図 12 は、プログラムのサンプルのコンパイルで有効になっているすべてのオプションのリストです。 CRTCBLMOD コマンドに指定したオプションは、PROCESS ステートメントによって修正されています。 OPTIONS パラメーターが指定されるとき、コンパイラー・オプションのリストはすべてのコンパイラー出力の最初になります。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/SAMPLE         ISERIES1   06/02/15 11:18:21      ページ  2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
          000100 PROCESS OPTIONS, SOURCE, VBSUM, MAP,
          000200         FLAG(00), MINIMUM, OBSOLETE, XREF
                    有効となっている COBOL コンパイラー・オプション
                         SOURCE
                         XREF
                         GEN
                         NOSEQUENCE
                         VBSUM
                         NONUMBER
                         MAP
                         OPTIONS
                         QUOTE
                         NOSECLVL
                         PRTCORR
                         MONOPRC
                         RANGE
                         NOUNREF
                         NOSYNC
                         NOCRTF
                         NODUPKEYCHK
                         NOINZDLT
                         NOBLK
                         STDINZ
                         NODDSFILLER
                         IMBEDERR
                         STDTRUNC
                         NOCHGPOSSGN
                         NOEVENTF
                         MONOPIC
                         NONATIONAL
                         NOLSPTRALIGN
                         NOCOMPASBIN
                         OUTPUT
                         GENLVL(30)
                         NOOPTIMIZE
                         MINIMUM
                         OBSOLETE
                         DFRWRT
                         UNDSPCHR
                         ACCUPDALL
                         FLAG(0)
                         LINKPGM
                         SRTSEQ(*HEX      )
                         LANGID(*JOBRUN  )
                         ENBPFRCOL(PEP)
                         PRFDTA(NOCOL)
                         CCSID(JOBRUN CCSID CCSID)
                         DATTIM(1900 40)
                         THREAD(NOTHREAD)
                         ARITHMETIC(NOEXTEND)
                         NTLPADCHAR(NX"0020" NX"3000" NX"3000")
                         OPTVALUE(NOOPT)
                         NOGRAPHIC
                    有効となっている COBOL 変換オプション
                         NOVARCHAR
                         NODATETIME
                         NOCVTPICXGRAPHIC
                         NOFLOAT
                         NODATE
                         NOTIME
                         NOTIMESTAMP
                         NOCVTTODATE
                         NOCVTPICNGRAPHIC
```

図 12. 有効なオプションのリスト

ソース・リスト

図 13 にソース・リストを示します。ソース・プログラムのステートメントは、REPLACE ステートメントで識別されるプログラム・ソース・テキストを除き、実行要求されたとおりに正確にリストされます。置換テキストはソース・リストに表示されます。 PROGRAM-ID 段落のリストのページの後、すべてのコンパイラー出力ページのシステム名の前のヘッディングのところには、リスト出力されるプログ

ラム ID 名が示されます。

```
 5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/SAMPLE          ISERIES1   06/02/15 11:18:21      ページ  4
  STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    A  B      C                                                                              D      E          F
     1     000300 IDENTIFICATION DIVISION.
     2     000500 PROGRAM-ID.     SAMPLE.
     3     000600   AUTHOR.       PROGRAMMER NAME.
     4     000700   INSTALLATION. COBOL DEVELOPMENT CENTRE.
     5     000800   DATE-WRITTEN. 02/24/94.
     6     000900   DATE-COMPILED. 02/02/05 11:18:21
     7     001100 ENVIRONMENT DIVISION.
     8     001300 CONFIGURATION SECTION.
     9     001400 SOURCE-COMPUTER. IBM-ISERIES.
    10     001500 OBJECT-COMPUTER. IBM-ISERIES.
    11     001700 INPUT-OUTPUT SECTION.
    12     001800 FILE-CONTROL.
    13     001900     SELECT FILE-1 ASSIGN TO DISK-SAMPLE.
    15     002100 DATA DIVISION.
    16     002300 FILE SECTION.
    17     002400 FD  FILE-1
           002500     LABEL RECORDS ARE STANDARD
*==>                       a
*=a> LNC0848   0  LABEL 文節は構文検査されて無視される。 G
           002600     RECORD CONTAINS 20 CHARACTERS
           002700     DATA RECORD IS RECORD-1.
*==>                       a
*=a> LNC0848   0  DATA RECORDS 文節は構文検査されて無視される。
    18     002800 01  RECORD-1.
    19     002900   02 FIELD-A      PIC X(20).
    20     003100 WORKING-STORAGE SECTION.
    21     003200 01  SUBSCRIPT-TYPE   TYPEDEF   PIC S9(2) COMP-3.
    22     003300 01  FILLER.
    23     003400   05 KOUNT        TYPE SUBSCRIPT-TYPE.
    24     003500   05 LETTERS      PIC X(26) VALUE "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
    25     003600   05 ALPHA REDEFINES LETTERS
           003700                   PIC X(1) OCCURS 26 TIMES.
    26     003800   05 NUMBR        TYPE SUBSCRIPT-TYPE.
    27     003900   05 DEPENDENTS   PIC X(26) VALUE "01234012340123401234012340".
    28     004000   05 DEPEND REDEFINES DEPENDENTS
           004100                   PIC X(1)  OCCURS 26 TIMES.
           004200 COPY WRKRCD.
    29    +000100 01  WORK-RECORD.                                                          WRKRCD
    30    +000200   05 NAME-FIELD   PIC X(1).                                              WRKRCD
    31    +000300   05 FILLER       PIC X(1)  VALUE SPACE.                                 WRKRCD
    32    +000400   05 RECORD-NO    PIC S9(3).                                             WRKRCD
    33    +000500   05 FILLER       PIC X(1)  VALUE SPACE.                                 WRKRCD
    34    +000600   05 LOCATION     PIC A(3)  VALUE "NYC".                                 WRKRCD
    35    +000700   05 FILLER       PIC X(1)  VALUE SPACE.                                 WRKRCD
```

図 13. ILE COBOL ソース・リストの例 (1/2)

```
 5722WDS V5R4M0 060210 LN  IBM ILE COBOL          CBLGUIDE/SAMPLE       ISERIES1   06/02/15 11:18:21        ページ      5
  STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
    36    +000800   05 NO-OF-DEPENDENTS                                                    WRKRCD
          +000900                 PIC X(2).                                               WRKRCD
    37    +001000   05 FILLER       PIC X(7)  VALUE SPACES.                               WRKRCD
    38     004300 77  WORKPTR USAGE POINTER.
           004500*******************************************************
           004600* THE FOLLOWING PARAGRAPH OPENS THE OUTPUT FILE TO  *
           004700* BE CREATED AND INITIALIZES COUNTERS               *
           004800*******************************************************
    39     004900 PROCEDURE DIVISION.
           005100 STEP-1.
    40     005200     OPEN OUTPUT FILE-1.
    41     005300     MOVE ZERO TO KOUNT, NUMBR.
           005500*******************************************************
           005600* THE FOLLOWING 3 PARAGRAPHS CREATE INTERNALLY THE  *
           005700* RECORDS TO BE CONTAINED IN THE FILE, WRITES THEM  *
           005800* ON THE DISK, AND DISPLAYS THEM                    *
           005900*******************************************************
           006000 STEP-2.
    42     006100     ADD 1 TO KOUNT, NUMBR.
    43     006200     MOVE ALPHA  (KOUNT) TO NAME-FIELD.
    44     006300     MOVE DEPEND (KOUNT) TO NO-OF-DEPENDENTS.
    45     006400     MOVE NUMBR          TO RECORD-NO.
           006600 STEP-3.
    46     006700     DISPLAY WORK-RECORD.
    47     006800     WRITE RECORD-1 FROM WORK-RECORD.
           007000 STEP-4.
    48     007100     PERFORM STEP-2 THRU STEP-3 UNTIL KOUNT IS EQUAL TO 26.
           007300*******************************************************
           007400* THE FOLLOWING PARAGRAPH CLOSES FILE OPENED FOR    *
           007500* OUTPUT AND RE-OPENS IT FOR INPUT                  *
           007600*******************************************************
           007700 STEP-5.
    49     007800     CLOSE FILE-1.
    50     007900     OPEN INPUT FILE-1.
           008100*******************************************************
           008200* THE FOLLOWING PARAGRAPHS READ BACK THE FILE AND   *
           008300* SINGLE OUT EMPLOYEES WITH NO DEPENDENTS           *
           008400*******************************************************
           008500 STEP-6.
    51     008600     READ FILE-1 RECORD INTO WORK-RECORD
    52     008700       AT END GO TO STEP-8.
           008900 STEP-7.
    53     009000     IF NO-OF-DEPENDENTS IS EQUAL TO "0"
    54     009100       MOVE "Z" TO NO-OF-DEPENDENTS.
    55     009200     GO TO STEP-6.
           009400 STEP-8.
    56     009500     CLOSE FILE-1.
    57     009600     STOP RUN.
 *==>                        a
 *=a> LNC0650   0  ファイル FILE-1 のブロック化 / ブロック解除はコンパイラー生成コードによって行われる。
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 13. ILE COBOL ソース・リストの例 (2/2)

78 ページの図 13 には、以下のフィールドが示されています。

A *コンパイラーが生成したステートメント番号:* ソース・プログラム・リストの左に表示されます。これらの番号は、FIPS リストを除くすべてのコンパイラー出力リストで参照されます。 1 つのステートメントが複数行にわたることもあり、1 行に複数のステートメントが含まれることもあります。入力ソース・メンバーに一連の ILE COBOL ソース・プログラムが存在する場合、ステートメント番号は新しいコンパイル単位ごとに 1 にリセットされます。ステートメント番号は、1 つまたは複数のネストされた COBOL プログラムが含まれている単一のコンパイル単位ではリセットされません。

B *プログラム・ネスト・レベル:* このフィールドに表示される数字は、プログラムのネストの程度を示します。

C *参照番号:* ソース・ステートメントの左に表示されます。このフィールドと列見出しに表示される番号 (このリストでは SEQNBR として表示) は、CRTCBLMOD または CRTBNDCBL コマンド、あるいは PROCESS ステー

トメントで指定されたオプションによって決定されます。次の表のとおりです。

オプション	見出し	出所
NONUMBER	SEQNBR	ソース・ファイル・シーケンス番号
NUMBER	NUMBER	ユーザー提供のシーケンス番号
LINENUMBER	LINNBR	コンパイラー生成のシーケンス番号

D *シーケンス・エラー標識欄:* この欄の S は、その行がシーケンスから外れていることを示します。参照番号フィールドの順序検査は、SEQUENCE オプションが指定される場合に限り実行されます。

E *コピー名:* コピー名が ILE COBOL COPY ステートメントで指定されている場合に、その COPY ステートメントによってソース・プログラムに含められたすべてのレコードについて、コピー名がここに示されます。DDS-ALL-FORMATS 句が使用される場合、<--ALL-FMTS という名前が COPYNAME の下に表示されます。

F *変更 / 日付フィールド:* ここには、行が最後に修正された日付が示されます。

G *組み込みエラー:* 第 1 レベルのエラー・メッセージは、リスト中のエラーが発生した行の後に示されます。エラーの原因となる文節、ステートメント、または句が示されます。

verb の使用カウントのリスト

図 14 に、ソース・プログラム内で使用されたすべての verb で構成されるアルファベット順のリストを示します。各 verb が使用された回数も示されます。このリストは、VBSUM オプションが指定される場合に作成されます。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL              CBLGUIDE/SAMPLE         ISERIES1   06/02/15 11:18:21       Page     7
                      動 詞 使 用 状 況 (カウント別)
動詞                   カウント
ADD                         1
CLOSE                       2
DISPLAY                     1
GOTO                        2
IF                          1
MOVE                        5
OPEN                        2
PERFORM                     1
READ                        1
STOP                        1
WRITE                       1
```

図 14. verb の使用カウントのリスト

データ部マップ

データ部 (DATA DIVISION) マップのリストは、MAP オプションが指定された場合に作成されます。このリストには、ILE COBOL ソース・プログラム内の名前に関する情報が含まれています。ファイル・セクション (FILE SECTION) および作業用記憶域セクション (WORKING-STORAGE SECTION) に必要な最小バイト数が、データ部マップの終わりに表示されます。

```
 5722WDS V5R4M0  060210 LN  IBM ILE COBOL              CBLGUIDE/SAMPLE         ISERIES1   06/02/15 11:18:21       ページ    8
                                 デ ー タ 部 マ ッ プ
  STMT LVL  ソース名                         セクション DISP    長さ   タイプ   属性
   H    I      J                                K        L       M      N       O
    17  FD  FILE-1                           FS                                 DEVICE DISK, ORGANIZATION SEQUENTIAL,
                                                                                ACCESS SEQUENTIAL, RECORD CONTAINS 20
                                                                                CHARACTERS
    18  01  RECORD-1                         FS  00000000       20  GROUP
    19  02  FIELD-A                          FS  00000000       20  AN
    21  01  SUBSCRIPT-TYPE                   WS  00000000        2  PACKED      TYPEDEF
    22  01  FILLER                           WS  00000000       56  GROUP
    23  05  KOUNT                            WS  00000000        2  PACKED      TYPE SUBSCRIPT-TYPE
    24  05  LETTERS                          WS  00000002       26  AN          VALUE
    25  05  ALPHA                            WS  00000002        1  AN          REDEFINES LETTERS, DIMENSION(26)
    26  05  NUMBR                            WS  00000028        2  PACKED      TYPE SUBSCRIPT-TYPE
    27  05  DEPENDENTS                       WS  00000030       26  AN          VALUE
    28  05  DEPEND                           WS  00000030        1  AN          REDEFINES DEPENDENTS, DIMENSION(26)
    29  01  WORK-RECORD                      WS  00000000       19  GROUP
    30  05  NAME-FIELD                       WS  00000000        1  AN
    31  05  FILLER                           WS  00000001        1  AN          VALUE
    32  05  RECORD-NO                        WS  00000002        3  ZONED
    33  05  FILLER                           WS  00000005        1  AN          VALUE
    34  05  LOCATION                         WS  00000006        3  A           VALUE
    35  05  FILLER                           WS  00000009        1  AN          VALUE
    36  05  NO-OF-DEPENDENTS                 WS  00000010        2  AN
    37  05  FILLER                           WS  00000012        7  AN          VALUE
    38  77  WORKPTR                          WS  00000000       16  POINTR
FILE SECTION は少なくとも 20 バイトの記憶域を使用します。
WORKING-STORAGE SECTION は少なくとも 91 バイトの記憶域を使用します。
                 * * * * *   デ ー タ 部 マ ッ プ の 終 わ り   * * * * *
```

図 15. データ部マップ

データ部マップには、次のフィールドが示されます。

H *ステートメント番号:* データ項目が定義されたコンパイラー生成のステートメント番号のリストが、データ部マップ内にある各データ項目ごとに示されます。

I *データ項目のレベル:* ここには、ソース・プログラムで指定されているデータ項目のレベル番号が示されます。索引名は、レベル番号内の *IX*、および SECTION、DISP、LENGTH、TYPE フィールド内のブランク・フィールドによって識別されます。

J *ソース名:* ここには、ソース・プログラムで指定されているデータ名が示されます。

K *セクション:* ここには、項目が定義されたセクションが、次のコードを使用して表示されます。

```
FS  File Section
WS  Working-Storage Section
LO  Local-Storage Section
LS  Linkage Section
SM  Sort/Merge Section
SR  Special Register.
```

L *変位:* ここには、レベル 01 グループ項目からの項目のオフセット (バイト単位) が表示されます。

M *長さ:* ここには、項目の 10 進数での長さ (バイト単位) が示されます。

N *タイプ:* ここには、項目のデータ・クラス・タイプが、次のコードを使用して表示されます。

コード	データ・クラス・タイプ
GROUP	グループ項目

コード	データ・クラス・タイプ
A	英字
AN	英数字
ANE	英数字編集
DT	日付
TM	時刻
TMS	タイム・スタンプ
INDEX	指標データ項目 (USAGE INDEX)
BOOLN	ブール
ZONED	ゾーン 10 進数 (外部 10 進数)
PACKED	パック 10 進数 (内部 10 進数) (USAGE COMP、COMP-3、または PACKED-DECIMAL)
BINARY	2 進数 (USAGE COMP-4 または BINARY) ネイティブ 2 進数 (USAGE COMP-5)
FLOAT	内部浮動小数点 (USAGE COMP-1 または COMP-2)
EFLOAT	外部浮動小数点 (USAGE DISPLAY)
NE	数字編集
POINTR	ポインター・データ項目 (USAGE POINTER)
PRCPTR	プロシージャー・ポインター・データ項目 (USAGE PROCEDURE-POINTER)
G	DBCS
GE	DBCS 編集

0 *属性:* ここには、項目の属性が以下のように示されます。

- ファイルの場合、次の情報が提供される可能性があります。
  - DEVICE タイプ
  - ORGANIZATION 情報
  - ACCESS モード
  - BLOCK CONTAINS 情報
  - RECORD CONTAINS 情報
  - LABEL 情報
  - RERUN が必要
  - SAME AREA が必要
  - CODE-SET が必要
  - SAME RECORD AREA が必要
  - LINAGE が必要
  - NULL CAPABLE が必要
- データ項目の場合は、属性は次の情報が項目に指定されたかどうかを示します。
  - REDEFINES 情報
  - VALUE
  - JUSTIFIED
  - SYNCHRONIZED
  - BLANK WHEN ZERO
  - SIGN IS LEADING

  - SIGN IS LEADING SEPARATE CHARACTER
  - SIGN IS SEPARATE CHARACTER
  - INDICATORS
  - SIZE
  - TYPEDEF
  - TYPE 文節情報
  - LOCALE 情報
- テーブル項目の場合は、項目の大きさが DIMENSION (nn) の形式でここに示されます。各ディメンションごとに、最大 OCCURS 値が示されます。ディメンションが変数である場合、そのことが示され、最低および最高の OCCURS 値が示されます。

FIPS メッセージ

84 ページの図 16 に示す FIPS メッセージのリストは、FLAGSTD パラメーターが指定された場合に出力されます。 FIPS フラグのオプションの指定についての詳細は 46 ページの『FLAGSTD パラメーター』を参照してください。リストに示されるのは、要求された FIPS サブセットと、オプショナル・モジュールまたは古くなったエレメント (あるいはその両方) のメッセージだけです。

注: メッセージが出されるたびに、シーケンス番号と列番号が示されます。

```
5722WDS V5R4M0   060210 LN  IBM ILE COBOL              CBLGUIDE/SAMPLE         ISERIES1   06/02/15 11:18:21       ページ    9
                        C O B O L   F I P S   メ ッ セ ー ジ
 FIPS-ID    説明およびフラグの付いた順序番号 Q
   P
LNC8100     次の項目が旧式の言語要素である。
LNC8102     AUTHOR 段落。
                000600    10
LNC8103     DATE-COMPILED 段落。
                000900    10
LNC8104     INSTALLATION 段落。
                000700    10
LNC8105     DATE-WRITTEN 段落。
                000800    10
LNC8117     LABEL RECORDS 文節。
                002500    12
LNC8177     DATA RECORDS 文節。
                002700    12
LNC8200     次の標準に非準拠の項目は FIPS 中間レベルまたはそれ以上のみに有効である。 R
LNC8201     COPY ステートメント。
                004200     8
LNC8500     次の例外的な標準に非準拠の項目は IBM 定義または IBM 拡張機能である。
LNC8504     ASSIGN 文節の割り当て名。
                001900    36
LNC8518     USAGE IS COMPUTATIONAL-3。
                003200    49
LNC8520     USAGE IS POINTER または PROCEDURE-POINTER。
                004300    26
LNC8561     COPY ステートメントの省略時のライブラリーと見なされる。
                004200     8
LNC8572     SKIP1/2/3 ステートメント。
                000400    13
                001000    13
                001200    13
                001600    13
                002000    13
                002200    13
                003000    13
                004400    13
                005000    13
                005400    13
                006500    13
                006900    13
                007200    13
                008000    13
                008800    13
                009300    13
LNC8616     TYPEDEF 文節。
                003200    29
LNC8617     TYPE 文節。
                003400    26
                003800    26
  30 FIPS 違反のフラグが付けられた。 S
              * * * * *  C O B O L   F I P S   メ ッ セ ー ジ の 終 わ り  * * * * *
```

図 16. FIPS メッセージ

FIPS メッセージは、次のフィールドから構成されます。

P *FIPS-ID:* このフィールドには、FIPS メッセージ番号が示されます。

Q *フラグ付き記述および参照番号:* このフィールドには、条件フラグの説明が示されます。その後に、その条件が見つかった位置のソース・プログラムからの参照番号のリストが示されます。

使用される参照番号のタイプ、およびヘッディングの中でのそれらの名前 (このリストの中の SEQUENCE NUMBERS として示されている) は、CRTCBLMOD コマンド、CRTBNDCBL コマンド、または PROCESS ステートメント内に指定されるオプションによって、以下の表のようにして決まるものです。

オプション	見出し
NONUMBER	DESCRIPTION AND SEQUENCE NUMBERS FLAGGED
NUMBER	DESCRIPTION AND USER-SUPPLIED NUMBERS FLAGGED

オプション	見出し
LINENUMBER	DESCRIPTION AND LINE NUMBERS FLAGGED

R *レベルごとにグループ化した項目:* これらのヘッディングは、レベルおよびカテゴリーによって FIPS メッセージを区分するものです。

S *FIPS 違反フラグ:* FIPS 違反フラグの合計数が、FIPS リストの終わりに表示されます。

相互参照リスト

図 17 に、XREF オプションが指定される場合に作成される相互参照リストを示します。これには、ソース・プログラム内のステートメント番号による、すべてのデータ参照、プロシージャー名参照、およびプログラム名参照のリストが示されます。

```
 5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/SAMPLE         ISERIES1   06/02/15 11:18:21       ページ   10
                            相  互  参  照  表
データ参照:
データ・タイプはデータ名定義の後の英字によって示されます。
これらの英字とその意味:
    E = EXTERNAL
    G = GLOBAL
    X = EXTERNAL および GLOBAL
データ名                          定義   参照 (* = 変更)
  T                               U      V
ALPHA                              25     43
DEPEND                             28     44
DEPENDENTS                         27     28
FIELD-A                            19
FILE-1                             17     40     49     50     51     56
KOUNT                              23     41*    42*    43     44     48
LETTERS                            24     25
LOCATION                           34
NAME-FIELD                         30     43*
NO-OF-DEPENDENTS                   36     44*    53     54*
NUMBR                              26     41*    42*    45
RECORD-NO                          32     45*
RECORD-1                           18     47*
WORK-RECORD                        29     46     47     51*
WORKPTR                            38
タイプ名                          定義   参照 (* = 変更)
SUBSCRIPT-TYPE                     21     23     26
プロシージャー参照:
コンテキストの使用はプロシージャー名参照に続く英字によって示されます。
これらの英字とその意味:
    A = ALTER (プロシージャー名)
    D = GO TO (プロシージャー名) DEPENDING ON
    E = (PERFORM) から (プロシージャー名) の範囲の終わり
    G = GO TO (プロシージャー名)
    P = PERFORM (プロシージャー名)
    T = (ALTER) TO PROCEED TO (プロシージャー名)
プロシージャー名                  定義   参照
STEP-1                             39
STEP-2                             41     48P
STEP-3                             45     48E
STEP-4                             47
STEP-5                             48
STEP-6                             50     55G
STEP-7                             52
STEP-8                             55     52G
プログラム参照:
外部プログラムのプログラム・タイプは、プログラム名定義内の語によって示されます。
これらの語とその意味:
    EPGM = 動的にリンクされるプログラム・オブジェクト
    BPRC = COBOL プログラム、あるいはバインドされる C 関数または RPG プログラム
    SYS  = システム・プログラム
プログラム名                      定義   参照
SAMPLE                             2
                 *****  相  互  参  照  表  の  終  わ  り  *****
```

図 17. 相互参照リスト

相互参照リストには以下のフィールドが示されます。

T *名前フィールド:* 参照されるデータ名、プロシージャー名、またはプログラム名のリストがここに示されます。名前はアルファベット順です。プログラム名はライブラリー名で修飾されていることもあります。

U *定義済みフィールド:* 名前がソース・プログラム内に定義されたステートメント番号がここに示されます。

V *参照フィールド:* すべてのステートメント番号が、ソース・プログラムで参照される名前と同じ順序で示されます。ステートメント番号の後ろに付いている * は、項目がそのステートメント内で修正されたことを示しています。

メッセージ

図 18 に、プログラムのコンパイル中に生成されるメッセージを示します。

```
 5722WDS V5R4M0  060210 LN  IBM ILE COBOL              CBLGUIDE/SAMPLE         ISERIES1   06/02/15 11:18:21       ページ   12
                                      メ ッ セ ー ジ
   STMT
  W                     Y                       X
*    17  MSGID: LNC0848  重大度 :   0  SEQNBR:  002500
         メッセージ. . . :   LABEL 文節は構文検査されて無視される。 Z
*    17  MSGID: LNC0848  重大度 :   0  SEQNBR:  002700
         メッセージ. . . :   DATA RECORDS 文節は構文検査されて無視される。
           ignored.
*    57  MSGID: LNC0650  重大度 :   0  SEQNBR:  009600
         メッセージ  . . :   ファイル FILE-1 のブロック化 / ブロック解除は
           コンパイラー生成コードによって行われる。
                                           メッセージの要約
メッセージ合計 :  AA
   情報        (00-04) . . . . . . . . . :      3
   警告        (05-19) . . . . . . . . . :      0
   エラー      (20-29) . . . . . . . . . :      0
   重大        (30-39) . . . . . . . . . :      0
   打ち切り    (40-99) . . . . . . . . . :      0
   -------------------------------------------
   合計                                         3
                        * * * * *   メ  ッ  セ  ー  ジ  の  終  わ  り   * * * * *
統計 :  BB
   読み取ったソース・レコード数. . . . :     96
   読み取ったコピー・レコード数. . . . :     10
   処理したコピー・メンバー数. . . . . :      1
   シーケンス・エラー  . . . . . . . . :      0
   出された最高重大度メッセージ. . . . :      0
  LNC0901  0  11:18:23 の 06/02/17 にライブラリー CBLGUIDE にプログラム SAMPLE が作成された。
                      * * * * *   コ  ン  パ  イ  ル  の  終  わ  り   * * * * *
```

図 18. 診断メッセージ

表示されるフィールドは、次のとおりです。

W *ステートメント番号:* このフィールドに示されるのは、メッセージが出されたソース・プログラム中のステートメントに関連したコンパイラー生成のステートメント番号です。[1]

X *参照番号:* 参照番号がここに示されます。[1] このフィールドと列見出しに表示される番号 (このリストでは SEQNBR として表示) は、次の表にあるとおり、CRTCBLMOD または CRTBNDCBL コマンド、あるいは PROCESS ステートメントで指定されたオプションによって判別されます。

1. ステートメント番号および参照番号は、脱落した項目を参照する特定のメッセージには示されません。たとえば、PROGRAM-ID 段落が脱落している場合、メッセージ LNC0031 は、ステートメント番号および参照番号なしで、リストに表示されます。

	見出し	出所
NONUMBER	SEQNBR	ソース・ファイル・シーケンス番号
NUMBER	NUMBER	ユーザー提供のシーケンス番号
LINENUMBER	LINNBR	コンパイラー生成のシーケンス番号

コピー・ファイルからのレコードについてのメッセージが出された場合、番号の前に + が付きます。

Y *MSGID* および*重大度レベル:* これらのフィールドには、メッセージ番号とそれに関連した重大度レベルが表示されます。重大度レベルは、次のように定義されます。

00 通知
10 警告
20 エラー
30 重大エラー
40 リカバリー不能 (通常は、ユーザー・エラー)
50 リカバリー不能 (通常は、コンパイラー・エラー)

Z *メッセージ:* メッセージは、どのような条件かを識別し、コンパイラーが取るアクションを示します。

AA *メッセージ合計:* このフィールドには、メッセージの合計数および重大度レベルごとのメッセージ数が示されます。

ここに示されるメッセージの合計数は、コンパイラーによって各重大度レベルごとに生成されたメッセージ数であり、必ずしもリストに示される数と同じではありません。たとえば、FLAG(10) が指定されている場合、10 より低い重大度レベルのメッセージはリストに示されません。 しかし、メッセージが抑制されていない場合に印刷されるのと同じメッセージの数がカウントとして示されます。

BB *コンパイラー統計:* このフィールドは、読み取ったソース・レコード、読み取ったコピー・レコード、処理したコピー・メンバー、検出したシーケンス・エラー、および出された最高重大度メッセージの合計数を示します。

第 4 章 プログラム・オブジェクトの作成

ここでは、次のことがらについて説明します。

- 1 つまたは複数のモジュール・オブジェクトをバインドしてプログラム・オブジェクトを作成する方法
- ILE COBOL ソース・ステートメントからプログラム・オブジェクトを作成する方法
- CRTBNDCBL コマンドおよびそのパラメーター
- バインダー・リストの読み方
- 1 つまたは複数のモジュール・オブジェクト、サービス・プログラム、および ILE バインディング・ディレクトリーを使用してプログラム・オブジェクトを作成する方法

WebSphere Development Studio for i5/OS を使用してください。 これが推奨される方法であり、プログラム・オブジェクトの作成に関する資料は、その製品のオンライン・ヘルプにあります。

クライアント・ツール入門については、『クライアント製品のアプリケーション開発ツールの使用』を参照してください。

プログラム・オブジェクトの定義

プログラム・オブジェクトとは、タイプ *PGM の実行可能システム・オブジェクトのことです。 ILE COBOL の場合、プログラム・オブジェクトの名前は、最外部の COBOL ソース・プログラムの CRTBNDCBL コマンド、CRTPGM コマンド、または PROGRAM-ID 段落によって決定されます。 1 つまたは複数のモジュール・オブジェクトおよび参照サービス・プログラムからプログラム・オブジェクトを作成するプロセスは、**バインディング**といいます。 1 つまたは複数のモジュール・オブジェクトは、CRTCBLMOD コマンドによって作成されるか、または CRTBNDCBL コマンドによって一時的に作成されて、1 つまたは複数のバインド済みプログラム・オブジェクトになります。バインディングとは、CRTCBLMOD または CRTBNDCBL コマンドによって作成されるモジュール・オブジェクトを入手し、それらを結合して実行可能なバインド済みプログラム・オブジェクトまたはサービス・プログラムを作成するプロセスです。

プログラム・オブジェクトが作成されると、ILE ソース・デバッガーによって、デバッグ・データを含むモジュール・オブジェクト内の ILE プロシージャーだけをデバッグすることができます。デバッグ・データは、実行中のプログラム・オブジェクトのパフォーマンスに影響を与えません。デバッグ・データにより、生成されるプログラム・オブジェクトのサイズは大きくなります。

プログラム・オブジェクトは、動的プログラム呼び出しを使用することによって実行されます。プログラム・オブジェクトへのエントリー・ポイントは PEP です。

バインディング・プロセス

プログラム・オブジェクトとして実行時のパフォーマンスを向上させるバインディング・プロセスでは、プログラム・オブジェクトの一部としてバインドされたルーチンの静的プロシージャー呼び出しを使用することができます。動的プログラム呼び出しは、ルーチンにアクセスする必要はありません。異なる ILE HLL コンパイラーによって作成された個々のモジュール・オブジェクトを同じプログラム・オブジェクトにバインドすることができ、それによって、最も適した言語でルーチンをコーディングしてから、それを必要なプログラム・オブジェクトにバインドすることができるようになります。

以前にコンパイルされたモジュール・オブジェクトは、さまざまな順序でバインドして、新しい実行可能プログラム・オブジェクトを作成することができます。以前にコンパイルされたモジュール・オブジェクトを再使用して、元のソース・プログラムを再コンパイルせずに、新しい実行可能プログラム・オブジェクトを作成することができます。これによって、必要に応じてモジュール・オブジェクトを再使用できます。

モジュール・オブジェクトを変更するたびにプログラムを再作成するかわりに、サービス・プログラムを使用することができます。共通ルーチンはサービス・プログラムとして作成できます。ルーチンを変更してもそのインターフェースを変更しない場合、またはインターフェースに対して上方に互換性のある変更だけが行われる場合、サービス・プログラムを再作成することによってその変更を取り込むことができます。これらの共通ルーチンを使用するプログラム・オブジェクトおよびサービス・プログラムは、再作成する必要はありません。

バインディング・プロセスを使用して、プログラム・オブジェクトを作成できる 2 つのパスがあります。下記の図は、使用可能な 2 つのパスを示しています。

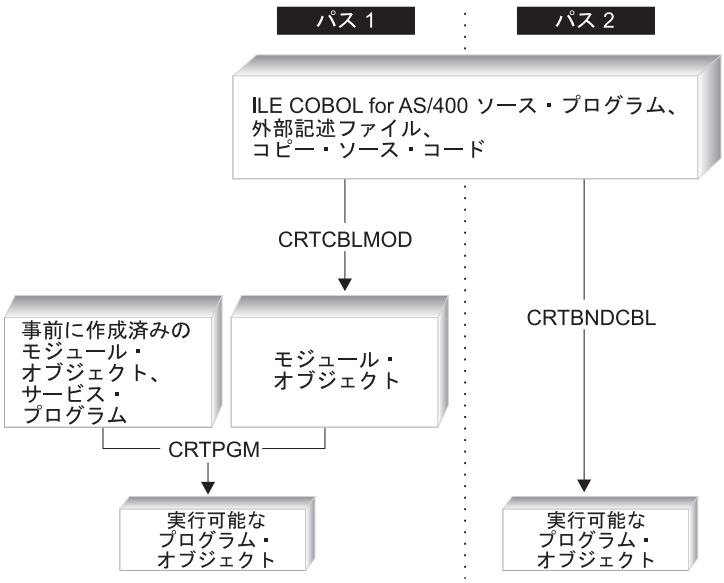


図 19. プログラム・オブジェクトを作成するための 2 つのパス

これらの 2 つのパスは両方ともバインディング・プロセスを使用します。プログラム作成 (CRTPGM) コマンドは、COBOL モジュールの作成 (CRTCBLMOD) コマンド、およびゼロまたはそれ以上のサービス・プログラムを使用して、ILE COBOL ソース・プログラムから作成されたモジュール・オブジェクトからプログラム・オブジェクトを作成します。

注: さらに、RPG モジュール作成 (CRTRPGMOD) コマンド、C モジュール作成 (CRTCMOD) コマンド、および CL モジュール作成 (CRTCLMOD) コマンドを使用して作成されたモジュール・オブジェクトも、プログラム作成 (CRTPGM) コマンドを使用してバインドできます。

バインド COBOL PGM の作成 (CRTBNDCBL) コマンドは、1 つまたは複数の ILE COBOL コンパイル単位から 1 つまたは複数の一時モジュール・オブジェクトを作成して、それから 1 つまたは複数のプログラム・オブジェクトを作成します。プログラム・オブジェクトが作成されると、CRTBNDCBL は作成したモジュール・オブジェクトを削除します。このコマンドは、COBOL モジュールの作成 (CRTCBLMOD) コマンドおよびプログラム作成 (CRTPGM) コマンドの 2 つを合わせた作業を単一のステップで実行します。前に作成されたモジュール・オブジェクトおよびサービス・プログラムは、バインディング・ディレクトリーを使用してバインドすることができます。ただし、このステップを使用してバインドされる入力ソース・メンバーは、PEP モジュールでなければなりません。

注: すべてのプログラム・オブジェクトは、1 つの PEP (および 1 つの UEP) しか認識しません。 CRTPGM を使用していくつかのモジュール・オブジェクトを 1 つにバインドしてプログラム・オブジェクトを作成し、それらの各モジュール・オブジェクトに 1 つずつ PEP がある場合、ENTMOD パラメーターで、どのモジュール・オブジェクトの PEP をプログラム・オブジェクトの PEP として使用するかを指定しなければなりません。また、モジュール・オブジェクトおよびサービス・プログラムが CRTPGM コマンドで指定される順序は、バインディング・プロセス中の記号の解決方法に影響を与えます。したがって、バインディングが実行される方法を理解することは重要です。バインディング・プロセスの詳細については、「*ILE 概念*」を参照してください。

バインディング・ディレクトリーには、ILE プログラムやサービス・プログラムの作成に必要になるモジュールの名前と ILE サービス・プログラムの名前が入っています。バインディング・ディレクトリーに含まれているモジュールおよびサービス・プログラムが使用されるのは、それらが現在未解決のインポート要求を満足できるエクスポートを提供している場合です。バインディング・ディレクトリーはシステム・オブジェクトであり、記号 *BNDDIR によりシステムに対して識別されます。

バインディング・ディレクトリーの指定はオプショナルです。バインディング・ディレクトリーを使用する理由は、便宜性とプログラム・サイズにあります。

- バインド・ディレクトリーは、独自の ILE プログラムやサービス・プログラムの作成時に場合によって必要となるモジュールやサービス・プログラムをパッケージにするのに便利な方法です。たとえば、1 つのバインディング・ディレクトリーには、数値計算の機能を提供するすべてのモジュールとサービス・プログラムを入れることができます。このような機能のいくつかを使用したい場合、使用す

るモジュールやサービス・プログラムをすべて指定するのではなく、その 1 つのバインディング・ディレクトリーだけを指定します。
- バインディング・ディレクトリーを使用する場合は、使用しないモジュールやサービス・プログラムを指定しないですむため、プログラムのサイズを小さくすることができます。

バインディング・ディレクトリーの項目については、ほとんど制約がありません。オブジェクトが存在していなくても、モジュールやサービス・プログラムの名前をバインディング・ディレクトリーに追加できます。

バインディング・ディレクトリーに関連して使用される CL コマンドのリストについては、「*ILE 概念*」を参照してください。 *BNDDIR オブジェクトには、以下のような特性があります。
- ILE プログラムやサービス・プログラムの作成に必要となる場合があるサービス・プログラムやモジュールの名前をまとめるのに便利です。
- バインディング・ディレクトリーの項目は名前だけなので、オブジェクト・リストはシステム上になくてもかまいません。
- 有効なライブラリー名は *LIBL または特定のライブラリーだけです。
- リスト中のオブジェクトの指定はオプショナルです。名前の指定されたオブジェクトが使用されるのは、未解決のインポートがあり、しかもそのオブジェクトがその未解決のインポート要求を満足できるエクスポートを提供している場合だけです。

プログラム作成 (CRTPGM) コマンドの使用

プログラム作成 (CRTPGM) コマンドは、以前に作成された 1 つまたは複数のモジュール・オブジェクト、および要求がある場合には 1 つまたは複数のサービス・プログラムから、プログラム・オブジェクトを作成します。 ILE モジュール作成コマンドの CRTCBLMOD、CRTCMOD、CRTRPGMOD、または CRTCLMOD のいずれかで作成されたモジュール・オブジェクトをバインドすることができます。

注: CRTPGM コマンドを使用するには、このコマンドを使用する権限がなければならず、必要なモジュールは、最初に CRTCBLMOD、CRTCMOD、CRTRPGMOD、または CRTCLMOD コマンドを使用して作成されていなければなりません。

CRTPGM コマンドを使用してプログラム・オブジェクトを作成する前に、以下のことを行ってください。
1. プログラム名を設定する。
2. バインドしてプログラム・オブジェクトにしたいモジュール・オブジェクト、および必要であればサービス・プログラムを指定する。
3. 使用する予定のバインディング・ディレクトリーを指定する。 ILE COBOL 実行時サービス・プログラムおよび ILE バインド可能 API の場合、CRTCBLMOD および CRTBNDCBL によって作成されたモジュール・オブジェクトからバインディング・ディレクトリーへの暗黙の参照が行われます。

4. どのモジュール・オブジェクトの PEP が作成中のプログラム・オブジェクト用の PEP として使用されるかを指定する。このモジュール・オブジェクトは、CRTPGM の ENTMOD パラメーターで指定します。

   ENTMOD パラメーターでモジュール・オブジェクトを明示的に指定する代わりに ENTMOD(*FIRST) を指定した場合は、作成中のプログラム・オブジェクトの PEP としてどのモジュール・オブジェクトのものを使用するかを決定する際に、バインディングが行われる順序が重要になります。 1 つの PEP しか使用できない場合でも、MODULE パラメーターに指定するモジュール・オブジェクト、またはバインディング・ディレクトリーによって見つかるモジュール・オブジェクトに、1 つまたは複数の PEP が含まれていることがあります。バインディングの順序は、記号解決などの他の理由からも重要です。バインディングの詳細については、「*ILE 概念*」を参照してください。
5. コマンドにおいて、プロシージャー名または変数名 (あるいはその両方) の重複が可能かどうかを設定する。

   複数の異なる方法で同じ変数名およびプロシージャー名をそれぞれ定義している複数のモジュール・オブジェクトを、1 つのプログラム・オブジェクトにバインドすることがあります。
6. プログラムが実行される活動化グループを指定する。活動化グループの説明については 235 ページの『活動化および活動化グループ』を参照してください。

CRTPGM コマンドを使用してプログラム・オブジェクトを作成するには、次のステップを実行してください。

1. CRTPGM コマンドを入力する。
2. コマンド・パラメーターに適切な値を入力する。

表 5 に、CRTPGM コマンド・パラメーターとそのデフォルト値を示します。
CRTPGM コマンドおよびそのパラメーターの詳細については、Web サイト
http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「プログラミング」カテゴリーの中の『*CL および API*』セクションを参照してください。

表 5. CRTPGM コマンドのパラメーターとそのデフォルト値

パラメーター・グループ	**パラメーター (デフォルト値)**
識別	PGM(*ライブラリー名/プログラム名*) MODULE(*PGM)
プログラム・アクセス	ENTMOD(*FIRST)
バインディング	BNDSRVPGM(*NONE) BNDDIR(*NONE)
実行時	ACTGRP(*NEW)

表 5. *CRTPGM* コマンドのパラメーターとそのデフォルト値 (続き)

パラメーター・グループ	パラメーター (デフォルト値)
その他	OPTION(*GEN *NODUPPROC *NODUPVAR *WARN *RSLVREF) DETAIL(*NONE) ALWUPD(*YES) ALWRINZ(*NO) REPLACE(*YES) AUT(*LIBCRTAUT) TEXT(*ENTMODTXT) TGTRLS(*CURRENT) USRPRF(*USER) ALWLIBUPD(*NO)

CRTPGM コマンドを入力すると、プログラム・オブジェクトは次のように作成されます。

1. リストに示されたモジュール・オブジェクトが、プログラム・オブジェクトになるものの中にコピーされる。
2. PEP を含むモジュール・オブジェクトが識別され、このモジュール内への最初のインポートが検索される。
3. モジュール・オブジェクトをリストの順序で調べていき、最初のインポートとモジュール・エクスポートとを突き合わせる。
4. 最初のモジュール・オブジェクトが返され、次のインポートが検索される。
5. 最初のモジュール・オブジェクトへのすべてのインポートが解決される。
6. 次のモジュール・オブジェクトが継続され、すべてのインポートが解決される。
7. すべてのインポートが解決されるまで、以降の各モジュール・オブジェクト内へのすべてのインポートが解決される。
8. OPTION(*RSLVREF) が指定されていて、インポートがエクスポートで解決できない場合、バインディング・プロセスは、プログラム・オブジェクトを作成せずに終了する。 OPTION(*UNRSLVREF) が指定される場合、すべてのインポートは必ずしも作成されるプログラム・オブジェクトのエクスポートに解決される必要はありません。プログラム・オブジェクトが実行時にこれらの未解決のインポートの 1 つを使用する場合、MCH4439 例外メッセージが出されます。
9. すべてのインポートが解決されると、バインディング・プロセスは完了し、プログラム・オブジェクトが作成されます。

複数のモジュールをバインドして 1 つのプログラム・オブジェクトにする例

この例は、CRTPGM コマンドを使用して、モジュール・オブジェクト A、B、および C をバインドして、ABC というプログラム・オブジェクトにする方法を示すものです。プログラム・オブジェクトの PEP および UEP を含むモジュール・オブジェクトは、ENTMOD パラメーターに指定されているモジュール・オブジェクトです。

複数のモジュールをバインドしてプログラム・オブジェクトにするには、すべての外部参照を CRTPGM コマンドで解決してください。 ILE COBOL 実行時関数への参照は、ILE COBOL モジュール・オブジェクトを含むプログラム・オブジェクトの中に自動的にバインドされる際に解決されます。

1. 複数のモジュール・オブジェクトをバインドしてプログラム・オブジェクトを作成するには、次のように入力します。

   ```
   CRTPGM PGM(ABC) MODULE(A B C) ENTMOD(*FIRST) DETAIL(*FULL)
   ```

   そして、実行キーを押します。

バインド COBOL PGM の作成 (CRTBNDCBL) コマンドの使用

バインド COBOL PGM の作成 (CRTBNDCBL) コマンドは、単一の ILE COBOL ソース・ファイル・メンバーから 1 つまたは複数のプログラム・オブジェクトを 1 つのステップで作成します。このコマンドは ILE COBOL コンパイラーを開始して一時モジュール・オブジェクトを作成してから、それらをバインドしてタイプ *PGM の 1 つまたは複数のプログラム・オブジェクトを作成します。

CRTPGM コマンドとは異なり、CRTBNDCBL コマンドを使用すると、CRTCBLMOD コマンドによる 1 つまたは複数のモジュール・オブジェクトを作成する別個の先行ステップは必要ありません。 CRTBNDCBL コマンドは、ソース・ファイル・メンバーから一時モジュール・オブジェクトを作成してから、1 つまたは複数のプログラム・オブジェクトを作成することによって、COBOL モジュールの作成 (CRTCBLMOD) コマンドとプログラム作成 (CRTPGM) コマンドを組み合わせたタスクを実行します。プログラム・オブジェクトを作成すると、CRTBNDCBL は作成したモジュール・オブジェクトを削除します。

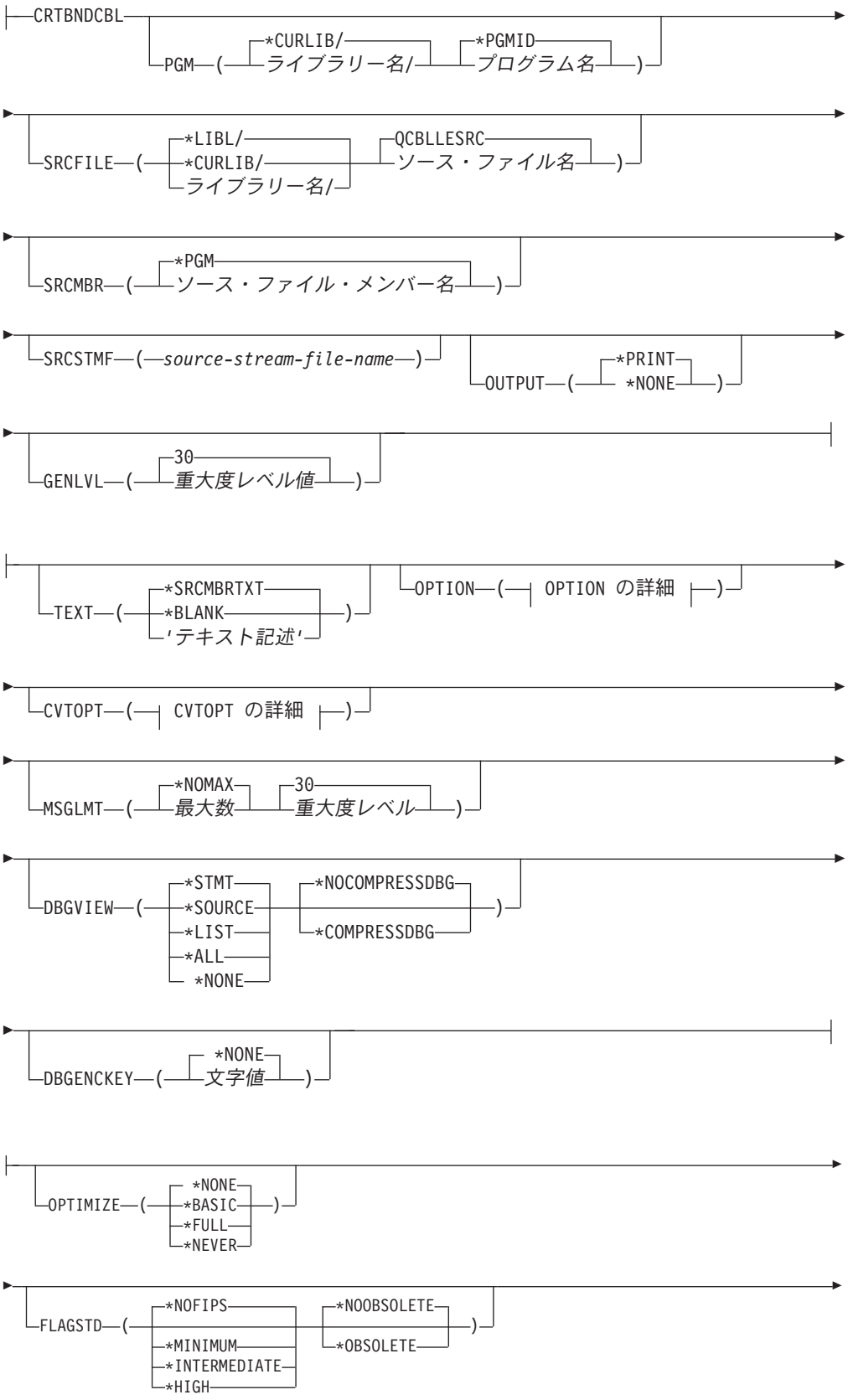
注: モジュール・オブジェクトを保存したい場合、CRTBNDCBL の代わりに CRTCBLMOD を使用してください。 CRTCBLMOD を使用する場合、CRTPGM を使用してモジュール・オブジェクトをバインドし、1 つまたは複数のプログラム・オブジェクトにする必要があります。また、CRTBNDCBL によって暗黙指定されたもの以外の CRTPGM のオプションを使用する場合には、CRTCBLMOD および CRTPGM を使用してください。

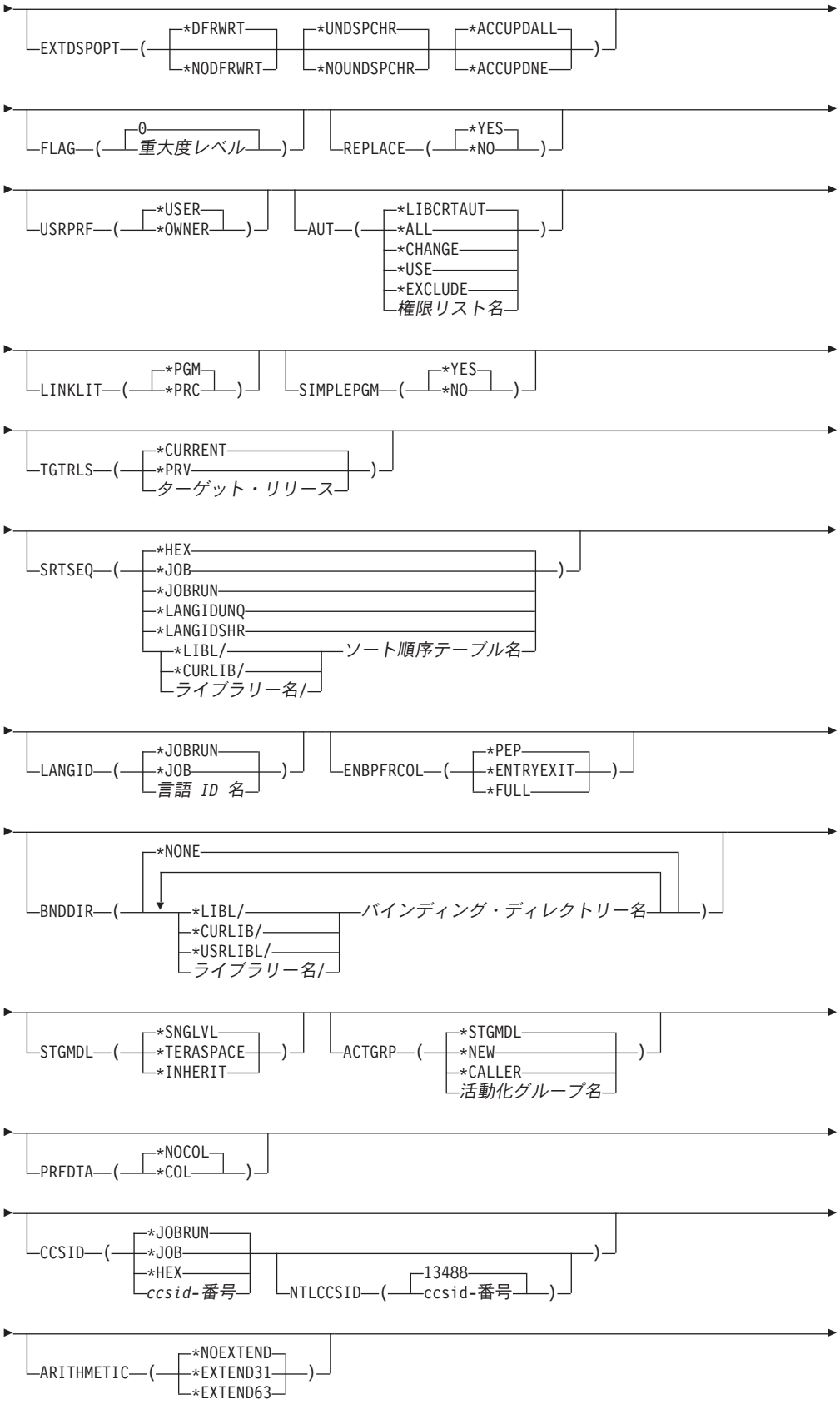
プロンプト画面での CRTBNDCBL コマンドの使用

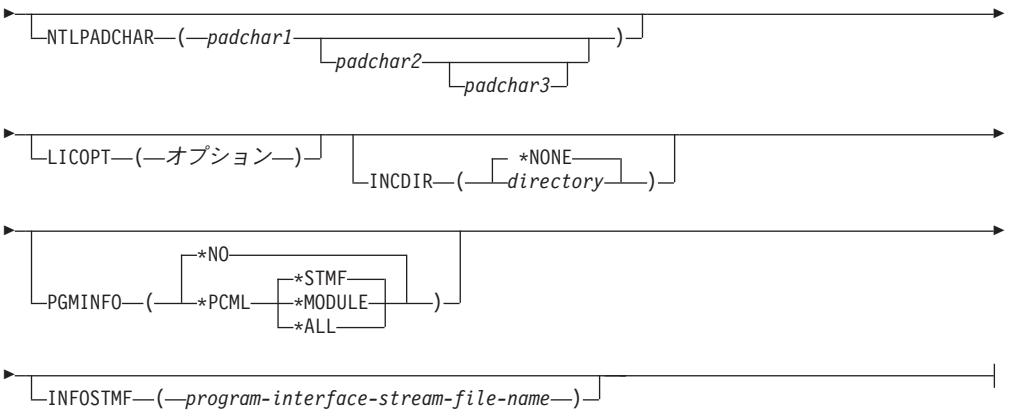
プロンプトが必要な場合、CRTBNDCBL を入力して、F4 (プロンプト) を押します。 CRTBNDCBL 画面が表示され、パラメーターがリストされ、そのデフォルト値が示されます。プロンプトを要求する前にすでにパラメーター値を指定している場合には、表示される画面にそのパラメーター値がすでに入っています。ヘルプが必要な場合には、CRTBNDCBL と入力して、F1 (ヘルプ) を押します。

CRTBNDCBL コマンドの構文

CRTBNDCBL コマンド - 形式

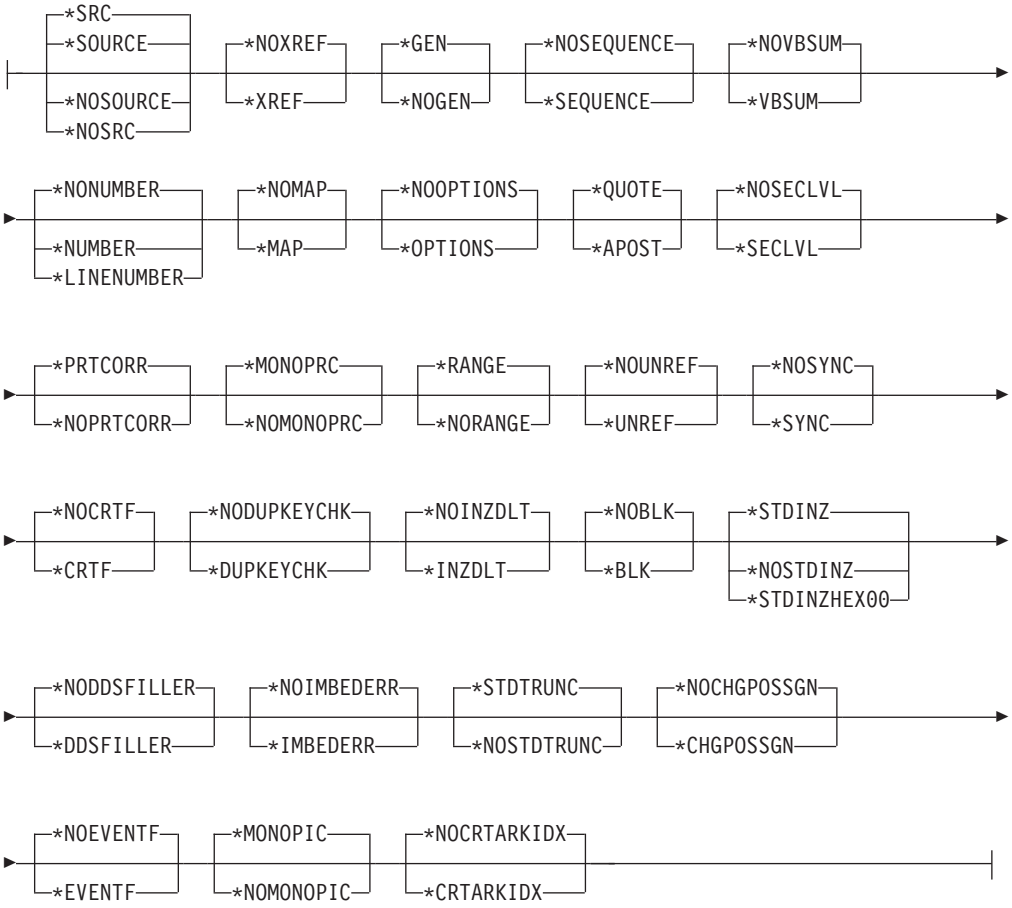

CRTBNDCBL
PGM
*CURLIB/
ライブラリー名/
*PGMID
プログラム名
SRCFILE
*LIBL/
*CURLIB/
ライブラリー名/
QCBLLESRC
ソース・ファイル名
SRCMBR
*PGM
ソース・ファイル・メンバー名
SRCSTMF
source-stream-file-name
OUTPUT
*PRINT
*NONE
GENLVL
30
重大度レベル値
TEXT
*SRCMBRTXT
*BLANK
'テキスト記述'
OPTION
OPTION の詳細
CVTOPT
CVTOPT の詳細
MSGLMT
*NOMAX
最大数
30
重大度レベル
DBGVIEW
*STMT
*SOURCE
*LIST
*ALL
*NONE
*NOCOMPRESSDBG
*COMPRESSDBG
DBGENCKEY
*NONE
文字値
OPTIMIZE
*NONE
*BASIC
*FULL
*NEVER
FLAGSTD
*NOFIPS
*MINIMUM
*INTERMEDIATE
*HIGH
*NOOBSOLETE
*OBSOLETE


EXTDSPOPT
*DFRWRT
*NODFRWRT
*UNDSPCHR
*NOUNDSPCHR
*ACCUPDALL
*ACCUPDNE
FLAG
0
重大度レベル
REPLACE
*YES
*NO
USRPRF
*USER
*OWNER
AUT
*LIBCRTAUT
*ALL
*CHANGE
*USE
*EXCLUDE
権限リスト名
LINKLIT
*PGM
*PRC
SIMPLEPGM
*YES
*NO
TGTRLS
*CURRENT
*PRV
ターゲット・リリース
SRTSEQ
*HEX
*JOB
*JOBRUN
*LANGIDUNQ
*LANGIDSHR
*LIBL/
*CURLIB/
ライブラリー名/
ソート順序テーブル名
LANGID
*JOBRUN
*JOB
言語 ID 名
ENBPFRCOL
*PEP
*ENTRYEXIT
*FULL
BNDDIR
*NONE
*LIBL/
*CURLIB/
*USRLIBL/
ライブラリー名/
バインディング・ディレクトリー名
STGMDL
*SNGLVL
*TERASPACE
*INHERIT
ACTGRP
*STGMDL
*NEW
*CALLER
活動化グループ名
PRFDTA
*NOCOL
*COL
CCSID
*JOBRUN
*JOB
*HEX
ccsid-番号
NTLCCSID
13488
ccsid-番号
ARITHMETIC
*NOEXTEND
*EXTEND31
*EXTEND63

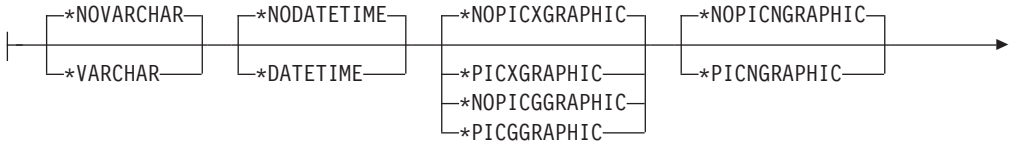


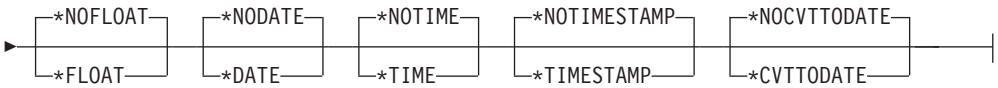
#

OPTION の詳細:



CVTOPT の詳細:





CRTBNDCBL コマンドのパラメーター

CRTBNDCBL のパラメーターのほとんどすべてが、以前に CRTCBLMOD について示されたパラメーターと同じです。ここでは 2 つのコマンドでの相違点だけを説明します。

CRTBNDCBL は次の点で CRTCBLMOD と異なっています。

- MODULE パラメーターに代わる PGM パラメーターの導入
- SRCMBR パラメーターでの *PGM オプションの使用 (*MODULE オプションに代わるもの)
- REPLACE パラメーターの使用法の違い
- USRPRF パラメーターの導入
- SIMPLEPGM パラメーターの導入
- PRFDTA パラメーターの影響を受けるオブジェクトに関する制限
- BNDDIR パラメーターの導入
- ACTGRP パラメーターの導入
- STGMDL パラメーターのデフォルト値。

PGM パラメーター:

作成するプログラム・オブジェクトのプログラム名およびライブラリー名を指定します。プログラム名およびライブラリー名は、i5/OS の命名規則に従ったものでなければなりません。 指定できる値は次のとおりです。

***PGMID**

コンパイル済みプログラム・オブジェクトの名前は、ILE COBOL ソース・プログラム内の PROGRAM-ID 段落から取られます。 SIMPLEPGM(*NO) が指定されると、コンパイル済みプログラム・オブジェクトの名前は、一連のソース・プログラム (単一ソース・ファイル・メンバー内の複数のコンパイル単位) 内の最初の ILE COBOL ソース・プログラムの PROGRAM-ID から取られます。

プログラム名

コンパイルされた ILE COBOL プログラムを識別するための名前を入力します。このパラメーターにプログラム名を指定して、一連のソース・プログラムをコンパイルする場合、SIMPLEPGM(*YES) が指定されていると、その一連のソース・プログラム内の最初のプログラム・オブジェクトにその名前が使用されます。その他のプログラム・オブジェクトでは、対応する ILE COBOL ソース・プログラム内の PROGRAM-ID 段落内に指定された名前が使用されます。

指定できるライブラリー値は次のとおりです。

***CURLIB**

作成されたプログラム・オブジェクトは、現行のライブラリー内に保管されます。特定のライブラリーを現行ライブラリーとして割り当てていない場合、QGPL が使用されます。

ライブラリー名
作成されたプログラム・オブジェクトを保管するライブラリーの名前を入力します。

REPLACE パラメーター
明示的または暗黙に指定されたライブラリーに同じ名前のプログラム・オブジェクトがすでに存在するとき、新しいプログラム・オブジェクトを作成するかどうかを指定します。 CRTBNDCBL コマンドの処理中に作成された中間モジュール・オブジェクトは、REPLACE の指定に従わず、QTEMP ライブラリーに対して暗黙のうちに REPLACE(*NO) になります。 CRTBNDCBL コマンドが処理を完了すると、中間モジュール・オブジェクトは削除されます。 REPLACE パラメーターに指定できる値は次のとおりです。

***YES**
新しいプログラム・オブジェクトが作成され、明示的または暗黙のうちに指定されたライブラリー内の同じ名前の既存のプログラム・オブジェクトに置き換わります。明示的または暗黙のうちに指定されたライブラリー内の同じ名前の既存のプログラム・オブジェクトは、ライブラリー QRPLOBJ に移されます。

***NO**
指定されたライブラリー内に同じ名前のプログラム・オブジェクトがすでに存在する場合、新しいプログラム・オブジェクトは作成されません。既存のプログラム・オブジェクトは置き換わらず、メッセージが表示され、コンパイルは停止します。

USRPRF パラメーター
作成されたプログラム・オブジェクトを実行するユーザー・プロファイルを指定します。プログラム所有者またはプログラム・ユーザーのプロファイルを使用して、プログラム・オブジェクト (プログラム・オブジェクトが各オブジェクトに対して持つ権限を含む) によってオブジェクトを使用できるプログラム・オブジェクトおよび制御を実行します。このパラメーターは、プログラム・オブジェクトがすでに存在する場合には更新されません。 USRPRF の値を変更するには、プログラム・オブジェクトを削除し、正しい値を使用して再コンパイルします (または、構成要素 *MODULE オブジェクトが存在する場合には、CRTPGM コマンドの呼び出しを選択することができます)。

指定できる値は次のとおりです。

***USER**
プログラム・ユーザーのユーザー・プロファイルが、プログラム・オブジェクトの実行時に使用されます。

***OWNER**
プログラム・オブジェクトの所有者とユーザーの両方のユーザー・プロファイルが、プログラム・オブジェクトの実行時に使用されます。所有者プロファイルとユーザー・プロファイルの両方のオブジェクト権限を集めたものが、プログラム・オブジェクトの実行中にオブジェクトを検索し、アクセスするために使用されます。プログラムが実行しているときに作成されるオブジェクトは、そのプログラムのユーザーが所有します。

SIMPLEPGM パラメーター
プログラム・オブジェクトが一連のソース・プログラム内のコンパイル単位ごと

に作成される場合に指定します。このオプションは、このコマンドへの入力ソース・メンバーに、複数のモジュール・オブジェクトを生成する一連のソース・プログラムが含まれている場合にしか意味がありません。このオプションを指定するが、入力ソース・メンバーに一連のソース・プログラムがない場合、このオプションは無視されます。指定できる値は次のとおりです。

***YES**
プログラム・オブジェクトは、一連のソース・プログラム内のコンパイル単位ごとに作成されます。 REPLACE(*NO) が指定され、同じ名前のプログラム・オブジェクトが一連のソース・プログラム内のコンパイル単位にすでに存在している場合、プログラム・オブジェクトは置き換えられず、コンパイルは次のコンパイル単位を続行します。

***NO**
単一のプログラム・オブジェクトが一連のソース・プログラム内のすべてのコンパイル単位から作成され、その最初のコンパイル単位がプログラム入り口を表します。 SIMPLEPGM(*NO) を指定すると、一連のソース・プログラム内の 1 つのソース・プログラムがモジュール・オブジェクトの生成に失敗した場合、その一連のソース・プログラム内の以降のソース・プログラムすべてもモジュール・オブジェクトの生成に失敗します。

PRFDTA パラメーター
このパラメーターは 53 ページの『PRFDTA パラメーター』に記載されているのと同じに機能しますが、下記について注意する必要があります。

注: BNDDIR パラメーターを使用して追加のモジュールとサービス・プログラムをバインドした場合には、プログラムに *COL または *NOCOL が指定されてもこれらの追加オブジェクトには影響しません。モジュールのプログラム・プロファイリング・データ属性は、そのモジュールが作成されるときに設定されます。

BNDDIR パラメーター
記号の解決に使用するバインディング・ディレクトリーのリストを指定します。最大 50 個までのバインディング・ディレクトリーを指定できます。

***NONE**
バインディング・ディレクトリーを指定しません。

バインディング・ディレクトリー名
記号の解決に使用するバインディング・ディレクトリーの名前を指定します。ディレクトリー名を修飾できるのは、次のライブラリー値のうちのいずれかです。

***LIBL**
システムはライブラリー・リストを検索して、バインディング・ディレクトリーが保管されているライブラリーを探します。これはデフォルトです。

***CURLIB**
ジョブの現行ライブラリーを検索します。ジョブの現行ライブラリーとしてライブラリーを指定していない場合には、ライブラリー QGPL が使用されます。

***USRLIBL**
ジョブのライブラリー・リストのユーザー部分にあるライブラリーだけを検索します。

ライブラリー名
検索するライブラリーの名前を指定します。

STGMDL パラメーター:
プログラムのストレージ・モデル属性を指定します。

***SNGLVL**
プログラムは、単一レベル・ストレージ・モデルを使用して作成されます。単一レベル・ストレージ・モデル・プログラムが活動化および実行されると、自動および静的ストレージ用に単一レベル・ストレージが提供されます。単一レベル・ストレージ・プログラムは、単一レベル・ストレージ活動化グループでのみ実行できます。

***TERASPACE**
プログラムは、テラスペース・ストレージ・モデルを使用して作成されます。テラスペース・ストレージ・モデル・プログラムが活動化および実行されると、自動および静的ストレージ用にテラスペース・ストレージが提供されます。テラスペース・ストレージ・プログラムは、テラスペース・ストレージ活動化グループでのみ実行できます。

***INHERIT**
プログラムは、継承ストレージ・モデルを使用して作成されます。プログラムは、活動化されると、活動化先の活動化グループのストレージ・モデルを採用します。つまり、呼び出し元のストレージ・モデルを継承します。*INHERIT ストレージ・モデルを選択した場合には、活動化グループ (ACTGRP) パラメーターに *CALLER を指定する必要があります。

ACTGRP パラメーター
プログラムの呼び出し時にそのプログラムが関連付けられる活動化グループを指定します。

***STGMDL**
STGMDL(*TERASPACE) を指定した場合には、プログラムは呼び出し時に、QILETS 活動化グループに活動化されます。それ以外の場合には、このプログラムは呼び出し時に、QILE 活動化グループに活動化されます。これはデフォルトです。

***NEW**
このプログラムが呼び出されると、新しい活動化グループの中で活動化されます。

***CALLER**
このプログラムが呼び出されると、呼び出し側の活動化グループの中で活動化されます。

活動化グループ名
このプログラムが呼び出される時に使用される活動化グループの名前を指定します。

CRTBNDCBL からの CRTPGM の暗黙呼び出し

CRTBNDCBL コマンドから暗黙のうちに呼び出された CRTPGM ステップには、暗黙のデフォルト値があります。以下でこれらを説明します。

CRTBNDCBL から呼び出されるときに CRTPGM で使用されるパラメーターは次のとおりです。

PGM

SIMPLEPGM(*YES) が明示的または暗黙のうちに指定される場合、CRTPGM は一連のソース・プログラム内のコンパイル単位ごとに呼び出されます。
CRTPGM の PGM パラメーターには、呼び出しのたびに、各コンパイル単位の対応する最外部の ILE COBOL ソース・プログラム内の PROGRAM-ID 段落内に指定されたプログラム名が使用されます。個々のバインド済みプログラム・オブジェクトは、コンパイル単位ごとに作成されます。

SIMPLEPGM(*NO) が指定されると、CRTPGM は、ソース・プログラムの順序列内のすべてのコンパイル単位に対して一度だけ呼び出されます。ソース・プログラムの順序列内の最初のコンパイル単位に対応する最外部の ILE COBOL ソース・プログラム内の PROGRAM-ID 段落に指定されたプログラム名だけが、呼び出し時に CRTPGM の PGM パラメーターで使用されます。すべてのコンパイル単位はバインドされて、1 つのプログラム・オブジェクトを作成します。

MODULE

SIMPLEPGM(*YES) が明示的または暗黙のうちに指定された場合、CRTPGM の MODULE パラメーターには、呼び出しのたびに、コンパイル単位ごとに QTEMP 内に作成されるモジュール名が使用されます。

SIMPLEPGM(*NO) が指定されると、CRTPGM コマンドの MODULE パラメーターには、呼び出しのたびに、コンパイル単位の QTEMP 内で作成されるモジュールのすべての名前が指定されます。

BNDDIR

記号の解決に使用するバインディング・ディレクトリーのリストを指定します。

*NONE (デフォルト) を指定すると、どのバインディング・ディレクトリーも使用されません。

バインディング・ディレクトリー名 を指定した場合、指定したバインディング・ディレクトリーの名前が記号解析に使用されます。ディレクトリー名を修飾できるのは、次のライブラリー値のうちのいずれかです。

***LIBL** システムはライブラリー・リストを検索して、バインディング・ディレクトリーが保管されているライブラリーを探します。これはデフォルトです。

***CURLIB**
ジョブの現行ライブラリーを検索します。ジョブの現行ライブラリーとしてライブラリーを指定していない場合には、ライブラリー QGPL が使用されます。

***USRLIBL**
ジョブのライブラリー・リストのユーザー部分にあるライブラリーだけを検索します。

ライブラリー名
検索するライブラリーの名前を指定します。

ACTGRP
指定した活動化グループが使用されます

REPLACE
CRTBNDCBL コマンドで指定される REPLACE オプションが使用されます

USRPRF
CRTBNDCBL コマンドで指定される USRPRF オプションが使用されます

AUT
CRTBNDCBL コマンドで指定される AUT オプションが使用されます

TEXT
CRTBNDCBL コマンドで指定される TEXT オプションが使用されます

TGTRLS
CRTBNDCBL コマンドで指定される TGTRLS オプションが使用されます

| **STGMDL**
| CRTBNDCBL コマンドで指定される STGMDL オプションが使用されます

CRTBNDCBL コマンドから呼び出されるときに、CRTPGM の残りのパラメーターすべてについてデフォルト値が使用されます。これらのデフォルト値の説明については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「プログラミング」カテゴリーの中の『*CL および API*』セクションの CRTPGM コマンドを参照してください。

1 つのモジュールをバインドして 1 つのプログラム・オブジェクトにする例

この例は、CRTBNDCBL コマンドを使用して、1 つのモジュールから 1 つのプログラム・オブジェクトを作成する方法を示すものです。

1. プログラム・オブジェクトを作成するには、次のように入力します。

```
CRTBNDCBL PGM(MYLIB/XMPLE1)
SRCFILE(MYLIB/QCBLLESRC)  SRCMBR(XMPLE1)
OUTPUT(*PRINT)
TEXT('ILE COBOL Program')
CVTOPT(*FLOAT)
```

そして、実行キーを押します。

CRTBNDCBL コマンドは、MYLIB 内のプログラム XMPLE1 を作成します。OUTPUT(*PRINT) オプションは、コンパイル・リストを出力させることを指定するものです。

2. リストを表示するには、次の CL コマンドのどれかを入力します。

注: コンパイル・リストを表示するためには、次にリストされているコマンドを使用する権限がなければなりません。

- DSPJOB、続いてオプション 4 (スプール・ファイルの表示) を選択する
- WRKJOB
- WRKOUTQ 待ち行列名

- WRKSPLF

CRTBNDCBL の国別言語ソート・シーケンスの指定

COBOL ソース・プログラムをコンパイルする時点で、プログラムの実行時に使用する照合順序を明示的に指定したり、プログラムの実行時に照合順序が判別される方法を指定したりすることができます。

CRTBNDCBL での照合順序の指定は、CRTCBLMOD での指定と同じです。 NLS 照合順序列の指定方法の詳細な説明については 57 ページの『CRTCBLMOD の国別言語ソート・シーケンスの指定』を参照してください。

バインダー・リストの読み方

バインディング処理では、使用されたリソース、検出した記号およびオブジェクト、およびバインディング処理で解決した問題または未解決の問題を説明するリストを作成することができます。このリストは、CRTPGM コマンドを入力するために使用するジョブのスプール・ファイルとして作成されます。コマンド・デフォルトである *NONE ではこの情報が作成されませんが、DETAIL パラメーターの値に次の 3 つのレベルの明細のいずれかを指定することにより、印刷出力としてこれを生成することができます。

- *BASIC
- *EXTENDED
- *FULL

バインダー・リストには、DETAIL に指定された値に応じて、次のセクションが含まれます。

表 6. DETAIL パラメーターに基づくバインダー・リストのセクション

セクション名	*NONE	*BASIC	*EXTENDED	*FULL
コマンド・オプションの要約		X	X	X
簡易要約表		X	X	X
拡張要約表		X	X	
BIND プログラムの情報リスト			X	X
相互参照リスト				X
バインディング統計				X

このリスト内の情報は、バインディングが失敗した場合に診断したり、またはバインダーが処理中に検出したものをフィードバックするために役立ちます。

サンプルのバインダー・リスト

次のサンプル・リストは、CRTPGM コマンドを使用して作成されたバインダー・リストです。図については、それに続く本文で参照されています。これらの参照は、反転文字 (たとえば **Z**) で参照先が示されています。本文の中の反転文字は、図の中にある文字と対応しています。

コマンド・オプションの要約

コマンド・オプションの要約は、バインダー・リストが要求される際に必ず作成されます。コマンド・オプションの要約は、ILE プログラムの作成時に使用されたオプションを示します。後で再びプログラムを作成する必要がある場合に参照できるように、コマンドに関するこの情報記述を保管しておくことができます。 図 20 に、コマンド・オプションの要約リストを示します。

```
                                        プログラムの作成                                        ページ   1
5722SS1 V5R4M0  060210                                          CBLGUIDE/EXTLFL    ISERIES1  06/02/15  13:14:03
 プログラム . . . . . :   EXTLFL
   ライブラリー . . . . . :     CBLGUIDE
 プログラム入力プロシージャー・モジュール . . . . . :   *FIRST
   ライブラリー . . . . . :
 活動化グループ . . . . . :   *NEW
 作成オプション . . . . . :   *GEN        *NODUPPROC  *NODUPVAR   *WARN       *RSLVREF
 明細のリスト . . . . . :   *FULL
 更新可能 . . . . . :   *YES
 バインド済み *SRVPGM ライブラリー名更新可能. . . . :   *NO
 ユーザー・プロファイル . . . . . :   *USER
 既存のプログラムの置き換え . . . . . :   *YES
 権限 . . . . . :   *LIBCRTAUT
 ターゲット・リリース . . . . . :   *CURRENT
 再初期設定可能 . . . . . :   *NO
 記憶域モデル . . . . . :   *SNGLVL
 プロシージャー間分析 . . . . . :   *NO
 IPA 制御ファイル . . . . . :   *NONE
 IPA 置き換え IL データ . . . . . :   *NO
 テキスト . . . . . :   *ENTMODTXT
 モジュール     ライブラリー     モジュール     ライブラリー     モジュール     ライブラリー     モジュール     ライブラリー
 EXTLFL         CBLGUIDE
 サービス・                      サービス・                      サービス・                      サービス・
 プログラム     ライブラリー     プログラム     ライブラリー     プログラム     ライブラリー     プログラム     ライブラリー
 *NONE
 バインド・                      バインド・                      バインド・                      バインド・
 ディレクトリー ライブラリー     ディレクトリー ライブラリー     ディレクトリー ライブラリー     ディレクトリー ライブラリー
 *NONE
```

図 20. CRTPGM コマンド・オプションの要約リスト

拡張要約表

拡張要約表は、*EXTENDED または *FULL が指定される場合に提供されます。この表には、バインダーが解決したインポートおよびエクスポートの一般的な表示を提供する統計情報が含まれています。図 21 に、拡張要約表のレイアウトを示します。

```
                                        プログラムの作成                                        ページ   3
5722SS1 V5R4M0  060210                                          CBLGUIDE/EXTLFL    ISERIES1  06/02/15  13:14:03

                                        拡張要約表
   有効な定義 . . . . . :          341 A
     強 . . . . . :          340
     弱 . . . . . :            1
   解決済み参照 . . . . . :           16 B
     終了強定義 . . . . . :           15
     終了弱定義 . . . . . :            1
                    * * * * *  拡 張 要 約 表 の 終 わ り  * * * * *
```

図 21. CRTPGM リスト - 拡張要約表

拡張要約表には、次の項目についての統計情報が示されます。

A *有効な定義:* このフィールドには、エクスポートに利用できる指定された変数またはプロシージャーの数が示されます。この定義は、さらに**強定義**と**弱**

定義に分類されます。強定義の場合、変数またはプロシージャーのためのストレージが割り振られます。弱定義の場合、変数またはプロシージャーのためのストレージが参照されます。

ILE COBOL では、最外部の COBOL ソース・プログラムおよびそれと関連した CANCEL プロシージャーの定義は強定義です。 EXTERNAL データおよび EXTERNAL ファイルは弱定義になります。外部静的プロシージャーへの CALL、CANCEL、および SET ENTRY のモジュール・インポートは、強定義になります。 EXTERNAL データおよび EXTERNAL ファイルへの参照のモジュール・インポートは、弱定義になります。

B *解決済み参照:* このフィールドには、内部モジュール・エクスポートと対応するインポートの数が示されます。

106 ページの図 21 に示されている使用カウントは、10 進数形式です。

簡易要約表

*BASIC、*EXTENDED、または *FULL を指定した場合に使用できる簡易要約表は、バインディング処理中のエラーを示す情報を提供します。図 22 に、簡易要約表のレイアウトを示します。

```
                                        プログラムの作成                                        ページ   4
5722SS1 V5R4M0  060210                                                CBLGUIDE/EXTLFL    ISERIES1  06/02/15  13:14:03

                                          簡略要約表
 プログラム入力プロシージャー . . . . . . . . . . :   1  C
   記号       タイプ    ライブラリー オブジェクト バインド    識別コード
     F         G         H         I         J          K
             *MODULE   CBLGUIDE    EXTLFL      *YES        _Qln_pep
 複数強定義  . . . . . . . . . . . . . . . . . . . :   0  D
 未解決の参照 . . . . . . . . . . . . . . . . . . . :   0  E
                        * * * * *  簡 略 要 約 表 の 終 わ り  * * * * *
```

図 22. CRTPGM リスト - 簡易要約表

このテーブルは 3 つのリストで構成されており、次の各カテゴリーごとに複数の項目が含まれています。

C *プログラム入力プロシージャー:* 呼び出し側プログラムから制御を取得するプロシージャーの数。

D *複数強定義:* 同じ名前のモジュール・エクスポート・プロシージャーの数。これは、0 でなければなりません。

E *未解決の参照:* インポートされるプロシージャーまたは変数で、エクスポートが見つからなかったものの数。これは、0 でなければなりません。

F *シンボル #:* シンボル番号は、 108 ページの『BIND プログラムの情報リスト』に示されているバインダー情報リストからのものです。 *BASIC が DETAIL パラメーターに指定された場合、この部分はブランクになります。

G *タイプ:* タイプ・フィールドに表示される ID を含むオブジェクトのタイプ。

H *ライブラリー:* ライブラリー・フィールドに表示されるオブジェクトを含むライブラリーの名前。

I *オブジェクト:* オブジェクト・フィールドに表示されるプログラム入力プロシージャー、未解決の参照、または強定義を含むオブジェクトの名前。

J *バインド:* このフィールドにモジュール・オブジェクトに値 *YES が表示されているなら、モジュール・オブジェクトはコピーによってバインドされます。プログラムの場合に、このフィールドに値 *YES が表示されているなら、そのプログラムは参照によってバインドされます。このフィールドでモジュール・オブジェクトまたはプログラムに値 *NO が表示されているなら、そのオブジェクトはバインドに含められません。

K *ID:* ID フィールドには、モジュール・ソースからのプロシージャーまたは変数の名前が表示されます。

この例では、プログラム入力プロシージャー、未解決の参照、または複数強定義の合計数はそれぞれ、1、0、0 です。 107 ページの図 22 に示されている使用カウントは、10 進数形式です。

BIND プログラムの情報リスト

このリストは、バインディング処理に関する詳細な情報を示すものであり、*EXTENDED または *FULL を指定した場合に出力されます。 109 ページの図 23 に、バインド・プログラムの情報リストのレイアウトを示します。

```
                                   プログラムの作成                                    ページ   5
5722SS1 V5R4M0  060210                                  CBLGUIDE/EXTLFL    ISERIES1  06/02/15  13:14:03

                                BIND プログラムの情報リスト
 モジュール . . . . . . . . . :   EXTLFL     L
   ライブラリー . . . . . . . :     CBLGUIDE
   バインド . . . . . . . . . :     *YES
   変更日 / 時刻  . . . . . . :     00/08/15  13:11:40
   テラスペース記憶域使用可能
     化 . . . . . . . . . . . :    *YES
   記憶域モデル . . . . . . . :     *SNGLVL
       数値      記号        参照      識別コード                          タイプ    有効範囲    エクスポート キー
        M         N           O         P                                 Q         R           S          T
       00000001  定義                  EF1_ffd                             データ    モジュール  弱              190
****                                    最弱定義
       00000002  定義                  EF1MAIN                             PROC      モジュール  強
       00000003  定義                  EF1MAIN_reset                       PROC      モジュール  強
       00000004  参照        000000A7  _Qln_rut                            データ
       00000005  参照        00000001  EF1_ffd                             データ
       00000006  参照        000000C6  _Qln_cancel_handler                 PROC
       00000007  参照        000000D6  _Qln_cancel_handler_pep             PROC
       00000008  参照        000000A8  _Qln_init_mod                       PROC
       00000009  参照        000000A9  _Qln_init_mod_bdry                  PROC
       0000000A  参照        000000AA  _Qln_init_oprg                     Proc
       0000000B  参照        000000B9  _Qln_recurse_msg                   PROC
       0000000C  参照        00000022  _Qln_disp_norm                     PROC
       0000000D  参照        000000BE  _Qln_stop_prg                      PROC
       0000000E  参照        000000BB  _Qln_cancel_msg                    PROC
       0000000F  参照        000000BD  _Qln_fc_hdlr                       PROC
       00000010  参照        0000012A  Q_LE_leDefaultEh                   PROC
       00000011  参照        00000131  Q_LE_leBdyCh                       PROC
       00000012  参照        00000161  Q_LE_leActivationInit              PROC
       00000013  参照        00000132  Q_LE_leBdyEpilog                   PROC
 サービス・プログラム . . . . :   QLNRACPT
   ライブラリー . . . . . . . :     QSYS
   バインド . . . . . . . . . :     *YES
   変更日 / 時刻  . . . . . . :     00/08/14  18:54:04
   テラスペース記憶域使用可能
     化モジュール . . . . . . :     *ALL
   記憶域モデル . . . . . . . :     *SNGLVL
       数値      記号        参照      識別コード                          タイプ    有効範囲    エクスポート キー
       00000014  定義                  _Qln_DateISODescriptor              データ              強
       00000015  定義                  _Qln_TimeISODescriptor              データ              強
       00000016  定義                  _Qln_acpt_norm                      PROC                強
       00000017  定義                  _Qln_acpt_console                   PROC                強
       00000018  定義                  _Qln_acpt_session                   PROC                強
       00000019  定義                  _Qln_acpt_time                      PROC                強

⋮
       0000015B  定義                  CEESECI                             PROC                強
       0000015C  定義                  CEEDYWK                             PROC                強
       0000015D  定義                  CEELOCT                             PROC                強
       0000015E  定義                  CEEUTC                              PROC                強
       0000015F  定義                  CEEGMT                              PROC                強
       00000160  定義                  CEEUTCO                             PROC                強
       00000161  定義                  Q_LE_leActivationInit               PROC                強
       00000162  定義                  Q_LE_leActivationInitRouter         PROC                強
       00000163  定義                  QleActBndPgm                        PROC                強
       00000164  定義                  QleGetExp                           PROC                強
       00000165  定義                  Q_LE_leCheck                        PROC                強
                        * * * * *  B I N D プ ロ グ ラ ム の 情 報 リ ス ト の 終 わ り   * *
```

図 23. CRTPGM リスト - バインド・プログラムの情報リスト

L *モジュールおよびライブラリー:* このフィールドは、処理されたライブラリー、モジュール・オブジェクト名、およびサービス・プログラム名を示します。

M *番号:* このプログラム内の各データ項目または ILE プロシージャーに割り当てられた固有の ID。この番号は相互参照に使用されます。

N *記号:* このフィールドには、エクスポートかインポートかを示す記号が示されます。このフィールドの値が定義の場合、記号はエクスポートです。このフィールドの値が参照の場合、記号はインポートです。

O *参照:* このフィールドは、記号が定義の場合はブランクであり、記号欄の値

が参照である場合は記号番号が示されます。記号欄が参照の場合、このフィールドには、インポート要求を満たすエクスポート (定義) を識別する固有の番号が示されます。

P *ID:* これは、エクスポートまたはインポートされるシンボルの名前です。

Q *タイプ:* シンボル名が ILE プロシージャーの場合、このフィールドには PROC が表示されます。シンボル名がデータ項目である場合、このフィールドにはデータが表示されます。

R *有効範囲:* このフィールドは、エクスポートされるシンボル名にアクセスできるレベルを示します。

S *エクスポート:* このフィールドは、エクスポートされるデータ項目が弱定義か強定義かを示します。

T *キー:* このフィールドには、弱エクスポートの項目の長さが示されます。このフィールドに表示される値は、16 進形式です。

このリストの欄には、次の情報が含まれています。

相互参照リスト

相互参照リストは、*FULL が指定される場合にのみ提供されます。これは、バインダー・リストが長大になったため、そのための簡潔な索引を手に入れたいプログラマーにとって便利です。相互参照リストでは、バインダー・リスト中のすべての固有の ID のリストがアルファベット順に示され、さらにその ID のすべての定義と解決済み参照との対応リストが示されます。 111 ページの図 24 に、相互参照リストのレイアウトを示します。

```
                                        プログラムの作成                                           ページ  15
5722SS1 V5R4M0  060210                                                  CBLGUIDE/EXTLFL    ISERIES1  06/02/15  13:14:03

                                          相互参照表
                                                          -------- 参照 --------
         識別コード                               定義         参照       参照          タイプ       ライブラリー  オブジェクト
           U                                       V                   W             X           Y             Z
         __CEEDOD                                 0000014F                           *SRVPGM      QSYS          QLEAWI
         __CEEGSI                                 00000150                           *SRVPGM      QSYS          QLEAWI
         __CEEHDLR                                0000013C                           *SRVPGM      QSYS          QLEAWI
         __CEEHDLU                                0000013D                           *SRVPGM      QSYS          QLEAWI
         __CEERTX                                 00000135                           *SRVPGM      QSYS          QLEAWI
         __CEETSTA                                0000014E                           *SRVPGM      QSYS          QLEAWI
         __CEEUTX                                 00000136                           *SRVPGM      QSYS          QLEAWI
         _C_session_cleanup                       00000144                           *SRVPGM      QSYS          QLEAWI
         _C_session_open                          00000145                           *SRVPGM      QSYS          QLEAWI
         _Qln_acos                                000000D7                           *SRVPGM      QSYS          QLNRMATH
         _Qln_acpt_attribute                      0000001D                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_console                        00000017                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_da                             0000002B                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_date                           0000001A                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_date_yyyy                      0000002C                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_day                            0000001B                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_day_of_week                    0000001C                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_day_yyyy                       0000002D                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_io_feed                        00000021                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_lda                            0000001F                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_norm                           00000016                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_open_feed                      00000020                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_pip                            0000001E                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_session                        00000018                           *SRVPGM      QSYS          QLNRACPT
         _Qln_acpt_time                           00000019                           *SRVPGM      QSYS          QLNRACPT

 :
         Q LE leBdyCh                             00000131    00000011               *SRVPGM      QSYS          QLEAWI
         Q LE leBdyEpilog                         00000132    00000013               *SRVPGM      QSYS          QLEAWI
         Q LE leCheck                             00000165                           *SRVPGM      QSYS          QLEAWI
         Q LE leDefaultEh                         0000012A    00000010               *SRVPGM      QSYS          QLEAWI
         Q LE AG_prod_rc                          00000129                           *SRVPGM      QSYS          QLEAWI
         Q LE AG_user_rc                          00000128                           *SRVPGM      QSYS          QLEAWI
         Q LE HdlrRouterEh                        00000138                           *SRVPGM      QSYS          QLEAWI
         Q LE RtxRouterCh                         00000137                           *SRVPGM      QSYS          QLEAWI
         QleActBndPgm                             00000163                           *SRVPGM      QSYS          QLEAWI
         QleGetExp                                00000164                           *SRVPGM      QSYS          QLEAWI
         QlnDumpCobol                             000000C0                           *SRVPGM      QSYS          QLNRMAIN
         QlnRtvCobolErrorHandler                  000000C1                           *SRVPGM      QSYS          QLNRMAIN
         QlnSetCobolErrorHandler                  000000C2                           *SRVPGM      QSYS          QLNRMAIN
                         * * * * *   相 互 参 照 表 の 終 わ り   * * * * *
```

図 24. CRTPGM リスト - 相互参照リスト

フィールドには、次の情報が含まれています。

U *ID:* 記号解決中に処理されたエクスポートの名前。

V *定義:* 各エクスポートに関連した固有の識別番号。

W *参照:* 解決されてこのインポート (定義) となったエクスポート (参照) の固有の識別番号のリスト。

X *タイプ:* モジュール・オブジェクト (*MODULE) またはサービス・プログラム (*SRVPGM) からエクスポートが可能かどうかを示します。

Y *ライブラリー:* この行で説明されている記号が定義されているライブラリーの名前。

Z *オブジェクト:* この行で説明されている記号が定義されているモジュール・オブジェクトまたはサービス・プログラムの名前。

バインディング統計

バインディング統計セクションは、DETAIL パラメーターに *FULL 値が使用された場合にのみ作成されます。このセクションは、プログラムの特定部分をバインドするために使用されたシステム CPU 時間の量を示します。他の ILE プログラムか

らの類似出力と比較する場合、またはその他の場合として特定プログラムが作成された場合にのみ、これらの値は意味を持ちます。バインディング言語コンパイルの CPU 時間の値は、ILE プログラムの場合には常にゼロになります。図 25 に、バインディング統計のレイアウトを示します。

```
                                    プログラムの作成                                    ページ  22
5722SS1 V5R4M0  060210                                    CBLGUIDE/EXTLFL    ISERIES1  06/02/15  13:14:03

                                    バインド統計
 記号収集 CPU 時間  . . . . . . . . . . . . . . . . . . . . . . :              .001
 記号分析解決 CPU 時間  . . . . . . . . . . . . . . . . . . . . :              .000
 バインド・ディレクトリー分析解決 CPU 時間. . . . . . . . . . . :              .009
 BIND プログラム言語コンパイル CPU 時間 . . . . . . . . . . . . :              .000
 リスト作成 CPU 時間  . . . . . . . . . . . . . . . . . . . . . :              .082
 プログラム / サービス・プログラム作成 CPU 時間 . . . . . . . . :              .020
 合計 CPU 時間  . . . . . . . . . . . . . . . . . . . . . . . . :              .322
 合計経過時間 . . . . . . . . . . . . . . . . . . . . . . . . . :             1.145
                          * * * * *  バ イ ン ド 統 計 の 終 わ り  * * * * *
*CPC5D07 - プログラム EXTLFL がライブラリー CBLGUIDE に作成された。
                       * * * * *  プ ロ グ ラ ム 作 成 リ ス ト の 終 わ り  * * * * *
```

図 25. CRTPGM リスト - バインディング統計

モジュール・オブジェクトの修正およびプログラム・オブジェクトの再バインディング

プログラム・オブジェクトは、問題にアドレスするため、または変更要件を満たすために、作成後に変更が必要になる場合があります。プログラム・オブジェクトはモジュール・オブジェクトから作成されるので、プログラム・オブジェクト全体を変更する必要はありません。変更する必要があるモジュール・オブジェクトだけを特定してからそれを変更し、再びプログラム・オブジェクトをバインドすることができます。モジュール・オブジェクトを変更する方法は、何を変更する必要があるかによって異なります。

モジュール・オブジェクトを変更するには、以下の 5 つの方法があります。
- モジュール・オブジェクトの ILE COBOL ソース・プログラムを変更する
- モジュール・オブジェクトの最適化レベルを変更する
- モジュール・オブジェクトのプログラム識別情報を変更する
- 使用可能なパフォーマンス収集の量を変更する
- 使用可能なプロファイリング・データを変更する

注: モジュール・オブジェクトに上記のどれかの変更を行うには、ソース・コードおよび必要なコマンドに対する権限が必要です。

モジュール・オブジェクトの最適化レベルまたはプログラム識別情報を変更したい場合、再びプログラム・オブジェクトを作成する必要はありません。これは、プログラム・オブジェクトをデバッグしたい場合や、プログラム・オブジェクトを実行用にする準備ができている場合などによくあることです。そのような変更は、モジュール・オブジェクトを再作成するよりも迅速に行うことができ、システム・リソースの使用量も少なくて済みます。

そのような状況では、多くのモジュール・オブジェクトを同時に作成することができます。モジュール処理 (WRKMOD) コマンドを使用して、ライブラリー、名前、

総称記号、または *ALL によって選択されたモジュール・オブジェクトのリストを入手することができます。また、ILE COBOL コンパイラーによって作成されたモジュール・オブジェクトだけのリストに限定することもできます。

モジュール・オブジェクトに変更を加えると、CRTPGM コマンドまたは UPDPGM コマンドを使用して、プログラム・オブジェクトを再バインドしなければなりません。

ILE COBOL ソース・プログラムの変更

ILE COBOL ソース・プログラムに変更を行う必要がある場合には、次のことを行います。

1. SEU を使用して、ILE COBOL ソース・プログラムの必要な個所を変更する。ソース・コードの変更の詳細については 14 ページの『原始ステートメント入力ユーティリティーを用いてソース・ステートメントを入力する』を参照してください。

   クライアント・ツール入門については、『クライアント製品のアプリケーション開発ツールの使用』を参照してください。
2. CRTCBLMOD コマンドを使用して ILE COBOL ソース・プログラムをコンパイルし、新しいモジュール・オブジェクトを作成する。 ILE COBOL ソース・プログラムのコンパイルについては 29 ページの『COBOL モジュールの作成 (CRTCBLMOD) コマンドの使用』を参照してください。
3. CRTPGM コマンドまたは UPDPGM コマンドを使用してモジュール・オブジェクトをバインドし、新しいプログラム・オブジェクトを作成する。プログラム・オブジェクトの作成については 92 ページの『プログラム作成 (CRTPGM) コマンドの使用』を参照してください。

最適化レベルの変更

システム上での実行のために、生成コードを最適化するレベルを変更することができます。コンパイラーがコードを最適化する場合、同じ出力でも、必要なシステム・リソースの量が少なくなるように、処理がより簡略に実行されるショートカットを探します。次に、ショートカットをマシン・コードに変換します。

たとえば、

```
a = (x + y) + (x + y) + 10
```

a の解決において、コンパイラーは 2 つの式 (x + y) が同じものであることを認識し、2 番目の式にはすでに計算済みの値を使用します。

最適化をさらに進めると、プログラム・オブジェクトのシステム上での実行効率がそれだけ良くなります。しかし、最適化をさらに進めると、コンパイル時間が増え、最適化された変数を見られない場合もでてきます。モジュール・オブジェクトの最適化レベルを変更して、プログラム・オブジェクトのデバッグ時の変数を正確に表示し、それからプログラム・オブジェクトが実行用に準備ができた段階で、最適化レベルを変更することができます。

ILE コンパイラーでは、一連の最適化レベルをサポートしています。現在 4 つの最適化レベルがあり、ILE COBOL のユーザーが使用できるのはそのうちの 3 つです。その 3 つの最適化レベルとは、以下のとおりです。

***NONE または 10**

生成されたコードに対して付加的な最適化は実行されません。この最適化レベルでは、プログラム・オブジェクトのデバッグ時に、変数を表示して変更することができます。この値により、実行時のパフォーマンスは最低レベルとなります。

***BASIC または 20**

生成されるコードについて、ローカル・ブロック・レベルだけの一部の最適化が実行されます。プログラム・オブジェクトのデバッグ時には、変数の表示はできますが、変更することはできません。この最適化レベルでは、実行時のパフォーマンスが少し向上します。

***FULL または 30**

生成されたコードに対して、完全な最適化 (グローバル・レベル) が実施されます。プログラム・オブジェクトのデバッグ中は、変数を変更することはできませんが、表示することはできます。しかし、デバッグ中に表示される変数の値は、その現行値でない場合があります。

実行時のパフォーマンスを最適化する効果は、アプリケーション・プログラムのタイプによって変わります。たとえば、数値計算のアプリケーション・プログラムの場合、最適化は実行時パフォーマンスをかなり向上させることがありますが、入出力を主とするアプリケーション・プログラムの場合、最適化を行っても実行時パフォーマンスはほんのわずかしか向上しない場合があります。

プログラム・オブジェクト内のモジュール・オブジェクトの最適化レベルを変更するには、モジュール処理 (WRKMOD) コマンドを使用してください。コマンド行に WRKMOD と入力すると、「モジュールの処理」画面が表示されます。「モジュールの処理」画面でオプション 5 (表示) を選択し、変更の必要がある属性値を表示します。「モジュール情報の表示」画面は、図 26 に示されています。

```
                          モジュール情報の表示
                                                              画面 7 の 1
 モジュール . . . . . :   COPYPROC
   ライブラリー . . . . . :     TESTLIB
 詳細 . . . . . :   *BASIC
 モジュール属性 . . . . . :   CBLLE

 モジュール情報 :
   モジュール作成日 / 時刻 . . . . . :   98/08/25  12:57:17
   ソース・ファイル  . . . . . :   QCBLLESRC
     ライブラリー. . . . . :     TESTLIB
   ソース・メンバー  . . . . . :   COPYPROC
   ソース・ファイル変更日 / 時刻 . . . . . :   98/08/19  12:04:57
   所有者. . . . . :   TESTLIB
   コード化文字セット ID . . . . . :   37
   テキスト記述  . . . . . :   PG - COPY WITHIN PR
OCESS STATEMENT EXAMPLE
   作成データ  . . . . . :   *YES
   中間言語データ  . . . . . :   *NO
                                                              続く...
 続行するには、実行キーを押してください。

 F3=終了    F12=取り消し
```

図 26. 「モジュール情報の表示」画面の最初の画面

まず、*作成データ* の値が *YES になっているか調べてください。これは、*最適化レベル* 値が変更されても、モジュール・オブジェクトを変換できることを意味して

います。値が *NO の場合、最適化レベルを変更するためには、モジュール・オブジェクトを再作成し、マシン・インストラクション・テンプレートを含めなければなりません。

次に、次ページ・キーを押すと、モジュール・オブジェクトの情報がさらに表示されます。

```
                          モジュール情報の表示
                                                              画面 7 の 1
モジュール . . . . . :   COPYPROC
  ライブラリー . . . . . :     TESTLIB
詳細 . . . . . :   *BASIC
モジュール属性 . . . . . :   CBLLE

  RTVCLSRC 可能 (CL モジュール)  . . . . . :   *NO
  ソート順序テーブル  . . . . . :   *HEX
  言語 ID . . . . . :   *JOBRUN
  最適化レベル  . . . . . :   *NONE
  最大最適化レベル  . . . . . :   *FULL
  デバッグ・データ. . . . . :   *YES
  圧縮  . . . . . :   *NO
  プログラム入りロプロシージャー名. . . . . :   _Qln_pep
  パラメーターの数  . . . . . :   0
  モジュールの状態. . . . . :   *USER
  モジュール定義域  . . . . . :   *SYSTEM
  エクスポートされた定義済み記号の数. . . . . :   2
                                                              続く...
続行するには、実行キーを押してください。

F3= 終了    F12= 取り消し
```

図 27. 「モジュール情報の表示」画面の 2 番目の画面

最適化レベル 値を調べてください。最適化レベル値がすでに希望するレベルになっている場合もあります。

モジュールにマシン・インストラクション・テンプレートがあり、最適化レベルを変更したい場合には、F12 (取り消し) を押します。「モジュールの処理」画面が表示されます。最適化レベルを変更するモジュール・オブジェクトの場合には、オプション 2 (変更) を選択します。 116 ページの図 29 に示されているとおりに、CHGMOD コマンド・プロンプトが表示されます。*モジュールの最適化* プロンプトに指定されている値を上書き入力してください。

次に、次ページ・キーを押すと、モジュール・オブジェクトの情報の最後のものが表示されます。

```
                          モジュール情報の表示
                                                              画面 7 の 1
モジュール . . . . . :   COPYPROC
  ライブラリー . . . . . :     TESTLIB
詳細 . . . . . :   *BASIC
モジュール属性 . . . . . :   CBLLE

  インポートされた（未解決）記号の数  . . . . . :   14
  プロファイル・データ. . . . . :   *NOCOL
  パフォーマンス収集使用可能  . . . . . :   *PEP

  テラスペース記憶域使用可能化. . . . . :   *YES
  記憶域モデル. . . . . :   *SNGLVL
  ライセンス内部コード・オプション. . . . . :   *NONE
モジュールの互換性 :
  モジュールの作成時期. . . . . :   V7R1M0
  モジュールの作成目的. . . . . :   V7R1M0
  モジュールを復元できる最も初期のリリース  . . . . . :   V7R1M0
  必要な変換  . . . . . :   *NO
  変換の詳細  . . . . . :   *COMPAT



                                                                  終わり
続行するには、実行キーを押してください。

F3= 終了    F12= 取り消し
```

図 28. 「モジュール情報の表示」画面の 3 番目の画面

パフォーマンス収集使用可能 プロンプトは、そのモジュールがプログラム入り口点への入り口とそこからの出口だけについてのパフォーマンス測定コードで作成されていることを示します。モジュールの互換性プロンプトは、そのモジュールと互換性のあるオペレーティング・システムのリリースおよびバージョンを示します。

```
                        モジュールの変更 (CHGMOD)

選択項目を入力して、実行キーを押してください。

  モジュール . . . . . .   > COPYPROC       名前, 総称*, *ALL
    ライブラリー . . . . . .   >   TESTLIB_     名前 , *USRLIBL, *LIBL
  モジュールの最適化 . . . . . .   *NONE_        *SAME, *FULL, *BASIC...
  識別情報の除去 . . . . . .   *DBGDTA       *SAME, *NONE, *ALL...
                値の続きは＋
  プロファイリング・データ   . . .     *COL          *SAME, *NOCOL, *COL
  モジュールの強制再作成   . . . .     *NO           *NO, *YES
  テキスト '記述'  . . . . . .    'PROCESS ステートメント例内の PG - COPY
       '_____________________
   LIC オプション :
     オプション  . . . . . .    *SAME
     処置  . . . . . .                  *REPLACE, *ADD
   パフォーマンス収集使用可能 :
     収集レベル  . . . . . .    *PEP          *SAME, *NONE, *PEP, *FULL...
     プロシージャー  . . . . . .                *ALLPRC, *NONLEAF


                                                                  終わり
F9= すべてのパラメーター     F11= キーワード     F14= コマンド・ストリング
F24= キーの続き
```

図 29. CHGMOD コマンドのプロンプト

モジュール・オブジェクトを低いレベルの最適化に変更すると、デバッグ中に変数の値を表示して、場合によっては変更することができるようになります。

最適化を変更する追加モジュール・オブジェクトのプロセスを繰り返します。同じプログラム・オブジェクト内で変更するモジュール・オブジェクトが 1 つであっても複数であっても、システムがそれらのモジュール・オブジェクトを検出すると、すべてのインポートが解決されるので、プログラム作成時間は同じです。

プログラム・オブジェクト内でモジュール・オブジェクトの最適化レベルの変更を終了したら、CRTPGM コマンドを使用してプログラム・オブジェクトを再作成するか、または UPDPGM コマンドを使用して、新しいモジュール・オブジェクトによって既存のプログラム・オブジェクトを更新してください。

モジュール識別情報の除去

モジュール識別情報とは、モジュール・オブジェクトとともに記憶できる 3 種類のデータのことです。このデータにより、モジュール・オブジェクトはデバッグするか、再作成せずに変更するか、あるいは中間言語最適化プログラムで最適化することができます。モジュール・オブジェクトが作成されているなら、このデータは除去することしかできません。データが除去されると、そのデータに代わるモジュール・オブジェクトを再作成しなければなりません。データには、次の 3 つのタイプがあります。

中間言語データ

*ILDTA 値によって表されます。このデータは、モジュール・オブジェクトを中間言語最適化プログラムによって最適化できるようにするために必要です。 現在、このタイプのデータの作成をサポートしているのは、ILE C コンパイラーだけです。

作成データ

*CRTDTA 値によって表されます。このデータは、コードをマシン・インストラクションに変換するために必要です。モジュール・オブジェクトが CRTCBLMOD コマンドを使用して作成される場合、マシン・インストラクション (MI) テンプレートがモジュール・オブジェクトに含められます。MI テンプレートは、明示的に除去されるまで表示され続けます。モジュール・オブジェクトの最適化レベルを変更するためには、モジュール・オブジェクトにこのデータがなければなりません。

デバッグ・データ

*DBGDTA 値によって表されます。このデータは、モジュール・オブジェクトをデバッグできるようにするために必要です。モジュール・オブジェクトが CRTCBLMOD コマンドを使用して作成される場合、デバッグ・データがモジュール・オブジェクトに含められます。デバッグ・データのタイプと量は、DBGVIEW パラメーターによって決定されます。

すべての識別情報を除去すると、モジュール・オブジェクトはその最小サイズに圧縮されます。すべての識別情報を除去すると、モジュール・オブジェクトを再作成しない限り、どんな方法でもモジュール・オブジェクトを変更することはできません。

モジュール・オブジェクトからあるタイプのデータを除去したり、すべてのタイプのデータを除去したり、またはいずれも除去しないようにしたりするには、モジュール処理 (WRKMOD) コマンドを使用します。コマンド行に WRKMOD と入力す

ると、「モジュールの処理」画面が表示されます。オプション 5 (表示) を選択し、変更の必要がある属性値を表示します。「モジュール情報の表示」画面は、 114 ページの図 26 に示されています。

まず、*作成データ・パラメーター*の値を調べてください。値が *YES の場合、作成データが存在し、除去可能です。値が *NO の場合、除去するための作成データはありません。モジュール・オブジェクトは、作成データを再作成してマシン・インストラクション・テンプレートを組み込むのでない限り、再変換することはできません。

次に、次ページ・キーを押すと、モジュール・オブジェクトの情報がさらに表示されます。*デバッグ・データ・ パラメーター*の値を調べてください。値が *YES の場合、デバッグ・データが存在し、モジュール・オブジェクトはデバッグできます。値が *NO の場合、デバッグ・データは存在せず、モジュール・オブジェクトは、デバッグ・データを再作成してデバッグ・データを組み込むのでない限り、デバッグすることはできません。識別情報を変更するモジュール・オブジェクトについて、オプション 2 (変更) を選択してください。 CHGMOD コマンド・プロンプトが表示されます。*識別情報除去* プロンプトに指定されている値を上書き入力してください。

モジュールの強制再作成 パラメーターを使用すると、モジュール・オブジェクトの再作成を可能にすることができます。最適化レベルが変更された場合、作成データが除去されていない限り、モジュール・オブジェクトは常に再作成されます。デバッグ・データを除去し、最適化レベルは変更せずにプログラム・オブジェクトを再変換する場合には、*モジュールの強制再作成* パラメーター値を *YES に変更しなければなりません。

変更する追加モジュール・オブジェクトのプロセスを繰り返します。同じプログラム・オブジェクト内で変更するモジュール・オブジェクトが 1 つであっても複数であっても、システムがそれらのモジュール・オブジェクトを検出すると、すべてのインポートが解決されるので、プログラム作成時間は同じです。

プログラム・オブジェクト内でモジュール・オブジェクトの最適化レベルの変更を終了したら、CRTPGM コマンドを使用してプログラム・オブジェクトを再作成するか、または UPDPGM コマンドを使用して、新しいモジュール・オブジェクトによって既存のプログラム・オブジェクトを更新してください。

パフォーマンス収集の使用可能化

コンパイル単位に対してパフォーマンス測定を行う場合に指定できるオプションは、以下のとおりです。

収集レベル

収集レベルには、以下のものがあります。

***PEP** プログラム入力プロシージャーの入り口および出口でのみ、パフォーマンス統計情報が収集されます。 アプリケーションに関する全体的なパフォーマンス情報を収集したい場合には、この値を選択してください。このサポートは、以前 TPST ツールで提供されていたものと同じです。これはデフォルトです。

***ENTRYEXIT**

プログラムのすべてのプロシージャーの入り口と出口だけについてのパフォーマンス統計が収集されます。これには、プログラムの PEP ルーチンが含まれます。

この選択項目が役立つのは、全ルーチンに関する情報を把握したい場合です。アプリケーションで呼び出されるプログラムがすべて、*PEP、*ENTRYEXIT、または *FULL オプションを使用してコンパイルされていることが分かっている場合は、このオプションを使用してください。そうでない場合、アプリケーションが、パフォーマンス測定のできない他のプログラムを呼び出しているなら、パフォーマンス測定ツールは、リソースをその呼び出し側のアプリケーションが使用しているものとします。このため、リソースが現在どこで実際に使用されているのかを判別することは困難になります。

***FULL**

すべてのプロシージャーの入り口と出口のパフォーマンス統計が収集されます。また、外部プロシージャー呼び出しの前後についても統計が収集されます。

アプリケーションから呼び出されるプログラムが、*PEP、*ENTRYEXIT、または *FULL オプションを使用してコンパイルされていない場合には、このオプションを使用してください。このオプションによりパフォーマンス測定ツールは、アプリケーションが使用するリソースとアプリケーションで呼び出されるプログラム (パフォーマンス測定が可能でない場合であっても) が使用するリソースとを区別します。このオプションは最もコストがかかるものですが、これによって 1 つのアプリケーション内のさまざまなプログラムを選択して分析することができます。

プロシージャー

プロシージャー・レベルとして指定できる値は以下のとおりです。

***ALLPRC**

全プロシージャーのパフォーマンス・データが収集されます。

***NONLEAF**

リーフ以外のプロシージャーと PEP についてのパフォーマンス・データが収集されます。

注: *NOLEAF は、ILE COBOL プログラムでは無効です。

第 5 章 サービス・プログラムの作成

サービス・プログラムは、バインドされる ILE プログラム・オブジェクトにサービスを提供する、特殊な種類のシステム・オブジェクトです。

この章では次のことについて説明します。
- サービス・プログラムの使用法
- サービス・プログラム用のバインダー言語コマンドを作成する方法
- CRTSRVPGM コマンドを使用してサービス・プログラムを作成する方法
- サービス・プログラムでデータを呼び出し、共用する方法

WebSphere Development Studio Client for i5/OS を使用してください。 これは推奨
される方法であり、サービス・プログラムの作成に関する説明は、製品のオンライ
ン・ヘルプに記載されています。

クライアント・ツール入門については、『クライアント製品のアプリケーション開発ツールの使用』を参照してください。

サービス・プログラムの定義

サービス・プログラムとは、他の ILE プログラム・オブジェクトおよびサービス・プログラムによって使用される、実行可能なプロシージャーと利用可能なデータ項目の集まりです。サービス・プログラムは、タイプ *SRVPGM のシステム・オブジェクトであり、サービス・プログラムが作成される時点で名前が指定されます。

サービス・プログラムは、サービス・プログラム作成 (CRTSRVPGM) コマンドを使用して作成します。サービス・プログラムは、バインドして実行可能オブジェクトとなる 1 つまたは複数のモジュール・オブジェクトで構成されるという点で、プログラム・オブジェクトに似ています。ただし、サービス・プログラムには PEP がないという点で異なっています。 PEP がないために、呼び出すことも取り消すこともできません。 PEP の代わりに、サービス・プログラムはプロシージャーをエクスポートすることができます。サービス・プログラムの外側から行われる静的プロシージャー呼び出しによって、サービス・プログラムからエクスポートされたプロシージャーの呼び出しのみ可能です。サービス・プログラムのエクスポートは、バインダー言語を使用して定義されます。

サービス・プログラムの詳細については、「*ILE 概念*」を参照してください。

サービス・プログラムの使用

一般に、サービス・プログラムは、アプリケーション・プログラム内でアプリケーション・プログラムから頻繁に呼び出される共通ルーチンのために使用されます。たとえば、ILE COBOL コンパイラーは、サービス・プログラムを使用して、数値計算の機能や入出力ルーチンなどの実行時サービスを提供します。サービス・プログラムを使用することによって、ソース・プログラムの再使用が可能になり、保守

が簡単になり、さらにストレージ要件が小さくなります。多くの点で、サービス・プログラムは、サブルーチン・ライブラリーまたはプロシージャー・ライブラリーと類似しています。

インターフェースを変更しないようにするか、または上位互換でのみ変更するようにする限り、サービス・プログラムは、そのサービス・プログラムを使用する他のプログラム・オブジェクトまたはサービス・プログラムを作成し直すことなく、更新することができます。変更を、サービス・プログラムによって提供されている既存のサポートと互換性のあるものにするかどうかは、プログラマーが制御することです。サービス・プログラムに互換性のある変更を行うには、新しいプロシージャー名またはデータ名の追加はエクスポート・リストの最後にしなければならず、またシグニチャーは同じに保たなければなりません。

ILE COBOL サービス・プログラムのバインダー言語コマンドの作成

バインダー言語を使用することによって、サービス・プログラムからエクスポートできるプロシージャー名およびデータ項目のリストを定義することができます。バインダー言語およびバインダー言語コマンドの詳細な説明については、「*ILE 概念*」を参照してください。

プロシージャーとデータ項目の名前、およびそれらの名前のバインダー言語での指定順序から、**シグニチャー**が生成されます。シグニチャーは、サービス・プログラムによってサポートされるインターフェースを識別する値です。また、バインダー言語内で SIGNATURE パラメーターを使用して、シグニチャーを明示的に指定することもできます。

ILE COBOL ソース・プログラムから作成されるサービス・プログラムの場合、バインダー言語のエクスポート・リスト内に含めることのできるモジュール・エクスポートは、以下の言語エレメントです。

- ILE COBOL プログラムの中で、コンパイル単位の最外部の PROGRAM-ID 段落にある名前。
- プログラムに INITIAL 属性がないなら、ILE COBOL プログラムのうち、コンパイル単位の最外部の PROGRAM-ID 段落にある名前から派生した ILE COBOL コンパイラー生成の名前。この名前は、PROGRAM-ID 段落にある名前に接尾部 `_reset` を追加することによって派生したものです。この名前をエクスポート・リストに含めることが必要なのは、サービス・プログラム内の ILE COBOL プログラムが取り消される必要がある場合だけです。

サービス・プログラム作成 (CRTSRVPGM) コマンドの使用

サービス・プログラムを作成するには、サービス・プログラム作成 (CRTSRVPGM) コマンドを使用します。どんな ILE モジュール・オブジェクトでも、サービス・プログラムの中にバインドできます。モジュール・オブジェクトは、サービス・プログラムを作成する前に存在していなければなりません。 ILE COBOL ソース・プログラムからモジュール・オブジェクトを作成するには、CRTCBLMOD コマンドを使用します。 CRTCBLMOD コマンドを使用してモジュール・オブジェクトを作成する方法については 29 ページの『COBOL モジュールの作成 (CRTCBLMOD) コマンドの使用』を参照してください。

表 7 に、CRTSRVPGM パラメーターとそのデフォルトを示します。 CRTSRVPGM コマンドおよびそのパラメーターの詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** のカテゴリー「プログラミング」の中の『*CL* および *API*』セクションを参照してください。

表 7. CRTSRVPGM コマンドのパラメーターとそのデフォルト値

パラメーター・グループ	パラメーター (デフォルト値)
識別	SRVPGM(*CURLIB/サービス・プログラム名) MODULE(*SRVPGM)
プログラム・アクセス	EXPORT(*SRCFILE) SRCFILE(*LIBL/QSRVSRC) SRCMBR(*SRVPGM)
バインディング	BNDSRVPGM(*NONE) BNDDIR(*NONE)
実行時	ACTGRP(*CALLER)
その他	OPTION(*GEN *NODUPPROC *NODUPVAR *WARN *RSLVREF) DETAIL(*NONE) REPLACE(*YES) AUT(*LIBCRTAUT) ALWUPD(*YES) ALWRINZ(*NO) TEXT(*BLANK) ALWLIBUPD(*NO) USRPRF(*USER) TGTRLS(*CURRENT)

サービス・プログラムの作成の例

この例では、バインダー言語を使用することによって、金融計算を実行するためのサービス・プログラムを作成する方法を示します。

サービス・プログラムを構成するモジュール・オブジェクトは、次の ILE COBOL ソース・プログラムから構成されるものとします。

- RATE

  貸付金額、支払期日、および支払金額を入力すると、利率が計算されます。
- AMOUNT

  利率、支払期日、および支払金額を入力すると、貸付金額が計算されます。
- PAYMENT

  利率、支払期日、および貸付金額を入力すると、支払金額が計算されます。
- TERM

  利率、貸付金額、および支払金額を入力すると、支払期日が計算されます。

1. RATE、AMOUNT、PAYMENT、および TERM の ILE COBOL プログラムを作成するためのサービス・プログラム用バインダー言語は、以下のようになります。

```
FILE: MYLIB/QSRVSRC  MEMBER: FINANCIAL
STRPGMEXP  PGMLVL(*CURRENT)
  EXPORT SYMBOL('TERM')
  EXPORT SYMBOL('RATE')
  EXPORT SYMBOL('AMOUNT')
  EXPORT SYMBOL('PAYMENT')
ENDPGMEXP
```

SEU を使用して、バインダー言語ソース・ステートメントを入力することができます。 BND ソース・タイプを指定すると、SEU 内の構文検査機能は、バインダー言語を入力するようプロンプトで要求し、その入力の妥当性を検査します。バインダー言語ソースを入力するために編集セッションを開始するには、次のように入力します。

```
STRSEU SRCFILE(MYLIB/QSRVSRC) SRCMBR(FINANCIAL)
TYPE(BND) OPTION(2)
```

そして、実行キーを押します。

2. CRTCBLMOD コマンドを使用して 4 つの ILE COBOL ソース・プログラムをモジュール・オブジェクトにコンパイルします。モジュール・オブジェクトの名前も RATE、AMOUNT、PAYMENT、および TERM であるとします。

   サービス・プログラムを作成するには、次のコマンドを使用することによって、必要なバインダー・ステートメントを実行することができます。

```
CRTSRVPGM  SRVPGM(MYLIB/FINANCIAL)
           MODULE(MYLIB/TERM MYLIB/RATE MYLIB/AMOUNT MYLIB/PAYMENT)
           EXPORT(*SRCFILE)
           SRCFILE(MYLIB/QSRVSRC)
           SRCMBR(*SRVPGM)
```

   注:

   a. ライブラリー MYLIB 内のソース・ファイル QSRVSRC は、バインダー言語ソースの含まれているファイルです。

   b. サービス・プログラムを作成する必要があるすべてのモジュール・オブジェクトが MODULE パラメーターで指定されているため、ここではバインディング・ディレクトリーは不要です。

   バインダー言語の使用およびサービス・プログラムの作成に関するさらに多くの例が、「*ILE 概念*」に記載されています。

入力としてのバインダー・ソース検索 (RTVBNDSRC) コマンドの使用

バインダー・ソース検索 (RTVBNDSRC) コマンドを使用すると、1 つのモジュールまたはモジュールの集合からエクスポートを検索し、それらを (そのエクスポートに必要なバインダー言語ステートメントと一緒に) 指定のファイル・メンバーに入れることができます。バインダー言語を検索してソース・ファイル・メンバーに入れた後で、そのバインダー言語を編集して、必要な変更を行えます。このファイル・メンバーを、後で、サービス・プログラム作成 (CRTSRVPGM) コマンドの SRCMBR パラメーターへの入力として使用することができます。

デフォルトでは、CRTSRVPGM コマンドでは、サービス・プログラムからのエクスポートを識別するために EXPORT パラメーターと SRCFILE パラメーターにバイ

ンダー言語ファイルが指定されます。 RTVBNDSRC コマンドは、このバインダー言語を自動的に作成する際に役立ちます。

バインダー・ソース検索 (RTVBNDSRC) コマンドの詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「プログラミング」カテゴリーの中の『*CL および API*』セクションを参照してください。

サービス・プログラム内のエクスポート済み ILE プロシージャーの呼び出し

サービス・プログラム内のエクスポート済み ILE プロシージャーは、静的プロシージャー呼び出しを使用して ILE プログラム・オブジェクトまたは別のサービス・プログラムからのみ呼び出し可能です。

ILE COBOL プログラムからサービス・プログラム内のエクスポート済み ILE プロシージャーを呼び出すには、CALL *リテラル* ・ステートメント (*リテラル* は、サービス・プログラム内の ILE プロシージャーの名前) を使用します。サービス・プログラム内のエクスポート済み ILE プロシージャーを呼び出すために ILE COBOL プログラムの中で CALL ステートメントをコーディングする方法については 248 ページの『CALL リテラルを使用した静的プロシージャー呼び出しの実行』を参照してください。

サービス・プログラムとのデータの共用

外部データは、1 つのサービス・プログラムの中の複数のモジュール・オブジェクトの間で共用したり、複数のサービス・プログラムの間で共用したり、複数のプログラム・オブジェクトの間で共用したり、さらにはサービス・プログラムとプログラム・オブジェクトの間で共用したりできます。

ILE COBOL プログラムでは、異なるモジュール・オブジェクトの間で共用されるデータ項目は、WORKING-STORAGE セクションの EXTERNAL 文節に記述する必要があります。外部データが ILE COBOL プログラム内で使用される方法については 266 ページの『EXTERNAL データの共用』または「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の中の EXTERNAL 文節の部分を参照してください。

サービス・プログラムの ILE COBOL プログラムの中で EXTERNAL として宣言されるデータおよびファイルは、サービス・プログラム用バインダー言語のエクスポート・リストに入れることはできません。サービス・プログラムより外側の ILE COBOL プログラムの中の EXTERNAL として宣言されるデータおよびファイルは、EXTERNAL データおよび EXTERNAL ファイルへの活動化時解決によって、サービス・プログラムの内側にある ILE COBOL プログラムとこのデータを共用することができます。この同じメカニズムによって、同じ活動化グループ内で活動化されるまったく別個の 2 つのプログラム・オブジェクトの間で、EXTERNAL データおよび EXTERNAL ファイルを共用することもできます。

サービス・プログラム内の ILE COBOL プログラムの取り消し

サービス・プログラムの外側からそのサービス・プログラムの一部である ILE COBOL プログラムを取り消すには、バインダー言語のエクスポート・リストに ILE COBOL プログラムの CANCEL プロシージャー名を指定してください。

第 6 章 ILE COBOL プログラムの実行

この章では、ILE COBOL プログラムを実行するために必要なことがらについて説明します。

ILE COBOL プログラムを実行するための最も一般的な方法は、次のとおりです。
- 制御言語 (CL) CALL コマンドを使用する
- 高水準言語 CALL ステートメント (たとえば、ILE COBOL の CALL ステートメント) を使用する
- メニュー方式のアプリケーション・プログラムを使用する
- ユーザー作成コマンドを発行する
- WebSphere Development Studio Client for i5/OS ワークベンチの、メニュー・アクションの「実行」またはツールバー・アイコンの「実行」を選択する WebSphere Development Studio ILE COBOL を使用してください。これが推奨される方法であり、ILE COBOL プログラムの実行に関する資料は、その製品のオンライン・ヘルプにあります。

  クライアント・ツール入門については、『クライアント製品のアプリケーション開発ツールの使用』を参照してください。

CL CALL コマンドを使用した COBOL プログラムの実行

CL CALL コマンドを使用して、ILE COBOL プログラムを実行することができます。バッチ・ジョブの一部として CL CALL コマンドを対話式に使用するか、CL プログラム内に CL CALL コマンドを含めることができます。 CL CALL コマンドの例は、次のとおりです。

```
CALL program-name
```

「プログラム名」によって指定されるプログラム・オブジェクトはライブラリー内に存在していなければならず、そのライブラリーは、ライブラリー・リスト *LIBL に含まれていなければなりません。また、次のように、CL CALL コマンドでライブラリーを明示的に指定することもできます。

```
CALL library-name/program-name
```

CL CALL コマンドの使用法の詳細については、Web サイト http://www.ibm.com/
systems/i/infocenter/ にある **i5/OS Information Center** の「*プログラミング*」カテゴ
リーの中の『*CL および API*』セクションを参照してください。

形式 1 の ACCEPT ステートメントを使用する ILE COBOL プログラムを呼び出すバッチ・ジョブを実行している場合、入力データはジョブ・ストリームから取られます。このデータは、ILE COBOL プログラムの CL CALL のすぐ後に入れる必要があります。プログラムは、(複数の ACCEPT ステートメントによって) 利用可能なデータ量と同じデータ量を要求するようにしなければなりません。詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の『ACCEPT ステートメント』のセクションを参照してください。

利用可能な量より多くのデータが要求されると、データの後の CL コマンドは入力データとして扱われます。利用可能な量より少ないデータが要求されると、余分な各データ行は CL コマンドとして扱われます。これによって、各インスタンスで望ましくない結果になる可能性があります。

CL CALL コマンドによる ILE COBOL プログラムへのパラメーターの受け渡し

実行時に ILE COBOL プログラムにパラメーターを渡すには、CL CALL コマンドの PARM オプションを使用します。

```
CALL PGM(program-name) PARM(parameter-1 parameter-2 parameter-3)
```

各パラメーター値は、次のいずれか 1 つの形式によってのみ指定できます。
- 文字ストリング定数
- 数値定数
- 論理定数
- 倍精度浮動小数点定数
- プログラム変数

パラメーターの処理方法の詳細な説明については、「*CL プログラミング*」のプログラム間でのパラメーターの受け渡しに関するセクションを参照してください。

HLL CALL ステートメントを使用した ILE COBOL プログラムの実行

ILE COBOL プログラムは、別の HLL プログラムから呼び出すことによって実行することができます。

ILE COBOL CALL ステートメントを使用すれば、ILE COBOL プログラム内で、別の ILE COBOL プログラムを呼び出すことができます。 ILE COBOL 呼び出しが動的プログラム呼び出しである場合、プログラム・オブジェクトは、IN LIBRARY 句を使用してライブラリー修飾が可能です。たとえば、ライブラリー LIBNAME にあるプログラム・オブジェクト PGMNAME を呼び出すには、次のように指定します。

```
CALL "PGMNAME" IN LIBRARY "LIBNAME" USING variable1.
```

IN LIBRARY 句がない場合は、ライブラリー・リスト *LIBL を検索してプログラム・オブジェクトを探します。詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の『CALL ステートメント』のセクションを参照してください。

ILE C から ILE COBOL プログラムを実行するには、ILE C 関数呼び出しを使用します。関数の名前は、ILE COBOL プログラムの名前に対応します。デフォルトでは、この関数呼び出しは静的プロシージャー呼び出しになります。動的プログラム呼び出しを実行するには、`#pragma linkage (PGMNAME, OS)` ディレクティブを使用してください。 `PGMNAME` は、ILE C プログラムから実行したい ILE COBOL プログラムの名前を表します。 `#pragma linkage (PGMNAME, OS)` ディレクティブを使用することによって `PGMNAME` が外部プログラムであることを ILE C コンパイラーに知らせたら、ILE C 関数呼び出しによって ILE COBOL プログラムを実行するこ

とができます。 詳細については、「*IBM Rational Development Studio for i: ILE C/C++ Programmer's Guide*」の中の他のプログラムを呼び出すプログラムの作成に関する章を参照してください。

ILE RPG プログラムから ILE COBOL プログラムを実行するには、CALL 命令コードを使用して動的プログラム呼び出しを行うか、または CALLB 命令コードを使用して静的プロシージャー呼び出しを行ってください。演算項目 2 の項目としてその名前を指定することによって呼び出し先プログラムを識別します。詳細については、「*IBM Rational Development Studio for i: ILE RPG プログラマーの手引き*」のプログラムおよびプロシージャーの呼び出しに関する章を参照してください。

C++ から ILE COBOL プログラムを実行するには、C++ 関数呼び出しを使用します。関数の名前は、ILE COBOL プログラムの名前に対応します。 C++ が内部的に関数の名前を変更しないようにするには、すなわち、VisualAge® C++ の関数名がマングルされないようにするには、`extern` キーワードを使用して関数をプロトタイプ化してください。何も戻さず、2 バイトの 2 進数値を 1 つ取り出す ILE COBOL プロシージャーを呼び出す場合、C++ プロトタイプは次のようになります。

```
extern "COBOL" void PGMNAME(short int);
```

同じ COBOL プログラム・オブジェクトを呼び出すには、″OS″ のリンケージを指定します。プロトタイプは次のようになります。

```
extern "OS" void PGMNAME(short int);
```

C++ の関数呼び出しにおける ″COBOL″ のリンケージによって、関数名がマングルされないようになるだけでなく、ILE COBOL のプロシージャーに渡される引き数が BY REFERENCE で渡されるようになります。 ILE COBOL プロシージャーが BY VALUE パラメーターの指定を予期している場合は、″C″ のリンケージを指定する必要があります。

メニュー方式のアプリケーションからの ILE COBOL プログラムの実行

ILE COBOL プログラムは、メニュー方式のアプリケーション・プログラムからも実行することができます。ワークステーション・ユーザーは、メニューからオプションを選択して、該当するプログラムを呼び出します。次の図は、アプリケーション・プログラム・メニューの例です。

```
                        PAYROLL DEPARTMENT MENU

   1.  Inquire into employee master
   2.  Change employee master
   3.  Add new employee
   4.  Return


 Option:
```

図 30. アプリケーション・プログラム・メニューの例

多くの場合、この図に示されているメニューは、各オプションが別々の COBOL プログラムを呼び出すようになっている CL プログラムによって表示されます。

上記の PAYROLL DEPARTMENT MENU のディスプレイ・ファイルの DDS は、次のようなものです。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* MENU PAYROLLD PAYROLL DEPARTMENT MENU
     A
     A          R MENU                        TEXT('PAYROLL DEPARTMENT MENU')
     A                                      1 29'PAYROLL DEPARTMENT MENU'
     A                                       5  4'1.  Inquire into employee master'
     A                                       6  4'2.  Change employee master'
     A                                       7  4'3.  Add new employee'
     A                                       8  4'4.  Return'
     A                                      12  2'Option:'
     A            RESP                      12 10VALUES(1 2 3 4)
     A                                           DSPATR(MDT)
```

図 31. アプリケーション・プログラム・メニューのデータ記述仕様

図 31 は 129 ページの図 30 に図示されているアプリケーション・プログラム・メニューの CL プログラムの例です。

```
           PGM /* PAYROLL Payroll Department Menu */
           DCLF FILE (PAYROLLD)
     START: SNDRCVF RCDFMT(MENU)
           IF (&RESP=1); THEN(CALL CBLINQ)
           /* Inquiry */
           ELSE +
             IF (&RESP=2); THEN(CALL CBLCHG)
             /* Change */
             ELSE +
               IF (&RESP=3); THEN(CALL CBLADD)
               /* Add */
               ELSE +
                 IF (&RESP=4); THEN(RETURN)
                 /* Return */
           GOTO START
           ENDPGM
```

図 32. ILE COBOL プログラムを呼び出す CL プログラムの例

ユーザーがアプリケーション・プログラム・メニューから 1、2、または 3 を入力すると図 32 の CL プログラムは、それぞれ ILE COBOL プログラム CBLINQ、CBLCHG、または CBLADD を呼び出します。ユーザーがアプリケーション・プログラム・メニューから 4 を入力すると、CL プログラムはそれを呼び出した元のプログラムに戻ります。

ユーザー作成コマンドを使用した ILE COBOL プログラムの実行

コマンド定義を使用することによって、ILE COBOL プログラムを実行するためのコマンドを自分で作成することもできます。**コマンド定義**はコマンドの定義 (コマンド名、パラメーター記述、および妥当性検査情報を含む) が入っているオブジェクトであり、コマンドによって要求される機能を実行するプログラムを識別します。システムが認識するオブジェクト・タイプは *CMD です。

たとえば、プログラム PAYROLL を呼び出すコマンド PAY を作成することができます。 PAYROLL は、呼び出されて実行される ILE COBOL プログラムの名前で

す。対話式に、またはバッチ・ジョブでコマンドを入力することができます。コマンド定義の使用法の詳細については、「*CL プログラミング*」を参照してください。

ILE COBOL プログラムの終了

ILE COBOL プログラムが正常に終了すると、システムは呼び出し側に制御を戻します。呼び出し側は、ワークステーション・ユーザー、CL プログラム (メニュー処理プログラムなど)、または別の HLL プログラムなどがあります。

ILE COBOL プログラムが実行時に異常終了すると、次のエスケープ・メッセージ CEE9901 が実行単位の呼び出し側に出されます。

アプリケーション・エラーです。メッセージ *ID* が *プログラム名* により、ステートメント *ステートメント番号*、命令 *命令番号* でモニターされていませんでした。

CL プログラムは、メッセージ・モニター (MONMSG) コマンドを使用して、この例外をモニターすることができます。 制御言語コマンドの詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「*プログラミング*」カテゴリーの中の『*CL および API*』セクションを参照してください。

プログラムの終了原因が、

- STOP RUN ステートメントの使用
- メイン・プログラムでの GOBACK ステートメントの使用
- メイン・プログラムでの EXIT-PROGRAM AND CONTINUE RUN UNIT ステートメントの使用
- プログラムの終わりに達したこと

などでなければ、RTNCDE ジョブ属性は 2 に設定されます。

戻りコードの詳細については、「*CL プログラミング*」の RTVJOBA および DSPJOB コマンドの説明を参照してください。

実行時照会メッセージへの応答

ILE COBOL プログラムを実行したときに、実行時照会メッセージが生成されることがあります。このメッセージは、プログラムが実行を続行する前に応答を必要とします。

システム応答リストに照会メッセージを追加して、メッセージに対する自動応答を提供することができます。これらのメッセージに対する応答は、個々にまたは共通に指定することができます。この照会メッセージへの応答の方式は、これがなければオペレーターが応答を出すことが必要になるようなバッチ・プログラムに特に適しています。

システム応答リストには、次の ILE COBOL 照会メッセージを追加することができます。

- LNR7200
- LNR7201
- LNR7203
- LNR7204

- LNR7205
- LNR7206
- LNR7207
- LNR7208
- LNR7209
- LNR7210
- LNR7211
- LNR7212
- LNR7213
- LNR7214
- LNR7604

応答リストが使用されるのは、照会メッセージ応答 (INQMSGRPY) 属性が INQMSGRPY(*SYSRPYL) として指定されているジョブによって送られる場合だけです。

INQMSGRPY パラメーターは、次の CL コマンドで指定できます。

- ジョブ変更 (CHGJOB)
- ジョブ記述変更 (CHGJOBD)
- ジョブ記述の作成 (CRTJOBD)
- ジョブの発行 (SBMJOB)

INQMSGRPY パラメーターに次の値の 1 つを指定することによって、4 つの応答モードのいずれか 1 つを選択することができます。

SAME 応答が照会メッセージに送信されるという方法に変更はありません。

RQD すべての照会メッセージには、照会メッセージの受け取り側による応答が必要です。

DFT デフォルト応答が出されます。

SYSRPYL システム応答リストの中に一致する応答リスト項目があるかどうかを調べます。一致するものがあるなら、その項目の応答値が使用されます。その照会メッセージのための項目が存在しない場合、応答が必要となります。

システム応答リスト項目追加 (ADDRPYLE) コマンドを使用してシステム応答リストに項目を追加したり、システム応答リスト項目の処理 (WRKRPYLE) コマンドを使用してシステム応答リスト内の項目を変更または除去したりすることができます。 また、ユーザー定義のエラー・ハンドラーを使用して、実行時照会メッセージに応答することもできます。

ADDRPYLE および WRKRPYLE コマンド、およびエラー処理 API の詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「*プログラミング*」カテゴリーの中の『*CL および API*』セクションを参照してください。

第 7 章 プログラムのデバッグ

デバッグを実行することにより、プログラム内のエラーを検出したり、診断したり、除去したりすることができます。

WebSphere Development Studio の統合 i5/OS デバッガーを使用してください。これが推奨される方法であり、ILE COBOL プログラムのデバッグに関する資料は、その製品のオンライン・ヘルプにあります。

統合 i5/OS デバッガーを使用すると、ワークステーション上のグラフィカル・ユーザー・インターフェースから、 i5/OS 上で実行しているプログラムのデバッグを行うことができます。また、デバッガーの実行前に、ソース中に直接停止点を設定することができます。統合 i5/OS デバッガーのクライアント・ユーザー・インターフェースを使用すると、プログラムの実行も制御することができます。 例えば、プログラムの実行、行の設定、監視ができ、またエントリー・ポイント・ブレークポイントの設定、プログラム命令のステップスルー、変数の値の表示、およびコール・スタック内容の表示ができます。また、たとえ異なる言語で書かれている場合でも、複数のアプリケーションを単一のデバッガー・ウィンドウ上でデバッグできます。デバッグする各セッションは、「デバッグ」ビュー内に個別に表示されます。

OPM および ILE COBOL プログラムは、ILE ソース・デバッガーを使用してもデバッグすることができます。この章では、ILE ソース・デバッガーの使用方法について説明します。

- ILE COBOL プログラムをデバッグするための準備
- デバッグ・セッションの開始
- デバッグ・セッションへのプログラムの追加およびデバッグ・セッションからのプログラムの除去
- デバッグ・セッションからのプログラム・ソースの表示
- 条件付き停止点および無条件停止点の設定と除去
- 監視条件の設定と除去
- プログラムのステップスルー
- 変数、レコード、グループ項目、および配列の値の表示
- 変数の値の変更
- 参照有効範囲の変更
- 変数、式、またはデバッグ・コマンドに短縮名を付ける

プログラムのデバッグおよびテストを行う時は、必ずライブラリー・リストを変更して、テスト・データを内容とするテスト・ライブラリーをプログラムの入出力先となるようにすることによって、既存の実データに影響を与えないようにしてください。

次の CL コマンドのどれかを使用すると、実行用ライブラリー内のデータベース・ファイルを間違って修正してしまうことを防ぐことができます。

- デバッグ開始 (STRDBG) コマンドを使用し、UPDPROD(*NO) パラメーターを指定する。

- デバッグの変更 (CHGDBG) コマンドを使用し、UPDPROD パラメーターに *NO 値を指定する。
- 「モジュール・ソースの表示」画面で SET デバッグ・コマンドを使用する。 ファイルを修正できないようにするための構文は次のとおりです。

  ```
  SET UPDPROD NO
  ```

  次のように短縮することもできます。

  ```
  SET U N
  ```

ILE ソース・デバッガーの詳細について (プログラム・オブジェクトをデバッグするために必要な権限、最適化レベルの効果を含む) は、「*ILE 概念*」のデバッグに関する章を参照してください。

ILE ソース・デバッガー

ILE ソース・デバッガーを使用して、プログラム・オブジェクトおよびサービス・プログラムのエラーを検出および除去できます。デバッグ・データの含まれている ILE プログラムでデバッグ・コマンドを使用すると、次のことが可能です。

- ILE COBOL または ILE 言語混在アプリケーションのデバッグ
- プログラム実行中における、デバッグ・コマンドを使用したプログラムのフローのモニター
- プログラム・ソースの表示またはデバッグ・ビューの変更
- 条件付き停止点および無条件停止点の設定と除去
- 監視条件の設定と除去
- 指定した数のステートメントのステップスルー
- 変数、レコード、グループ項目および配列の値の表示または変更

注: ILE COBOL COLLATING SEQUENCE は、ILE ソース・デバッガーではサポートされていません。 ILE COBOL プログラム内で ILE COBOL COLLATING SEQUENCE 文節を使用して照合順序を指定していても、ILE ソース・デバッガーはこの照合順序を使用しません。

停止点またはステップ・コマンドによってプログラムが停止した場合、プログラムが停止した個所において関連しているモジュール・オブジェクトのビューが表示されます。この時点で、さらに他のデバッグ・コマンドを入力することができます。

ソース・デバッガーを使用するためには、その前に CRTCBLMOD コマンドまたは CRTBNDCBL コマンドを使用してモジュール・オブジェクトまたはプログラム・オブジェクトを作成する際に、DBGVIEW パラメーターの値を *NONE 以外のものに指定しなければなりません。デバッガーの開始後に、停止点などの ILE ソース・デバッガー・オプションを設定してからプログラムを実行できます。

デバッグ・コマンド

ILE ソース・デバッガーでは、数多くのデバッグ・コマンドを使用することができます。デバッグ・コマンドとそのパラメーターは、「モジュール・ソースの表示」および「評価式」画面の下部に表示されるデバッグ・コマンド行に入力します。こ

れらのコマンドは、大文字、小文字、または大文字小文字混合のどれでも入力できます。デバッグ・コマンドの詳細については、「*ILE 概念*」を参照してください。

注: デバッグ・コマンド行で入力されるデバッグ・コマンドは、CL コマンドではありません。

表 8 に、これらのデバッグ・コマンドを要約しています。 ILE ソース・デバッガーのオンライン・ヘルプでは、デバッグ・コマンドとそれに使用できる省略形について説明されています。

表 8. ILE ソース・デバッガー・コマンド

デバッグ・コマンド	説明
ATTR	変数の属性が表示されるようにします。属性とは、デバッグ記号テーブルに記録されている変数のサイズおよびタイプです。属性と、ILE COBOL でのそれに相当するものを示すリストについては 136 ページの表 9 を参照してください。それらの属性は、ILE COBOL によって定義された属性と同じではありません。
BREAK	テストするプログラムの特定の場所へ条件付きまたは無条件のジョブ停止点を入れられるようにします。 条件付きジョブ停止点を入れるには、BREAK *位置* WHEN *式* を使用します。
CLEAR	条件付きまたは無条件停止点の除去または活動中の監視条件の 1 つもしくはすべてを除去します。
DISPLAY	EQUATE コマンドを使用して割り当てた名前と定義を表示することができます。また、現在「モジュール・ソースの表示」画面に示されているもの以外のソース・モジュールを表示することができます。現行のプログラム・オブジェクト内にモジュール・オブジェクトが存在していなければなりません。
EQUATE	式、変数、またはデバッグ・コマンドに短縮名を割り当てることができます。
EVAL	変数の値を表示または変更することができます。さらに、式、レコード、グループ項目、または配列の値を表示することができます。
QUAL	後続の EVAL または WATCH コマンドに現れる変数の有効範囲を定義できます。
SET	実働ファイルの更新を可能にしたり、検索操作で大文字小文字を区別するかどうかの指定をしたり、または OPM ソース・デバッグ・サポートを使用可能にしたりする、デバッグ・オプションを変更することができます。
STEP	デバッグするプログラムの 1 つまたは複数のステートメントを実行します。
TBREAK	テストするプログラムの特定の場所へ現行スレッドの条件付きまたは無条件停止点を入れられるようにします。
THREAD	「デバッグ済みスレッドの処理」画面の表示または現行スレッドの変更ができます。
WATCH	指定した保管場所の内容が現行値から変更された時点で停止点を要求します。
FIND	現在表示されているモジュールを前方検索して、指定した行番号、ストリング、テキストを探します。

表 8. ILE ソース・デバッガー・コマンド (続き)

デバッグ・コマンド	説明
UP	ソースの表示ウィンドウを、入力された量だけ上に移動します。
DOWN	ソースの表示ウィンドウを、入力された量だけ下に移動します。
LEFT	ソースの表示ウィンドウを、入力された文字数分だけ左に移動します。
RIGHT	ソースの表示ウィンドウを、入力された文字数分だけ右に移動します。
TOP	最初の行が表示されるまで移動します。
BOTTOM	最後の行が表示されるまで移動します。
NEXT	ソースのうち、画面に現在表示されている位置の次の停止点を表示します。
PREVIOUS	ソースのうち、画面に現在表示されている位置の直前の停止点を表示します。
HELP	使用可能なソース・デバッガー・コマンドのオンライン・ヘルプ情報を表示します。

変数の属性

ILE ソース・デバッガーの変数の属性の記述方法は、ILE COBOL とは異なります。 表 9 に、ILE ソース・デバッガーで記述される変数の属性と、それに相当する ILE COBOL データ・カテゴリーを示します。

表 9. ILE ソース・デバッガーの変数属性とそれに相当する ILE COBOL データ・カテゴリー

ILE ソース・デバッガー変数の属性	ILE COBOL データ・カテゴリー
FIXED LENGTH STRING	英字 英数字 英数字編集 数字編集 外部浮動小数点 日付 時刻 タイム・スタンプ
GRAPHIC	DBCS DBCS 編集
CHAR	ブール
INTEGER	2 進数
CARDINAL	符号なしネイティブ 2 進数 (USAGE COMP-5)
ZONED(2,0)	ゾーン 10 進数
PACKED(2,0)	パック 10 進数 パック日付 パック時刻
PTR	ポインター プロシージャー・ポインター
REAL	内部浮動小数点

デバッグ・セッションのためのプログラム・オブジェクトの準備

ILE ソース・デバッガーを使用する前に、DBGVIEW パラメーターを指定した CRTCBLMOD または CRTBNDCBL コマンドを使用する必要があります。

デバッグする ILE COBOL モジュール・オブジェクトごとに、3 つのビューのうちのいずれかを作成することができます。それらのビューは、以下のものです。

- リスト・ビュー
- ソース・ビュー
- ステートメント・ビュー

注: ILE ソース・デバッガーを使用してデバッグするためには、OPM プログラムは、OPTION(*SRCDBG) または OPTION(*LSTDBG) 指定でコンパイルする必要があります。詳細については 139 ページの『ILE ソース・デバッガーの開始』を参照してください。

リスト・ビューの使用

リスト・ビューは、ILE COBOL コンパイラーで作成したコンパイル・リストまたはスプール・ファイルのソース・リストの一部に似たものです。

リスト・ビューを使用して ILE COBOL モジュール・オブジェクトをデバッグするには、モジュール・オブジェクトまたはプログラム・オブジェクトを作成する際に、CRTCBLMOD コマンドまたは CRTBNDCBL コマンドのどちらかの DBGVIEW パラメーターに *LIST または *ALL 値を指定します。

リスト・ビューを作成する 1 つの方法として、次のように入力します。

```
CRTCBLMOD MODULE(MYLIB/xxxxxxxx)
SRCFILE(MYLIB/QCBLLESRC) SRCMBR(xxxxxxxx)
TEXT('CBL Program') DBGVIEW(*LIST)
```

CRTCBLMOD コマンドまたは CRTBNDCBL コマンドに DBGVIEW(*LIST) を指定してリスト・ビューを生成した場合、作成されるモジュール・オブジェクトの大きさは、リスト・ビューにより大きくなります。 リスト・ビューでは、モジュール・オブジェクトまたはプログラム・オブジェクトの作成時に ILE COBOL コンパイラーによってなされたすべての拡張 (たとえば、COPY および REPLACE ステートメント) が提供されます。リスト・ビューは、ソース・メンバーとは独立して存在しています。ソース・メンバーは、リスト・ビューに影響を与えることなく変更や削除を行うことができます。

ソース・メンバーに複数のコンパイル単位が含まれている場合、リスト・ビューには、その 1 つだけをデバッグする場合でも、すべてのコンパイル単位のソース・リストが含まれます。しかし、「モジュール・ソースの表示」画面から出されるデバッグ・コマンドは、デバッグの対象のコンパイル単位だけに適用されます。

ソース・ビューの使用

ソース・ビューには、ソース・メンバーのソース・ステートメントの参照が含まれます。

ソース・ビューを ILE ソース・デバッガーで使用するために、ILE COBOL コンパイラーは、モジュール・オブジェクト (*MODULE) の作成時にソース・メンバーへの参照を作成します。

注: モジュール・オブジェクトは、ソース・ステートメントをビューにコピーするのではなく、ルート・ソース・メンバー内のソース・ステートメントの位置を参照して作成されます。したがって、モジュールを作成してから、ルート・ソース・メンバーに基づいて作成したモジュールをデバッグするまでの間は、このルート・ソース・メンバーを修正したり名前を変更したり移動したりしないでください。

ソース・ビューを使用して ILE COBOL モジュール・オブジェクトをデバッグするためには、CRTCBLMOD コマンドまたは CRTBNDCBL コマンドの DBGVIEW パラメーターに値 *SOURCE または *ALL を指定します。

ソース・ビューを作成する 1 つの方法として、次のように入力します。

```
CRTCBLMOD MODULE(MYLIB/xxxxxxxx)
SRCFILE(MYLIB/QCBLLESRC) SRCMBR(xxxxxxxx)
TEXT('CBL Program') DBGVIEW(*SOURCE)
```

CRTCBLMOD コマンドまたは CRTBNDCBL コマンドに DBGVIEW(*SOURCE) を指定してソース・ビューを生成した場合、作成されるモジュール・オブジェクトのサイズはソース・ビューのために大きくはなりますが、リスト・ビューで生成される場合よりは小さくなります。生成されるモジュール・オブジェクトのサイズは、ステートメント・ビューの場合と同じです。 ソース・ビューでは、モジュール・オブジェクトまたはプログラム・オブジェクトの作成時に ILE COBOL コンパイラーによってなされた拡張は提供されません。ソース・ビューは、ソース・メンバーが変更されずに存在していることを前提に作成されます。ソース・メンバーに変更が加えられると、ソース・ビューも影響を受けます。

ソース・メンバーに複数のコンパイル単位が含まれている場合、ソース・ビューには、その 1 つだけをデバッグする場合でも、すべてのコンパイル単位のソース・コードが含まれます。しかし、「モジュール・ソースの表示」画面から出されるデバッグ・コマンドは、デバッグの対象のコンパイル単位だけに適用されます。

ステートメント・ビューの使用

ステートメント・ビューには、ソース・ステートメントは含まれません。それには行番号とステートメント番号が含まれています。 ステートメント・ビューを使用して ILE COBOL モジュール・オブジェクトをデバッグするには、コンパイラー・リストのハード・コピーが必要です。

注: ステートメント・ビューを使用して ILE COBOL モジュール・オブジェクトをデバッグする場合、「モジュール・ソースの表示」画面にソース・コードは表示されません。

ステートメント・ビューを使用して ILE COBOL モジュール・オブジェクトをデバッグするには、モジュールの作成時に、CRTCBLMOD または CRTBNDCBL コマンドの DBGVIEW パラメーターに *STMT、*SOURCE、*LIST または *ALL 値を指定します。

ステートメント・ビューを作成する 1 つの方法として、次のように入力します。

```
CRTCBLMOD MODULE(MYLIB/xxxxxxxx)
SRCFILE(MYLIB/QCBLLESRC) SRCMBR(xxxxxxxx)
TEXT('CBL Program') DBGVIEW(*STMT)
```

CRTCBLMOD コマンドまたは CRTBNDCBL コマンドに DBGVIEW(*STMT) を指定してステートメント・ビューを生成した場合、作成されるモジュール・オブジェクトのサイズは、ステートメント・ビューであるために最小限の大きさになります。作成されるモジュール・オブジェクトのサイズは、リスト・ビューまたはソース・ビューで生成されるモジュール・オブジェクトのサイズよりも小さくなります。ステートメント・ビューでは、作成されるモジュール・オブジェクトのサイズは最小限のものになりますが、デバッグは可能です。ステートメント・ビューでは、記号テーブル、およびステートメント番号とデバッグ行番号とのマッピングだけが提供されます。

ILE ソース・デバッガーの開始

デバッグ・ビューを作成したなら、アプリケーションのデバッグを開始することができます。

ILE ソース・デバッガーを開始するには、デバッグの開始 (STRDBG) コマンドを使用します。デバッガーを一度開始すると、デバッグの終了 (ENDDBG) コマンドを入力するまで活動化されたままです。ジョブの中でデバッグ変更 (CHGDBG) コマンドを使用することにより、後からデバッグ・モードの属性を変更することができます。

表 10 に、STRDBG および CHGDBG コマンドのパラメーターとそのデフォルト値を示します。 ENDDBG コマンドには、関連したパラメーターがありません。 STRDBG、CHGDBG、および ENDDBG コマンドおよびそのパラメーターの詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「*プログラミング*」カテゴリーの中の『*CL および API*』セクションを参照してください。

表 10. STRDBG コマンドおよび CHGDBG コマンドのパラメーターとそのデフォルト値

パラメーター・グループ	STRDBG コマンド・パラメーター (デフォルト値)	CHGDBG コマンド・パラメーター (デフォルト値)
識別	PGM(*NONE) DFTPGM(*PGM)	DFTPGM(*SAME)
トレース	MAXTRC(200) TRCFULL(*STOPTRC)	MAXTRC(*SAME) TRCFULL(*SAME)
その他	UPDPROD(*NO) OPMSRC(*NO) SRVPGM(*NONE) CLASS(*NONE) DSPMODSRC(*PGMDEP) SRCDBGPGM(*SYSDFT) UNMONPGM(*NONE)	UPDPROD(*SAME) OPMSRC(*SAME)

注: トレースが適用されるのは OPM プログラムだけです。 ILE プログラムおよびサービス・プログラムには適用されません。

プログラム (PGM) パラメーターを指定した STRDBG コマンドを使用することにより、最初にプログラム・オブジェクトを 20 個までデバッグ・セッションに追加することができます。 (OPM プログラムのコンパイル方法およびデバッグ環境の設定値に応じて、 ILE ソース・デバッガーを使用してそれらのプログラムをデバッグすることができます)。任意の ILE プログラムまたは OPM プログラムの組み合わせが可能です。

STRDBG コマンドの PGM パラメーターには、プログラム・オブジェクトしか指定できません。 STRDBG コマンドのサービス・プログラム (SRVPGM) パラメーターを使用すると、最初に、最大 20 のサービス・プログラムをデバッグ・セッションに追加できます。追加のサービス・プログラムは、デバッグ・セッションを開始した後でデバッグ・セッションに追加することができます。また、STRDBG コマンドのサービス・プログラム (SRVPGM) パラメーターを使用して、最初に、最大 20 のサービス・プログラム・オブジェクトをデバッグ・セッションに追加することもできます。サービス・プログラムのデバッグの規則は、プログラムのデバッグの場合と同じです。

- プログラムまたはサービス・プログラムはデバッグ・データをもっていなければならない
- プログラムまたはサービス・プログラムをデバッグ・セッションに組み込むためには、プログラマーがそのプログラムに対して *CHANGE 権限を持っている必要がある

STRDBG コマンドで指定した最初のプログラムが表示されるのは、そのプログラムにデバッグ・データがあり、しかも OPM の OPMSRC パラメーターが *YES の場合です。 ILE では、入力モジュールにデバッグ・データがある場合にその入力モジュールが表示されます。デバッグ・データがないなら、デバッグ・データを使用して ILE プログラムにバインドされた最初のモジュールが表示されます。

ILE ソース・デバッガーを使用して OPM プログラムをデバッグするには、次の条件が満たされていなければなりません。

1. OPM プログラムのコンパイルが OPTION(*LSTDBG) または OPTION(*SRCDBG) を指定して行われていること。 (3 つの OPM 言語である、RPG、COBOL、CL をサポートしています。 RPG および COBOL プログラムは *LSTDBG や *SRCDBG でコンパイル可能ですが、CL プログラムは *SRCDBG でなければコンパイルできません。)
2. ILE デバッグ環境は、OPM プログラムを受け入れられるように設定されています。そのためには、STRDBG コマンドで OPMSRC(*YES) を指定します。 (システム・デフォルトは OPMSRC(*NO) です。)

これら 2 つの条件が満たされていない場合に OPM プログラムをデバッグするには、OPM システム・デバッガーを使用する必要があります。

*LSTDBG または *SRCDBG を使用せずにコンパイルされた OPM プログラムが指定され、しかもサービス・プログラムが指定された場合、そのサービス・プログラムは、デバッグ・データをもっていれば表示されます。 デバッグ・データがなければ DSPMODSRC 画面は空になります。ILE プログラムとサービス・プログラムが指定された場合には、ILE プログラムが表示されます。

STRDBG の例

たとえば、サンプルのデバッグ・プログラム MYPGM1 と、呼び出し先 OPM プログラム MYPGM2 のデバッグ・セッションを開始するには、次のように入力します。

```
STRDBG PGM(TESTLIB/MYPGM1 MYLIB/MYPGM2) OPMSRC(*YES)
```

注: プログラム・オブジェクトをデバッグ・セッションに追加するには、そのプログラム・オブジェクトに対する *CHANGE 権限が必要です。

STRDBG コマンドを入力すると、「モジュール・ソースの表示」画面が表示されます。 ILE プログラムと ILE デバッガーの使用可能な OPM プログラムとの組み合わせを STRDBG コマンドに指定した場合、デバッグ・データのある最初のプログラムが表示されます。その最初のプログラムが ILE プログラムの場合、デバッグ・データを使用してプログラム・オブジェクトにバインドされた最初のモジュール・オブジェクトが、図 33 のように表示されます。

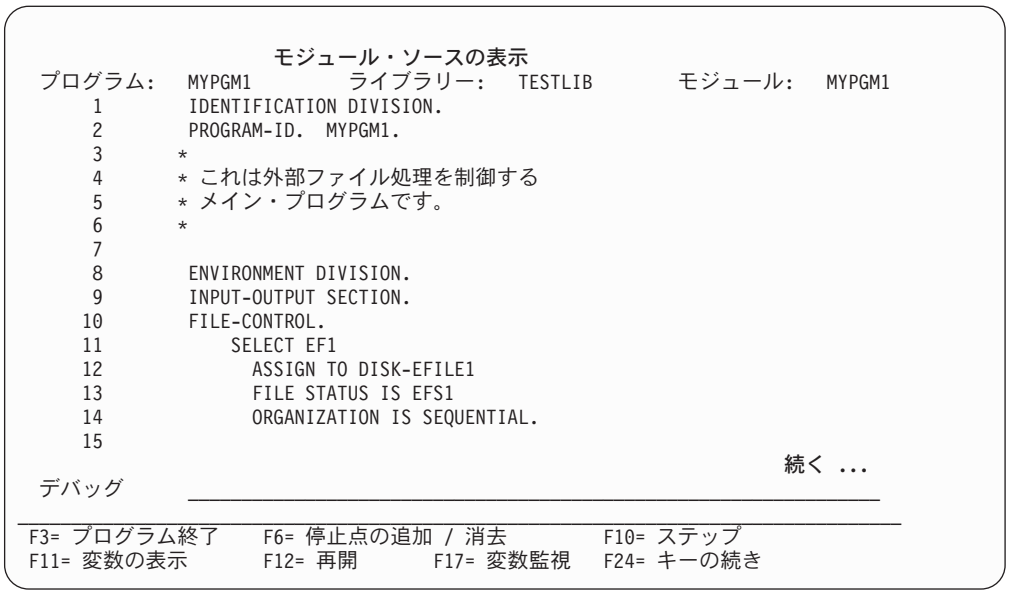


図 33. デバッグ・セッションの開始

デバッグ・オプションの設定

デバッグ・セッションの開始後、SET デバッグ・コマンドをデバッグ・コマンド入力行に入力することによって、次のようなデバッグ・オプションを設定または変更できます。

- プログラムのデバッグ中にデータベース・ファイルを更新できるようにするかどうか。 (このオプションは、STRDBG コマンドの UPDPROD パラメーターと対応しています)。
- FIND によるテキスト検索で大文字小文字を区別するかどうか。
- ILE ソース・デバッガーを使用して OPM プログラムをデバッグするかどうか。 (このオプションは OPMSRC パラメーターに対応するものです。)

デバッグ・コマンド SET を使用してデバッグ・オプションを変更すると、対応するパラメーターが STRDBG コマンドに指定されていれば、このパラメーターの値に影響を与えます。また、デバッグの変更 (CHGDBG) コマンドを用いてデバッグ・オプションを設定することもできます。ただし、CHGDBG コマンドによって OPMSRC オプションを変更することはできません。 OPMSRC を変更できるのは、SET デバッグ・コマンドだけです。

ILE プログラムでデバッグ・セッションを行っている際に、デバッグ・データの使用可能な OPM プログラムもデバッグしたい場合があります。その場合、ILE ソース・デバッガーが OPM プログラムを受け入れられるようにするには、次のようにします。

1. 現在の画面が「モジュール・ソースの表示」画面ではない 場合、STRDBG を入力した後で、次のように入力します。

   ```
   DSPMODSRC
   ```

2. 次のように入力します。

   ```
   SET
   ```

   「デバッグ・オプションの設定」画面が表示されます。

3. この画面の「*OPM* ソース・デバッグ・サポート」フィールドに Y と入力してから、実行キーを押して「モジュール・ソースの表示」画面に戻ります。

これで、「モジュールの処理」画面を使用することによって、またはそのプログラムの呼び出しステートメントを処理することによって、OPM プログラムを追加することができます。

デバッグ・セッションでのプログラム・オブジェクトの実行

デバッグ・セッションを開始した後、「モジュール・ソースの表示」画面で次のキーを押すことにより、デバッグ・セッションでプログラム・オブジェクトを実行することができます。

- F3 (プログラムの終了)、
- F12 (再開)、または
- F21 (コマンド行)

次に、CALL CL コマンドを使用してコマンド行からプログラム・オブジェクトを呼び出します。

デバッグ・セッション中にプログラム・オブジェクト内で例外が生じた場合、その例外はプログラム・オブジェクトに指定されたエラーおよび例外処理ルーチンにより処理されます。例外が例外処理ルーチンによって処理される前に機能チェックに変更された場合、デバッガーが呼び出され、「モジュール・ソースの表示」画面が表示されます。例外が生じたモジュール・オブジェクトが、例外の原因であるステートメントに表示されます。エラーおよび例外処理の詳細については 415 ページの『第 16 章 ILE COBOL のエラーおよび例外の処理』を参照してください。

停止点を設定するか、「モジュール・ソースの表示」画面で F3 (プログラムの終了) を押すと、プログラムの実行を停止することができます。停止点の設定については 148 ページの『停止点の設定と除去』を参照してください。

プログラム・オブジェクトおよびサービス・プログラムのデバッグ・セッションへの追加

デバッグ・セッションの開始後、さらにプログラム・オブジェクトとサービス・プログラムをセッションに追加することができます。

ILE プログラム・オブジェクトおよびサービス・プログラムをデバッグ・セッションに追加するには、オプション 1 (プログラムの追加) を使用し、「モジュール・リストの処理」画面の最初の行にプログラム・オブジェクトの名前を入力します(144 ページの図 34 を参照)。「モジュール・リストの処理」画面は、「モジュール・ソースの表示」画面で F14 (モジュール・リストの処理) を押すと表示されます。サービス・プログラムを追加するには、デフォルト・プログラム・タイプを *PGM から *SRVPGM に変更します。デバッグ・セッションに含めることのできる ILE プログラム・オブジェクトとサービス・プログラムの数に制限はなく、いつでもそれを実行できます。

OPM プログラム・オブジェクトをデバッグ・セッションに追加するには、OPMSRC の指定値に応じて 2 とおりの方法があります。 STRDBG、SET デバッグ・コマンド、または CHGDBG を使用することにより OPMSRC(*YES) を指定した場合、「モジュール・リストの処理」画面で OPM プログラムを追加します。(OPM プログラムのモジュール名はリストされないことに留意してください。) OPMSRC(*YES) が指定されていれば、デバッグ・セッションに同時に含めることのできる OPM プログラムの数に制限はありません。 OPMSRC(*NO) を指定した場合、プログラムの追加 (ADDPGM) コマンドを使用する必要があります。 OPMSRC(*NO) が指定されているなら、デバッグ・セッションに含めることのできる OPM プログラムは 20 個までに限定されます。

注: デバッグ・データを含む OPM プログラムを、 ILE デバッグ・セッションと OPM デバッグ・セッションの両方からデバッグすることはできません。 OPM プログラムがすでに OPM デバッグ・セッションにある場合、まずそれをそのセッションから除去してからでなければ、そのプログラムを ILE デバッグ・セッションに追加したり、呼び出しステートメントからそれに入ることはできません。同様に、そのプログラムを OPM デバッグ・セッションからデバッグする場合も、まずそれを ILE デバッグ・セッションから除去する必要があります。

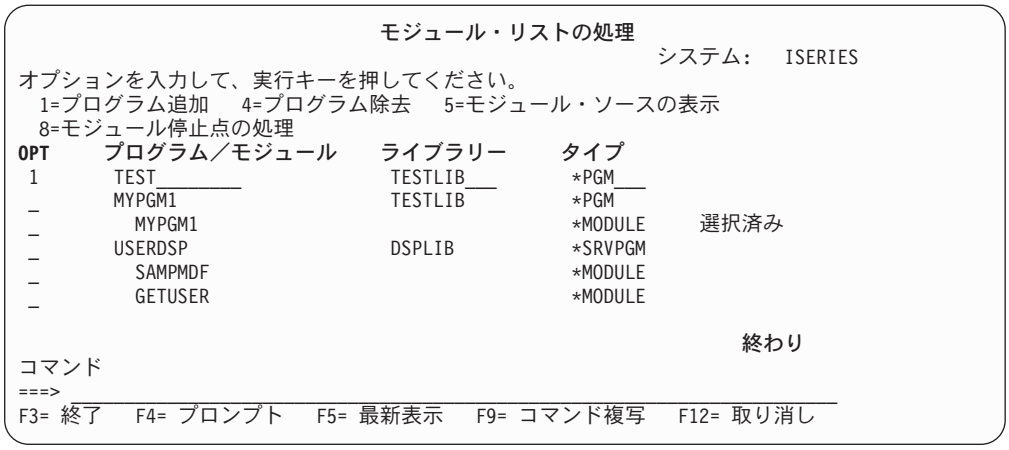


図 34. ILE プログラム・オブジェクトをデバッグ・セッションに追加する

プログラム・オブジェクトまたはサービス・プログラムのデバッグ・セッションへの追加が終了したら、「モジュール・リストの処理」画面で F3 (終了) を押すと、「モジュール・ソースの表示」画面に戻ります。

注: プログラムをデバッグ・セッションに追加するには、そのプログラムに対する *CHANGE 権限が必要です。 ILE サービス・プログラムをデバッグ・セッションに追加するには、「モジュール・リストの処理」画面でオプション 1 を使用する方法しかありません。 ILE サービス・プログラムを STRDBG コマンドに指定することはできません。

プログラム・オブジェクトまたはサービス・プログラムをデバッグ・セッションから除去する

セッションの開始後、デバッグ・セッションからプログラム・オブジェクトまたはサービス・プログラムを除去することができます。

ILE プログラム・オブジェクトおよびサービス・プログラムをデバッグ・セッションから除去するには、「モジュール・リストの処理」画面で、除去したいプログラム・オブジェクトまたはサービス・プログラムの横に、オプション 4 (プログラムの除去) を指定します (145 ページの図 35 を参照)。「モジュール・リストの処理」画面は、「モジュール・ソースの表示」画面で F14 (モジュール・リストの処理) を押すと表示されます。

OPM プログラム・オブジェクトをデバッグ・セッションから除去するには、OPMSRC の指定値に応じて 2 とおりの方法があります。 STRDBG、SET デバッグ・コマンド、または CHGDBG を使用することにより OPMSRC(*YES) を指定した場合、「モジュール・リストの処理」画面で OPM プログラムを除去します。 (OPM プログラムのモジュール名はリストされないことに留意してください。) OPMSRC(*YES) が指定されていれば、デバッグ・セッションから一度に除去することのできる OPM プログラムの数に制限はありません。 OPMSRC(*NO) を指定した場合、プログラムの除去 (RMVPGM) コマンドを使用する必要があります。 OPMSRC(*NO) が指定されている場合、デバッグ・セッションに含めることのできる OPM プログラムの数は 10 個までに限定されます。

```
                          モジュール・リストの処理
                                                        システム:   ISERIES
オプションを入力して、実行キーを押してください。
  1=プログラム追加    4=プログラム除去    5=モジュール・ソースの表示
  8=モジュール停止点の処理
OPT      プログラム／モジュール     ライブラリー      タイプ
 _       ____________               *LIBL_____       *PGM___
 4       TEST                       TESTLIB          *PGM
 _         SAMPMDF                                   *MODULE
 _       MYPGM1                     TESTLIB          *PGM
 _         MYPGM1                                    *MODULE      選択済み
 _       USERDSP                    DSPLIB           *SRVPGM
 _         SAMPMDF                                   *MODULE
 _         GETUSER                                   *MODULE
                                                                  終わり
コマンド
===> ____________________________________________________________________________
F3= 終了    F4= プロンプト    F5= 最新表示    F9= コマンド複写    F12= 取り消し
```

図 35. ILE プログラム・オブジェクトをデバッグ・セッションから除去する

プログラム・オブジェクトまたはサービス・プログラムをデバッグ・セッションから除去し終えたら、「モジュール・リストの処理」画面で F3 (終了) を押すと、「モジュール・ソースの表示」画面に戻ります。

注: プログラムをデバッグ・セッションから除去するには、そのプログラム・オブジェクトに対する *CHANGE 権限が必要です。

プログラム・ソースの表示

「モジュール・ソースの表示」画面では、プログラム・オブジェクトまたはサービス・プログラムのソースのモジュール・オブジェクトが一度に 1 つずつ表示されます。モジュール・オブジェクトのソースは、次のデバッグ・ビュー・オプションのいずれかを使用してデバッグ・データを含むモジュール・オブジェクトを作成した場合に表示可能です。

- DBGVIEW(*STMT)
- DBGVIEW(*SOURCE)
- DBGVIEW(*LIST)
- DBGVIEW(*ALL)

OPM プログラムのソースは、次の条件が満たされている場合に表示できます。

1. OPM プログラムのコンパイルが OPTION(*LSTDBG) または OPTION(*SRCDBG) を指定して行われていること (*LSTDBG でコンパイルできるのは、RPG および COBOL のプログラムだけです。)
2. ILE デバッグ環境が OPM プログラムを受け入れられるように設定されていること。すなわち、OPMSRC の値が *YES になっていること (システム・デフォルトは OPMSRC(*NO) です。)

「モジュール・ソースの表示」画面に表示されるものを変更するには、次の 2 つの方法があります。

- 表示されているモジュール・オブジェクトを変更する
- 表示されているモジュール・オブジェクトのビューを変更する

ILE ソース・デバッガーはモジュール・オブジェクトが表示された最後の位置を覚えており、モジュール・オブジェクトを再度表示する時には同じ位置で表示しま

す。停止点が設定された行番号は強調表示されます。停止点、ステップ、またはメッセージによってプログラムが停止して画面が表示された場合、そのイベントが発生したソース行は強調表示されます。

表示されているモジュール・オブジェクトの変更

「モジュール・ソースの表示」画面に表示されているモジュール・オブジェクトを変更するには、「モジュール・リストの処理」画面でオプション 5 (モジュール・ソースの表示) を使用します。「モジュール・リストの処理」画面は、「モジュール・ソースの表示」画面で F14 (モジュール・リストの処理) を押すと表示されます。「モジュール・リストの処理」画面を図 36 に示します。

モジュール・オブジェクトを選択するには、表示したいモジュール・オブジェクトの横に 5 (モジュール・ソースの表示) を入力します。このオプションを ILE プログラム・オブジェクトに対して使用した場合、ソース・ビューを含むモジュール・オブジェクトがあれば、それが表示されます。そうでない場合には、デバッグ・データを持つプログラム・オブジェクトにバインドされた、最初のモジュール・オブジェクトが表示されます。このオプションを OPM プログラム・オブジェクトに対して使用した場合、ソース・ビューまたはリスト・ビューが使用可能であれば、それが表示されます。

```
                          モジュール・リストの処理
                                                    システム :   ISERIES
 オプションを入力して、実行キーを押してください。
   1= プログラム追加    4= プログラム除去    5= モジュール・ソースの表示
   8= モジュール停止点の処理
OPT     プログラム／モジュール     ライブラリー      タイプ
 _       ____________              *LIBL_____       *PGM___
 _       TEST                      TESTLIB          *PGM
 5         SAMPMDF                                  *MODULE
 _       MYPGM1                    TESTLIB          *PGM
 _         MYPGM1                                   *MODULE     選択済み
 _       USERDSP                   DSPLIB           *SRVPGM
 _         SAMPMDF                                  *MODULE
 _         GETUSER                                  *MODULE
 _                                                              終わり
コマンド
===> ____________________________________________________________________
 F3= 終了    F4= プロンプト    F5= 最新表示    F9= コマンド複写    F12= 取り消し
```

図 36. モジュール・ビューの表示

見たいモジュール・オブジェクトを選択してから実行キーを押すと、選択したビューが「モジュール・ソースの表示」画面に表示されます。

表示されているモジュール・オブジェクトを変更するもう 1 つの方法は、DISPLAY デバッグ・コマンドの使用です。デバッグ・コマンド行で、次のように入力します。

```
DISPLAY MODULE module-name
```

モジュール名 というモジュール・オブジェクトが表示されます。このモジュール・オブジェクトは、デバッグ・セッションに追加されたプログラム・オブジェクト内に存在していなければなりません。

表示されているモジュール・オブジェクトのビューの変更

ILE COBOL モジュール・オブジェクトの作成時に指定した値に応じて、ILE COBOL モジュール・オブジェクトのいくつかの異なるビューを使用することができます。次のビューが使用可能です。

- ILE COBOL リスト・ビュー
- ILE COBOL ソース・ビュー

「モジュール・ソースの表示」画面に表示されているモジュール・オブジェクトのビューを変更するには、「ビューの選択」画面を使用します。「ビューの選択」画面を表示するには、「モジュール・ソースの表示」画面で F15 (ビューの選択) を押します。「ビューの選択」画面を図 37 に示します。現行のビューがウィンドウの先頭に示され、使用できる他のビューがその下に表示されています。プログラム・オブジェクトまたはサービス・プログラムの各モジュール・オブジェクトごとにそれぞれ異なるビューの集まりを使用することができます。これはモジュール・オブジェクトを作成する際に使用したデバッグ・オプションに応じて異なります。

ビューを選択するには、表示したいビューの横に 1 (選択) を入力します。

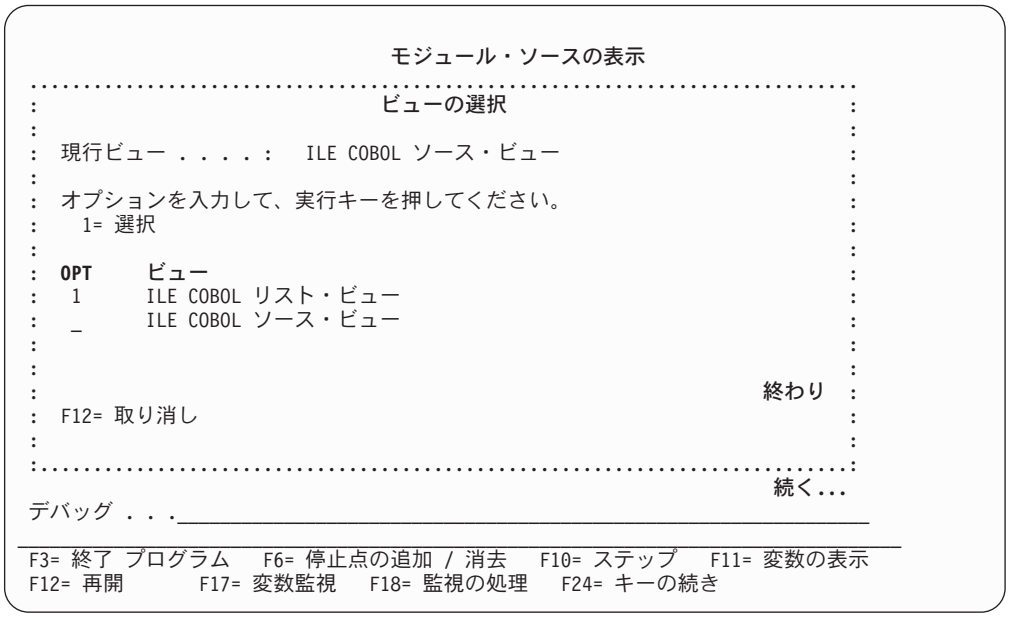


図 37. モジュール・オブジェクトのビューの変更

表示したいモジュール・オブジェクトのビューを選択してから実行キーを押すと、選択したモジュール・オブジェクトのビューが「モジュール・ソースの表示」画面に表示されます。

停止点の設定と除去

停止点を使用すると、実行中のプログラム・オブジェクトまたはサービス・プログラムを特定の位置で停止させることができます。**無条件停止点**では、プログラム・オブジェクトまたはサービス・プログラムを特定のステートメントで停止させることができます。**条件付き停止点**では、プログラム・オブジェクトまたはサービス・プログラムを特定のステートメントにおける特定の条件が満たされた場合に停止させることができます。

停止点には、ジョブとスレッドという 2 つのタイプがあります。スレッドされたアプリケーションの各**スレッド**は、その固有のスレッド停止点を同時に同じ位置にもつことができます。ジョブ停止点とスレッド停止点はどちらも、無条件でも条件付きでもかまいません。通常、デバッグ・コマンドとファンクション・キーの組み合わせが、ジョブ停止点用とスレッド停止点用とにそれぞれ 1 セットずつあります。停止点について説明している本セクションの残りの部分では、停止点という用語は、特に断わりがない限り、ジョブとスレッドの両方を指します。

プログラム・オブジェクトまたはサービス・プログラムが停止すると、「モジュール・ソースの表示」画面が表示されます。該当するモジュール・オブジェクトのソースのうち、停止点のある行の部分が表示されます。その行は強調表示されます。この時点で、変数の評価、他の停止点の設定、およびデバッグ・コマンドの実行を行うことができます。

停止点を使用するためには、停止点の次の特性を知っておく必要があります。

- 停止点が (たとえば GO TO ステートメントによって) う回された場合、その停止点は処理されません。
- 停止点がステートメントに設定されると、そのステートメントが処理される前に停止点が有効になります。
- 条件付き停止点を設定したステートメントに達すると、停止点に関連した条件式は、ステートメントが処理される前に評価されます。
- 停止点機能は、デバッグ・コマンドによって指定されます。

  これらの機能には、次のものが含まれます。
  - 停止点の追加
  - 停止点の除去
  - 停止点情報の表示
  - 停止点に達した後、プログラム・オブジェクトまたはサービス・プログラムの再開
- ジョブ停止点またはスレッド停止点のどちらかを指定の位置に同時にもつことはできますが、両方をもつことはできません。

無条件ジョブ停止点の設定と除去

無条件ジョブ停止点を設定したり除去したりするには、以下のものを使用します。

- 「モジュール・ソースの表示」画面で F6 (停止点の追加 / 消去)
- 「モジュール・ソースの表示」画面で F13 (モジュール停止点の処理)
- ジョブ停止点を設定する場合は BREAK デバッグ・コマンド
- ジョブ停止点を除去する場合は CLEAR デバッグ・コマンド

無条件ジョブ停止点の設定と除去を行う最も簡単な方法は、「モジュール・ソースの表示」画面で F6 (停止点の追加 / 消去) を使用することです。

F6 (停止点の追加 / 消去) を使用して無条件ジョブ停止点を設定するには、停止点を追加したい行の上にカーソルを移動して、F6 (停止点の追加 / 消去) を押します。無条件ジョブ停止点がその行に設定されます。

F6 (停止点の追加 / 消去) を使用して無条件ジョブ停止点を除去するには、ジョブ停止点を除去したい行にカーソルを移動して、F6 (停止点の追加 / 消去) を押します。その行からジョブ停止点が除去されます。

設定したい無条件ジョブ停止点ごとに、前述のステップを繰り返します。

ジョブ停止点を設定したい行に複数のステートメントが含まれている場合に F6 (停止点の追加 / 消去) を押すと、ジョブ停止点は行の最初のステートメントに設定されます。

注: ジョブ停止点を設定したい行が実行可能ステートメントでない場合は、ジョブ停止点はその次の実行可能ステートメントに設定されます。

F13 (モジュール停止点の処理) を使用して無条件停止点を除去するには、「モジュール・ソースの表示」画面で F13 (モジュール停止点の処理) を押します。停止点の設定または除去を行えるようにするオプションのリストが表示されます。 4 (消去) を選択すると、ジョブ停止点はその行から除去されます。

停止点の設定後、F3 (プログラムの終了) を押すと、「モジュール・ソースの表示」画面は終了します。「モジュール・ソースの表示」画面で F21 (コマンド行) を使用することによって、コマンド行からプログラムを呼び出すこともできます。

プログラム・オブジェクトを呼び出します。停止点に達すると、プログラム・オブジェクトまたはサービス・プログラムは停止し、「モジュール・ソースの表示」画面が再び表示されます。この時点で、変数の評価、他の停止点の設定、およびデバッグ・コマンドの実行を行うことができます。

無条件ジョブ停止点の設定と除去を行うもう 1 つの方法は、BREAK デバッグ・コマンドおよび CLEAR デバッグ・コマンドを使用することです。

BREAK デバッグ・コマンドを使用して無条件ジョブ停止点を設定するには、デバッグ・コマンド行で次のものを入力します。

```
BREAK line-number
```

行番号 は、モジュール・オブジェクトの現在表示されているビューの中で停止点を設定する番号です。

停止点を設定したい行に複数のステートメントが含まれている場合に BREAK デバッグ・コマンドを出すと、停止点は行の最初のステートメントに設定されます。

CLEAR デバッグ・コマンドを使って無条件ジョブ停止点を除去するには、デバッグ・コマンド行で次のように入力します。

```
CLEAR line-number
```

行番号 は、モジュール・オブジェクトの現在表示されているビューの中で停止点を除去したい番号です。ジョブ停止点が消去されるときは、すべてのスレッドについて消去されます。

無条件スレッド停止点の設定と除去

無条件スレッド停止点を設定したり除去したりするには、以下のものを使用します。

- 「モジュール・ソースの表示」画面で F13 (モジュール停止点の処理)
- 現行スレッド内にスレッド停止点を設定する場合は TBREAK デバッグ・コマンド
- スレッド停止点を除去する場合は CLEAR デバッグ・コマンド

設定

「モジュール停止点の処理」画面の使用: 「モジュール停止点の処理」画面を使用して無条件スレッド停止点を設定するには、次のように行います。

- *OPT* フィールドに 1 (追加) を入力します。
- スレッド・フィールドに、スレッド ID を入力します。
- 無条件ジョブ停止点の場合と同様に、残りのフィールドに入力します。
- 実行キーを押します。

注: スレッド・フィールドは、SPAWN コマンドの DEBUG オプションが 1 またはそれ以上である場合に表示されます。詳しくは 410 ページの『マルチスレッド化環境での ILE COBOL の使用例』を参照してください。

TBREAK コマンドの使用: TBREAK デバッグ・コマンドの構文は、BREAK デバッグ・コマンドと同じです。 BREAK デバッグ・コマンドはすべてのスレッド内の同じ位置にジョブ停止点を設定しますが、TBREAK デバッグ・コマンドは単一のスレッド、すなわち現行スレッド内にスレッド停止点を設定します。

現行スレッドとは、現在デバッグ中のスレッドのことです。デバッグ・コマンドはこのスレッドに対して出されます。停止点など、デバッグ停止が発生した時点で、現行スレッドはデバッグ停止が発生したスレッドに設定されます。現行スレッドの変更は、デバッグ THREAD コマンドおよび「デバッグ済みスレッドの処理」画面を使用して行えます。

除去

無条件スレッド停止点を除去するには、CLEAR デバッグ・コマンドを使用します。スレッド停止点が消去されるときは、現行スレッドについてのみ消去されます。

条件付きジョブ停止点の設定と除去

条件付きジョブ停止点の設定または除去を行うには、次のものを使用します。

- 「モジュール停止点の処理」画面
- ジョブ停止点を設定する場合は BREAK デバッグ・コマンド
- ジョブ停止点を除去する場合は CLEAR デバッグ・コマンド

注: 条件付き停止点についてサポートされている比較演算子は、<、>、=、=<、=>、および <> です。

条件付きジョブ停止点の設定または除去を行う方法の 1 つは、「モジュール停止点の処理」画面を使うことです。「モジュール停止点の処理」画面は、「モジュール・ソースの表示」画面で F13 (モジュール停止点の処理) を押すと表示されます。「モジュール停止点の処理」画面を図 38 に示します。

設定

条件付きジョブ停止点の設定は、「モジュール停止点の処理」画面を使用しても、あるいは BREAK デバッグ・コマンドを使用しても行えます。

「モジュール停止点の処理」画面を使用して条件付きジョブ停止点を設定するには、次のように行います。

1. *OPT* フィールドに 1 (追加) を入力します。
2. *行* フィールドに、停止点を設定したいデバッガー行番号を入力します。
3. *条件* フィールドに条件式を入力します。
4. スレッド欄が表示されている場合は、実行キーを押す前に、*スレッド・フィールド*に *JOB と入力します。
5. 実行キーを押します。

```
                          モジュール停止点の処理
                                                        システム:   ISERIES
 プログラム . . :   TEST                ライブラリー . :   TESTLIB
 モジュール . . :   SAMPMDF             タイプ . . . . :   *PGM
オプションを入力して、Enter キーを押してください。
   1= 追加   4= 消去
 OPT     行         条件
  1      35___      I=21_____________________
  _      _____      _________________________
```

図 38. 条件付き停止点の設定

停止点を設定したい行に複数のステートメントが含まれている場合、停止点は行の最初のステートメントに設定されます。

注: 停止点を設定したい行が実行可能ステートメントでない場合、停止点はその次の実行可能ステートメントに設定されます。

設定または除去したい停止点をすべて指定した後、 F3 (終了) を押すと、「モジュール・ソースの表示」画面が表示されます。

「モジュール・ソースの表示」画面を終了するには、F3 (プログラムの終了) を押します。「モジュール・ソースの表示」画面で F21 (コマンド行) を使用して、コマンド行からプログラム・オブジェクトを呼び出すこともできます。

プログラム・オブジェクトまたはサービス・プログラムを実行します。 条件付きジョブ停止点を設定したステートメントに達すると、その停止点に関連した条件式はステートメントが実行される前に評価されます。結果が偽の場合には、プログラム・オブジェクトは実行し続けます。結果が真の場合には、プログラム・オブジェクトが停止し、「モジュール・ソースの表示」画面が表示されます。この時点で、変数の評価、他の停止点の設定、およびデバッグ・コマンドの実行を行うことができます。

BREAK デバッグ・コマンドを使用して条件付きジョブ停止点を設定するには、デバッグ・コマンド行で次のものを入力します。

```
BREAK line-number WHEN expression
```

行番号 はモジュール・オブジェクトの現在表示されているビューの中で停止点を設定する行の番号であり、*式* はその停止点に達した時点で評価する条件式です。条件式は、単純式でなければなりません。すなわち、等号の右辺には 1 つの値しか指定できません。たとえば、I=21 は可能ですが、I=A+2 または I=3*2 は受け入れられません。

停止点を設定したい行に複数のステートメントが含まれている場合に BREAK デバッグ・コマンドを出すと、停止点は行の最初のステートメントに設定されます。

例: たとえば、デバッガー行 35 に条件付きジョブ停止点を設定するには、次のように行います。

1. *OPT* フィールドに 1 (追加) を入力します。
2. *行* フィールドに 35 と入力します。
3. 151 ページの図 38 に示されているとおりに、*条件* フィールドに I=21 と入力して、実行キーを押します。 (スレッド欄が表示されている場合は、実行キーを押す前に、*スレッド・フィールドに* *JOB と入力してください。)
4. 設定したい条件付きジョブ停止点ごとに、前述のステップを繰り返します。

除去

条件付きジョブ停止点の除去は、「モジュール停止点の処理」画面を使用しても、あるいは CLEAR デバッグ・コマンドを使用しても行えます。

「モジュール停止点の処理」画面を使用して条件付きジョブ停止点を除去するには、除去したい停止点の横の *OPT* に 4 (消去) を入力して、実行キーを押します。このようにして、無条件停止点を除去することもできます。 151 ページの図 38 に、*OPT* フィールドに 4 (消去) を入力できる一般的な画面を示します。

除去したい条件付きジョブ停止点ごとに、前述のステップを繰り返します。

CLEAR デバッグ・コマンドを使用して条件付きジョブ停止点を除去するには、デバッグ・コマンド行で次のものを入力します。

```
CLEAR line-number
```

行番号 は、モジュール・オブジェクトの現在表示されているビューの中でジョブ停止点を除去したい行番号です。

条件付きスレッド停止点の設定と除去

条件付きスレッド停止点の設定または除去を行うには、次のものを使用します。

- 「モジュール停止点の処理」画面
- 現行スレッド内に条件付きスレッド停止点を設定する場合は TBREAK デバッグ・コマンド
- 条件付きスレッド停止点を除去する場合は CLEAR デバッグ・コマンド

「モジュール停止点の処理」画面の使用

「モジュール停止点の処理」画面を使用して条件付きスレッド停止点を設定するには、次のように行います。

1. *OPT* フィールドに 1 (追加) を入力します。
2. スレッド・フィールドに、スレッド ID を入力します。
3. 条件付きジョブ停止点の場合と同様に、残りのフィールドに入力します。
4. 実行キーを押します。

「モジュール停止点の処理」画面を使用して条件付きスレッド停止点を除去するには、次のように行います。

1. 除去したい停止点の横の *OPT* に 4 (消去) を入力します。
2. 実行キーを押します。

TBREAK デバッグ・コマンドまたは CLEAR デバッグ・コマンドの使用

TBREAK デバッグ・コマンドには、TBREAK デバッグ・コマンドの場合と同じ構文を使用できます。この 2 つのコマンドの相違点は、BREAK デバッグ・コマンドが条件付きジョブ停止点をすべてのスレッドの同じ位置に設定するのに対して、TBREAK デバッグ・コマンドは現行スレッドに条件付きスレッド停止点を設定するところです。

条件付きスレッド停止点を除去する場合は、CLEAR デバッグ・コマンドを使用します。条件付きスレッド停止点が消去されるときは、現行スレッドについてのみ消去されます。

すべての停止点の除去

「モジュール・ソースの表示」画面に表示されているモジュール・オブジェクトを含むプログラム・オブジェクトから、条件付き、無条件を問わず、すべてのジョブ停止点およびスレッド停止点を除去するには、CLEAR PGM デバッグ・コマンドを使用します。このデバッグ・コマンドを使用するには、デバッグ・コマンド行で次のように入力します。

```
CLEAR PGM
```

プログラムにバインドされたすべてのモジュールから停止点が除去されます。

監視条件の設定と除去

指定した値の内容 (またはサブストリングや配列エレメントに関係する式) が現在の値から変更される時点でジョブ停止点を要求するには、**監視条件**を使用します。監視条件の設定は条件付きジョブ停止点の設定と同様ですが、次のような重要な違いがあります。

- 監視条件は、対象の式または変数が現在の値から変更されると、ただちにプログラムを停止します。
- 条件付きジョブ停止点は、変数が条件に指定された値に変更された場合にのみプログラムを停止します。

デバッガーは、監視条件の設定時に計算される**ストレージ・アドレス**の内容によって、式または変数を監視します。ストレージ・アドレスの内容が、監視条件の設定時の値、または最後の監視条件発生時の値から変更されると、プログラムは停止します。

注: 監視条件が登録された後、監視対象の保管場所での新しい内容が、対応する式または変数の新しい現在値として保管されます。その後で監視対象保管場所での新しい内容が変更されると、その次の監視条件が登録されることになります。

監視の特性

監視を使用する前に、次のような特性を知っておく必要があります。

- 監視はシステム全体を通じてモニターされ、同時に最大 256 の監視を活動状態にできます。これにはシステムが設定する監視も含まれます。

  システム全体の使用状況に応じて、所定の時間に設定できる監視条件の数が制限されることがあります。 システム内で活動中の監視が最大数を超えている場合に監視条件を設定しようとすると、エラー・メッセージが出て設定できません。

  注: 式または変数がページ境界をまたぐ場合は、内部で 2 つの監視を使用して保管場所をモニターします。したがって、システム全体を通じて同時に監視可能な式または変数の最大数は、128〜256 の範囲になります。
- 監視条件を設定できるのは、デバッグでプログラムが停止しており、監視される式または変数が有効範囲にある場合だけです。これにあてはまらない場合に監視が要求されると、対応する呼び出しスタック項目が存在しないことを示すエラー・メッセージが出されます。
- いったん監視条件を設定すると、監視される保管場所のアドレスは変わりません。したがって、一時記憶位置に監視を設定すると、実体の伴わない監視条件通知が生じる可能性があります。

  ILE COBOL プロシージャーの自動ストレージはその一例です。それはプロシージャーの終了後には再使用されることがあります。

  監視条件は、変数監視が有効範囲になくても登録できます。監視条件が報告されたというだけの理由で、変数が有効範囲にあると見なすことはできません。
- 同一ジョブ内の 2 つの監視位置は決して重なり合わないようにしてください。別々のジョブの 2 つの監視位置が同じストレージ・アドレスから始まらないようにしてください。そうしないと、オーバーラップする可能性があります。こうした制約に違反すると、エラー・メッセージが出されます。

  注: 監視される記憶域の位置が、この監視条件を設定したジョブとは異なるジョブで変更される場合、この変更は無視されます。
- コマンドが正常に実行された後、セッション内のプログラムが監視対象の保管場所の内容を変更すると、アプリケーションが停止し、「モジュール・ソースの表示」画面になります。

プログラムにデバッグ・データがある場合、使用可能なソース・テキスト・ビューがあれば表示されます。保管場所への変更を検出した際に実行目前であったステートメントのソース行が強調表示されます。メッセージは、どの監視条件が満たされているかを示します。

プログラムをデバッグすることができない場合、画面のテキスト部分はブランクになります。

- 適格なプログラムが監視を停止させると、このプログラムは、自動的にデバッグ・セッションに追加されます。
- 同じプログラム・ステートメントで複数の監視条件がヒットした場合、最初の監視条件だけが報告されます。
- デバッグのサービス・ジョブを使用している場合、すなわち、1 つのジョブを別のジョブからデバッグする場合にも監視条件を設定できます。

監視条件の設定

監視条件を設定する前に、プログラムをデバッグの制御下で停止する必要があり、しかも監視を行いたい式または変数が有効範囲内になければなりません。

- グローバル変数を監視するには、監視条件を設定する前に、その変数が定義されている COBOL プログラムが活動状態にあるようにする必要があります。
- ローカル変数を監視するには、監視条件を設定する前に、その変数が定義されている COBOL プログラムをステップイントゥする必要があります。

監視条件を設定するには、次のものを使用できます。

- F17 (変数監視)。カーソル位置の変数 (COBOL データ項目) に監視条件を設定します。
- WATCH デバッグ・コマンド (パラメーターあり、またはパラメーターなし)

WATCH コマンドの使用

WATCH コマンドを使用する場合には、単一のコマンドとして入力しなければなりません。同一のコマンド入力行で他のデバッグ・コマンドを使用することはできません。

- 下の「監視の処理」画面にアクセスするには、デバッグ・コマンド入力行に次のように入力します。

  ```
  WATCH
  ```

  パラメーターは指定しません。

```
                              監視の処理
                                                    システム :  DEBUGGER
オプションを入力して、実行キーを押してください。
  4= 消去 5= 表示
OPT    NUM      変数                    アドレス           長さ
 -     1        KOUNT                   080090506F027004   4









                                                                終わり
コマンド
===>_____________________________________________________________________
F3= 終了   F4= プロンプト   F5= 最新表示    F9= コマンドの複写    F12= 取り消し
```

図 39. 「監視の処理」画面の例

「監視の処理」画面には、デバッグ・セッションで現在活動中の全監視が表示されます。この画面で監視を除去または表示することができます。オプション 5 (表示) を選択すると図 40 に示されている「監視の表示」ウィンドウに現在活動中の監視に関する情報が表示されます。

```
                              監視の処理
.....................................................
:                        監視の表示                 :
:                                                   :
:  監視番号 . . . . . :   1                         :
:  アドレス . . . . . :   080090506F027004          :
:  長さ . . . . . :   4                             :
:  ヒットの数 . . . . :   0                         :
:                                                   :
:  監視送信時の範囲. . :                            :
:   プログラム/ライブラリー/タイプ: PAYROLL     ABC     *PGM       :
:                                                    :
:    モジュール . . : MAYROLL                       :
:    プロシージャー : MAIN                          :
:    変数 . . . . . : KOUNT                         :
:                                                   :
:  F12= 取り消し                                    :
:                                                   :
.....................................................
                                                    終わり
コマンド
===>_____________________________________________________________________
F3= 終了   F4=プロンプト   F5= 最新表示    F9= コマンドの複写    F12= 取り消し
```

図 40. 「監視の表示」ウィンドウの例

- 監視対象の変数または式を指定するには、次のうちのいずれかを行います。
  - 監視対象の変数を指定するには、デバッグ・コマンド入力行で次のように入力します。

    ```
    WATCH SALARY
    ```

    SALARY が対象となる変数です。

    このコマンドは、SALARY の値が現在の値から変更されている場合に停止点を設定するよう要求するものです。

監視の式変数の有効範囲は、最後に発行した QUAL コマンドによって定義されます。

– 監視対象の式を指定するには、次のように入力します。

```
WATCH %SUBSTR(A 1 3)
```

A はサブストリング式の一部です。サブストリングについては 166 ページの『文字ストリング変数のサブストリングの表示』を参照してください。

注: ILE COBOL では、配列エレメントまたはサブストリングに関係する式だけが監視可能です。

- 監視条件を設定し、監視長を指定するには、デバッグ・コマンド入力行で次のように入力します。

```
WATCH expression : watch length
```

各監視ごとに最大 128 バイトまでの隣接ストレージをモニターし比較することができます。 128 バイトの最大長を超えると、監視条件は設定されず、デバッガーがエラー・メッセージを出します。

デフォルトでは、式タイプの長さが監視比較操作の長さになります。*監視長パラメーター* によってこのデフォルトを指定変更することができます。このパラメーターは、値が変更されているかどうか判別するために比較する式のバイト数を指定するものです。

たとえば、4 バイトの 2 進整数が変数として指定されていて、監視長パラメーターが指定されていなければ、比較長は 4 バイトということになります。しかし、監視長パラメーターが指定されていれば、それによって式の長さが書き換えられ、監視長が決められます。

活動状態の監視の表示

活動状態の監視のシステム全体にわたるリストを表示し、どのジョブがそれらを設定したかを表示するには、次のように入力します。

```
DSPDBGWCH
```

このコマンドで、下記の「デバッグ監視の表示」画面が表示されます。

```
                          デバッグ監視の表示
                                                    システム:  DEBUGGER
------------ジョブ ------------      番号       長さ     アドレス
MYJOBNAME1  MYUSERPRF1  123456          1          4     080090506F027004
JOB4567890  PRF4567890  222222          1          4     09849403845A2C32
JOB4567890  PRF4567890  222222          2          4     098494038456AA00
JOB         PROFILE     333333         14          4     040689578309AF09
SOMEJOB     SOMEPROFIL  444444          3          4     005498348048242A



                                                                終わり
続行するためには、実行キーを押してください。

F3= 終了  F5= 最新表示    F12= 取り消し
```

図 41. 「デバッグ監視の表示」画面の例

注: システムが設定した監視条件はこの画面には表示されません。

監視条件の除去

次の方法で監視を除去することができます。

- WATCH キーワードを指定した CLEAR コマンドを使用することにより、1 つの監視を選択して終了させたり、全部終了させたりすることができます。たとえば、*監視番号* により識別される監視をクリアするには、次のように入力します。

  ```
  CLEAR WATCH watch-number
  ```

  監視番号は「監視の処理」画面から入手できます。

  セッションの監視を全部クリアするには、デバッグ・コマンド入力行で次のように入力します。

  ```
  CLEAR WATCH ALL
  ```

  注: CLEAR PGM コマンドは、表示中のモジュールを含んだプログラムの停止点をすべて除去しますが、監視には影響を及ぼしません。監視条件を除去するには、CLEAR コマンドを WATCH キーワードで明示的に使用する必要があります。
- CL デバッグの終了 (ENDDBG) コマンドは、ローカル・ジョブやサービス・ジョブの監視設定を除去します。

  注: 異常時には ENDDBG が自動的に呼び出され、影響を受けた監視がすべて除去されるようにします。
- IBM i システムの初期プログラム・ロード (IPL) では、システム全体にわたるすべての監視条件が除去されます。

監視条件の設定例

この例では、プログラム MYLIB/PAYROLL の変数 *kount* を監視します。監視条件を設定するには、デバッグ行で次のように入力します。

```
WATCH kount
```

監視長は、デフォルト値になります。

後で変数 *kount* の値が変わると、アプリケーションが停止し 159 ページの図 42 に示されている「モジュール・ソースの表示」画面になります。

```
                         モジュール・ソースの表示
プログラム:   PAYROLL          ライブラリー:   MYLIB      モジュール:  PAYROLL
    42        * THE FOLLOWING 3 PARAGRAPHS CREATE INTERNALLY THE  *
    43        * RECORDS TO BE CONTAINED IN THE FILE, WRITES THEM  *
    44        * ON THE DISK, AND DISPLAYS THEM                    *
    45        *****************************************************
    46         STEP-2.
    47             ADD 1 TO KOUNT, NUMBR.
    48             MOVE ALPHA  (KOUNT) TO NAME-FIELD.
    49             MOVE DEPEND (KOUNT) TO NO-OF-DEPENDENTS.
    50             MOVE NUMBR          TO RECORD-NO.
    51         STEP-3.
    52             DISPLAY WORK-RECORD.
    53             WRITE RECORD-1 FROM WORK-RECORD.
    54         STEP-4.
    55             PERFORM STEP-2 THRU STEP-3 UNTIL KOUNT IS =
                                                              続く...
デバッグ     __________________________________________________________________
_______________________________________________________________________________
F3= プログラム終了   F6= 停止点の追加 / 消去          F10= ステップ
F11= 変数の表示      F12= 再開       F17= 変数監視     F24= キーの続き
行 55 の監視番号 1、変数: KOUNT
```

図 42. 監視が正常に設定されたことを示すメッセージの例

- 変数監視への変更が検出されたステートメントの行番号が強調表示されます。一般に、これは変数を変更したステートメントの後に来る 最初の実行可能行です。
- メッセージに、どの監視条件が満たされているかが示されます。

注: テキスト・ビューが利用できない場合、ブランクの「モジュール・ソースの表示」画面になり、上のメッセージ域と同じメッセージが表示されます。

次のプログラムは、ILE デバッグ環境に追加することはできません。

1. デバッグ・データのない ILE プログラム
2. 非ソース・デバッグ・データしかない OPM プログラム
3. デバッグ・データのない OPM プログラム

最初の 2 つの場合、停止したステートメント番号が渡されます。3 番目の場合、停止した MI 命令が渡されます。情報は、下のようにブランクの「モジュール・ソースの表示」画面の下部に表示されます。行番号ではなく、ステートメントまたは命令の番号が示されます。

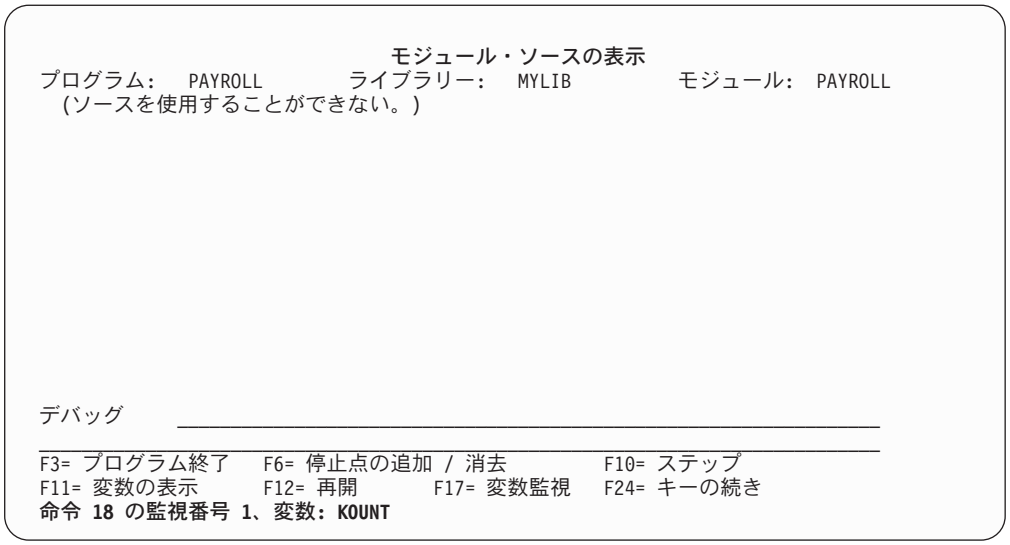


図 43. 「モジュール・ソースの表示」画面の例

停止点後のプログラム・オブジェクトまたは ILE プロシージャーの実行

停止点の検出後、プログラム・オブジェクトまたは ILE プロシージャーの実行を再開するには、次の 2 とおりの方法があります。

- 停止点の後の次のステートメントからプログラム・オブジェクトまたは ILE プロシージャーの実行を再開して、次の停止点で停止するか、あるいはプログラム・オブジェクトが終了したら停止する。
- 停止点の後の指定した数のステートメントをステップスルーしてから、プログラム・オブジェクトを再度停止する。

プログラム・オブジェクトまたは ILE プロシージャーの再開

停止点の検出後、「モジュール・ソースの表示」画面で F12 (再開) を押すと、プログラム・オブジェクトまたは ILE プロシージャーの実行を再開することができます。プログラム・オブジェクトまたは ILE プロシージャーは、プログラムが停止したモジュール・オブジェクトの次のステートメントから実行が開始されます。プログラム・オブジェクトまたは ILE プロシージャーは、次の停止点またはプログラム終了時に再度停止します。

プログラム・オブジェクトまたは ILE プロシージャーのステップスルー

停止点の検出後、プログラム・オブジェクトまたは ILE プロシージャーの指定された数のステートメントを実行してから、プログラムを再度停止し、次に「モジュール・ソースの表示」画面に戻ることができます。プログラム・オブジェクトまたは ILE プロシージャーは、プログラムが停止したモジュール・オブジェクトの次のステートメントから実行が開始されます。

OPM プログラムに使用可能なデバッグ・データが含まれている場合、およびデバッグ・セッションで OPM プログラムのデバッグが受け入れられる場合は、そのプログラムをステップイントゥできます。

次のものを使用すると、プログラム・オブジェクトまたは ILE プロシージャーをステップスルーすることができます。
- 「モジュール・ソースの表示」画面の F10 (ステップ) または F22 (ステップイントゥ)
- STEP デバッグ・コマンド

プログラム・オブジェクトまたは ILE プロシージャーをステップスルーしてステートメントごとに実行する最も簡単な方法は、「モジュール・ソースの表示」画面で F10 (ステップ) または F22 (ステップイントゥ) を使用することです。 F10 (ステップ) または F22 (ステップイントゥ) を押すと、「モジュール・ソースの表示」画面に表示されているモジュール・オブジェクトの次のステートメントが実行され、プログラム・オブジェクトまたは ILE プロシージャーは再度停止します。複数のステートメントが含まれている行で F10 (ステップ) または F22 (ステップイントゥ) を押した場合、その行のすべてのステートメントが実行され、プログラム・オブジェクトまたは ILE プロシージャーは、次の行の次のステートメントで停止します。

注: F10 (ステップ) または F22 (ステップイントゥ) を使用した場合は、ステップスルーするステートメントの数を指定することはできません。 F10 (ステップ) または F22 (ステップイントゥ) を押すと、1 ステップずつ実行されます。

プログラム・オブジェクトまたは ILE プロシージャーをステップスルーするもう 1 つの方法は、STEP デバッグ・コマンドを使用することです。 STEP デバッグ・コマンドを使用すると、1 つのステップで複数のステートメントを実行することができます。 STEP デバッグ・コマンドを使用する場合に実行されるステートメント数のデフォルトは 1 です。 STEP デバッグ・コマンドを使用してプログラム・オブジェクトまたは ILE プロシージャーをステップスルーするには、デバッグ・コマンド行で次のように入力します。

```
STEP number-of-statements
```

ステートメント数 は、アプリケーションが再停止するまでに次のステップで実行させるステートメントの数です。たとえば、デバッグ・コマンド行に次のように入力します。

```
STEP 5
```

この場合、プログラム・オブジェクトまたは ILE プロシージャーのうち、その次の 5 つのステートメントが実行された後に、プログラム・オブジェクトまたは ILE プロシージャーは再度停止し、続いて「モジュール・ソースの表示」画面が表示されます。

デバッグ・セッション内で他のプログラム・オブジェクトまたは ILE プロシージャーに指定された CALL ステートメントが検出された場合、次のことを行うことができます。
- 呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーのステップオーバー
- 呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーのステップイントゥ

呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーの**ステップオーバー**を選択した場合、CALL ステートメントと呼び出し先プログラム・オブジェクトとが、1 つのステップとして実行されます。呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーは、呼び出し側プログラム・オブジェクトまたは ILE プロシージャーが次のステップで停止するまで実行されます。ステップオーバーは、デフォルトのステップ・モードです。

呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーの**ステップイントゥ**を選択した場合、その呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーの各ステートメントが 1 つのステップとして実行されます。実行中のプログラム・オブジェクトまたは ILE プロシージャーが停止する次のステップが、呼び出し先のプログラム・オブジェクトまたは ILE プロシージャー内にある場合、呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーはその時点で停止し、「モジュール・ソースの表示」画面にその呼び出し先のプログラムまたは ILE プロシージャーが表示されます。

プログラム・オブジェクトまたは ILE プロシージャーのステップオーバー

プログラム・オブジェクトまたは ILE プロシージャーをステップオーバーするには、以下のものを使用します。

- 「モジュール・ソースの表示」画面で F10 (ステップ)
- STEP OVER デバッグ・コマンド

「モジュール・ソースの表示」画面で F10 (ステップ) を使用すると、デバッグ・セッション内の呼び出されたプログラム・オブジェクトまたは ILE プロシージャーをステップオーバーすることができます。次に実行するステートメントが他のプログラム・オブジェクトまたは ILE プロシージャーへの CALL ステートメントである場合、F10 (ステップ) を押すと、呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーは、呼び出し側プログラム・オブジェクトまたは ILE プロシージャーが再び停止するまで最後まで実行されます。

別の方法として、STEP OVER デバッグ・コマンドを使用して、デバッグ・セッション内の呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーをステップオーバーすることができます。 STEP OVER デバッグ・コマンドを使用するには、デバッグ・コマンド行で次のように入力します。

```
STEP number-of-statements OVER
```

ステートメント数 は、アプリケーションが再停止するまでに次のステップで実行させるステートメントの数です。 実行するステートメントのどれかに、他のプログラム・オブジェクトまたは ILE プロシージャーへの CALL ステートメントが含まれている場合、ILE ソース・デバッガーは、呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーをステップオーバーします。

プログラム・オブジェクトまたは ILE プロシージャーのステップイントゥ

プログラム・オブジェクトまたは ILE プロシージャーをステップイントゥするには、以下のものを使用します。

- 「モジュール・ソースの表示」画面の F22 (ステップイントゥ)
- STEP INTO デバッグ・コマンド

「モジュール・ソースの表示」画面で F22 (ステップイントゥ) を使用すると、デバッグ・セッション内の呼び出されたプログラム・オブジェクトまたは ILE プロシージャーをステップイントゥすることができます。次に実行するステートメントが他のプログラム・オブジェクトまたは ILE プロシージャーへの CALL ステートメントである場合に F22 (ステップイントゥ) を押すと、呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーの最初の実行可能ステートメントが実行されます。その後、プログラム・オブジェクトまたは ILE プロシージャーが「モジュール・ソースの表示」画面に表示されます。

注: 呼び出し先の ILE プログラム・オブジェクトまたはプロシージャーが「モジュール・ソースの表示」画面に表示されるには、ILE プログラム・オブジェクトまたはプロシージャーに、関連するデバッグ・データがなければなりません。ILE ソース・デバッガーが OPM プログラムを受け入れるようにセットアップされていて、その OPM プログラムにデバッグ・データが含まれている場合は、呼び出し先 OPM プログラム・オブジェクトが「モジュール・ソースの表示」画面に表示されます。 (OPM プログラムが OPTION(*SRCDBG) または OPTION(*LSTDBG) でコンパイルされたものであれば、デバッグ・データが含まれています。)

別の方法として、STEP INTO デバッグ・コマンドを使用して、デバッグ・セッション内の呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーをステップイントゥすることができます。 STEP INTO デバッグ・コマンドを使用するには、デバッグ・コマンド行で次のように入力します。

```
STEP number-of-statements INTO
```

ステートメント数 は、プログラム・オブジェクトまたは ILE プロシージャーが再停止するまでに次のステップで実行したいプログラム・オブジェクトまたは ILE プロシージャーのステートメントの数です。実行するステートメントのどれかに、他のプログラム・オブジェクトまたは ILE プロシージャーへの CALL ステートメントが含まれている場合、デバッガーは、呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーをステップオーバーします。ステップの中で、呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーの各ステートメントがカウントされます。呼び出し先のプログラム・オブジェクトまたは ILE プロシージャー内でステップが終了する場合、そのプログラム・オブジェクトまたは ILE プロシージャーが「モジュール・ソースの表示」画面に表示されます。たとえば、デバッグ・コマンド行に次のように入力します。

```
STEP 5 INTO
```

この場合、プログラム・オブジェクトまたは ILE プロシージャーの次の 5 つのステートメントが実行されます。 3 番目のステートメントが他のプログラム・オブジェクトまたは ILE プロシージャーへの CALL ステートメントである場合、呼び出し側プログラム・オブジェクトまたは ILE プロシージャーの 2 つのステートメントが実行され、呼び出し先のプログラム・オブジェクトまたは ILE プロシージャーの最初の 3 つのステートメントが実行されます。

変数、定数名、式、レコード、グループ項目、および配列の表示

次のものを使用することによって、変数、定数名、式、グループ項目、レコード、および配列の値を表示することができます。

- 「モジュール・ソースの表示」画面で F11 (変数の表示)
- EVAL デバッグ・コマンド

EVAL コマンドで使用される変数の有効範囲は、QUAL コマンドを使用して定義されます。

注: ILE COBOL 特殊レジスターは、ILE ソース・デバッガーではサポートされていません。したがって、ILE COBOL 特殊レジスターに含まれている値は、デバッグ・セッションでは表示できません。 ILE ソース・デバッガーでは、COBOL 関数 ID の結果を評価できません。

変数および式の表示

変数の値を表示する最も簡単な方法は、「モジュール・ソースの表示」画面で F11 (変数の表示) を使用することです。 F11 (変数の表示) を使用して変数を表示するには、カーソルを表示したい変数のところに移動して、F11 (変数の表示) を押します。変数の現行値が、「モジュール・ソースの表示」画面の最下部にあるメッセージ行に表示されます。たとえば図 44 に示されているモジュール・オブジェクトの 221 行目の変数 *COUNTER* の値を見たい場合は、その変数の先頭にカーソルを移動して、F11 (変数の表示) を押します。変数 *COUNTER* の現行値が、メッセージ行に表示されます。

```
                                モジュール・ソースの表示
プログラム:   TEST       ライブラリー:   TESTLIB     モジュール:   SAMPMDF
   213
   214         PROCEDURE-SECTION SECTION.
   215         FILL-TERMINAL-LIST.
   216             READ TERMINAL-FILE RECORD INTO LIST-OF-TERMINALS(COUNTER)
   217                 AT END
   218                     SET END-OF-TERMINAL-LIST TO TRUE
   219                     SUBTRACT 1 FROM COUNTER
   220                     MOVE COUNTER TO NO-OF-TERMINALS.
   221             ADD 1 TO COUNTER.
   222
   223         ACQUIRE-AND-INVITE-TERMINALS.
   224             ACQUIRE LIST-OF-TERMINALS(COUNTER) FOR MULTIPLE FILE.
   225             WRITE MULTIPLE-REC
   226                 FORMAT IS "SIGNON"
   227                 TERMINAL IS LIST-OF-TERMINALS(COUNTER).
                                                                  続く...
デバッグ      ___________________________________________________________________
F3= プログラム終了    F6= 停止点の追加 / 消去          F10= ステップ
F11= 変数の表示       F12= 再開        F17= 変数監視    F24= キーの続き
COUNTER = 89.
```

図 44. F11 (変数の表示) を使用して変数を表示する

レコード、グループ項目、または配列を評価している場合、F11 (変数の表示) を押すと、複数行にわたるメッセージが戻されることがあります。複数行にわたるメッセージは、その全テキストが「評価式」画面に表示されます。「評価式」画面に表示されたメッセージを読み終ったら、実行キーを押して、「モジュール・ソースの表示」画面に戻ります。

EVAL デバッグ・コマンドを使って、変数の値を判別することができます。まず、QUAL デバッグ・コマンドを使用して、表示する変数の行番号を指定します。変数の有効範囲に関する規則が、この行から適用されます。

注: デフォルトの QUAL の位置は、現在行です。
EVAL デバッグ・コマンドを使用して変数の値を表示するには、デバッグ・コマンド行で次のように入力します。

```
EVAL variable-name
```

変数名 は、表示したい変数の名前です。「モジュール・ソースの表示」画面から EVAL デバッグ・コマンドを入力し、変数の値が 1 行内に収まる場合、値はメッセージ行に表示されます。それ以外の場合、変数の値は「評価式」画面に表示されます。

たとえば、164 ページの図 44 に示されているモジュール・オブジェクトの 221 行の変数 *COUNTER* の値を表示するには、次のように入力します。

```
EVAL COUNTER
```

図 45 に示すように、「モジュール・ソースの表示」画面のメッセージ行に `COUNTER = 89` が表示されます。

```
                              モジュール・ソースの表示
 プログラム:   TEST       ライブラリー:   TESTLIB    モジュール:   SAMPMDF
    213
    214        PROCEDURE-SECTION SECTION.
    215        FILL-TERMINAL-LIST.
    216            READ TERMINAL-FILE RECORD INTO LIST-OF-TERMINALS(COUNTER)
    217                AT END
    218                    SET END-OF-TERMINAL-LIST TO TRUE
    219                    SUBTRACT 1 FROM COUNTER
    220                    MOVE COUNTER TO NO-OF-TERMINALS.
    221            ADD 1 TO COUNTER.
    222
    223        ACQUIRE-AND-INVITE-TERMINALS.
    224            ACQUIRE LIST-OF-TERMINALS(COUNTER) FOR MULTIPLE FILE.
    225            WRITE MULTIPLE-REC
    226                FORMAT IS "SIGNON"
    227                TERMINAL IS LIST-OF-TERMINALS(COUNTER).
                                                                 続く...
 デバッグ    ____________________________________________________________
_____________________________________________________________________________
 F3= プログラム終了   F6= 停止点の追加 / 消去       F10= ステップ
 F11= 変数の表示      F12= 再開       F17= 変数監視   F24= キーの続き
 COUNTER = 89.
```

図 45. EVAL デバッグ・コマンドを使用して変数を表示する

16 進値の変数の表示

EVAL デバッグ・コマンドを使用して、変数の値を 16 進型式で表示することができます。変数を 16 進型式で表示するには、デバッグ・コマンド行で次のように入力します。

```
EVAL variable-name: x 32
```

変数名 は、16 進型式で表示したい変数の名前です。「x」は変数を 16 進型式で表示するということ、「32」は変数の開始後に 32 バイトのダンプが表示されること

を指定するものです。変数の 16 進値が図 46 に示す「評価式」画面に表示されます。「x」の後に長さを指定しない場合、変数の長さが長さとして使用されます。最低 16 バイトの長さが、常に表示されます。変数の長さが 16 バイト未満の場合、16 バイトに達するまで残りのスペースにゼロが埋め込まれます。

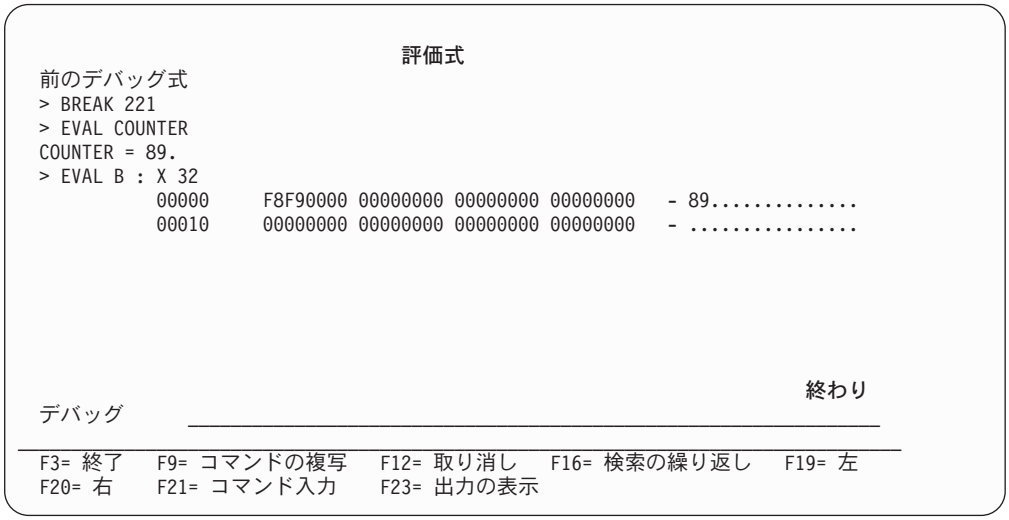


図 46. EVAL デバッグ・コマンドを使用して 16 進値で変数を表示する

文字ストリング変数のサブストリングの表示

ILE ソース・デバッガーは、ILE COBOL の参照変更の構文をサポートしていません。その代わりに、EVAL コマンドに %SUBSTR 演算子を使用して、文字ストリング変数のサブストリングを表示することができます。 %SUBSTR 演算子により、いくつかのエレメントのうちのエレメント開始位置から文字ストリング変数のサブストリングが獲得されます。

注: ILE ソース・デバッガーは、COBOL の参照変更の構文を使用したサブストリングの処理はサポートしていません。サブストリングの処理には、ILE ソース・デバッガーの %SUBSTR 演算子が必要です。

%SUBSTR 演算子の構文は、次のとおりです。

```
%SUBSTR(identifier start-element number-of-elements)
```

ここで、*id* は文字ストリング変数でなければならず、*開始エレメント* とエレメント数 は、ゼロ以外の正の整数リテラルでなければなりません。 *id* は、修飾変数、添え字付き変数、または指標付き変数です。*開始エレメント + エレメント数 - 1* の値は、*id* のエレメントの合計数以下でなければなりません。

たとえば、%SUBSTR(CHAR20 1 10) を使うと、20 個のエレメントを含む文字ストリングの最初の 10 個のエレメントを取得できます。 %SUBSTR(CHAR8 4 5) を使用すると、8 個のエレメントを含む文字ストリングの最後の 5 個のエレメントを取得できます。 DBCS または DBCS 編集項目の場合、エレメントは DBCS 文字 (すなわち 2 バイト文字) のことです。

%SUBSTR 演算子を使用するなら、文字ストリング変数のサブストリングを別の変数または変数のサブストリングに代入することができます。ソース変数のデータが

ターゲット変数に、左から右へコピーされます。ソース変数またはターゲット変数あるいはその両方がサブストリングの場合、オペランドは、文字ストリング変数の全体ではなく、サブストリングの部分だけです。ソース変数とターゲット変数のサイズが異なる場合、次の切り捨てまたは埋め込みの規則が適用されます。

- ソース変数の長さがターゲット変数よりも長い場合、文字ストリングはターゲット変数の長さに切り捨てられます。
- ソース変数の長さがターゲット変数よりも短い場合、文字ストリングはターゲット変数内で左そろえされて、残りの部分にはブランクが埋め込まれます。
- ソース変数の長さとターゲット変数の長さが等しい場合、代入後の 2 つの変数はまったく同じになります。

注: 同じ文字ストリング変数のサブストリングをソース変数とターゲット変数の両方で使用することができますが、ターゲット・ストリングの一部がソース・ストリングに重なるとエラーになります。

図 47 は、%SUBSTR 演算子の使用法を示す例です。

```
                                   評価式
前のデバッグ式
> EVAL CHAR10
   CHAR10 = '10CHARLONG'
> EVAL CHARA
   CHARA = 'A'
> EVAL CHARA = %SUBSTR(CHAR10 3 5)
   CHARA = 'C'
> EVAL %SUBSTR(CHAR10 1 2) = 'A'
   CHAR10 = 'A CHARLONG'
> EVAL %SUBSTR(CHAR10 1 2) = 'XYZ'
   CHAR10 = 'XYCHARLONG'
> EVAL %SUBSTR(CHAR10 7 4) = 'ABCD'
   CHAR10 = 'XYCHARABCD'
> EVAL %SUBSTR(CHAR10 1 2) = %SUBSTR(CHAR10 7 4)
   CHAR10 = 'ABCHARABCD'
                                                                  終わり
デバッグ     ________________________________________________________

F3= 終了   F9= コマンドの複写    F12= 取り消し   F16= 検索の繰り返し    F19= 左
F20= 右    F21= コマンド入力     F23= 出力の表示
```

図 47. %SUBSTR 演算子を使用して文字ストリング変数のサブストリングを表示する

レベル 01 またはレベル 77 のデータ項目のアドレスの表示

ILE ソース・デバッガーの EVAL コマンドの %ADDR 演算子を使用して、レベル 01 またはレベル 77 のデータ項目のアドレスを表示できます。レベル 01 またはレベル 77 の変数のアドレスを表示するには、次のように入力します。

```
EVAL %ADDR(variable-name)
```

ILE COBOL には間接参照演算子はありませんが、それでもポインター・データ項目が指す領域を複数の 16 進値または文字値として表示することはできます。ポインター・データ項目のターゲット領域を複数の 16 進値で表示するには、以下のように入力します。

```
EVAL pointer-name: x size
```

ポインター・データ項目のターゲット領域を複数の文字値で表示するには、以下のように入力します。

```
EVAL pointer-name: c size
```

レコード、グループ項目、および配列の表示

EVAL デバッグ・コマンドを使用して、構造、レコード、および配列を表示することができます。レコード、グループ項目、または配列が現在行にない場合は、表示したいレコード、グループ項目、または配列をまず修飾しなければなりません。それには、QUAL デバッグ・コマンドを使って行番号を指定します。 QUAL デバッグ・コマンドの使用法については 164 ページの『変数および式の表示』を参照してください。レコード、グループ項目、または配列を表示するには、デバッグ・コマンド行で次のように入力します。

```
EVAL data-name
```

データ名 は、表示したいレコード、グループ項目、または配列の名前です。レコード、グループ項目、または配列の値が「評価式」画面に表示されます。

次の例は、ILE COBOL グループ項目の内容を表示するための方法を示すものです。

```
     01  ACCOUNT.
         02  NUMBER             PIC 9(5).
         02  FULL-NAME.
             03  LAST-NAME      PIC X(20).
             03  FIRST-NAME     PIC X(10).
```

グループ項目 *ACCOUNT* の内容を表示するには、デバッグ・コマンド行で次のように入力します。

```
EVAL ACCOUNT
```

グループ項目 *ACCOUNT* の現在の内容が、 169 ページの図 48 に示すように「評価式」画面に表示されます。

グループ項目 *ACCOUNT* の単一エレメント、たとえば、エレメント FIRST-NAME OF FULL-NAME OF ACCOUNT を表示するには、デバッグ・コマンド行で次のように入力します。

```
EVAL FIRST-NAME OF FULL-NAME OF ACCOUNT
```

エレメント *FIRST-NAME OF FULL-NAME OF ACCOUNT* の現在の内容が、 169 ページの図 48 に示すように「評価式」画面に表示されます。実行キーを押すと、「モジュール・ソースの表示」画面に戻ります。他の名前と混同しさえしなければ、部分的な修飾名を使用してエレメントを表示することもできます。

```
                                評価式
前のデバッグ式
> EVAL ACCOUNT
  NUMBER OF ACCOUNT = 12345
  LAST-NAME OF FULL-NAME OF ACCOUNT = 'SMITH               '
  FIRST-NAME OF FULL-NAME OF ACCOUNT = 'JOHN      '
> EVAL FIRST-NAME OF FULL-NAME OF ACCOUNT
  FIRST-NAME OF FULL-NAME OF ACCOUNT = 'JOHN      '
```

図 48. EVAL デバッグ・コマンドを使用してグループ項目を表示する

次の例は、ILE COBOL の配列の内容を表示するための方法を示すものです。

```
05  A       PIC X(5) OCCURS 5 TIMES.
```

配列 *A* の内容を表示するには、デバッグ・コマンド行で次のように入力します。

```
EVAL A
```

配列 *A* の現在の内容が 図 49 に示すように「評価式」画面に表示されます。

配列 *A* のエレメントの範囲の内容を表示するには、デバッグ・コマンド行で次のように入力します。

```
EVAL A(2..4)
```

配列 *A* のエレメント A(2)、A(3)、および A(4) の現在の内容が 図 49 に示すように「評価式」画面に表示されます。

配列 *A* の単一エレメント、たとえば A(4) の内容を表示するには、デバッグ・コマンド行で次のように入力します。

```
EVAL A(4)
```

エレメント *A(4)* の現在の内容が図 49 に示すように「評価式」画面に表示されます。「モジュール・ソースの表示」画面に戻るには、F3 (終了) を押します。

注: EVAL デバッグ・コマンドに添え字として指定できるのは数値だけです。たとえば、A(4) は可能ですが、A(I+2) または A(2*3) は不可です。

```
                                評価式
前のデバッグ式
> EVAL A
  A(1) = 'ONE  '
  A(2) = 'TWO  '
  A(3) = 'THREE'
  A(4) = 'FOUR '
  A(5) = 'FIVE '
> EVAL A(2..4)
  A(2) = 'TWO  '
  A(3) = 'THREE'
  A(4) = 'FOUR '
> EVAL A(4)
  A(4) = 'FOUR '
```

図 49. EVAL デバッグ・コマンドを使用して配列を表示する

変数の値の変更

代入演算子を指定した EVAL コマンドを使用することによって、変数の値を変更することができます。変数が現在行にある場合を除き、変更したい変数は、まず QUAL デバッグ・コマンドを使用して、その行番号を指定しなければなりません。QUAL デバッグ・コマンドの使用法については 164 ページの『変数および式の表示』を参照してください。変数の値を変更するには、デバッグ・コマンド行で次のように入力します。

```
EVAL variable-name = value
```

変数名 は変更したい変数の名前であり、*値* は変数 *変数名* に代入する ID、リテラル、または定数値です。たとえば、

```
EVAL COUNTER=3
```

これにより、*COUNTER* の値が 3 に変更されます。この場合、次のものが表示されます。

```
COUNTER=3 = 3
```

これは「モジュール・ソースの表示」画面のメッセージ行に表示されます。

数値、英字、英数字、DBCS、ブール、浮動小数点、および日時データが変数の定義に一致している場合、EVAL デバッグ・コマンドを使用して、上記の値を変数に代入することができます。

注: EVAL デバッグ・コマンドを使用して変数に代入した値が変数の定義に一致していない場合は、警告メッセージが表示され、変数の値は変更されません。

変数に代入した値が文字ストリングの場合は、次の規則が適用されます。

- 変数に代入する文字ストリングの長さは、変数の長さと一致していなければなりません。
- 変数に代入する文字ストリングの長さが変数より短い場合、文字ストリングは左そろえされ、残りの位置にはブランクが埋め込まれます。
- 変数に代入する文字ストリングの長さが変数より長い場合、文字ストリングは変数の長さに切り捨てられます。

以下の例で、EVAL デバッグ・コマンドを使用して、異なるデータのタイプを変数に代入する方法を示します。

```
EVAL COUNTER=3                    (COUNTER is a numeric variable)
EVAL COUNTER=LIMIT                (LIMIT is another numeric variable)
EVAL END-OF-FILE='1'              (END-OF-FILE is a Boolean variable)
EVAL BOUNDARY=x'C9'               (BOUNDARY is an alphanumeric variable)
EVAL COMPUTER-NAME='ISERIES"      (COMPUTER-NAME is an alphanumeric variable)
EVAL INITIALS=%SUBSTR(NAME 17 3)  (INITIALS and NAME are alphanumeric variables)
EVAL DBCS-NAME= G'OEKIK2K3OF'      (K1K2K3 は DBCS 文字)

EVAL LONG-FLOAT(3) = -30.0E-3
        (LONG-FLOAT is an array of 3 double-precision floating-point
         data items - COMP-2)

EVAL SHORT-FLOAT = 10
        (SHORT-FLOAT is a single-precision floating-point data item -
         COMP-1)
```

注: EVAL デバッグ・コマンドを使用して形象定数を変数に代入することはできません。形象定数は EVAL デバッグ・コマンドではサポートされていません。EVAL デバッグ・コマンドを使用して PROCEDURE DIVISION の定数項目の値を変更することは可能な場合があります。しかし、結果は予測不能です。

変数、式、またはコマンドに名前を割り当てる

EQUATE デバッグ・コマンドを使用すると、変数、式、またはデバッグ・コマンドに名前を割り当てることにより、短縮名を使用することができます。その名前は、単独で使用したり、別の式で使用したりすることができます。名前を別の式で使用する場合、式が評価される前に名前の変数が判別されます。これらの名前は、デバッグ・セッションが終了するか、名前が除去されるまで引き続き使用することができます。

変数、式、またはデバッグ・コマンドに名前を割り当てるには、デバッグ・コマンド行で次のように入力します。

```
EQUATE shorthand-name definition
```

短縮名 は、変数、式、またはデバッグ・コマンドに割り当てる名前です。*定義* は、名前を割り当てる変数、式、またはデバッグ・コマンドです。

たとえば、*COUNTER* という変数の内容が *DC* という短縮名で表示するように定義するには、デバッグ・コマンド行で次のように入力します。

```
EQUATE DC EVAL COUNTER
```

これで、*DC* とデバッグ・コマンド行に入力するごとに、コマンド EVAL *COUNTER* が実行されるようになります。

EQUATE コマンドには、最大 144 文字まで入力できます。定義が指定されず、前の EQUATE コマンドが名前を定義している場合には、前の定義が除去されます。名前が前に定義されていなかった場合には、エラー・メッセージが表示されます。

EQUATE デバッグ・コマンドを使用して、デバッグ・セッション用に定義されていた名前を表示するには、デバッグ・コマンド行で次のように入力します。

```
DISPLAY EQUATE
```

活動状態の名前のリストが「評価式」画面に表示されます。

ILE ソース・デバッガーの各国語サポート

ILE ソース・デバッガーの各国語サポートの処理では、次の条件が適用されます。

- ビューが「モジュール・ソースの表示」画面に表示されている場合、ILE ソース・デバッガーは、すべてのデータをデバッグ・ジョブの CCSID に変換します。
- リテラルを変数に代入する場合、ILE ソース・デバッガーは、引用符付きのリテラル (たとえば、'abc') の CCSID 変換を実行しません。さらに、引用符付きのリテラルは、大文字と小文字を区別します。

ソース・ビューの処理において、ソース・ビューの元であるソース・ファイルの CCSID がモジュール・オブジェクトの CCSID と異なる場合、ILE ソース・デバッガーは、不変文字を含む ILE COBOL ID を認識できないことがあります。

次の条件のどちらかが成立している場合、

- デバッグ・ジョブの CCSID が 290、930、または 5026 (日本語カタカナ) である
- デバッグに使用する装置記述のコード・ページが、290、930、または 5026 (日本語カタカナ)

デバッグ・コマンド、関数、および 16 進値リテラルは、大文字で入力しなければなりません。 たとえば、

```
BREAK 16 WHEN var=X'A1B2'
EVAL var:X
```

しかし、ILE COBOL、ILE RPG、または ILE CL のモジュール・オブジェクトのデバッグでは、デバッグ・コマンドの識別名がソース・デバッガーにより大文字に変換されてしまうので、表示が異なることがあります。

ロケールに基づく変数の変更と表示

ILE COBOL では、日時クラスの項目、あるいは数字編集項目は、全体的に、あるいは一部分がロケールに基づきます。たとえば、日付項目を、次のように定義することができます。

```
01 group-item.
   05 date1 FORMAT DATE SIZE 10 LOCALE is locale-french.
```

この場合、日付項目の形式およびその日付項目の内容を構成する文字の CCSID は、ロケール locale-french に基づくものとなります。

ロケールを作成する場合、そのロケールは、ロケール・ソース・メンバーで記述する必要があります。ロケール・ソースは COBOL ソースと似ています。一定数のセクションと事前定義済みの構文とセマンティクスをもっており、COBOL ソースと同様に、ロケール・オブジェクトを作成するためにはコンパイルする必要があります。ロケール・オブジェクトを作成するためには、CCSID を指定し、さらにロケール・ソース・メンバー名、ファイル、およびライブラリーを指定する必要があります。ロケールの作成については 221 ページの『i5/OS でのロケールの作成』を参照してください。

これは、COBOL データ項目 date1 は、ジョブ CCSID とは別の CCSID をもつことができることを意味しています。 ILE ソース・デバッガーには date1 の CCSID を判別する方法がないため、データ項目の CCSID をジョブ CCSID に変換します。このために、データ項目の内容が間違って表示されることがあります。これらのタイプのデータ項目の正しい内容を見るために、それらを 16 進数で表示することができます。たとえば、date1 の内容を 16 進数で見るには、次のように入力します。

```
EVAL date-1:x
```

ユーザー定義データ・タイプについてのサポート

データ部 (DATA DIVISION) 内のデータ項目をユーザー定義データ・タイプとして定義しても、デバッガーによるデータの解釈方法は変更されません。 TYPE 文節を使用して定義されたデータ項目の行動は、TYPE 文節を使用せずに定義された場合とまったく同じです。

第 2 部 ILE COBOL プログラミング考慮事項

第 8 章 データ項目の処理

この章では ILE COBOL 数値データを処理する方法、数値データをうまく表示する方法、および効率的な算術演算の実行方法について説明します。その他、組み込み関数の使用と、日時クラスの項目の処理についても説明しています。トピックは、次のとおりです。

ILE COBOL における一般的な数値の表示 (PICTURE 文節)

一般に、ILE COBOL の数値データは、文字ストリング・データの場合と同様、一連の 10 進数字として表示することができます。しかし、数字項目には算術符号などの特殊な特性があります。

数字項目の定義

数字項目は、その数の 10 進数としての各桁を表す文字「9」をデータ記述部に使用することによって定義します (英数字項目の場合は「X」)。

```
05  COUNT-X         PIC 9(4)   VALUE 25.
05  CUSTOMER-NAME   PIC X(20)  VALUE "Johnson".
```

PICTURE 文節には最大 18 桁までの数字と、特別な意味を持つさまざまな文字をコーディングできます。次の例の「S」は、値に符号が付けられます。

```
05  PRICE           PIC S99V99.
```

このフィールドには正または負の値を入れることができます。「V」は暗黙の小数点の位置を示します。「S」も「V」も項目のサイズとしてはカウントされず、そのためのストレージは必要ありません。ただし、その項目が SIGN IS SEPARATE 文節の USAGE DISPLAY としてコーディングされている場合は除きます。例外は、内部浮動小数点データ (COMP-1 および COMP-2) の場合で、これには PICTURE 文節がありません。たとえば、内部浮動小数点データ項目は次のように定義されます。

```
05  GROMMET-SIZE-DEVIATION    USAGE COMP-1    VALUE 02.35E-5
```

コンパイラーの浮動小数点データ項目の処理を制御する方法については 41 ページの『CVTOPT パラメーター』および 60 ページの『PROCESS ステートメントを使用したコンパイラー・オプションの指定』の *FLOAT および *NOFLOAT を参照してください。

独立の符号位置 (移植性のため)

プログラムやデータを別のマシンに移植する計画があるなら、次のようにして符号を独立した 1 つの桁としてストレージ中にコーディングすることもできます。

```
05  PRICE             PIC S99V9   SIGN IS LEADING, SEPARATE.
```

このようにすると、符号を独立して保管する規則のないマシンを使用している場合に、それとは異なる規則のマシンを使用したときに異常な結果が出ないようにすることができます。

表示可能記号のための余分の記憶位置 (数字編集)

特定の編集記号 (小数点、コンマ、円記号など) を使用して数字項目を定義すると、データを表示したり報告書に印刷したりする際に読みやすく、分かりやすくなります。たとえば、

```
05 PRICE           PIC   9(5)V99.
05 EDITED-PRICE    PIC  $ZZ,ZZ9V99.
   .
   .
   .
   MOVE PRICE to EDITED-PRICE
   DISPLAY EDITED-PRICE
```

PRICE の内容が 0150099 (値 1,500.99 を表す) の場合、このコードを実行すると、¥ 1,500.99 と表示されるようになります。

数字編集項目を数値として使用する方法

数字編集項目は、数値ではなく英数字データ項目にクラス分けされています。したがって、数字編集項目を演算式や ADD、SUBTRACT、MULTIPLY、DIVIDE、および COMPUTE ステートメントでのオペランドにすることはできません。

数字編集項目を、数字項目および数字編集項目に移動 (MOVE) することができます。次の例では、数字編集項目が*編集解除* され、その値が数字データ項目に移動されます。

```
MOVE EDITED-PRICE to PRICE.
DISPLAY PRICE.
```

これら 2 つのステートメントが上記の例で示したステートメントの直後に続いている場合、PRICE は値 1,500.99 を表す 0150099 として表示されます。

数字編集項目は、ロケールと関連付けることもできます。ロケールに基づく数字編集項目に対して MOVE を行うと、結果は、そのロケールに応じて編集されます。ロケールと関連付けられた CCSID も編集結果に影響し、プログラムの実行時に、プログラムが使用するファイル、ロケール、および数字編集項目と関連付けられた CCSID は、変換が必要かどうかを判別するために比較されます。実行時の CCSID の取扱方法については 22 ページの『実行時 CCSID に関する考慮事項』を参照してください。

数値データのデータ記述に関する詳しい情報については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

計算データの表記 (USAGE 文節)

数値データ項目のコンピューター内部での保管方法を制御するには、データ記述記入項目に USAGE 文節をコーディングします。プログラムで使用する数値データは、ILE COBOL で使用可能な次の形式のうちのいずれかです。

- 外部 10 進数 (USAGE DISPLAY)
- 内部 10 進数 (USAGE PACKED-DECIMAL または COMP-3)
- 2 進数 (USAGE COMP-4 または BINARY)
- ネイティブ 2 進数 (USAGE COMP-5)
- 外部浮動小数点 (USAGE DISPLAY)
- 内部浮動小数点 (USAGE COMP-1、USAGE COMP-2)

COMP-4 は BINARY と、また COMP および COMP-3 は PACKED-DECIMAL とそれぞれ同義です。

コンピューターの値の内部表記を制御するためにどの USAGE 文節を使用しているかにはかかわりなく、同じ PICTURE 文節規則と 10 進数値を VALUE 文節に使用します。ただし、浮動小数点データを除きます。

外部 10 進数 (USAGE DISPLAY) 項目

USAGE DISPLAY をコーディングする場合、または USAGE 文節を省略した場合、ストレージの各位置 (すなわち各バイト) ごとに 1 つの 10 進数数字が入れられます。これは出力の印刷や表示に使用する形式に対応しており、その項目が表示可能な形式で保管されることを意味します。

USAGE DISPLAY 項目の用途

外部 10 進数項目は、主としてプログラムとファイル、端末、およびプリンターの間での数値のやり取りを目的にしています。しかし、それはプログラムの算術処理のオペランドおよび受け取り側としても使用でき、そのようなプログラミングを行う際にしばしば便利なものとなります。

算術処理に使用する場合

プログラムの行う演算が膨大な量であり、効率が優先される場合は、その演算で使用するデータ項目に対して、ILE COBOL の計算数値データ・タイプのうちのいずれかを使用するとよいでしょう。

コンピューターが表示可能数値を算術演算で使用するためには、その前にコンピューターは、その数値の*内部* 表記に自動的に変換する必要があります。したがって、多くの場合、データ項目を DISPLAY 項目ではなく、始めから計算項目として定義しておくほうが効率がよくなります。たとえば、

```
05 COUNT-X          PIC S9V9(5)   USAGE COMP   VALUE 3.14159.
```

内部 10 進数 (USAGE PACKED-DECIMAL または COMP-3)

パック 10 進数形式は、PICTURE 記述の 2 つの 10 進数ごとに 1 バイトのストレージを占有します。ただし、右端のバイトの内容は 1 つの数字と符号だけです。こ

の形式が非常に効率的なのは、PICTURE 記述に奇数桁の数字をコーディングする場合です。この場合には、左端のバイトをいっぱいに使用できます。パック 10 進数形式は、演算を目的として固定小数点数値として処理されます。

パック 10 進数を使用する理由

パック 10 進数形式は次のような理由で使用します。

- 1 桁あたりのストレージが DISPLAY 形式よりも少なくてすむ
- 2 進数形式に比べ、小数点位置合わせに適している
- 2 進数形式よりも DISPLAY 形式への変換および DISPLAY 形式からの変換が容易にできる
- 演算のオペランドまたは結果を入れるのに適している

2 進数 (USAGE BINARY または COMP-4) 項目

2 進数形式は、2、4、または 8 バイトのストレージを占有し、演算においては左端ビットが演算符号である固定小数点数値として処理されます。バイト配置が逆の 2 進数データの場合、符号ビットは右端バイト中の左端ビットになります。

BINARY が占有するストレージ量

4 桁以下の 10 進数の PICTURE 記述部は 2 バイト、5 ～ 9 桁は 4 バイト、10 ～ 18 桁は 8 バイトになります。

2 進数項目は、添え字や参照変更の開始位置および長さを入れるのに適しています。

しかし、BINARY 形式は小数点位置合わせには適していないので、ILE COBOL では、演算式中の BINARY 数値は PACKED DECIMAL 形式に変換されます。したがって、PACKED DECIMAL 形式を使用する方が演算式には適しています。

また、数値を表示形式に変換する場合も、BINARY 形式より PACKED DECIMAL 形式の方が適しています。数値を BINARY 形式から DISPLAY 形式に変換するよりも、PACKED DECIMAL 形式から DISPLAY 形式に変換する方が面倒です。

2 進数データの切り捨て (*STDTRUNC コンパイラー・オプション)

BINARY および COMP-4 データの切り捨て方法を指定するには、*STDTRUNC および *NOSTDTRUNC コンパイラー・オプション (35 ページの『OPTION パラメーター』の『OPTION パラメーター』を参照) を使用します。

ネイティブ 2 進数 (USAGE COMP-5) 項目

ネイティブ 2 進数形式は、2 進数形式 (USAGE BINARY または COMP-4) と似ていますが、以下の点が異なります。

- *STDTRUNC および *NOSTDTRUC コンパイラー・オプションは、ネイティブ 2 進数項目には適用されません。データ項目には、(USAGE BINARY データの場合のように) 項目のピクチャーにおける 9 の数により暗黙指定される値に制限されるのではなく、最大でネイティブ 2 進数表記の容量 (2、4、または 8 バイト) の値を入れることができます。数値データが COMP-5 項目に移動または保管され

| ると、COBOL ピクチャー・サイズ限界ではなく、2 進数フィールド・サイズで
| 切り捨てが行われます。 COMP-5 項目が参照される場合、その操作でフル 2 進
| 数フィールド・サイズが使用されます。

- COMP-5 ネイティブ 2 進数形式の項目が符号なしで定義されている場合、演算符
| 号用にビットは使用されません。その代わりに、数値データ用にすべてのビット
| が使用され、数値は負になりません。

内部浮動小数点 (USAGE COMP-1 および COMP-2) 項目

COMP-1 は短 (単精度の) 浮動小数点形式、COMP-2 は長 (倍精度の) 浮動小数点形式のことで、ストレージの占有はそれぞれ 4 バイトと 8 バイトです。左端ビットに符号、続く 7 ビットに指数、残りの 3 バイトまたは 7 バイトに小数部が入れられます。

IBM i では、データ項目 COMP-1 および COMP-2 は IEEE 形式で表示されます。

浮動小数点データ項目のデータ記述部で PICTURE 文節は使用できませんが、VALUE 文節で浮動小数点リテラルを使用することにより、初期値を与えることができます。

```
05 COMPUTE-RESULT     USAGE COMP-1   VALUE 06.23E-24.
```

浮動小数点形式と他の数値形式間の変換の特性については 187 ページの『データ・フォーマットの変換』で説明します。

浮動小数点形式は、算術演算のオペランドおよび結果を入れたり、算術演算で最高レベルの正確性を維持するのに適しています。

数値データのデータ記述に関する詳しい情報については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

外部浮動小数点 (USAGE DISPLAY) 項目

浮動小数点形式でコーディングされた表示可能数値のことを、**外部浮動小数点**項目と呼びます。外部 10 進数項目と同様、外部浮動小数点項目の定義は、USAGE DISPLAY を使用して明示的に行うか、USAGE 文節を省略することで暗黙に行います。

次の例では、COMPUTE-RESULT が外部浮動小数点項目として暗黙定義されています。ストレージのバイトごとに 1 つの文字が入れられます (V は除く)。

```
05 COMPUTE-RESULT     PIC -9V(9)E-99.
```

VALUE 文節は、外部浮動小数点項目のデータ記述部では使用できません。また、負符号 (-) は、必ずしも小数部および指数が負の数値であることを示すものではなく、表示においてこの記号は、正ならブランク、負なら負符号となります。正符号 (+) を使用している場合、正数は正符号で、負数は負符号でそれぞれ表示されます。

外部 10 進数数値の場合と同様、外部浮動小数点数値は数値の内部表記に変換 (コンパイラーで自動的に行う) してからでなければ演算できません。外部浮動小数点数値は、必ず長精度内部浮動小数点形式に変換されます。

ユーザー定義データ・タイプの作成

ILE COBOL では、TYPEDEF 文節を使用して、**ユーザー定義データ・タイプ**を作成できます。ユーザー定義データ・タイプは、英数字、数字、ブールなど、ILE COBOL データ・タイプですでに使用可能なものへの追加ではありません。ユーザー定義データ・タイプ (**型定義**または**型名**とも呼ばれる) は、実際は TYPEDEF 文節を使用して、プログラムの WORKING-STORAGE、LOCAL-STORAGE、LINKAGE、または FILE セクションで定義された、項目全体またはグループ項目です。これらの型定義はテンプレートと同様に行動するため、TYPE 文節を使用して新しいデータ項目を定義するのに使用できます。新しいデータ項目は、ユーザー定義データ・タイプのすべての特性を獲得します。ユーザー定義データ・タイプがグループ項目であるならば、その新しいデータ項目は、ユーザー定義データ・タイプに属しているものと同じ名前、記述、および階層の従属エレメントをもちます。

ユーザー定義データ・タイプでは、時間が節約でき、ソース・コードを最小化できます。それは、プログラム内の 2 つまたはそれ以上のデータ項目の定義の一部として発生する複雑なデータ構造を再定義する必要がないためです。必要なことは、定義を 1 つ作成し、それを、TYPE 文節を使用して、必要とするものと同じタイプの後続の定義のどれかに適用するだけです。

たとえば、次の 2 つのタイプの品目を配送する小規模流通業者用の在庫プログラムを開発するとします。
- 衣類。色やサイズが多様です。
- 書籍。タイトルが多種多様です。

在庫プログラムで、個々の衣類品目や書籍について手元にある数量をカウントし、別個のデータ項目にそれらを格納してから、衣類および書籍の在庫の累計も別個のデータ項目に入れることにします。

183 ページの図 50 は、TYPEDEF 文節と TYPE 文節を使用して、時間を節約し、これと同様のプログラムの WORKING-STORAGE セクションのソース・コードを最小化する方法の例です。

この例では、衣類と書籍用のユーザー定義データ・タイプを作成します。次に、ユーザー定義データ・タイプに基づいて、3 つの異なる衣類品目と 2 つの異なる書籍品目に別個のデータ項目を作成します。これは、在庫のタイプごとに定義をコーディングし直すよりも、簡単で効率が高い方法です。間違いをする確率も、少なくなります。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/SAMPTYPE        AISERIES  06/02/15 13:31:06       ページ    2

                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.    SAMPTYPE.
          000300
          000400*****************************************************************
          000500* The following program demonstrates some of the funcitons
          000600* available with the TYPE and TYPEDEF clauses.
          000700*****************************************************************
          000800
    3     000900 ENVIRONMENT DIVISION.
    4     001000 CONFIGURATION SECTION.
    5     001100 SOURCE-COMPUTER.  IBM-ISERIES.
    6     001200 OBJECT-COMPUTER.  IBM-ISERIES.
    7     001300 INPUT-OUTPUT SECTION.
    8     001400 FILE-CONTROL.
    9     001500     SELECT DATA-IN
   10     001600         ASSIGN TO Database-INVDATA
   11     001700         ORGANIZATION IS INDEXED
   12     001800         record key is inv-type
          001900            with duplicates
   13     002000         ACCESS MODE IS SEQUENTIAL.
          002100
   14     002200     SELECT PRINTER-FILE
   15     002300         ASSIGN TO PRINTER-QPRINT
   16     002400         ORGANIZATION IS SEQUENTIAL
   17     002500         ACCESS MODE IS SEQUENTIAL.
          002600
   18     002700 DATA DIVISION.
   19     002800 FILE SECTION.
   20     002900 FD PRINTER-FILE.
   21     003000 01 PRINTER-REC.
   22     003100    05 PRINTER-RECORD    PIC X(132).
          003200
          003300*****************************************************************
          003400* define inventory type
          003500*****************************************************************
   23     003600 01 INV-TYPE-T IS TYPEDEF PIC S9(3) VALUE 0.
   24     003700    88 INV-TYPE-BOOK             VALUE 4, 5.
   25     003800    88 INV-TYPE-BOOK-001         VALUE 4.
   26     003900    88 INV-TYPE-BOOK-002         VALUE 5.
   27     004000    88 INV-TYPE-CLOTHES          VALUE 1, 2, 3.
   28     004100    88 INV-TYPE-CLOTHES-SWEATERS VALUE 1.
   29     004200    88 INV-TYPE-CLOTHES-SOCKS    VALUE 2.
   30     004300    88 INV-TYPE-CLOTHES-PANTS    VALUE 3.
          004400
   31     004500 FD DATA-IN.
   32     004600 01 DATA-IN-REC.
   33     004700    05 INV-TYPE          TYPE INV-TYPE-T.
   34     004800    05 FILLER            PIC X(80).
          004900
   35     005000 WORKING-STORAGE SECTION.
          005100*****************************************************************
          005200* Initialize END-OF-FILE flag to FALSE
          005300*****************************************************************
```

図 50. プログラムにおける TYPEDEF 文節と TYPE 文節の使用法を示す例 (1/4)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/SAMPTYPE        AISERIES   06/02/15 13:31:06        ページ     3
STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
       005400
  36   005500 01 END-OF-FILE PIC 1 VALUE B"0".
  37   005600    88 AT-END-OF-FILE VALUE B"1".
  38   005700 01 ITEM-PRICE-T TYPEDEF PIC S9(4)V9(2) value 0.
  39   005800 01 ITEM-COLOR-T TYPEDEF PIC S9(2) VALUE 1.
  40   005900    88 ITEM-COLOR-BLUE  VALUE 1.
  41   006000    88 ITEM-COLOR-RED   VALUE 2.
  42   006100    88 ITEM-COLOR-GREEN VALUE 3.
  43   006200 01 ITEM-SIZE-T    TYPEDEF PIC S9(2) VALUE 10.
  44   006300 01 ITEM-COUNTER-T TYPEDEF PIC S9(6) VALUE 0.
       006400
  45   006500 01 ITEM-B-T TYPEDEF.
  46   006600    05  ITEM-B-VALUE PIC s9(2).
  47   006700    88 ITEM-B-BLUE  VALUE 1.
  48   006800    88 ITEM-B-RED   VALUE 2.
  49   006900    88 ITEM-B-GREEN VALUE 3.
  50   007000 01 TEST-ITEM TYPE ITEM-B-T.
       007100
  51   007200 01 WORK-INV-TYPE  TYPE INV-TYPE-T.
       007300*****************************************************************
       007400* User-defined data type for items of clothing.
       007500* Items of clothing are INVENTORY-TYPE 1 through 3.
       007600*****************************************************************
       007700
  52   007800 01 CLOTHING-ITEM IS TYPEDEF.
  53   007900    05 CLOTHING-TYPE   TYPE INV-TYPE-T.
  54   008000    05 PRICE           TYPE ITEM-PRICE-T.
  55   008100    05 COLOR           TYPE ITEM-COLOR-T.
  56   008200    05 CLOTHING-SIZE   TYPE ITEM-SIZE-T.
  57   008300    05 FILLER PIC X(70).
       008400
  58   008500 01 SWEATERS TYPE CLOTHING-ITEM.
  59   008600 01 SOCKS TYPE CLOTHING-ITEM.
  60   008700 01 PANTS TYPE CLOTHING-ITEM.
       008800
       008900*****************************************************************
       009000* User-defined data type for books.
       009100* Books are INVENTORY-TYPE 4 through 5.
       009200*****************************************************************
       009300
  61   009400 01 BOOK-ITEM IS TYPEDEF.
  62   009500    05 BOOK-TYPE    TYPE INV-TYPE-T.
  63   009600    05 PRICE        TYPE ITEM-PRICE-T.
  64   009700    05 FILLER       PIC X(20).
  65   009800    05 BOOK-TITLE   PIC X(40).
  66   009900    05 FILLER       PIC X(14).
       010000
  67   010100 01 BOOK-001 TYPE BOOK-ITEM.
  68   010200 01 BOOK-002 TYPE BOOK-ITEM.
       010300
       010400*****************************************************************
       010500* Initialize all of the inventory counters.
       010600*****************************************************************
       010700
  69   010800 01 sweaters-count TYPE item-counter-t.
```

図 50. プログラムにおける TYPEDEF 文節と TYPE 文節の使用法を示す例 (2/4)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/SAMPTYPE         AISERIES  06/02/15 13:31:06        ページ    4
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
   70     010900 01 socks-count TYPE item-counter-t.
   71     011000 01 pants-count TYPE item-counter-t.
          011100
   72     011200 01 book-001-count TYPE item-counter-t.
   73     011300 01 book-002-count TYPE item-counter-t.
          011400
   74     011500 01 clothes-count TYPE item-counter-t.
   75     011600 01 book-count TYPE item-counter-t.
          011700
          011800*****************************************************************
          011900* Declare report variables.
          012000*****************************************************************
          012100
   76     012200 01 header-line.
   77     012300    05 FILLER   pic x(40) value spaces.
   78     012400    05 FILLER   pic x(52) value "Detailed Inventory Report".
   79     012500    05 FILLER   pic x(40) value spaces.
          012600
   80     012700 01 DETAIL-LINE.
   81     012800    05 FILLER             pic x(10) value spaces.
   82     012900    05 ITEM-DESCRIPTION   pic x(25) value spaces.
   83     013000    05 ITEM-QUANTITY      pic 9(6) blank when zero.
   84     013100    05 FILLER             pic x(92) value spaces.
          013200
          013300
   85     013400 PROCEDURE DIVISION.
          013500 MAIN-PAR.
   86     013600     OPEN INPUT DATA-IN
          013700          OUTPUT PRINTER-FILE.
          013800
          013900
          014000*****************************************************************
          014100* Read the first record.
          014200*****************************************************************
          014300
   87     014400     READ DATA-IN
          014500         AT END
   88     014600            SET AT-END-OF-FILE TO TRUE
          014700         NOT AT END
   89     014800            MOVE INV-TYPE TO WORK-INV-TYPE
          014900     END-READ.
          015000
          015100*****************************************************************
          015200* Tally each of the inventory types and move the amounts into
          015300* separate totals.
          015400*****************************************************************
          015500
   90     015600     PERFORM UNTIL AT-END-OF-FILE
   91     015700         EVALUATE TRUE
          015800         WHEN INV-TYPE-CLOTHES-SWEATERS OF WORK-INV-TYPE
   92     015900            ADD 1 TO sweaters-count
   93     016000            ADD 1 TO clothES-count
          016100         WHEN INV-TYPE-CLOTHES-SOCKS OF WORK-INV-TYPE
   94     016200            ADD 1 TO socks-count
   95     016300            ADD 1 TO clothES-count
```

図 50. プログラムにおける TYPEDEF 文節と TYPE 文節の使用法を示す例 (3/4)

```
5722WDS V5R4M0 060210 LN  IBM ILE COBOL                CBLGUIDE/SAMPTYPE        AISERIES   06/02/15 13:31:06        Page      5
STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
        016400         WHEN INV-TYPE-CLOTHES-PANTS OF WORK-INV-TYPE
  96    016500            ADD 1 TO pants-count
  97    016600            ADD 1 TO clothES-COUNT
        016700         WHEN INV-TYPE-BOOK-001 OF WORK-INV-TYPE
  98    016800            ADD 1 TO book-001-count
  99    016900            ADD 1 TO book-count
        017000         WHEN INV-TYPE-BOOK-002 OF WORK-INV-TYPE
 100    017100            ADD 1 TO book-002-count
 101    017200            ADD 1 TO book-count
        017300         END-EVALUATE
        017400
 102    017500         READ DATA-IN
        017600            AT END
 103    017700               SET AT-END-OF-FILE TO TRUE
        017800            NOT AT END
 104    017900               MOVE INV-TYPE TO WORK-INV-TYPE
        018000         END-READ
        018100     END-PERFORM.
        018200
        018300*****************************************************************
        018400* Write report.
        018500*****************************************************************
        018600
 105    018700     PERFORM REPORT-WRITE.
 106    018800     CLOSE DATA-IN
        018900           PRINTER-FILE.
 107    019000     STOP RUN.
        019100
        019200*****************************************************************
        019300* Procedure to write report.
        019400*****************************************************************
        019500
        019600 REPORT-WRITE.
 108    019700     WRITE PRINTER-REC FROM HEADER-LINE AFTER ADVANCING PAGE.
        019800
 109    019900     MOVE "BOOKS:"               TO ITEM-DESCRIPTION.
 110    020000     MOVE ZEROS                  TO ITEM-QUANTITY.
 111    020100     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
        020200
 112    020300     MOVE "Best-seller Number 1:" TO ITEM-DESCRIPTION.
 113    020400     MOVE BOOK-001-COUNT          TO ITEM-QUANTITY.
 114    020500     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
        020600
 115    020700     MOVE "Best-seller Number 2:" TO ITEM-DESCRIPTION.
 116    020800     MOVE BOOK-002-COUNT          TO ITEM-QUANTITY.
 117    020900     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
        021000
 118    021100     MOVE "Total Books:"          TO ITEM-DESCRIPTION.
 119    021200     MOVE BOOK-COUNT              TO ITEM-QUANTITY.
 120    021300     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
        021400
 121    021500     WRITE PRINTER-REC FROM HEADER-LINE AFTER ADVANCING PAGE.
        021600
 122    021700     MOVE "CLOTHES:"             TO ITEM-DESCRIPTION.
 123    021800     MOVE ZEROS                  TO ITEM-QUANTITY.
 124    021900     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
        022000
 125    022100     MOVE "Sweaters:"            TO ITEM-DESCRIPTION.
 126    022200     MOVE SWEATERS-COUNT         TO ITEM-QUANTITY.
 127    022300     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
        022400
 128    022500     MOVE "Socks:"               TO ITEM-DESCRIPTION.
 129    022600     MOVE SOCKS-COUNT            TO ITEM-QUANTITY.
 130    022700     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
        022800
 131    022900     MOVE "Pants:"               TO ITEM-DESCRIPTION.
 132    023000     MOVE PANTS-COUNT            TO ITEM-QUANTITY.
 133    023100     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
        023200
 134    023300     MOVE "Total Clothes:"       TO ITEM-DESCRIPTION.
 135    023400     MOVE CLOTHES-COUNT          TO ITEM-QUANTITY.
 136    023500     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
        023600
                       * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 50. プログラムにおける TYPEDEF 文節と TYPE 文節の使用法を示す例 (4/4)

データ・フォーマットの変換

プログラムのコードに別のデータ形式を持つ項目との相互作用が含まれている場合、コンパイラーはそのような項目を次のように変換します。

- 比較演算および算術演算では、一時的に変換する
- MOVE または COMPUTE ステートメントの受け取り側への代入では、永続的に変換する

変換の意味

変換とは、実際には、ある値をあるデータ項目から別のデータ項目に移動することです。コンパイラーは、算術ステートメントおよび比較ステートメントの実行中に変換が必要になったなら、MOVE および COMPUTE ステートメントで使用しているのと同じ規則に従って行います。移動の規則は、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」で定義されています。コンパイラーは、桁ごとに移動するのではなく、可能な限りその数値の*値* を保持するように移動します。 (切り捨てと予測される有効数字の脱落については『変換と精度』を参照してください。)

変換に要する時間

変換が実行される場合、データが内部作業域に移されて変換されてから演算が実行されるため、多くの場合、付加的なストレージおよび処理時間が必要になります。また、その結果をもう一度作業域に戻して、再変換しなければならない場合もあります。

変換と精度

異なる固定小数点データ形式 (外部 10 進数、パック 10 進数、および 2 進数) の間の変換は、ターゲット・フィールドにソース・オペランドのすべての数字が入りさえすれば、精度を落とすことなく完了できます。

データが失われる可能性のある変換

固定小数点データ形式と浮動小数点データ形式 (短精度浮動小数点、長精度浮動小数点、および外部浮動小数点) との間の変換では、精度が落ちる可能性があります。こうした変換は、固定小数点オペランドと浮動小数点オペランドが混在している演算評価の際に生じます。 (以下のいくつかの例で固定小数点項目について述べていますが、固定小数点項目と外部浮動小数点項目には共に 10 進数の特性があるため、特に明記されていない限り外部浮動小数点項目についても同じことが言えます。)

固定小数点から内部浮動小数点形式に変換する場合、基数 10 の固定小数点数値は、内部で使用している基数 16 の数値システムに変換されます。

コンパイラーは短形式を長形式に変換して比較しますが、短い数値のための埋め込みには 0 を使用します。

USAGE COMP-1 データ項目が 7 桁以上の固定小数点データ項目に移動される場合、固定小数点データ項目の受け取る有効数字は 6 桁だけであり、残りの桁は 0 になります。

精度を保存する変換: 6 桁以下の固定小数点データ項目が USAGE COMP-1 データ項目に移動されてから、再度固定小数点データ項目に戻されると、それは元の値になります。

USAGE COMP-1 データ項目が 6 桁以上の固定小数点データ項目に移動されてから、再度 USAGE COMP-1 データ項目に戻されると、それは元の値になります。

15 桁以下の固定小数点データ項目が USAGE COMP-2 データ項目に移動されてから、再度固定小数点データ項目に戻されると、それは元の値になります。

USAGE COMP-2 データ項目が 18 桁の固定小数点 (外部浮動小数点ではない) データ項目に移されてから、再度 USAGE COMP-2 データ項目に戻されると、それは元の値になります。

丸めが行われる変換: USAGE COMP-1 データ項目、USAGE COMP-2 データ項目、外部浮動小数点データ項目、または浮動小数点リテラルが固定小数点データ項目に移動されると、ターゲット・データ項目の低位桁で丸めが生じます。

USAGE COMP-2 データ項目が USAGE COMP-1 データ項目に移動されると、ターゲット・データ項目の低位桁で丸めが生じます。

固定小数点データ項目が外部浮動小数点データ項目に移される場合、その固定小数点データ項目の PICTURE に含まれる桁数が、外部浮動小数点データ項目の PICTURE に含まれる桁数より多いならば、ターゲット・データ項目の低位桁で丸めが生じます。

外部浮動小数点データが表示されるか、受け入れられたとき、あるいは外部浮動小数点リテラルが外部浮動小数点データ項目に移動されたときに、表示されるか、受け入れられるか、あるいは受け取られた外部浮動小数点データ項目が、不正確な値である可能性があります。これは、浮動小数点データ・タイプが近似値であるためです。外部浮動小数点リテラルは受け入れ、表示、または移動されたときに、最初に、真の浮動小数点値 (IEEE) に変換され、これがその正確性に影響することはあり得ます。たとえば、次の MOVE の例を考えてみます。

```
77 external-float-1 PIC +9(3).9(13)E+9(3).
   MOVE +123455779012.34523E+297 to external-float-1.
   DISPLAY "EXTERNAL-FLOAT-1=" external-float-1.
```

表示される MOVE の結果は、次のとおりです。

```
EXTERNAL-FLOAT-1=+123.4557790123452E+306
```

符号表記と処理

符号表記は、数字データの処理および対話に影響を及ぼします。

X'sd' (s は符号表記、d は数字) に対して、外部 10 進数 (SIGN IS SEPARATE 文節のない USAGE DISPLAY) の有効な符号表記は次のようになります。

正: A、C、E、および F

負: B および D

内部で生成される符号は、正および符号なしの場合は F、負の場合は D です。

X'ds' (d は数字、s は符号表記) に対して、内部 10 進数の ILE COBOL データ (USAGE PACKED-DECIMAL) の有効な符号表記は次のようになります。

正: A、C、E、および F

負: B および D

内部で生成される符号は、正および符号なしの場合は F、負の場合は D です。

*CHGPOSSN コンパイラー・オプションの指定

ILE COBOL コンパイラー・オプション *CHGPOSSGN は、外部 10 進データおよび内部 10 進データの符号処理に影響します。 *CHGPOSSGN は 2 進数データや浮動小数点データには影響を及ぼしません。詳細については 40 ページの『*NOCHGPOSSGN および *CHGPOSSGN』の *CHGPOSSN の説明を参照してください。

コンパイラーで生成される正の符号は通常 F であり、VALUE 文節同様、MOVE および演算ステートメントでは C になります。

*CHGPOSSGN コンパイラー・オプションは、デフォルトの *NOCHGPOSSGN ほど効率的ではないので、それを使用するのは、符号の扱いが異なる MVS、VM、あるいはその他のシステムとデータを共用する場合だけにしてください。

互換性のないデータのチェック (数値クラス・テスト)

コンパイラーは、データ項目に対して指定された値がその項目の PICTURE および USAGE 文節に有効なものであると見なし、指定された値の妥当性検査を行わず、そのまま代入します。ある項目に、そのデータ記述部と互換性のない値を指定すると、PROCEDURE DIVISION にあるその項目への参照は未定義になり、結果は予測不能です。

値がプログラムに渡されて、データ記述部にその値との互換性のない項目に代入されることがよくあります。たとえば、非数値データが、プログラムの中で符号なし数値として定義されているフィールドに移動されたり、渡されたりすることがあります。いずれの場合も、それらのフィールドの内容は無効なデータです。データ項目の処理ステップを進める前に、データ項目の内容がその PICTURE および USAGE 文節と一致していることを確認するようにしてください。

数値クラス・テストの実行方法

数値クラス・テストによってデータの妥当性検査を行うことができます。たとえば、

```
LINKAGE SECTION.
01   COUNT-X     PIC 999.
   .
   .
   .
   PROCEDURE DIVISION USING COUNT-X.
     IF COUNT-X IS NUMERIC THEN DISPLAY "DATA IS GOOD".
   .
   .
   .
```

数値クラス・テストは、データ項目の内容が、そのデータ項目の特定の PICTURE および USAGE に対して有効な値になっているかどうかをチェックすることです。たとえば、パック 10 進数項目の場合、16 進数の桁位置が X'0' ～ X'9' になっているかどうか、また符号位置が有効な符号値かどうか (独立か非独立か) を検査することになります。

算術演算の実行

ILE COBOL と ILE の実行時には、算術演算を実行するための次のような機能があります。

- ADD、SUBTRACT、MULTIPLY、DIVIDE、および COMPUTE ステートメント (『COMPUTE およびその他の演算ステートメント』を参照)。
- 演算式 (191 ページの『演算式』を参照)。
- 組み込み関数 (191 ページの『数値組み込み関数』を参照)。
- ILE 呼び出し可能サービス (API)

  ILE には、どの ILE コンパイラーでも使用できる、複数のグループからなるバインド可能 API があります。数学 API には階乗を計算する CEE4SIFAC とコサイン値を計算する CEESDCOS が含まれています。

  使用できるバインド可能 API の詳細については、 Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「プログラミング」カテゴリーの中の『*CL および API*』セクションを参照してください。

ILE COBOL 言語構成要素の構文および使用法の詳しい説明については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

COMPUTE およびその他の演算ステートメント

一般に、ほとんどの算術演算の評価には、ADD、SUBTRACT、MULTIPLY、および DIVIDE ステートメントではなく、COMPUTE ステートメントを使用します。それは、複数の個別のステートメントに相当するものを、1 つの COMPUTE ステートメントでコーディングできることが多いためです。

COMPUTE ステートメントは、次のようにして演算式の結果をデータ項目に代入します。

```
COMPUTE Z = A + B / C ** D - E
```

また、次のように複数のデータ項目にも代入できます。

```
COMPUTE X Y Z = A + B / C ** D - E
```

その他の演算ステートメントの使用

ある種の演算では、その他の演算ステートメントを使用するほうがもっと直感的である場合があります。たとえば、

```
ADD 1 TO INCREMENT.
```

上記のほうが、次のものより直感的です。

```
COMPUTE INCREMENT = INCREMENT + 1.
```

また、

```
SUBTRACT OVERDRAFT FROM BALANCE.
```

上記のほうが、次のものより直感的です。

```
COMPUTE BALANCE = BALANCE - OVERDRAFT.
```

また、

```
ADD 1 TO INCREMENT-1, INCREMENT-2, INCREMENT-3.
```

上記のほうが、次のものより直感的です。

```
COMPUTE INCREMENT-1 = INCREMENT-1 + 1
COMPUTE INCREMENT-2 = INCREMENT-2 + 1
COMPUTE INCREMENT-3 = INCREMENT-3 + 1
```

このほか、DIVIDE ステートメント (とその REMAINDER 句) を使用して除算の剰余を処理する場合も考えられます。剰余は、REM 組み込み関数でも処理できます。

演算式

上記の COMPUTE の例で、等号の右辺はどれも演算式を表しています。演算式を構成するのは、単一の数字リテラル、単一の数値データ項目、または単一の組み込み関数の参照です。また、それらの項目のいくつかを算術演算子で結合したもので構成されることもあります。このような演算子は、階層順に評価されます。

表 11. 演算子の評価

演算子	意味	評価順序
単項 + または -	代数符号	最初
**	累乗	2 番目
/ または *	除算または乗算	3 番目
2 項 + または -	加算または減算	最後

同じレベルの演算子は、左から右へ評価されますが、演算子に括弧を使用すれば評価順序を変更できます。括弧内の式は、個々の演算子に先だって評価されます。必要かどうかを別としても、括弧があるとプログラムが読みやすくなります。

演算式は、COMPUTE ステートメント以外でも、数字データ項目を指定できる他の場所でも使用できます。たとえば、演算式を比較条件での被比較項目として使用できます。

```
IF (A + B) > (C - D + 5) THEN...
```

数値組み込み関数

組み込み関数は、英数字、DBCS、数値、ブール、または日時の値を戻します。

数値組み込み関数は、以下のとおりです。

- 符号付き数値を戻します。
- 一時数値データ項目と見なされます。
- 式を指定できる言語構文中でのみ使用可能です。

- それらの関数が網羅している多くの一般的な計算を自分で用意する必要がないので、時間を節約できます。組み込み関数の活用については 193 ページの『組み込み関数の例』を参照してください。

数字関数の種類

数字関数は次のカテゴリーにクラス分けされます。

整数　　整数を戻すもの。

浮動小数点　　長精度浮動小数点値を戻すもの。

引き数依存　　リターン・タイプは指定する引き数によって異なります。

表 12 に、ILE COBOL で使用可能なこれらのカテゴリーの数字関数のリストを示します。

表 12. 数字関数が戻すデータの型

整数	浮動小数点	引き数依存
DATE-OF-INTEGER	ACOS	MAX *
DATE-TO-YYYYMMDD	ANNUITY	MIN *
DAY-OF-INTEGER	ASIN	RANGE
DAY-TO-YYYYDDD	ATAN	SUM
EXTRACT-DATE-TIME	COS	
FACTORIAL	LOG	
FIND-DURATION	LOG10	
INTEGER	MEAN	
INTEGER-OF-DATE	MEDIAN	
INTEGER-OF-DAY	MIDRANGE	
INTEGER-PART	NUMVAL	
LENGTH	NUMVAL-C	
MOD	PRESENT-VALUE	
ORD	RANDOM	
ORD-MAX	REM	
ORD-MIN	SIN	
YEAR-TO-YYYY	SQRT	
	STANDARD-DEVIATION	
	TAN	
	VARIANCE	

注: * MAX および MIN は英数字にすることができます。

関数のネストと演算式

数字関数はネスト可能です。ある関数を別の関数の引き数として参照できます。ネストされた関数は、外部関数とは別に評価されます。

数字関数と演算式では構文に関する状況がよく似ているので、演算式を数字関数の引き数としてネストすることもできます。

```
COMPUTE X = FUNCTION MEAN (A, B, C / D).
```

この例では、関数引き数は A、B、および演算式 (C / D) の 3 つだけです。

ALL 添え字と特殊レジスター

組み込み関数の便利な機能として、このほかに ALL 添え字と特殊レジスターの 2 つがあります。

ALL 添え字を使用すれば、ある配列の全エレメントを関数引き数として参照できます。この機能はテーブルで使用します。

整数タイプの特殊レジスターは、整数引き数を使用できるところでは引き数としてどこでも使用できます。

組み込み関数の例

組み込み関数を使用することにより、以下の表に示されている各種の演算を実行できます。

表 13. 数字関数が処理する演算のタイプ

数値処理	日付 / 時刻	金融	数学	統計
LENGTH	CURRENT-DATE	ANNUITY	ACOS	MEAN
MAX	DATE-OF-INTEGER	PRESENT	ASIN	MEDIAN
MIN	DAY-TO-YYYYDDD	-VALUE	ATAN	MIDRANGE
NUMVAL	DATE-TO-YYYYMM		COS	RANDOM
NUMVAL-C	DD		FACTORIAL	RANGE
ORD-MAX	DAY-OF-INTEGER		INTEGER	STANDARD
ORD-MIN	EXTRACT-DATE-		INTEGER-PART	-DEVIATION
	TIME		LOG	VARIANCE
	FIND-DURATION		LOG10	
	INTEGER-OF-DATE		MOD	
	INTEGER-OF-DAY		REM	
	WHEN-COMPILED		SIN	
	YEAR-TO-YYYY		SQRT	
			SUM	
			TAN	

次の例と説明で、上記の表に示されている各カテゴリーの組み込み関数を示します。

一般的な数値処理: 3 つの価格の平均値 (ドル記号の付いた英数字項目として表示) を調べたい場合、その値を出力レコードの数字フィールドに入れ、出力レコードの長さを調べます。それには、NUMVAL-C (英数字ストリングの数値の値を戻す関数) と MEAN 関数を使用できます。

```
01 X                PIC 9(2).
01 PRICE1           PIC X(8)     VALUE "$8000".
01 PRICE2           PIC X(8)     VALUE "$4000".
01 PRICE3           PIC X(8)     VALUE "$6000".
01 OUTPUT-RECORD.
   05 PRODUCT-NAME   PIC X(20).
   05 PRODUCT-NUMBER PIC 9(9).
   05 PRODUCT-PRICE  PIC 9(6).
.
```

```
.
.
PROCEDURE DIVISION.
   COMPUTE PRODUCT-PRICE =
       FUNCTION MEAN (FUNCTION NUMVAL-C(PRICE1)
                      FUNCTION NUMVAL-C(PRICE2)
                      FUNCTION NUMVAL-C(PRICE3)).
       COMPUTE X = FUNCTION LENGTH(OUTPUT-RECORD).
```

さらに、PRODUCT-NAME の内容を大文字にするため、次のようなステートメントを使用できます。

```
    MOVE FUNCTION UPPER-CASE(PRODUCT-NAME) TO PRODUCT-NAME.
```

日付および時刻: 次の例は、90 日後の期限を計算する方法を示しています。CURRENT-DATE 関数が戻す最初の 8 文字は、4 桁の年、2 桁の月、そして 2 桁の日付の形式 (YYYYMMDD) を表しています。この例では、その日付をその整数値に変換します。その後、その値に 90 を加えます。整数が YYYYMMDD 形式に再び変換されます。

```
01 YYYYMMDD              PIC 9(8).
01 INTEGER-FORM          PIC S9(9).
   .
   .
   .
   MOVE FUNCTION CURRENT-DATE(1:8) TO YYYYMMDD.
   COMPUTE INTEGER-FORM = FUNCTION INTEGER-OF-DATE(YYYYMMDD).
   ADD 90 TO INTEGER-FORM.
   COMPUTE YYYYMMDD = FUNCTION DATE-OF-INTEGER(INTEGER-FORM).
   DISPLAY 'Due Date: ' YYYYMMDD.
```

満期日をカテゴリー日時データ項目として計算することもできます。このタイプの計算の例については 205 ページの『満期日の計算の例』を参照してください。

金融: 事業投資の判断を行うためには、将来見込まれる現金流入の現在の値を計算して、計画中の投資のもうけを見積もらなければならない場合がしばしばあります。金額の現在値とは、その日の値のことです。将来の特定の時に受け取ることが期待できる額の現在値は、その日に投資して、一定の利率で蓄積されて達する将来の額のことです。

たとえば、$1,000 投資すると今後 3 年間に $100、$200、$300 というように毎年支払いを受けるものとします。次の ILE COBOL ステートメントは、このような現金流入の現在値の計算方法を示しています (利率を 10% とします)。

```
01 SERIES-AMT1              PIC 9(9)V99     VALUE 100.
01 SERIES-AMT2              PIC 9(9)V99     VALUE 200.
01 SERIES-AMT3              PIC 9(9)V99     VALUE 300.
01 DISCOUNT-RATE            PIC S9(2)V9(6)  VALUE .10.
01 TODAYS-VALUE             PIC 9(9)V99.
.
.
.
COMPUTE TODAYS-VALUE =
FUNCTION
PRESENT-VALUE(DISCOUNT-RATE SERIES-AMT1 SERIES-AMT2
SERIES-AMT3).
```

ANNUITY 関数は、融資の元利を返済するための分割払い額を決めなければならないような事業上の問題に使用できます。支払いは、毎期同額、毎期同期間、毎期同

利率を特徴としています。次の例は、$15,000 の融資を 3 年 (36 カ月) で返済するのに必要な月額の計算方法です (年利 12%、月利 = .12/12 とします)。

```
01 LOAN                PIC 9(9)V99.
01 PAYMENT             PIC 9(9)V99.
01 INTEREST            PIC 9(9)V99.
01 NUMBER-PERIODS      PIC 99.
.
.
.
COMPUTE LOAN = 15000.
COMPUTE INTEREST = .12
COMPUTE NUMBER-PERIODS = 36.
COMPUTE PAYMENT =
LOAN * FUNCTION ANNUITY((INTEREST / 12) NUMBER-PERIODS).
```

数学: 次の ILE COBOL ステートメントは、組み込み関数のネスト方法、引き数を演算式にする方法、および従来複雑だった数学計算を簡単に行う方法を示しています。

```
COMPUTE Z = FUNCTION LOG(FUNCTION SQRT (2 * X + 1))
+ FUNCTION REM(X 2)
```

ここで、X を 2 で割った剰余は、REMAINDER 文節を使用する DIVIDE ステートメントではなく、組み込み関数を使用して求めています。

統計: 組み込み関数を使用すると、データに関する統計情報の計算も容易になります。さまざまな市民税を計算して、平均、中間、および範囲 (最大額と最少額の差) を出したい場合、次のようにします。

```
01 TAX-S               PIC 99V999 VALUE .045.
01 TAX-T               PIC 99V999 VALUE .02.
01 TAX-W               PIC 99V999 VALUE .035.
01 TAX-B               PIC 99V999 VALUE .03.
01 AVE-TAX             PIC 99V999.
01 MEAN-TAX            PIC 99V999.
01 TAX-RANGE           PIC 99V999.
   .
   .
   .
COMPUTE AVE-TAX = FUNCTION MEAN(TAX-S TAX-W TAX-B)
COMPUTE MEDIAN-TAX = FUNCTION MEDIAN(TAX-S TAX-W TAX-B)
COMPUTE TAX-RANGE = FUNCTION RANGE(TAX-S TAX-W TAX-B)
```

データ項目の変換 (組み込み関数)

組み込み関数を使用することによって、文字ストリング・データ項目を次のものに変換できます。

- 大文字または小文字
- 逆順
- 数値
- 日時データ項目
- UTF-8

NATIONAL-OF および DISPLAY-OF 組み込み関数を使用すると、国別 (Unicode) ストリングとの間で相互に変換することができます。

ストリングから先頭または末尾 (あるいはその両方の) 文字を除去するには、TRIM、TRIML、TRIMR 組み込み関数を使用します。

組み込み関数で文字変換を行うほかに、INSPECT ステートメントも使用できます。

大文字または小文字への変換 (UPPER-CASE、LOWER-CASE)

以下のコーディングの場合、

```
01 ITEM-1              PIC X(30)     VALUE "Hello World!".
01 ITEM-2              PIC X(30).
   .
   .
   .
   DISPLAY ITEM-1.
   DISPLAY FUNCTION UPPER-CASE(ITEM-1).
   DISPLAY FUNCTION LOWER-CASE(ITEM-1).
   MOVE FUNCTION UPPER-CASE(ITEM-1) TO ITEM-2.
   DISPLAY ITEM-2.
```

次のメッセージを端末に表示します。

```
Hello World!
HELLO WORLD!
hello world!
HELLO WORLD!
```

DISPLAY ステートメントは、ITEM-1 の実際の内容を変更することなく、文字の表示方法だけに影響します。しかし、MOVE ステートメントは大文字を ITEM-2 の実際の内容に移動します。

逆順への変換 (REVERSE)

以下のコードは、

```
   MOVE FUNCTION REVERSE(ORIG-CUST-NAME) TO ORIG-CUST-NAME.
```

ORIG-CUST-NAME の文字の順序を逆にします。たとえば、開始値が JOHNSON であれば、ステートメントの実行後に NOSNHOJ になります。

数値への変換 (NUMVAL、NUMVAL-C)

NUMVAL 関数および NUMVAL-C 関数は、文字ストリングを数値に変換します。これらの関数は、フリー・フォーマット文字表記を含んだ英数字データ項目を数値形式に変換し、それらを数値的に処理するのに使用します。たとえば、

```
01 R            PIC X(20)   VALUE "- 1234.5678".
01 S            PIC X(20)   VALUE "-$12,345.67CR".
01 TOTAL        USAGE IS COMP-2.
   .
   .
   .
   COMPUTE TOTAL = FUNCTION NUMVAL(R) + FUNCTION NUMVAL-C(S).
```

NUMVAL と NUMVAL-C との違いは、例にも見られるように、引き数に通貨記号やコンマが含まれる場合には NUMVAL-C を使用するという点です。先頭または末尾に代数符号を付けても、問題なく処理されます。引き数は、18 桁以下 (ただし、編集記号は含みません) にする必要があります。正確な構文規則については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

注: NUMVAL も NUMVAL-C も、長精度 (倍精度) 浮動小数点値を戻します。したがって、これらのいずれかの関数への参照は、数字データ項目への参照になります。

NUMVAL および NUMVAL-C を使用する理由: NUMVAL や NUMVAL-C を使用すると、数字データを固定形式で静的に宣言したり、入力データを正確に宣言したりする必要がありません。たとえば、次のコーディングの場合、

```
01 X         PIC S999V99   LEADING SIGN IS SEPARATE.
.
.
.
ACCEPT X FROM CONSOLE.
```

アプリケーションのユーザーは、数値の入力を PICTURE 文節の定義どおり正確に行う必要があります。たとえば、

```
+001.23
-300.00
```

しかし、NUMVAL 関数を使用する場合は、次のようにコーディングできます。

```
01 A         PIC X(10).
01 B         PIC S999V99.
.
.
.
ACCEPT A FROM CONSOLE.
COMPUTE B = FUNCTION NUMVAL(A).
```

以下のように入力することができます。

```
1.23
-300
```

日時データ項目への変換 (CONVERT-DATE-TIME)

CONVERT-DATE-TIME 関数は、英数字、数字、または日時項目を受けとり、それを日時データ項目に変換します。組み込み関数を使用して、次のことが行えます。

- 日付、時刻、またはタイム・スタンプを英数字 (ストリング) 項目から日時項目へ変換する。
- ある形式のカテゴリー日付の項目を、ロケールに基づく形式の、別のカテゴリー日付項目へ変換する。

たとえば、以下のステートメントは、非数値リテラル (英数字定数) をカテゴリー日付データ項目へ変換します。

```
MOVE FUNCTION CONVERT-DATE-TIME ('98/08/09' DATE '%y/%m/%d')
     TO DATE-1.
```

変換は、日付が含まれている数値データ項目を日時データ項目に比較または移動する際にも発生します。これらのタイプの移動に関する考慮事項については 216 ページの『日時データ項目の MOVE に関する考慮事項』を参照してください。

日付を含む英数字データ項目を日時データ項目へ移動する際には、変換は行われません。英数字データ項目に含まれている文字はすべて、日時データ項目へ移動されます。英数字データ項目に含まれている日付を日時データ項目へ移動される前に正しい形式にするのは、プログラマーの責任において行ってください。

UTF-8 (UTF8STRING) への変換

UTF8STRING 関数は、文字ストリングを UTF-8 (UCS Transformation Format 8) に変換します。UTF-8 コード化形式は、CCSID 1208 によって表されます。たとえば、次のとおりです。

```
01 STR1 PIC X(3) VALUE "ABC".
01 VRR-X3 PIC X(3).
.
.
.
MOVE FUNCTION UTF8STRING(STR1) TO VRR-X3.
```

VRR-X3 の内容は X"414243" になります。

英数字または DBCS から国別データへの変換 (NATIONAL-OF)

英字、英数字、または DBCS の項目を Unicode (UCS-2) で表現される文字ストリングに変換するには、NATIONAL-OF 組み込み関数を使用します。ソースが、CCSID コンパイラー・オプションによって有効になっているものとは異なるコード・ページでエンコードされている場合、ソース・コード・ページを引数として指定します。

国別データから英数字または DBCS データへの変換 (DISPLAY-OF)

引数として指定したコード・ページ、または CCSID コンパイラー・オプションを使用して指定したコード・ページで表される文字ストリングに国別項目を変換するには、DISPLAY-OF 組み込み関数を使用します。 SBCS と DBCS を結合している EBCDIC コードを指定した場合、戻されるストリングには SBCS 文字と DBCS 文字が混在している場合があります。この場合、DBCS サブストリングは、シフトインおよびシフトアウト文字で区切られます。

デフォルトのコード・ページの指定変更

場合によっては、CCSID オプション値として指定された CCSID とは異なる CCSID との間でデータを相互に変換する必要が生じることもあります。これを行うには、変換関数を使用し、この関数でその項目のコード・ページを明示的に指定します。

CCSID コンパイラー・オプションで指定したコード・ページとは異なるコード・ページを DISPLAY-OF の引数として指定した場合は、暗黙の変換を伴う操作 (国別データ項目との間の割当や比較など) には DISPLAY-OF 関数の結果を使用しないでください。そのような操作では、EBCDIC コード・ページは CCSID コンパイラー・オプションで指定されていることを想定しています。

変換の例外

国別データと英数字データとの暗黙的または明示的変換が失敗した場合、重大度 40 のエラーが生成されることがあります。このエラーは、以下のいずれかに起因する可能性があります。

- (暗黙的または明示的に) 指定したコード・ページが無効なコード・ページである。
- 明示的または暗黙的に指定した CCSID (例えば、CCSID コンパイラー・オプションを指定した結果) と UCS-2 Unicode CCSID (例えば、National CCSID コンパ

イラー・オプションを指定して、明示的または暗黙的に指定した結果) との組み
合わせが、当該のオペレーティング・システムでサポートされていない。

ターゲットの CCSID に対応するものがない文字は、変換例外にはなりません。そのような文字は、ターゲットのコード・ページの置換文字に変換されます。

次の例は、NATIONAL-OF および DISPLAY-OF 組み込み関数を使用して、Unicode ストリングとの間で変換を行う方法を示しています。またこの例は、同一のプログラムで、複数のコード・ページでエンコードされたストリングを操作するときには、明示的な変換が必要であることも示しています。

```
 PROCESS CCSID(37)
*...
 01  Data-in-Unicode pic N(100) usage national.
 01  Data-in-Greek pic X(100).
 01  other-data-in-US-English pic X(12) value "PRICE in $=".
*...
     Read Greek-file into Data-in-Greek
     Move function National-of(Data-in-Greek, 00875)
         to Data-in-Unicode
*...process Data-in-Unicode here ...
     Move function Display-of(Data-in-Unicode, 00875)
         to Data-in-Greek
     Write Greek-record from Data-in-Greek
```

上の例は、正常に動作します。Data-in-Greek は、CCSID 00875 (ギリシャ語) で明示的に表されるデータとして変換されます。ただし、(項目内のすべての文字が、ギリシャ語と英語のコード・ページ内に共通の表記を持つ文字の中にない限り) 次のステートメントは誤変換になります。

```
Move Data-in-Greek to Data-in-Unicode
```

Data-in-Greek は、CCSID 00037 (U.S. English) から UCS-2 への変換に基づいて、MOVE ステートメントによって Unicode に変換されます。 Data-in-Greek は実際には CCSID 00875 でエンコードされているので、この変換は失敗します。

CCSID コンパイラー・オプションを正しく CCSID 00875 に設定できる場合 (すなわち、プログラムの残りの部分も EBCDIC データをギリシャ語で処理している場合)、上の例は以下のように正しくコーディングすることができます。

```
 PROCESS CCSID(00875)
*...
 01  Data-in-Unicode pic N(100) usage national.
 01  Data-in-Greek pic X(100).
     Read Greek-file into Data-in-Greek
*... process Data-in-Greek here ...
*... or do the following (if need to process data in Unicode)
     Move Data-in-Greek to Data-in-Unicode
*... process Data-in-Unicode
     Move function Display-of(Data-in-Unicode) to Data-in-Greek
     Write Greek-record from Data-in-Greek
```

先頭または末尾 (あるいはその両方) の文字の除去 (TRIM、TRIML、TRIMR)

TRIM、TRIML、TRIMR 関数は、ブランクまたは指定された文字をストリングから除去します。たとえば、

```
01 ADDR.
  05 STREET-NO  PIC X(5)  VALUE "120".
  05 STEET-NAME PIC X(50) VALUE "Young Street".
```

```
  05 CITY       PIC X(20) VALUE "Toronto".
  05 STATE      PIC X(15) VALUE "Ontario".
  05 ZIP        PIC X(6)  VALUE "M1C5D9".
01 ADDRESS-LINE PIC X(80).
    STRING FUNCTION TRIM(STREET-NO) " "
           FUNCTION TRIM(STREET-NAME) ", "
           FUNCTION TRIM(CITY) ", " FUNCTION TRIM(STATE) " "
           FUNCTION TRIM(ZIP) DELIMITED BY SIZE
           INTO ADDRESS-LINE.
    DISPLAY ADDRESS-LINE.
```

この出力は、以下のようになります。

```
120 Young Street, Toronto, Ontario M1C5D9
```

データ項目の評価 (組み込み関数)

いくつかの組み込み関数は、データ項目の評価に使用できます。

- CHAR および ORD は、プログラムで使用している照合順序に関して、整数および単一英数字を評価するのに使用します。
- 一連のデータ項目で最大項目および最小項目を検索するための MAX、MIN、ORD-MAX、および ORD-MIN。
- LENGTH はデータ項目の長さを調べるのに使用します。
- WHEN-COMPILED はプログラムのコンパイル日時を調べるのに使用します。
- TEST-DATE-TIME は、日時、英数字、数値パック、またはゾーン項目が、有効な日付、時刻、またはタイム・スタンプであるかどうかを判別するために使用します。

照合順序に関する単一文字の評価 (CHAR、ORD)

特定の文字の照合順序における順序位置を知りたい場合、その文字を引き数にして ORD 関数を参照すると、ORD はその順序位置を示す整数を戻します。これを行うための 1 つの便利な方法は、データ項目のサブストリングを ORD の引き数として使用する方法です。

```
IF FUNCTION ORD (CUSTOMER-RECORD(1:1)) IS > 194 THEN ...
```

一方、照合順序における順序位置は分かっているが、それに対応する文字が分からないという場合、次のように整数の順序位置を引き数にして CHAR 関数を参照すれば、CHAR がその文字を戻します。

```
INITIALIZE CUSTOMER-NAME REPLACING ALPHABETIC BY FUNCTION CHAR(65).
```

英数字関数によって可変長の結果を戻す

英数字関数の結果は、その長さや値が関数の引き数によって変わる場合があります。

次の例の場合、R3 に移動されたデータの量と COMPUTE ステートメントの結果は、R1 および R2 の値とサイズによって変わります。

```
01 R1            PIC X(10) VALUE "e".
01 R2            PIC X(05) VALUE "f".
01 R3            PIC X(05) VALUE "g".
01 R4            PIC X(20) VALUE SPACES.
01 L             PIC 99.
.
.
.
MOVE FUNCTION MEAN(R1 R2 R3) TO R4.
COMPUTE L = FUNCTION LENGTH(FUNCTION MEAN(R1 R2 R3)).
```

ここで R2 の評価結果は中間値となります。したがって、記号 b でブランク・スペースを表すとすると、ストリング「fbbbb」が R4 に移動され、(R4 のうち文字が入っていない桁にはスペースが入れられる)、L の評価結果は値 5 になります。 R1 が値「f」なら、R1 が中間値となり、ストリング「fbbbbbbbbb」が R4 に移動され (R4 のうち文字が入っていない桁にはスペースが入れられる)、値 10 が L に代入されます。

英数字関数からの可変長結果を処理する場合があります。プログラム・コードをそれにしたがって計画してください。たとえば次のように、書き込むレコードの長さが異なる可能性がある場合、可変長レコード・ファイルの使用を考慮する必要があります。

```
   FILE SECTION.
FD OUTPUT-FILE.
01 CUSTOMER-RECORD     PIC X (80).
WORKING-STORAGE SECTION.
01 R1                      PIC X (50).
01 R2                      PIC X (70).
.
.
.
WRITE CUSTOMER-RECORD FROM FUNCTION MEAN(R1 R2 R3).
```

最大データ項目または最小データ項目の検索 (MAX、MIN、ORD-MAX、ORD-MIN)

複数の英数字データ項目があり、どのデータ項目が最大値 (照合シーケンスに従って評価される) を含んでいるかを知りたい場合は、MAX 関数または ORD-MAX 関数を使用して、そのデータ項目を引き数として提供します。どのデータ項目が最小値を含んでいるかを知りたい場合は、MIN 関数または ORD-MIN 関数を使用します。

MAX および MIN: MAX 関数および MIN 関数は、単に、提供した変数のどれかの内容を戻すだけです。たとえば、以下のデータ定義では、

```
05 Arg1 Pic x(10) Value "THOMASSON ".
05 Arg2 Pic x(10) Value "THOMAS ".
05 Arg3 Pic x(10) Value "VALLEJO ".
```

次のステートメント

```
Move Function Max(Arg1 Arg2 Arg3) To Customer-record(1:10)
```

は、「VALLEJObbb」を、Customer-record の最初の 10 文字の位置に割り当てます。

注: ブランクは「b」で表しています。

代わりに MIN を使用すると、「THOMASbbbb」が戻されます。

ORD-MAX および ORD-MIN: ORD-MAX 関数および ORD-MIN 関数は、指定した引き数リスト内での、最大値または最小値をもった引き数の (左から数えた) 順番を表す整数を戻します。上記の例で ORD-MAX 関数を使用すると、コンパイル時に構文エラー・メッセージを受け取ります。これは、正しくない場所で数字関数を参照しようとするためです (「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照)。次の例は、ORD-MAX 関数の正しい例です。

```
Compute x = Function Ord-max(Arg1 Arg2 Arg3)
```

前の例と同じ引き数を使用した場合、これによって、整数 3 が x に割り当てられます。ORD-MIN を使用すると、整数 2 が戻されます。

注: この関数グループは、数値に対しても使用できます。この場合、引き数の代数値が比較されます。詳細については、191 ページの『演算式』を参照してください。上記の例は、Arg1、Arg2、および Arg3 が、配列テーブルの連続するエ

レメントである場合に、より現実的なものになります。テーブル・エレメントを関数引き数として使用する場合の説明については、209 ページの『テーブル項目の処理』を参照してください。

英数字関数によって可変長の結果を戻す: 英数字関数の結果は、その長さや値が関数の引き数によって変わる場合があります。次の例の場合、R3 に移動されたデータの量と COMPUTE ステートメントの結果は、R1 および R2 の値とサイズによって変わります。

```
01 R1 Pic x(10) value "e".
01 R2 Pic x(05) value "f".
01 R3 Pic x(20) value spaces.
01 L Pic 99.
.
.
Move Function Max(R1 R2) to R3
Compute L = Function Length(Function Max(R1 R2))
```

ここで、R2 は、R1 より大きい値と評価されます。したがって、記号 b がブランク・スペースを表すとすると、ストリング「fbbbb」が R3 に移動され (R3 のうち埋め込みのない文字位置にはスペースが入れられる)、L の評価結果は値 5 になります。 R1 が値「g」の場合は、R1 は R2 より大きくなり、ストリング「gbbbbbbbbb」が R3 に移動され (R3 のうち埋め込みのない文字位置にはスペースが入れられる)、値 10 が L に代入されます。

英数字関数からの可変長結果を処理する場合があります。プログラム・コードをそれにしたがって計画してください。たとえば次のように、書き込むレコードの長さが異なる可能性がある場合、可変長レコード・ファイルの使用を考慮する必要があります。

```
File Section.
FD Output-File.
01 Customer-Record Pic X(80).
Working-Storage Section.
01 R1 Pic x(50).
01 R2 Pic x(70).
.
.
Write Customer-Record from Function Max(R1 R2)
```

データ項目の長さを調べる (LENGTH)

LENGTH 関数は、多くのプログラミング・コンテキストでストリング項目の長さを調べるのに役立ちます。次の ILE COBOL ステートメントは、得意先名などのデータ項目を、得意先名用レコードの特定のフィールドに移動します。

```
MOVE CUSTOMER-NAME TO CUSTOMER-RECORD(1:FUNCTION LENGTH(CUSTOMER-NAME)).
```

注: LENGTH 関数は、数字データ項目やテーブル項目でも使用できます。

LENGTH OF 特殊レジスター: LENGTH 関数のほかに、データ項目の長さを検出するための方法は、LENGTH OF 特殊レジスターを使用する方法です。

LENGTH OF 特殊レジスターと LENGTH 組み込み関数は、基本的に異なっています。 FUNCTION LENGTH はある項目の長さを文字位置数で戻すのに対して、LENGTH OF はバイト数で戻します。大抵の場合、DBCS クラスの項目を別にすれば、結果は大差ありません。

たとえば、

```
77 CUSTOMER-NAME PIC X(30).
77 CUSTOMER-LOCATION-ASIA PIC G(50).
```

FUNCTION LENGTH(CUSTOMER-NAME) とコーディングしても LENGTH OF CUSTOMER-NAME とコーディングしても 30 を戻しますが、FUNCTION LENGTH(CUSTOMER-LOCATION-ASIA) は 50 を戻し、LENGTH OF CUSTOMER-LOCATION-ASIA は 100 を戻します。

LENGTH 関数は演算式を指定できる場合しか使用できませんが、LENGTH OF 特殊レジスターはさまざまなコンテキストで使用できます。たとえば、LENGTH OF 特殊レジスターを、整数引き数の可能な組み込み関数の引き数として使用することができます。 (組み込み関数は、LENGTH OF 特殊レジスターのオペランドとしては使用できません。) LENGTH OF 特殊レジスターは、CALL ステートメントのパラメーターとしても使用できます。

コンパイル日付を調べる (WHEN-COMPILED)

システムによって提供されるプログラムのコンパイル日時を知るには、WHEN-COMPILED 関数を使用できます。戻される結果は 21 桁で、最初の 16 桁の形式は以下のとおりです。

```
YYYYMMDDhhmmsshh
```

これは、コンパイル日時を 4 桁年、月、日、時刻 (時、分、秒、1/100 秒) で示したものです。

WHEN-COMPILED 特殊レジスター: WHEN-COMPILED 特殊レジスターは、コンパイル日時を調べるためのもう一つの方法です。形式は次のとおりです。

```
MM/DD/YYhh.mm.ss
```

WHEN-COMPILED 特殊レジスターがサポートする年は 2 桁だけで、時刻も秒までです。この特殊レジスターは、MOVE ステートメントの送り出しフィールドとしてのみ使用できます。

日時データ項目についてテストする (TEST-DATE-TIME)

日時、英数字、数値パック、またはゾーン項目が有効な日付、時刻、またはタイム・スタンプであるかどうかを知りたい場合には、TEST-DATE-TIME 組み込み関数を使用できます。 ADD-DURATION、または SUBTRACT-DURATION など別の日時組み込み関数を使用して移動または計算を完了する前に有効な日時データ項目についてテストすることは有用です。次の例は、日時データ項目についてのテスト方法を示しています。

```
IF FUNCTION TEST-DATE-TIME (date-3 DATE) = B'1'
   MOVE DATE-3 TO CUTOFF-DATE.
```

日付および時刻期間を処理する (ADD-DURATION、FIND-DURATION、SUBTRACT-DURATION)

組み込み関数を使用して、日付、時刻、およびタイム・スタンプの期間を処理することができます。たとえば、日付が 2 つあり、その 2 つの日付の間が何カ月であるかを知りたい場合には、FIND-DURATION 関数を使用して、これを計算すること

ができます。また、ADD-DURATION および SUBTRACT-DURATION 組み込み関数を使用して、満期日および失効日 (すでに経過した日付) を計算することもできます。

日時データ項目の使用法については 212 ページの『日時データ・タイプを処理する』を参照してください。

2 つの日付間の期間を見つける例: 次の例は、日時形式の 2 つの日付間の日数を計算する方法を示しています。

```
01 YYYYMMDD          FORMAT DATE "@Y%m%d".
01 EXPIRY-DATE       FORMAT DATE "%m/%d/@Y"  VALUE "10/31/1997".
01 DURATION          PIC S9(5).
   .
   .
   .
    MOVE FUNCTION CURRENT-DATE(1:8) TO YYYYMMDD.
    COMPUTE DURATION = FUNCTION FIND-DURATION (YYYYMMDD EXPIRY-DATE DAYS).
        IF DURATION <= 0 THEN
          DISPLAY 'Expiry date, ' EXPIRY-DATE ' has passed.'
        END-IF.
```

上記の FIND-DURATION 組み込み関数は、EXPIRY-DATE から YYYYMMDD を減じます。 YYYYMMDD の日付が 1997 年 10 月 31 日より後になった場合、期間は負の値として戻されます。ゼロ日数または負の日数の期間は、満了を表します。

現在日付が 1997 年 11 月 1 日であるとすれば、上記プログラムの出力は、次のようになります。

```
Expiry date 10/31/1997 has passed.
```

満期日の計算の例: 次の例は、日時形式の満期日の計算方法を示しています。

```
01 YYYYMMDD          FORMAT DATE "@Y%m%d".
01 DATE-TIME-FORM    FORMAT DATE "%m/%d/@Y".
   .
   .
   .
    MOVE FUNCTION CURRENT-DATE(1:8) TO YYYYMMDD.
    MOVE FUNCTION ADD-DURATION (YYYYMMDD DAYS 90) TO DATE-TIME-FORM.
    DISPLAY 'Due Date: ' DATE-TIME-FORM.
```

現在日付が 1997 年 10 月 8 日であるとすれば、上記プログラムの出力は、次のようになります。

```
Due Date: 01/06/1998
```

失効日の計算の例: 日付が過ぎてしまったために、その日付の記載されているもの (たとえば、小切手) が失効したかどうかを計算するには、次のように SUBTRACT-DURATION 組み込み関数を使用できます。

```
01 YYYYMMDD                    FORMAT DATE "@Y%m%d".
01 STALE-DATE                  FORMAT DATE "%m/%d/@Y".
01 cheque-date                 FORMAT DATE "%m/%d/@Y"  VALUE "03/09/1997".
   .
   .
   .
    MOVE FUNCTION CURRENT-DATE(1:8) TO YYYYMMDD.
    MOVE FUNCTION SUBTRACT-DURATION (YYYYMMDD DAYS 180) TO STALE-DATE.
    IF STALE-DATE > cheque-date THEN
       DISPLAY 'Cheque date, ' cheque-date ', is stale-dated.'
       DISPLAY 'The stale-date is: ' STALE-DATE
    END-IF.
```

現在日付が 1997 年 10 月 8 日であるとすれば、上記プログラムの出力は、次のようになります。

```
Cheque date, 03/09/1997, is stale-dated.
The stale date is: 04/11/1997
```

ロケールに基づいて日付と時刻をフォーマット設定する (LOCALE-DATE、LOCALE-TIME)

LOCALE 関数を使用して、日付または時刻を、特定の文化圏で一般的に使用している方法でフォーマット設定することができます。以下に示す例では、COBOL プログラムを実行する前に、ロケール・オブジェクト (タイプ *LOCALE) EN_US をライブラリー QSYSLOCALE に作成する必要があります。ロケール・オブジェクトの作成方法については 221 ページの『i5/OS でのロケールの作成』を参照してください。

LOCALE 関数は、LOCALE-DATE 組み込み関数の場合は日付の形式、また、LOCALE-TIME 組み込み関数の場合は時刻形式の英数字項目 (文字ストリング) をとり、特定の文化圏で一般的に使用されている方法でフォーマット設定された日付または時刻をもつ別の英数字項目を戻します。

LOCALE-DATE の引き数は、CURRENT-DATE 組み込み関数で指定された日付形式の 8 バイトの文字ストリングでなければなりません。 LOCALE-TIME の引き数は、CURRENT-DATE 組み込み関数で指定された時刻形式の、9 桁目から 21 桁目までの 13 バイトの文字ストリングでなければなりません。

たとえば、

```
SPECIAL-NAMES.
    LOCALE "EN_US" IN LIBRARY "QSYSLOCALE" IS usa.
:
DISPLAY "Date is:"  FUNCTION LOCALE-DATE("19970908" usa).
DISPLAY "Time is:"  FUNCTION LOCALE-TIME("06345200+0000" usa).
```

表示は、次のようになります。

```
Date is:  08/09/1997
Time is:  06:34:52
```

注: 上記の結果を獲得するには、ロケール USA が GMT 時間帯内になければなりません。

上の例では、LOCALE-DATE 関数の引き数-1 である 19970908 は、4 桁の年と、その後に続く月、月間通算日を表します。 LOCALE-TIME 関数の引き数-1 である 06345200+0000 は、次のものを表します。

- 最初の 6 桁は時、分、秒です。
- 7 番目および 8 番目の文字は、100 分の 1 秒です。
- 9 番目の文字はプラスまたはマイナスです。
- 10 番目および 11 番目の数字は、GMT (グリニッジ標準時) との時間単位の差です。 (これらの 2 つの数字は LOCALE-TIME 関数では使用されません。)
- 12 番目および 13 番目の数字は分を表します。

固定小数点演算と浮動小数点演算

プログラムの多くのステートメントには演算が含まれています。たとえば、次の各 COBOL ステートメントには、ある種の演算評価が必要です。

- 一般演算

```
COMPUTE REPORT-MATRIX-COL = (EMP-COUNT ** .5) + 1
ADD REPORT-MATRIX-MIN TO REPORT-MATRIX-MAX GIVING
REPORT-MATRIX-TOT.
```

- 式および関数

```
COMPUTE REPORT-MATRIX-COL = FUNCTION SQRT(EMP-COUNT) + 1
COMPUTE CURRENT-DAY = FUNCTION DAY-OF-INTEGER(NUMBER-OF-DAYS + 1)
```

- 算術比較

```
IF REPORT-MATRIX-COL < FUNCTION SQRT(EMP-COUNT) + 1
IF CURRENT-DAY not = FUNCTION DAY-OF-INTEGER(NUMBER-OF-DAYS + 1)
```

  ステートメント、組み込み関数、式、互いにネストされたそれらの組み合わせなど、いずれであれ、プログラムでの演算評価ごとに、演算のコーディング方法によって浮動小数点として評価するか、固定小数点として評価するかが決まります。

以下に、算術演算と算術比較が、どのような固定小数点および浮動小数点として評価されるかを説明します。演算評価の精度に関する詳細は 187 ページの『変換と精度』を参照してください。

浮動小数点評価

一般に、演算評価が次のリストのうちのいずれかの特性に該当する場合、それは、浮動小数点演算においてコンパイラーによって評価されます。

- オペランドまたは結果のフィールドが浮動小数点の場合

  データ項目を浮動小数点リテラルとしてコーディングした場合、あるいは USAGE COMP-1、USAGE COMP-2 または外部浮動小数点 (浮動小数点 PICTURE を指定した USAGE DISPLAY) としてそのデータ項目を定義する場合、そのデータ項目は浮動小数点です。

  ネストされた演算式であるか数値組み込み関数への参照であるオペランドは、次の場合に浮動小数点になります。

  – 演算式の引き数は浮動小数点になります。
  – 関数は浮動小数点関数です。
  – 関数は、1 つまたは複数の浮動小数点引き数を持つ混合関数です。

- 浮動小数点関数の引き数である場合

  COS や SIN などの関数は、1 つの引き数を取る浮動小数点関数です。これらの関数は浮動小数点関数なので、引き数は浮動小数点で計算されます。

固定小数点の評価

一般に、演算命令に上記のような浮動小数点の特性が該当しない場合、その演算命令はコンパイラーによって固定小数点演算として評価されます。すなわち、コンパイラーが演算の評価結果を固定小数点として処理するには、すべてのオペランドが

固定小数点であって、しかも結果のフィールドが固定小数点として定義されている場合だけです。 ネストされた演算式および関数参照は、固定小数点値を表すものでなければなりません。

算術比較 (関係条件)

演算が比較 (比較演算子を含んでいるもの) の場合、比較する数値式は、データ項目、演算式、関数参照、あるいはそれらの組み合わせのいずれであれ、評価全体のコンテキストの中での実際のオペランド (被比較項目) になります。これは省略比較にも当てはまります。 1 つの被比較項目は明示的に示されない場合がありますが、両方とも比較のオペランドです。 ILE COBOL で比較を含む式を使用する際に、被比較項目の 1 つが浮動小数点であるか、または浮動小数点に解決されるものである場合には、その式は浮動小数点として評価されます。そうでない場合は、固定小数点として計算されます。

たとえば、次の例を考えてみます。

```
IF (A + B) = C or D = (E + F)
```

ここには 2 つの比較があるため、被比較項目は 4 つになります。 4 つの被比較項目のうちのいずれかが浮動小数点値または浮動小数点値に解決されるものなら、この IF ステートメントは浮動小数点で実行されます。そうでない場合は、すべての演算が固定小数点で実行されます。

次の EVALUATE ステートメントの場合、

```
EVALUATE (A + D)
  WHEN (B + E) THRU C
  WHEN (F / G) THRU (H * I)
   .
   .
   .
END-EVALUATE.
```

EVALUATE ステートメントはそれに相当する 1 個の IF ステートメント、または一連の IF ステートメントに書き換え可能です。この例では、相当する IF ステートメントは次のようになります。

```
if ( (A + D) >= (B + E) ) AND
   ( (A + D) <= C)
if ( (A + D) >= (F / G) ) AND
   ( (A + D) <= (H * I) )
```

このように、上記の IF ステートメントの規則に従って各 IF ステートメントの被比較項目を調べ、その IF ステートメントのすべての演算が固定小数点か浮動小数点かを判別する必要があります。

固定小数点評価と浮動小数点評価の例

207 ページの『固定小数点演算と浮動小数点演算』に示されている例において、次の方法でデータ項目を定義すると、

```
01  EMPLOYEE-TABLE.
    05  EMP-COUNT           PIC  9(4).
    05  EMPLOYEE-RECORD OCCURS 1 TO 1000 TIMES
                        DEPENDING ON EMP-COUNT.
        10 HOURS             PIC +9(5)E+99.
.
```

```
.
.
01  REPORT-MATRIX-COL       PIC  9(3).
01  REPORT-MATRIX-MIN       PIC  9(3).
01  REPORT-MATRIX-MAX       PIC  9(3).
01  REPORT-MATRIX-TOT       PIC  9(3).
01  CURRENT-DAY             PIC  9(7).
01  NUMBER-OF-DAYS          PIC  9(3).
```

- 次の評価は浮動小数点演算で実行されます。

```
COMPUTE REPORT-MATRIX-COL = FUNCTION SQRT(EMP-COUNT) + 1
IF REPORT-MATRIX-TOT < FUNCTION SQRT(EMP-COUNT) + 1
```

- 次の評価は固定小数点演算で実行されます。

```
 ADD REPORT-MATRIX-MIN TO REPORT-MATRIX-MAX GIVING REPORT-MATRIX-TOT.
 IF CURRENT-DAY NOT = FUNCTION DAY-OF-INTEGER((NUMBER-OF-DAYS) + 1)
COMPUTE REPORT-MATRIX-MAX =
FUNCTION MAX(REPORT-MATRIX-MAX REPORT-MATRIX-TOT)
```

テーブル項目の処理

テーブル項目のデータ記述が組み込み関数の引き数要件と互換性のある限り、そのような組み込み関数を使用して英数字テーブル項目や数値テーブル項目を処理できます。各種の組み込み関数の引き数に必要なデータ形式については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」で説明してあります。

関数引き数として個々のデータ項目を参照するには、添え字または指標を使用します。 TABLE-ONE を数字項目の 3 × 3 の配列とすると、次のステートメントで中間エレメントの平方根を求めることができます。

```
COMPUTE X = FUNCTION SQRT(TABLE-ONE(2,2)).
```

複数テーブル項目の処理 (ALL 添え字)

テーブルのデータを繰り返し処理する必要のある場合がよくあります。複数の引き数を受け取る組み込み関数の場合、テーブル内のすべての項目やテーブルの単一ディメンションを参照するために、 ALL 添え字を使用できます。反復は自動的に処理されるため、コードは短く、単純になります。

例 1:

この例は、Table-Two のクロスセクションを合計します。

```
Compute Table-Sum = FUNCTION SUM (Table-Two(ALL, 3, ALL)))
```

テーブル 2 が 2x3x2 の配列とすると、上記のステートメントによって、以下の要素が合計されます。

```
Table-Two(1,3,1)
Table-Two(1,3,2)
Table-Two(2,3,1)
Table-Two(2,3,2)
```

例 2:

この例は、すべての従業員の値を計算します。

```
01 Employee-Table.
05 Emp-Count Pic s9(4) usage binary.
05 Emp-Record occurs 1 to 500 times
depending on Emp-Count.
10 Emp-Name Pic x(20).
```

```
10 Emp-Idme Pic 9(9).
10 Emp-Salary Pic 9(7)v99.
.
.
Procedure Division.
Compute Max-Salary = Function Max(Emp-Salary(ALL))
Compute I = Function Ord-Max(Emp-Salary(ALL))
Compute Avg-Salary = Function Mean(Emp-Salary(ALL))
Compute Salary-Range = Function Range(Emp-Salary(ALL))
Compute Total-Payroll = Function Sum(Emp-Salary(ALL))
```

例 3:

複数の引き数を受け取る関数では、スカラーと配列の引き数を混合できます。

```
Compute Table-Median = Function Median(Arg1 Table-One(ALL))
```

西暦 2000 年問題について

西暦 2000 年問題は、2 桁で年を表すことに関係するものです。プログラムの日付フィールドが年の末尾 2 桁だけの場合、2000/1/1 日には、現在年が 00 年として表示されることになります。 00 は 99 よりも小さいので、その年がその前の年よりも小さいことになってしまいます。

21 世紀の世紀サポートが ILE COBOL に追加されました。 これは、40〜99 までの範囲の末尾 2 桁で年を検索すると「19」が接頭部として追加され、1940 から 1999 までの範囲の 4 桁年として検索されることになります。これに対して、00〜39 までの範囲の 2 桁年が検索されると「20」が接頭部に追加され、2000〜2039 までの範囲の 4 桁年として検索されることになります。

長期的解決策

プログラムで 9999 年までを使用するには、結局は次のようにする必要があります。

1. 次のいずれかの方法で、2 桁年の代わりに 4 桁年が検索されるようアプリケーションを変更します。
   - ACCEPT ステートメントの新しい YYYYMMDD 句および YYYYDDD 句を使用して、4 桁年を取得する。または
   - CURRENT-DATE、DATE-OF-INTEGER、および DAY-OF-INTEGER などの組み込み関数を使用して、4 桁年日付を取得する。または
   - Integrated Language Environment呼び出し可能サービスを使用して、4 桁年日付を取得する。
2. 4 桁年を格納できるように日付を含むデータ項目のサイズを増やすか、あるいは、データ項目を、4 桁年を保持する日付データ項目に変更します。
3. 4 桁年を使用してデータベースを再構築します。

しかし、より簡単な短期的解決策もあります。

短期的解決策

アプリケーションおよびデータのすべてを 2000 年までに変更できない場合には、データだけを残しておき、2 桁年を 4 桁年として解釈するようアプリケーションを変更することができます。このタイプの手法を、通常、**ウィンドウ操作**といいま

す。この手法により、2 桁年を採用しながら、事前定義済みの 100 年ウィンドウに基づいて 4 桁年を判別できます。たとえば、1940 ～ 2039 というウィンドウを考えてみます。

- 92 という 2 桁年は 1992 年になります。
- 02 という 2 桁年は 2002 年になります。

ILE COBOL でウィンドウ操作を行う方法は 2 つあります。 ILE COBOL 組み込み関数を使用することによってユーザー自身でウィンドウ操作を実行することもできますし、あるいは数値日付または文字日付を日付データ項目に変更することによって ILE COBOL にウィンドウ操作を実行させることもできます。

ユーザー自身でウィンドウ操作を行いたい場合、ILE COBOL は世紀ウィンドウ組み込み関数の集まりを提供します。これにより、2 桁年は 1 つの 100 年ウィンドウの中で解釈されます (どの 2 桁数値も 100 年に一度しか現れないからです)。期間を選択し、組み込み関数に 2 桁年、または 2 桁年を使用した日付または日数を指定すると、組み込み関数はその 100 年ウィンドウの範囲内の 4 桁年の該当する値を戻します。

ILE COBOL コンパイラーは、YEAR-TO-YYYY、DAY-TO-YYYYDDD、DATE-TO-YYYYMMDD という 3 つの世紀ウィンドウ組み込み関数を提供しています。 YEAR-TO-YYYY 組み込み関数は 2 桁年で、指定した 100 年ウィンドウの 4 桁年を戻します。他の 2 つの組み込み関数は 2 桁年を含んだ日付を引き数として取り、指定した 100 年ウィンドウの 4 桁年を含んだ日付を戻します。DAY-TO-YYYYDDD 組み込み関数の場合、その日付として ACCEPT FROM DAY ステートメントの日付形式と同じ 5 桁数値を指定します。同様に、DATE-TO-YYYYMMDD 組み込み関数には、ACCEPT FROM DATE ステートメントの日付形式と同じ 6 桁数値を指定します。

世紀ウィンドウ組み込み関数については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

ILE COBOL にウィンドウ操作を実行させるには、文字または数値での日付を日付データ項目に変更する必要があります。以下に示す一部分のコーディングには、日付を表す数字項目が 2 つあります。このコードは、現在日付が満了日を過ぎるとメッセージを表示します。

```
  01  my-dates.
* expiration-date is year 1997, month 10, day 9
      05  expiration-date  PIC S9(6)  VALUE 971009
              USAGE PACKED-DECIMAL.
* current-date-1 is year 2002, month 8, day 5
      05  current-date-1   PIC S9(6)  VALUE 020805
              USAGE PACKED-DECIMAL.
     IF current-date-1 > expiration-date THEN
       DISPLAY "items date is past expiration date"
     END-IF.
```

上記のコードでは、2002 が 1997 より大きいにもかかわらず、数値 020805 は 971009 より大きくないため、IF の評価は FALSE となり、DISPLAY ステートメントは実行されません。しかし、数値日付を日付データ項目に変更すると、DISPLAY ステートメントは実行されます。 `expiration-date` と `current-date-1` は両方ともサイズ (バイト単位) が変わっていないことに注意してください。

```
  01  my-dates.
* expiration-date is year 1997, month 10, day 9
     05  expiration-date  FORMAT DATE "%y%m%d" VALUE "971009"
              USAGE PACKED-DECIMAL.
* current-date-1 is year 2002, month 8, day 5
     05  current-date-1   FORMAT DATE "%y%m%d" VALUE "020805"
              USAGE PACKED-DECIMAL.
     IF current-date-1 > expiration-date THEN
       DISPLAY "items date is past expiration date"
     END-IF.
```

短期的解決策の利点

短期的解決策の利点は、変更を要するプログラムが少なくてすみ、データベースは変更しなくてもよい点にあります。この方法のほうが、長期的解決策よりも安上がりで、手早く、簡単です。

しかし、世紀ウィンドウ組み込み関数を使用して、データベースやファイルを 2 桁年日付から 4 桁年日付に変換することもできます。これを行うには、2 桁年日付を読み取って 4 桁年に変換し、オリジナルのコピーを 4 桁年日付用に拡張して、そこにデータを再び書き込みます。こうして、すべての新データを新ファイルまたは新データベースに入れます。

短期的解決策の欠点

この方法では、アプリケーションによってはほんの数年しかもちません。そうなると、やはりすべての日付プログラムおよびデータベースを変更する必要が生じます。

2 桁年が固有であるのは特定の 100 年間だけなので、世紀ウィンドウは永久に使用できるわけではありません。いずれ、100 年を超えるデータ・ウィンドウが必要になります。実際に現在それを必要としている企業は少なくありません。

世紀ウィンドウが役立つのは、ILE COBOL コードの特定のセクションに関して、ある日付が古い (過去のもの) かどうか、あるいは期限がまだ来ていない (未来のもの) かどうかといったことを識別するかどうかがわかっているときです。その場合、その知識を使用して世紀ウィンドウの設定方法を決めることができます。

しかし限界があります。たとえば、世紀ウィンドウでは、タイム・スパンが 100 年を超えるような場合に、得意先がどれほどの間、自分の会社と取引しているのか割り出すことはできず、年号の下 2 桁しか知ることができません。もう一つの例はソートです。日付でソートするレコードは、みな 4 桁年日付でなければなりません。これらの問題やその他の問題を考慮するなら、4 桁年を戻す ACCEPT ステートメント、組み込み関数、または ILE 日付サービスを使用する必要があります。

日時データ・タイプを処理する

COBOL 日時クラスの項目には、日付、時刻、およびタイム・スタンプ項目が含まれます。これらの項目は、データ記述項目の FORMAT 文節で宣言されます。たとえば、

```
01 group-item.
   05 date1 FORMAT DATE "%m/%d/@Y".
   05 date2 FORMAT DATE.
```

```
05 time1 FORMAT TIME SIZE 8 LOCALE german-locale.
05 time2 FORMAT TIME "%H:%M:%S".
05 time3 FORMAT TIME.
05 timestamp1 FORMAT TIMESTAMP.
```

日時クラスの項目の場合、FORMAT 文節は PICTURE 文節の代わりに使用されます。上の例では、キーワード FORMAT の後に、語 DATE、TIME、または TIMESTAMP がきます。これらの語は、日時項目のカテゴリーを識別します。

注: 語 DATE および TIME は予約語です。ただし、語 TIMESTAMP はコンテキストに依存した語です。

日時項目のカテゴリーを示す予約語またはコンテキストに依存した語の後には、**フォーマット設定リテラル**を入れることができます。フォーマット設定リテラルとは、日付項目または時刻項目の形式を記述する非数値リテラルです。

データ項目 date1 の場合、%m は月、%d は日、また @Y は年 (2 桁の世紀を含む) を表します。文字 % および @ は指定子の先頭にきます。 date1 というフォーマット設定リテラルで使用される 3 つの指定子は、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」に記載されている指定子の集まりの一部です。フォーマット設定リテラルは、指定子と区切り文字を組み合わせたものです。そこで、もう一度 date1 を見てみると、区切り文字が 2 つあり、両方とも文字 / です。

上の例では、各指定子は事前決定されたサイズをもっています。たとえば、データ項目 time2 は、%H、%M、および %S という 3 つの指定子をもっています。それらは、時 (2 つの数字)、分 (2 つの数字)、および秒 (2 つの数字) です。このほかに、区切り記号が 2 つあり、どちらも文字 : です。したがって、time2 の合計サイズは 8 文字です。

フォーマット設定リテラルでは、区切り文字と指定子の順序は任意ですが、次の規則に従う必要があります。

- 指定子と区切り文字の合計サイズが 256 バイトを超えてはならない。
- 区切り文字は、サイズは任意で、反復できる。
- 各指定子は、1 つのフォーマット設定リテラル内で一度だけ使用できる。
- 指定子は、特定の日時カテゴリーに限定される。たとえば、指定子 %H (時) を日付項目に使用することはできません。
- 指定子がオーバーラップしてはならない。たとえば、2 桁の世紀をもつ年 @Y と一緒に @C (1 桁の世紀) を指定することはできません。

上の例では、date2 と timestamp1 のどちらもフォーマット設定リテラルが指定されていません。タイム・スタンプのカテゴリーの項目は、ユーザー定義フォーマット設定リテラルをもつことができません。ただし、@Y-%m-%d-%H.%M.%S.@Sm というデフォルトのフォーマット設定リテラルをもっています。日付または時刻のカテゴリーの項目の場合、フォーマット設定リテラルがデータ記述項目に明示的に指定されていなければ、SPECIAL-NAMES 段落に指定することができます。以下に、その例を示します。

```
SPECIAL-NAMES.  FORMAT OF DATE IS "@C:%y:%j",
                FORMAT OF TIME IS "%H:%M:%S:@Sm".
```

上の SPECIAL-NAMES 段落がグループ項目 group-item と同じプログラムに指定されている場合には、date2 の日付形式は @C:%y:%j です。しかし、SPECIAL-NAMES 段落が存在しなかった場合には、日付項目の形式はデフォルトとして ISO がとられます。 ISO 日付の形式は @Y-%m-%d です。フォーマット設定リテラルのないカテゴリー時刻の唯一の項目 (暗黙的または明示的に定義されているもの) は time3 であるため、上の SPECIAL-NAMES 段落が存在しなければ、time3 は時刻形式 %H:%M:%S:@Sm をもつことになります。ただし、FORMAT OF TIME 文節が SPECIAL-NAMES 段落にある場合には、形式はデフォルトとして ISO がとられます。 ISO 時刻の形式は %H.%M.%S です。

デフォルトでは、COPY DDS は日時クラスの項目を宣言すると、英数字項目について PICTURE 文節を生成します。 PICTURE 文節を FORMAT 文節に変更するために、いくつかの新しい CVTOPT パラメーター値が定義されました。それらは、次のものです。

- *DATE
- *TIME
- *TIMESTAMP

*DATE が指定されていれば、DDS 日付データ・タイプはすべて COBOL 日付項目に変換されます。たとえば、PICTURE 文節ではなく、FORMAT 文節が生成されます。

DDS では、日付フィールドの形式を指定するために、DATFMT キーワードを指定することができます。 DATFMT キーワードは、ゾーン、パック、および文字の各フィールドに対しても指定できます。これらのタイプのフィールドでは、COPY DDS は、通常、数字ゾーン、数字パック、および英数字の各データ項目について、それぞれ PICTURE 文節を生成します。 CVTOPT パラメーターの *CVTTODATE 値を指定すると、これらの項目について COPY DDS に強制的に FORMAT 文節を生成させることができます。

ゾーン、パック、および文字の各 DDS フィールドに許されている DATFMT パラメーターと、CVTOPT(*CVTTODATE) 変換パラメーターが指定されたときに DDS から生成される、それぞれに相当する ILE COBOL 形式のリストについては 492 ページの『日時クラス』を参照してください。

上に述べたとおり、データ記述項目の FORMAT 文節は、日時クラスの項目を定義します。このデータ記述項目には、以下に示す文節のうち 1 つまたは複数の文節を含めることもできます。

- OCCURS
- REDEFINES
- RENAMES
- SYNCHRONIZED
- TYPEDEF
- USAGE
- VALUE

この同じデータ記述項目が、1 つまたは複数の 88 (条件名) をそれと関連付けることができます。条件名の VALUE 文節には THRU 句を含めることができます。以下に示す文節は、日時クラスのデータ記述項目を参照できます。

- LIKE

- REDEFINES
- RENAMES
- TYPE

次のコーディングの部分は、日時クラスの項目の各種定義を示しています。

```
01  TimestampT IS TYPEDEF
        FORMAT TIMESTAMP VALUE "1997-12-01-05.52.50.000000".
01 group-item.
    05  date1 FORMAT DATE OCCURS 3 TIMES VALUE "1997-05-08".
    05  date2 FORMAT DATE "@Y-%m-%d" VALUE "2001-09-08".
    05  date3 REDEFINES date2 FORMAT DATE.
       88  date3-key-dates VALUE "1997-05-01" THRU "2002-05-01".
    05  time1 FORMAT TIME "%H:%M" VALUE "14:10".
    05  time2 LIKE time1.
    05  timestamp1 TYPE TimestampT.
```

上の文節はいずれも、日時クラスの項目と一緒に使用する際にはさまざまな規則があります。

SYNCHRONIZED 文節は日時項目に指定できますが、文書化用としてしか扱われません (項目の位置合わせは行いません)。

日時項目の USAGE 文節は日付または時刻項目の DISPLAY または PACKED-DECIMAL でかまいませんが、タイム・スタンプは USAGE DISPLAY だけです。日時項目に PACKED-DECIMAL という USAGE がある場合には、フォーマット設定リテラルには指定子しか入れられず (区切り文字は不可)、しかもその指定子は結果として数字を生じるものでなければなりません。

日時項目の VALUE 文節は、日時項目の形式の非数値リテラルにします。コンパイル時に、VALUE 文節の非数値リテラルの形式が FORMAT 文節に一致しているかどうかを確かめる検査は行われません。 VALUE 文節の非数値リテラルが正しいものであるようにするのは、プログラマーが行う必要があります。

日時項目と関連付けられたレベル 88 (条件名) は THRU 句をもつことができます。日時項目と関連付けられたレベル 88 項目の VALUE 文節には、親項目と一致する形式の非数値リテラルを含めてください。レベル 88 項目が比較条件で使用されると、結果として日時比較が発生します。

日時項目を参照する LIKE 文節は、そのサイズを変更できません。 LIKE 文節により、新しい項目は、SIZE 文節および LOCALE 文節を含め、FORMAT 文節の属性をすべて継承します。

日時データ項目は、次に示すステートメント、文節、および組み込み関数と一緒に使用できます。

- MOVE
- 次のものにおける暗黙的な移動:
  - READ INTO
  - WRITE FROM
  - REWRITE FROM
  - RETURN INTO
  - RELEASE FROM
- 比較条件

- ACCEPT (形式 2)
- DISPLAY (拡張 DISPLAY を除くすべての形式)
- CALL USING および CALL GIVING
- 手続き部 USING および GIVING
- OCCURS 文節のキーとして
- SORT/MERGE のキーとして
- RECORD KEY 文節
- ADDRESS OF、LENGTH OF、FORMAT OF、LOCALE OF を使用する式
- 次の組み込み関数:
  - ADD-DURATION
  - CONVERT-DATE-TIME
  - EXTRACT-DATE-TIME
  - FIND-DURATION
  - SUBTRACT-DURATION
  - TEST-DATE-TIME
  - LENGTH

日時データ・タイプは SORT (および MERGE) 操作でも使用できますが、いくつかの制限が適用されます。これらの制限については 483 ページの『日時データ・タイプに関する考慮事項』を参照してください。

日時データ項目の MOVE に関する考慮事項

ここでは、MOVE ステートメント、および暗黙的な移動 (READ INTO、WRITE FROM、REWRITE FROM、RETURN INTO、および RELEASE FROM) のあるステートメント、および比較条件で日時データ項目を使用するための考慮事項のいくつかを説明します。

- @p (am または pm) の大文字 (AM または PM) への変換
- 2 桁年の 4 桁年または世紀への変換
- DATTIM 処理ステートメントのオプションを使用したデフォルト日付ウィンドウの指定変更
- マイクロ秒への時間の変換
- 時間帯

@p を大文字に変換する

時刻項目は、@p 変換指定子で定義できます。この指定子は、AM または PM のいずれかで置き換えることができます。ただし、AM および PM は、大文字と小文字を任意に組み合わせたものでかまいません。これは、@p 変換指定子をもつ変換前と受け取り側の両方を含むステートメントでは、変換前には大文字と小文字を混在させることができるが、受け取り側では、必ず、大文字に変換されるということを意味します。たとえば、

```
01 group-item.
   05 time1 FORMAT TIME "%I:%M @p" VALUE "06:20 am".
   05 time2 LIKE time1.
   MOVE time1 TO time2.
   DISPLAY "time2 = " time2.
```

上のコーディングでは、time1 は MOVE ステートメントの変換前であるため、その @p 指定子は大文字と小文字が混在していて構いません。この例では、小文字の

am です。受け取り側 time2 は、time1 と同じ形式をもっているため、@p は大文字で含まれます。したがって、上のプログラムの出力は、次のようになります。

```
time2 = 06:20 AM
```

2 桁年の 4 桁年または世紀への変換

2 桁年を 4 桁年または世紀に移動または比較する場合、あるいは 4 桁年または世紀を 2 桁年に比較する場合に、ILE COBOL は、まず最初に、ウィンドウ操作アルゴリズムを使用して 2 桁年を変換します。使用されるデフォルト・ウィンドウ操作アルゴリズムは、次のとおりです。

- 2 桁年が 4 桁年に移動された場合、世紀 (年の最初の 2 桁) は次のように選ばれます。
  - 2 桁年が 40 以上であれば、使用される世紀は 1900 です。すなわち、19 が 4 桁年の最初の 2 桁になります。
  - 2 桁年が 40 未満であれば、使用される世紀は 2000 です。すなわち、20 が 4 桁年の最初の 2 桁になります。
- 4 桁年または世紀をもつデータ項目が 2 桁年に移動された場合、年 (世紀) の最初の 2 桁は切り捨てられます。後に、日付が変更され、その 2 桁年が 4 桁年に移動された場合には、2 桁年と 4 桁年との間の移動について上記のアルゴリズムが使用されるため、結果が不正確になる可能性があります。プログラマーは、これらのタイプの移動が行われた場合に結果が不正確にならないようにする必要があります。すなわち、結果が不正確になる可能性がある場合には、移動は 2 桁年同士および 4 桁年同士だけで行ってください。

注: 日付を含む英数字データ項目が日時データ項目に移動される際には、検査や変換は行われません。プログラマーは、移動される英数字日付が正しい形式になるようにする必要があります。

この変換がどのように行われるかを示すために、次のプログラムでは 3 つの日付移動を行います。

```
ID DIVISION.
PROGRAM-ID. datmoves.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
     FORMAT DATE IS '%m/%d/%y'.
DATA DIVISION.
WORKING-STORAGE SECTION.
77  date1  format date Value '07/12/39'.
77  date2  format date '@Y/%m/%d'.
77  date3  format date '%y/%m/%d'.
01  ALPHA_USA_S    PIC X(08).
PROCEDURE DIVISION.
PARA1.
     move date1 to date2. 1
     display "date2 =" date2.
*
     move date2 to date3. 2
     display "date3 =" date3.
*
     move FUNCTION ADD-DURATION (date3 YEARS 1) to date2. 3
     display "date2 =" date2.
```

このプログラムの出力は、次のとおりです。

```
date2 =2039/07/12
date3 =39/07/12
date2 =1940/07/12
```

move **1** では、date1 (値 07/12/39 を含むもの) が date2 に移動されます。date1 には 40 未満の 2 桁年が含まれており、それが、4 桁年をもつ date2 に移動されるため、使用される世紀は 2000 で、4 桁年は 2039 となります。

move **2** では、4 桁年が 2 桁年に移動され、世紀 (年の最初の 2 桁) は切り捨てられるだけです。

move **3** では、date3 に 1 年加算されてから、それが元どおり 4 桁年に移動されます。加算された年により、日付の 2 桁年は 21 世紀の範囲外に移動されるため、日付は 20 世紀の日付になり、正確ではありません。この移動は、ウィンドウ操作アルゴリズムがどのように働くか、また、4 桁年形式と 2 桁年形式との間で日付を移動するとどのように結果が不正確になるかを示しています。

DATTIM 処理ステートメントのオプションを使用して、デフォルト日付ウィンドウを指定変更する: 4 桁と 2 桁の年号間でデータを移動する必要があり、移動する時に、ILE COBOL が使用するデフォルトのウィンドウ操作アルゴリズムに基づくと、間違った結果になることが明らかである場合があります。 DATTIM 処理ステートメントのオプションを使用すると、デフォルト日付ウィンドウを変更できます。

DATTIM 処理ステートメントの構文は、次のとおりです。

構文

4 桁の基本世紀
これは最初の引き数でなければなりません。 ILE COBOL がそのウィンドウ操作のアルゴリズムで使用する基本世紀を定義します。 DATTIM 処理ステートメント・オプションが指定されていない場合は、1900 が使用されます。

「4 桁の基本世紀」は、@C 変換指定子の変換処理にも影響します。 @C 変換指定子は 1 桁の世紀を表すもので、その範囲は 0 から 9 までです。 1 桁世紀の 0 は 1900 という基本世紀を表し、1 = 2000、... 9 = 2800 という具合です。したがって、形式が @C/%y/%m で、しかも値が 1/12/05 である日付データ項目は、2012 年の 5 番目の月 (5 月) の最初の日を表します。ただし、@C が 0 であると、実際に 4 桁の基本世紀に等しくなります。したがって、DATTIM(2200, 40) は、0 = 2200、1 = 2300 ...、9 = 3100 という具合になります。

2 桁の基本年
これは 2 番目の引き数でなければなりません。 ILE COBOL がそのウィンドウ操作のアルゴリズムで使用する基本年を定義します。 DATTIM 処理ステートメント・オプションが指定されていない場合は、40 が使用されます。

デフォルトの DATTIM(1900, 40) では、1940 から 2039 までの 100 年ウィンドウになります。先の例で見つかった問題を解決するためには、以下のいずれかの方法で DATTIM オプションを設定できます。

- DATTIM(1900 70) を指定すると、1970 から 2069 までの 100 年ウィンドウになります。
- すべての 2 桁年が 21 世紀にあると想定すると、DATTIM(2000 00) を指定することができ、この場合は 2000 から 2099 までの 100 年ウィンドウになります。

先の例の PROCESS ステートメントにこれらのオプションのいずれかを指定すると、出力は次のようになります。

```
date2 =2039/07/12
date3 =39/07/12
date2 =2040/07/12
```

出力で変ったのは、move **3** の結果だけです。先の例の出力は `date2 =1940/07/12` で、これは、デフォルト ILE COBOL ウィンドウ操作アルゴリズム (1900 という基本世紀と 40 という基本年) に基づいて更新済み 2 桁日付を 4 桁日付に移動した結果であり、不正確なものでした。

日時比較条件のパフォーマンスに関する考慮事項: 日時項目が ILE COBOL の別の日時項目に比較される際には、まず最初に、すべての区切り文字が取り払われて、ISO 形式 (区切り文字なし) に変換されます。次に、2 つの日時項目が、数字ごとに比較されます。日付同士の比較の場合は、日付が両方とも `@Y%m%d` に変換されてから、比較されます。時間は、`8/1/07M%S` に変換されてから、比較されます。これは、日時比較で最良のパフォーマンスを得るためには、日時項目を ISO 形式にする必要があるということを意味しています。

日時移動のパフォーマンスに関する考慮事項: 日時項目の形式は、日時移動が完了するのに要する時間の長さに影響します。基本的な形式を、パフォーマンスの良いものから順に 3 つ挙げると、次のようになります。

1. 日時項目。この形式は、i5/OS DDS がサポートしている日付、時刻、またはタイム・スタンプ形式と同じです。 このグループの中では、一般的に、ISO 形式のパフォーマンスが最高となります。
2. 非ロケール日時項目。この形式は 1 には含まれません。
3. ロケールに基づく日時項目。

マイクロ秒への時間の変換

時刻データ項目には、1 秒未満の秒数を含む指定子を 1 つまたは複数個含めることができます。 1 秒未満の秒数を保持する 4 つの変換指定子は、次のものです。

@Sh 1 秒の 100 分の 1

@Sm 1 秒の 100 万分の 1 (マイクロ秒)

@So 1 秒の 1000 分の 1 (ミリ秒)

@St 1 秒の 10 分の 1

たとえば、時刻データ項目の形式が `%H:%M:%S @So/@Sm` の場合があります。

時刻データ項目が 2 つあり、一方をもう一方に移動した場合、送り側のデータ項目の 1 秒未満の秒数を保持する指定子はすべて、マイクロ秒数を表す 1 つの数値に変換されます。マイクロ秒は、次に、受け取り側の時刻データ項目の適切な 1 秒未満の秒数に変換されます。

時間帯: 時刻およびタイム・スタンプのカテゴリーのデータ項目には時間帯の影響があります。 ILE COBOL は、System i 値 QUTCOFFSET (協定世界時オフセット (グリニッジ標準時オフセットとも呼ばれる)) および LC_TOD ロケール から時間帯情報を検索します。 ロケールと関連付けられた時刻データ項目は、LC_TOD ロケール・カテゴリーのキーワード tzdiff で時間帯を使用します。ロケールと関連付けられていない時刻データ項目およびタイム・スタンプ・データ項目は、System i の時間帯、つまり、QUTCOFFSET システム値で指定した時間帯にあるものと見なされます。

したがって、たとえば、次の場合を考えます。

```
SPECIAL-NAMES.
     LOCALE "EN_US" IN LIBRARY "QSYSLOCALE" IS USA.

⋮
01 GROUP-ITEM.
   05 SYSTEM-TIME-1 FORMAT TIME "%H:%M:%S" VALUE "14:32:10".
   05 LOCALE-TIME-1 FORMAT TIME SIZE 8 IS LOCALE USA.

⋮
   MOVE SYSTEM-TIME-1 TO LOCALE-TIME-1.
```

i5/OS に付属している EN_US のロケール・ソースには 0 というデフォルト tzdiff 値が備わっています。しかし、これは、そのロケール・ソースを別のソース物理ファイルにコピーすることにより、ユーザーが変更できます。 上記の MOVE ステートメントでは、データ項目 SYSTEM-TIME-1 はどのロケールとも関連付けられていないため、その時間帯は QUTCOFFSET システム値により示されます。データ項目 LOCALE-TIME-1 はライブラリー QSYSLOCALE 内のロケール EN_US と関連付けられています。これは、その時間帯が、このロケールの LC_TOD ロケール・カテゴリーで示されることを意味しています。 MOVE ステートメントの実行中、LOCALE-TIME-1 の結果として生じる時刻は、SYSTEM-TIME-1 と LOCALE-TIME-1 との間の時間帯の差で調整されます。

ILE COBOL では、夏時間を考慮に入れません。これを適応させるには、LC_TOD ロケール・カテゴリーおよび QUTCOFFSET の協定世界時オフセットを更新して、この時間差を調整する必要があります。

時間帯を考慮に入れる他の組み込み関数は、FIND-DURATION、LOCALE-TIME、および CONVERT-DATE-TIME です。

ロケールの処理

ロケールは、特定の文化圏固有のフォーマット設定情報を識別します。この情報は、特定の文化圏について、有効な英字、照合順序、数字形式と通貨数量、および日付と時刻の形式を記述します。

ロケール情報は、プログラムの実行時の特定の局面を制御するロケール・カテゴリーにグループ分けされます。これらのロケール・カテゴリーは、次のものです。

ロケール・カテゴリー名	影響される動作
LC_CTYPE	大文字、小文字、スペース、数字、句読点といった

文字タイプを定義します。ロケール・ベースの数字編集、日付、および時刻項目のほか、組み込み関数の動作に影響します。

LC_COLLATE 照合順序を定義します。

LC_TIME 使用されるカレンダー、時間帯、および曜日といった日付と時刻の規則を定義します。ロケールに基づく形式をもつ日付と時刻のデータ項目の動作、および日付と時刻の項目を戻す組み込み関数の動作に影響します。

LC_NUMERIC 数字形式を定義します。

LC_MONETARY 通貨名、記号、句読点、その他の詳細を定義します。ロケールに基づく数字編集項目に影響します。

LC_MESSAGES 通知メッセージと診断メッセージ、および対話式応答の形式を定義します。

LC_TOD 時間帯の差、時間帯名、および夏時間の始まりと終わり (i5/OS 固有) を定義します。 また、ロケールに基づく時刻データ項目、時刻項目を戻す組み込み関数、およびロケールに基づいて時刻をフォーマット設定する組み込み関数の動作にも影響します。

LC_ALL このリストで以前に定義されたものすべてを含む、すべてのロケール・カテゴリー。このカテゴリーには、ILE COBOL で使用しないカテゴリーや文化圏固有のエレメントも含まれます。

ロケール・カテゴリー LC_MESSAGES、LC_COLLATE、および LC_NUMERIC を ILE COBOL が直接使用することはありません。しかし、これらのカテゴリーを、アプリケーションが使用できるように、SET で設定して照会することができます。

i5/OS でのロケールの作成

i5/OS システム上では、*LOCALE オブジェクトは、ロケールの定義ソース、および
ロケールの文字セットの文字をそれぞれの 16 進値にマップするために使用される
CCSID を含むファイル・メンバーの名前を指定して、CRTLOCALE コマンドで作
成されます。

ロケール定義ソース・メンバーには、言語環境に関する情報が含まれます。この情報は、前のセクションで説明した多数の異なるカテゴリーに分けられます。 1 つのロケール定義ソース・メンバーが 1 つの言語環境の特性を表します。

文字は、ロケール定義ソース・メンバー内ではそれぞれのシンボル名で表されます。シンボル名、それらが表す文字、およびそれぞれの関連付けられた 16 進値間のマッピングは、CRTLOCALE コマンドに指定された CCSID 値に基づいて行われます。ロケール定義ソース・メンバーは、ライブラリー QSYSLOCALE 内の i5/OS システム上にあります。

タイプ *LOCALE のロケールの作成方法については 19 ページの『コード化文字セット ID の使用』を参照してください。

アプリケーション用の現行ロケールの設定

AS/400 上で実行し、タイプ *LOCALE のロケールを使用している ILE COBOL アプリケーションはすべて、そのプログラムの活動化グループを有効範囲とする現行ロケールをもっています。現行ロケールは、ロケールに基づく数字編集データ項目、ロケールに基づく日付と時刻のデータ項目、およびロケール組み込み関数の動作を決定します。ただし、それらがロケール簡略名を指定することはありません。現行ロケールは、SET LOCALE ステートメントを使用して明示的に設定できます。SET LOCALE ステートメントの使用法については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

現行ロケールが SET LOCALE を使用して明示的に設定されない場合には、プログラムを活動化する時に ILE COBOL ランタイムによって暗黙的に設定されます。これは、SET ステートメントの形式 8 にキーワード DEFAULT を使用して設定できるものと同じデフォルト・ロケールです。以下に、プログラムが活動化されるときに ILE COBOL ランタイムが現行ロケールを設定する方法について示します。

- ユーザー・プロファイルに *NONE (デフォルト) または *SYSVAL 以外の LOCALE パラメーターの値がある場合は、その値がアプリケーションの現行ロケールに使用されます。
- ユーザー・プロファイル内の LOCALE パラメーターの値が *NONE である場合は、デフォルトの ILE COBOL ロケールが現行ロケールになります。
- ユーザー・プロファイル内の LOCALE パラメーターの値が *SYSVAL である場合は、システム値 QLOCALE と関連付けられたロケールがプログラムの現行ロケールに使用されます。
- QLOCALE の値が *NONE である場合には、デフォルトの ILE COBOL ロケールが現行ロケールになります。

ILE COBOL が使用する現行ロケールは、ILE C コンパイラーおよび ILE C++ コンパイラーで共用されます。これは、ILE C コンパイラー・プログラムの現行ロケールを変更する ILE C コンパイラーの setlocale 関数が ILE COBOL プログラムの現行ロケールにも影響し、その逆もあることを意味します。

タイプ *LOCALE のロケールを使用可能にする方法については 19 ページの『コード化文字セット ID の使用』を参照してください。

ロケールの識別と有効範囲

ロケールが識別される時点と、識別された後のその影響の有効範囲は、次のとおりです。

- 実行単位が活動化されるときにデフォルト・ロケールが識別され、SET ステートメントによってその実行単位内で変更されるまで、それが現行ロケールになります。ロケールがデフォルトから変更された後で、SET ステートメントの形式 8 でキーワード DEFAULT を使用すると、再度、そのデフォルトを現行ロケールにすることができます。
- LOCALE-DATE および LOCALE-TIME 組み込み関数の場合、現行ロケールは、これらの関数のいずれかを参照する各ステートメントの始めで識別され、そのステートメントの実行中は関数の評価に使用されます。これらの組み込み関数については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

- LOCALE 句を PICTURE 文節または FORMAT 文節で使用し、簡略名-1 を指定しない 場合には、現行ロケールは、データ項目の編集または編集解除を行う各ステートメントの始めで一度識別されます。

  注: 特定のデータ項目の編集と編集解除との間でロケールを切り替えると、予想できない動作が生じる可能性があります。編集解除に使用されるロケールが、編集に使用されるロケールと必ず同じものになるようにしてください。
- LOCALE 句を PICTURE 文節または FORMAT 文節で使用し、簡略名-1 を指定した 場合には、現行ロケールは、SPECIAL-NAMES 段落内の簡略名と関連付けられているものです。このロケールは、ソース単位内でのデータ項目への最初の参照でそれを使用する必要が生じる前に識別しておく必要があります。その有効範囲は、そのソース単位です。
- SET ステートメントの場合、FROM 句に指定されたロケールが、別の SET ステートメントによって再度変更されるまで、実行単位の現行ロケールとなります。

LC_MONETARY ロケール・カテゴリー

ILE COBOL では、LC_MONETARY ロケール・カテゴリーで作成可能な定義に対応する、ロケールに基づいた数字編集データ項目の PICTURE 編集記号のサブセットがあります。これらの記号は、9、.、¥、および cs (通貨記号) です。これらのロケールに基づいた PICTURE 編集記号について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。ここでは、LC_MONETARY ロケール・カテゴリーについて説明し、ILE COBOL のロケールに基づいた PICTURE 編集記号をそれぞれ、使用されるキーワードと関連付けて、このロケール・カテゴリーを定義します。

ロケール定義ソース・ファイルの LC_MONETARY カテゴリーは、通貨数値情報のフォーマット設定を行うための規則と記号を定義します。このカテゴリーは、LC_MONETARY カテゴリー・ヘッダーで始まり、END LC_MONETARY カテゴリー・トレーラーで終わります。

LC_MONETARY カテゴリー・キーワードのオペランドはすべて、ストリング値または整数値として定義されます。ストリング値は、二重引用符 (″″) で結合されます。値はすべて、1 つまたは複数のスペースで、それらが定義するキーワードと区切られます。 2 つの二重引用符が隣接している場合は、未定義ストリング値であることを示します。 -1 は未定義整数値であることを示します。 LC_MONETARY カテゴリーでは、以下に示すキーワードが認識されます。

int_curr_symbol
国際通貨記号に使用されるストリングを指定します。キーワード int_curr_symbol のオペランドは、4 文字のストリングです。最初の 3 文字には、英字の国際通貨記号が含まれます。 4 番目の文字は、国際通貨記号と通貨数量との区切り文字を指定します。ローカルの通貨記号に使用されるストリングを指定します。このキーワードは、ILE COBOL では使用されません。

currency_symbol
ローカルの通貨記号に使用されるストリングを指定します。 ILE COBOL では、このキーワードは、ロケールに基づく PICTURE 編集記号 cs をフォ

ーマット設定するために、いくつかの他のキーワードと一緒に使用します。キーワード p_cs_precedes、p_sep_by_space、n_cs_precedes、および n_sep_by_space を参照してください。

mon_decimal_point
通貨数量をフォーマット設定するのに使用される小数点に使用されるストリングを指定します。 ILE COBOL では、これは、ロケールに基づく PICTURE 編集記号 . に対応します。

mon_thousands_sep
フォーマット設定された通貨数量の小数点の左側にある桁のグループ分けに使用されるストリングを指定します。

mon_grouping
フォーマット設定された通貨数量の各グループの桁数を定義します。キーワード mon_grouping のオペランドは、セミコロンで区切られた一連の整数で構成されます。各整数は、1 つのグループ内の桁数を指定します。冒頭の整数は、小数点のすぐ左にあるグループの桁数を定義します。その後の整数は、直前のグループの左にある後続グループを定義します。最後の数字が -1 でなければ、直前の数字を使用して、その後のグループ分けが実行されます。最後の数字が -1 の場合には、グループ分けは、指定されたグループ数分だけ実行されます。

以下に、キーワード mon_grouping の解釈の例を示します。フォーマット設定される値は 123456789 であり、キーワード mon_thousands_sep のオペランドはコンマ (,) であると想定すると、結果は次のようになります。

mon_grouping の値	**フォーマット設定された値**
3;-1	123456,789
3	123,456,789
3;2	12,34,56,789
3;2;-1	134,56,789

positive_sign
フォーマット設定された負でない値をもつ通貨数量を示すのに使用されるストリングを指定します。 ILE COBOL では、これは、ロケールに基づく PICTURE 編集記号 + に対応します。

注: ILE COBOL では、このキーワードは、ロケールに基づく PICTURE 編集記号 + をフォーマット設定するためにいくつかの他のキーワードと一緒に使用します。キーワード negative_sign、p_sign_posn、および n_sign_posn を参照してください。

negative_sign
フォーマット設定された負の値をもつ通貨数量を示すのに使用されるストリングを指定します。

注: ILE COBOL では、このキーワードは、ロケールに基づく PICTURE 編集記号 + をフォーマット設定するためにいくつかの他のキーワードと一緒に使用します。キーワード positive_sign、p_sign_posn、および n_sign_posn を参照してください。

int_frac_digits

int_curr_symbol 値を使用してフォーマット設定された通貨数量に表示される小数部の桁数 (小数点の後にあるもの) を表す整数値を指定します。このキーワードは、ILE COBOL では使用されません。

frac_digits

currency_symbol 値を使用してフォーマット設定された通貨数量に表示される小数部の桁数 (小数点の後にあるもの) を表す整数値を指定します。このキーワードは、ILE COBOL では使用されません。

p_cs_precedes

int_curr_symbol ストリングまたは currency_symbol ストリングがフォーマット設定された負でない通貨数量の値より前にくるか、後にくるかを示す整数値を指定します。認識される整数値は、次のものです。

0 通貨記号が通貨数量の後にくることを指示します。

1 通貨記号が通貨数量の前にくることを指示します。

注: ILE COBOL では、このキーワードは、ロケールに基づく PICTURE 編集記号 cs をフォーマット設定するためにいくつかの他のキーワードと一緒に使用します。キーワード currency_symbol、p_sep_by_space、n_cs_precedes、および n_sep_by_space を参照してください。

p_sep_by_space

int_curr_symbol ストリングまたは currency_symbol ストリングと、フォーマット設定された負でない通貨数量とをスペースで区切るかどうかを示す整数値を指定します。認識される整数値は、次のものです。

0 通貨記号と通貨数量をスペースで区切らないことを指示します。

1 通貨記号と通貨数量をスペースで区切ることを指示します。

2 通貨記号と positive_sign ストリングが隣接した場合にスペースで区切ることを指示します。

注: ILE COBOL では、このキーワードは、ロケールに基づく PICTURE 編集記号 cs をフォーマット設定するためにいくつかの他のキーワードと一緒に使用します。キーワード currency_symbol、p_cs_precedes、n_cs_precedes、および n_sep_by_space を参照してください。

n_cs_precedes

int_curr_symbol ストリングまたは currency_symbol ストリングがフォーマット設定された負の通貨数量の値より前にくるか、後にくるかを示す整数値を指定します。認識される整数値は、次のものです。

0 通貨記号が通貨数量の後にくることを指示します。

1 通貨記号が通貨数量の前にくることを指示します。

注: ILE COBOL では、このキーワードは、ロケールに基づく PICTURE 編集記号 cs をフォーマット設定するためにいくつかの他のキーワードと一緒に使用します。キーワード currency_symbol、p_cs_precedes、p_sep_by_space、および n_sep_by_space を参照してください。

n_sep_by_space

int_curr_symbol ストリングまたは currency_symbol ストリングをフォーマッ

ト設定された負の通貨数量とスペースで区切るかどうかを示す整数値を指定します。認識される整数値は、次のものです。

0 通貨記号と通貨数量をスペースで区切らないことを指示します。

1 通貨記号と通貨数量をスペースで区切ることを指示します。

2 通貨記号と negative_sign ストリングが隣接した場合にスペースで区切ることを指示します。

注: ILE COBOL では、このキーワードは、ロケールに基づく PICTURE 編集記号 cs をフォーマット設定するためにいくつかの他のキーワードと一緒に使用します。キーワード currency_symbol、p_cs_precedes、p_sep_by_space、および n_cs_precedes を参照してください。

p_sign_posn

フォーマット設定された負でない通貨数量の positive_sign ストリングの位置を示す整数値を指定します。認識される整数値は、次のものです。

0 通貨数量と int_curr_symbol ストリングまたは currency_symbol ストリングの両方を括弧で囲むことを指示します。

1 positive_sign ストリングが、数量および int_curr_symbol ストリングまたは currency_symbol ストリングの前にくることを指示します。

2 positive_sign ストリングが、数量および int_curr_symbol ストリングまたは currency_symbol ストリングの後にくることを指示します。

3 positive_sign ストリングが int_curr_symbol ストリングまたは currency_symbol ストリングの直前にくることを指示します。

4 positive_sign ストリングが int_curr_symbol ストリングまたは currency_symbol ストリングの直後にくることを指示します。

注: ILE COBOL では、このキーワードは、ロケールに基づく PICTURE 編集記号 + をフォーマット設定するためにいくつかの他のキーワードと一緒に使用します。キーワード positive_sign、negative_sign、および n_sign_posn を参照してください。

n_sign_posn

フォーマット設定された負の通貨数量の negative_sign ストリングの位置を示す整数値を指定します。認識される整数値は、次のものです。

0 通貨数量と int_curr_symbol ストリングまたは currency_symbol ストリングの両方を括弧で囲むことを指示します。

1 negative_sign ストリングが、数量および int_curr_symbol ストリングまたは currency_symbol ストリングの前にくることを指示します。

2 negative_sign ストリングが、数量および int_curr_symbol ストリングまたは currency_symbol ストリングの後にくることを指示します。

3 negative_sign ストリングが int_curr_symbol ストリングまたは currency_symbol ストリングの直前にくることを指示します。

4 negative_sign ストリングが int_curr_symbol ストリングまたは currency_symbol ストリングの直後にくることを指示します。

注: ILE COBOL では、このキーワードは、ロケールに基づく PICTURE 編集記号 + をフォーマット設定するためにいくつかの他のキーワードと一緒に使用します。キーワード positive_sign、negative_sign、および p_sign_posn を参照してください。

固有の通貨形式の作成 - 例

1 つのステートメントの値を変更することによって、カスタマイズされた固有の通貨形式を作成することができます。たとえば、以下の表に、p_cs_precedes、p_sep_by_space、および p_sign_posn の各ステートメントの定義された値のあらゆる組み合わせを使用した結果を示します。

表 14. 各種ロケール可変値の組み合わせの結果

		p_sep_by_space		
		2	**1**	**0**
p_cs_precedes = 1	p_sign_posn = 0	(¥1.25)	(¥ 1.25)	(¥1.25)
	p_sign_posn = 1	+ ¥1.25	+¥ 1.25	+¥1.25
	p_sign_posn = 2	¥1.25 +	¥ 1.25+	¥1.25+
	p_sign_posn = 3	+ ¥1.25	+¥ 1.25	+¥1.25
	p_sign_posn = 4	¥ +1.25	¥+ 1.25	¥+1.25
p_cs_precedes = 0	p_sign_posn = 0	(1.25 ¥)	(1.25 ¥)	(1.25¥)
	p_sign_posn = 1	+1.25 ¥	+1.25 ¥	+1.25¥
	p_sign_posn = 2	1.25¥ +	1.25 ¥+	1.25¥+
	p_sign_posn = 3	1.25+ ¥	1.25 +¥	1.25+¥
	p_sign_posn = 4	1.25¥ +	1.25 ¥+	1.25¥+

LC_MONETARY - 例

以下に、ロケール定義ソース・ファイルにリストされている LC_MONETARY カテゴリーの例を示します。

```
LC_MONETARY
#
int_curr_symbol    "<U><S><D>"
currency_symbol    "<dollar-sign>"
mon_decimal_point  "<period>"
mon_thousands_sep  "<comma>"
mon_grouping       3;-1
positive_sign      "<plus-sign>"
negative_sign      "<hyphen>"
int_frac_digits    2
frac_digits        2
p_cs_precedes      1
p_sep_by_space     2
n_cs_precedes      1
n_sep_by_space     2
p_sign_posn        3
n_sign_posn        3
#
END LC_MONETARY
```

LC_TIME カテゴリー

ILE COBOL では、LC_TIME カテゴリーは、ロケールに基づく日付および時刻をフォーマット設定するのに使用されます。他のロケール・カテゴリーと同様に、LC_TIME はそれらのオペランドが後に続く一連のキーワードで構成されます。LC_TIME キーワードの ″d_fmt″ は、ロケールに基づく日付データ項目の形式を指定します。 LC_TIME キーワードの ″t_fmt″ は、ロケールに基づく時刻データ項目の形式を指定します。

以下のセクションでは、すべての LC_TIME カテゴリーのキーワード (ILE COBOL が現在使用しなくなったものを含めて) の詳細を説明します。以下の説明では、ILE COBOL が現在サポートしていない %a および %c のようないくつかの変換指定子を説明します。

ロケール定義ソース・ファイルの LC_TIME カテゴリーは、時刻と日付情報のフォーマット設定を行うための規則と記号を定義します。このカテゴリーは、LC_TIME カテゴリー・ヘッダーで始まり、END LC_TIME カテゴリー・トレーラーで終わります。

LC_TIME カテゴリー・キーワードのオペランドはすべて、ストリング値または整数値として定義されます。ストリング値は、二重引用符 (″″) で結合されます。値はすべて、1 つまたは複数のスペースで、それらが定義するキーワードと区切られます。 2 つの二重引用符が隣接している場合は、未定義ストリング値であることを示します。 -1 は未定義整数値であることを示します。フィールド記述子は、時刻と日付の形式のエレメントを表示するために LC_TIME カテゴリーに照会するコマンドおよびサブルーチンによって使用されます。 LC_TIME カテゴリーでは、以下に示すキーワードが認識されます。

abday %a フィールド記述子に一致する省略した曜日名を定義します。 7 つのセミコロンで区切られたストリングで構成された値が認識されます。最初のストリングは、週の最初の日 (日曜) の省略名に相当し、2 番目は、週の 2 日目の省略名に相当し、... というようになります。

day %A フィールド記述子に一致する曜日名を完全なスペルで定義します。 7 つのセミコロンで区切られたストリングで構成された値が認識されます。最初のストリングは、週の最初の日 (日曜) の完全なスペルの名前に相当し、2 番目は、週の 2 日目の名前に相当し、... というようになります。このキーワードは、ILE COBOL では使用されません。

abmon %b フィールド記述子に一致する省略した月の名前を定義します。 12 のセミコロンで区切られたストリングで構成された値が認識されます。最初のストリングは、年の最初の月 (1 月) の省略名に相当し、2 番目は、年の 2 カ月目の省略名に相当し、... というようになります。このキーワードは、ILE COBOL では使用されません。

mon %B フィールド記述子に一致する月の名前を完全なスペルで定義します。 12 のセミコロンで区切られたストリングで構成された値が認識されます。最初のストリングは、年の最初の月 (1 月) の完全なスペルの名前に相当し、2 番目は、年の 2 カ月目の完全なスペルの名前に相当し、.... というようになります。このキーワードは、ILE COBOL では使用されません。

d_t_fmt
%c フィールド記述子に一致する標準の日付と時刻の形式を使用したストリ

ングを定義します。ストリングには、どのような組み合わせの文字、フィールド記述子、またはエスケープ・シーケンスも入れることができます。このキーワードは、ILE COBOL では使用されません。

d_fmt %x フィールド記述子に一致する標準の日付形式を使用したストリングを定義します。ストリングには、どのような組み合わせの文字、フィールド記述子、またはエスケープ・シーケンスも入れることができます。以下の例は、d_fmt キーワードがどのように構成されているかを示します。

%D %D は %m/%d/%y の日付形式を示します。

%d-%m-%y

%m/%d/%Y

t_fmt %X フィールド記述子に一致する標準時刻形式を使用したストリングを定義します。ストリングには、どのような組み合わせの文字、フィールド記述子、またはエスケープ・シーケンスも入れることができます。以下の例は、t_fmt キーワードがどのように構成されているかを示します。

%H:%M:%S

%H.%M.%S

am_pm

%p フィールド記述子に対応する、AM (午前) と PM (午後) を表すために使用するストリングを定義します。 ; (セミコロン) で区切られた 2 つのストリングで構成された値が認識されます。最初のストリングは、AM (午前) の指定に相当し、最後のストリングは、PM (午後) の指定に相当します。

t_fmt_ampm

am_pm 値 (%p フィールド記述子) を含む標準 12 時間の時刻形式を使用するストリングを定義します。このステートメントは、%r フィールド記述子に相当します。ストリングには、どのような組み合わせの文字およびフィールド記述子も入れることができます。このキーワードは、ILE COBOL では使用されません。

era %E フィールド記述子の修飾子に相当するロケール内のそれぞれの紀元ごとに、どのように年をカウントし、表示するかを定義します。それぞれの紀元ごとに、以下の形式で 1 つのストリングが必要です。

direction:offset:start_date:end_date:era_name:era_format

このキーワードは、ILE COBOL では使用されません。

紀元のストリング形式の変数は、以下のように定義されます。

direction

- (負符号) または + (正符号) 文字で指定します。正符号は、開始の日付から終了の日付に移る場合に、正の方向にカウントする年を指示します。負符号は、開始の日付から終了の日付に移る場合に、負の方向にカウントする年を指示します。

offset 紀元の最初の年を表す数値を指定します。

start_date

yyyy/mm/dd 形式で紀元の開始日を指定します。ここで、yyyy、mm、および dd は、それぞれ年、月、日を表します。 AD 1 年よ

り前の年は、負の数字で表されます。たとえば、BC 100 年の 3 月 5 日に開始する紀元は、-100/03/05 として表されます。

end_date

start_date 変数、または 2 つの特殊値 -* または +* のいずれか 1 つを使用する同じ形式で、紀元の終了日を指定します。 -* 値は、紀元の終了日が、逆方向の時間で展開されることを示します。 +* 値は、紀元の終了日が、順方向の時間で展開されることを示します。したがって、終了の日付は、時間的な順序に従って、紀元の開始日の前または後になります。たとえば、西暦紀元 AD および BC のストリングは、以下のように入力します。

```
+:0:0000/01/01:+*:AD:%o %N
+:1:-0001/12/31:-*:BC:%o %N
```

era_name

%EC フィールド記述子を置換する紀元の名前を表すストリングを指定します。

era_format

%EY フィールド記述子をフォーマット設定するためのストリングを指定します。

1 つの**紀元**値は、それぞれの紀元ごとに 1 つのストリングで構成されています。複数の紀元を指定する場合、各ストリングは ; (セミコロン) で区切ります。

era_d_fmt

%Ex フィールド記述子に相当する代替紀元の形式で日付を表すのに使用するストリングを定義します。ストリングには、どのような組み合わせの文字およびフィールド記述子も入れることができます。

era_t_fmt

%EX フィールド記述子に相当する代替紀元の形式で時刻を表すのに使用するストリングを定義します。ストリングには、どのような組み合わせの文字およびフィールド記述子も入れることができます。

era_d_t_fmt

%Ec フィールド記述子に相当する代替紀元の形式で日付と時刻を表すのに使用するストリングを定義します。ストリングには、どのような組み合わせの文字およびフィールド記述子も入れることができます。

alt_digits

%O フィールド記述子に相当する数字の代替ストリングを定義します。 ; (セミコロン) で区切られたストリングのグループで構成された値が認識されます。最初のストリングは、ゼロの代替ストリングを表し、2 番目のストリングは 1 の代替ストリングを表し、.... というようになります。指定できるのは、最大 100 代替ストリングです。

エスケープ・シーケンス

d_t_fmt、d_fmt、および t_fmt キーワード値に許可されるエスケープ・シーケンスは、以下のとおりです。

**** 円記号を表します。

¥a 警報文字を表します。

¥b バックスペース文字を表します。

¥f 用紙送り文字を表します。

¥n 改行 (newline) 文字を表します。

¥r 復帰 (キャリッジ・リターン) 文字を表します。

¥t タブ文字を表します。

¥v 垂直タブ文字を表します。

LC_TIME 例

以下に、ロケール定義ソース・ファイルにリストされている LC_TIME カテゴリーの例を示します。

```
LC_TIME
#
#Abbreviated weekday names (%a)
abday   "<S><u><n>";"<M><o><n>";"<T><u><e>";"<W><e><d>";¥
        "<T><h><u>";"<F><r><i>";"<S><a><t>"
#
#Full weekday names (%A)
day     "<S><u><n><d><a><y>";"<M><o><n><d><a><y>";¥
        "<T><u><e><s><d><a><y>";"<W><e><d><n><e><s><d><a><y>";¥
        "<T><h><u><r><s><d><a><y>";"<F><r><i><d><a><y>";¥
        "<S><a><t><u><r><d><a><y>"
#
#Abbreviated month names (%b)
abmon   "<J><a><n>";"<F><e><b>";"<M><a><r>";"<A><p><r>";¥
        "<M><a><y>";"<J><u><n>";"<J><u><l>";"<A><u><g>";¥
        "<S><e><p>";"<O><c><t>";"<N><o><v>";"<D><e><c>"
#
#Full month names (%B)
mon     "<J><a><n><u><a><r><y>";"<F><e><b><r><u><a><r><y>";¥
        "<M><a><r><c><h>";"<A><p><r><i><l>";"<M><a><y>";¥
        "<J><u><n><e>";"<J><u><l><y>";"<A><u><g><u><s><t>";¥
        "<S><e><p><t><e><m><b><e><r>";"<O><c><t><o><b><e><r>";¥
        "<N><o><v><e><m><b><e><r>";"<D><e><c><e><m><b><e><r>"
#
#Date and time format (%c)
d_t_fmt "%a_%bf%d %H:%M:%S %Y"
#
#Date format (%x)
d_fmt   "%m/%d/%y"
#
#Time format (%X)
t_fmt   "%H:%M:%S"
#
#Equivalent of AM/PM (%p)
am_pm   "<A><M>";"<P><M>"
#
#12-hour time format (%r)
t_fmt_ampm   "%I:%M:%Sm%p"
#
era     "+:0:0000/01/01:+*:AD:%EC";¥
        "+:1:-0001/12/31:-*:BC:%Ey";
era_d_fmt  ""
alt_digits    "<0><t><h>";"<1><s><t>";"<2><n><d>";"<3><r><d>";¥
              "<4><t><h>";"<5><t><h>";"<6><t><h>";"<7><t><h>";¥
              "<8><t><h>";"<9><t><h>";"<1><0><t><h>"
#
END LC_TIME
```

LC_TOD カテゴリー

ILE COBOL では、LC_TOD ロケール・カテゴリーは、ロケールに基づく時刻項目の時間帯を指示します。特に、tzdiff キーワードは、現地時間とグリニッジ標準時間の差を指定します。この情報は、ロケールに基づく時刻項目を他の時刻 (ロケールまたは非ロケールに基づくもの) に移動したり、比較したりする時に使用されます。 ILE COBOL が現在使用する tzdiff キーワードは、LC_TOD キーワードだけです。

LC_TOD カテゴリーは、夏時間の開始および終了時刻、現地時間とグリニッジ標準時間の差、時間帯の名前、および夏時間の名前を定義するために使用する規則を定義します。このカテゴリーは、IBM 拡張であり、ソース・ファイル内の他のすべてのカテゴリー定義より後ろに表示する必要があります。

LC_TOD カテゴリーのオペランドはすべて、ストリング値または整数値として定義されます。ストリング値は、二重引用符 (″″) で結合されます。値はすべて、1 つまたは複数のスペースで、それらが定義するキーワードと区切られます。 2 つの二重引用符が隣接している場合は、未定義ストリング値であることを示します。 0 (ゼロ) は未定義整数値であることを示します。 LC_TOD カテゴリーでは、以下に示すキーワードが認識されます。

tzdiff 時間帯の差を分単位で表す整数値を指定します。これは、現地時間とグリニッジ標準時間の差です。

tname 時間帯の名前に使用されるストリングを指定します。

dstname
夏時間の名前に使用されるストリングを指定します。

dststart
夏時間の開始日を表す 4 つの整数のセットを指定します。 dststart キーワードのオペランドは、以下の形式のように 4 つのコンマで区切られた一連の整数で構成されます。

```
month,week,day,time
```

dststart 形式の変数は、以下のように定義されます。

month 夏時間 (DST) の開始月を表す、整数値を指定します。この値は 1 から 12 までの範囲で、1 は 1 月に相当し、12 は 12 月に相当します。

week DST の開始月の週を表す整数値を指定します。この値は、-4 から 4 までの範囲で、-4 は月末から数えた 4 番目の週に相当し、4 は月の初めから数えた 4 番目の週に相当します。

day DST の開始月の日にちを表す整数値を指定します。また、week キーワードが 0 (ゼロ) でない場合、これは、DST が有効になった曜日になります。この値は、1 からその月の最後の日、または 1 からその週の最後の曜日の範囲です。

time DST が有効になった時、ローカル標準時刻の真夜中の 12 時以後の秒数を表す整数値を指定します。この値は 0 から 86399 までの範囲です。

dstend 夏時間の終了日を表す 4 つの整数のセットを指定します。 dstend キーワードのオペランドは、以下の形式のように 4 つのコンマで区切られた一連の整数で構成されます。

month,week,day,time

dstend 形式の変数は、以下のように定義されます。

month 夏時間 (DST) 終了月を表す、整数値を指定します。この値は 1 から 12 までの範囲で、1 は 1 月に相当し、12 は 12 月に相当します。

week DST が終了した月の週を表す整数値を指定します。この値は、-4 から 4 までの範囲で、-4 は月末から数えた 4 番目の週に相当し、4 は月の初めから数えた 4 番目の週に相当します。

day DST が終了した月の日にちを表す整数値を指定します。また、week キーワードが 0 (ゼロ) でない場合、これは、DST が終了した週の日にちになります。この値は、1 からその月の最後の日、または 1 からその週の最後の曜日の範囲です。

time DST が有効になった時、ローカル標準時刻の真夜中の 12 時以後の秒数を表す整数値を指定します。この値は 0 から 86399 までの範囲です。

dstshift

夏時間のシフトを秒数で表す整数値を指定します。

LC_TOD の例

以下に、ロケール定義ソース・ファイルにリストされている LC_TOD カテゴリーの例を示します。

```
LC_TOD
#
tzdiff    360
tname     "<C><e><n><t><r><a><l>"
dstname   "<P><D><T>"
#Set daylight savings time to start on 3rd week of October at
#midnight on Saturday.
dststart  10,3,6,0
#Set daylight savings time to end on April 23, at midnight.
dstend    4,0,23,0
dstshift  3600
#
END LC_TOD
```

ヌル終了ストリングの操作

さまざまな仕組みを使用して、ヌル終了ストリング (例えば、C プログラムとの間で受け渡しされるストリング) を構成し、操作することができます。

例えば、以下のことができます。

- ヌル終了リテラル定数 (Z". . . ") を使用する。
- INSPECT ステートメントを使用して、ヌル終了ストリングの文字数をカウントする。

```
MOVE 0 TO char-count
INSPECT source-field TALLYING char-count
                     FOR CHARACTERS
                     BEFORE X"00"
```

- UNSTRING ステートメントを使用して、ヌル終了ストリング内の文字をターゲット・フィールドに移動し、文字カウントを取得する。

```
WORKING-STORAGE SECTION.
01  source-field          PIC X(1001).
01  char-count     COMP   PIC 9(4).
01  target-area.
    02 individual-char OCCURS 1 TO 1000 TIMES DEPENDING ON char-count
                         PIC X.
. . .
PROCEDURE DIVISION.
    . . .
    UNSTRING source-field DELIMITED BY X"00"
                         INTO target-area
                         COUNT IN char-count
      ON OVERFLOW
        DISPLAY "source not null terminated or target too short"
        . . .
    END-UNSTRING
```

- SEARCH ステートメントを使用して、末尾のヌルまたはスペース文字を見つける。検査対象のストリングを、単一文字のテーブルとして定義します。
- ループ内のフィールドの各文字を調べる (PERFORM)。フィールド内の各文字は、source-field (I:1) などの参照修飾子を使用して調べることができます。

『例: ヌル終了ストリング』

関連リファレンス
『ヌル終了非数値リテラル』(「*ILE COBOL 言語解説書*」)

例: ヌル終了ストリング

次の例は、ヌル終了ストリングの処理方法をいくつか示しています。

```
 01  L pic X(20) value z'ab'.
 01  M pic X(20) value z'cd'.
 01  N pic X(20) value z'xyz'.
 01  N-Length pic 99 value zero.
 01  N pic X(20) value z'xyz'.
 01  X pic X(20).
 01  Y pic X(13) value 'Hello, World!'.
 . . .
* Display null-terminated string
     Inspect N tallying N-length
       for characters before initial x'00'
     Display 'N: ' N(1:N-Length) ' Length: ' N-Length
     . . .
* Move null-terminated string to alphanumeric, strip null
     Unstring N delimited by X'00' into X
     . . .
* Create null-terminated string
     String Y     delimited by size
            X'00' delimited by size
            into N.
     . . .
* Concatenate two null-terminated strings to produce another
     String L     delimited by x'00'
            M     delimited by x'00'
            X'00' delimited by size
            into N.
```

第 9 章 ILE COBOL プログラム相互間での呼び出しとデータ共用

アプリケーションの中には、それだけが単独で実行するようにコーディングされた単純なものもあります。しかし、ほとんどの場合、1 つのアプリケーションのソリューションは、別個にコンパイルされた複数のプログラムを一緒に使用するように構成されています。

IBM i システムには、ILE COBOL プログラム相互間の通信、および ILE COBOL と非 ILE COBOL プログラムとの間の通信を行う機能があります。

この章では次のことについて説明します。

- 他の ILE COBOL プログラムを呼び出すためのさまざまな方法
- 呼び出し先プログラムの実行が終了した時に、制御を呼び出し側プログラムに戻す方法
- 呼び出し側プログラムと呼び出し先プログラムとの間でのデータのやりとりの方法
- ILE COBOL プログラムの取り消し方法

実行時間に関係したさまざまな概念

プログラム・オブジェクトは、1 つまたは複数のモジュール・オブジェクトから作成されます。各プログラム・オブジェクトには、活動化されるときの主な入り口点として常に 1 つの (唯一の) モジュール・オブジェクトが指定されています。 ILE COBOL コンパイラーによってモジュール・オブジェクトが生成されると、 PEP が生成されます。この PEP は、コンパイル単位内に含まれる最外部の ILE COBOL プログラムを呼び出します。複数部分からなるモジュール・オブジェクトをバインドしてプログラム・オブジェクトを作成する場合、どのモジュール・オブジェクトに作成されるプログラム・オブジェクトの PEP が入るかを指定しなければなりません。これは CRTPGM コマンドの ENTMOD パラメーターでモジュール・オブジェクトを識別することによって実行します。これによりモジュール・オブジェクトの PEP はプログラム・オブジェクトの PEP になります。

動的プログラム呼び出しを使ってプログラム・オブジェクトを活動化した場合、PEP に制御が渡されます。次に PEP は、最初に実行されるモジュール・オブジェクトの最外部の ILE COBOL プログラムである UEP を呼び出します。 PEP および UEP に関する詳細については、「*ILE 概念*」を参照してください。

活動化および活動化グループ

プログラム・オブジェクトまたはサービス・プログラムを実行できるようにすることを**活動化**といいます。活動化は、システムによりプログラム・オブジェクトが呼び出された時点で実行されます。サービス・プログラムの場合、その全体が呼び出されるわけではないので、サービス・プログラムが活動化されるのは、プログラム・オブジェクトが直接または間接にサービス・プログラムを必要とする時になり

ます。サービス・プログラム内の ILE プロシージャーは、静的プロシージャー呼び出しを使用して呼び出されます。それらは動的プログラム呼び出しを使用して呼び出すことはできません。

活動化によって、次の機能が実行されます。

- プログラム・オブジェクトまたはサービス・プログラムが必要とする静的データを固有に割り振る。
- 使用するサービス・プログラムのシンボリック・リンクを物理アドレスのリンクに変更する。

活動化によってプログラム・オブジェクトが使用する静的変数に必要なストレージが割り振られると、スペースが**活動化グループ**から割り振られます。それぞれの活動化グループには名前があります。活動化グループの名前は、ユーザーが指定します (または *NEW を指定するとシステムにより提供されます)。そのプログラム・オブジェクトまたはサービス・プログラムを活動化する活動化グループは、CRTPGM または CRTSRVPGM を使用してプログラム・オブジェクトまたはサービス・プログラムを作成する時に指定できます。活動化および活動化グループに関する詳細については、「*ILE 概念*」を参照してください。

COBOL 実行単位

COBOL **実行単位**とは、実行時に問題解決の 1 つの単位として機能する 1 つまたは複数のプログラムの集まりです。 COBOL 実行単位は他の実行単位から独立したエンティティーなので、実行時に他の実行単位と通信または調整をする必要はありません (他の実行単位によって使用されるデータ・ファイルとメッセージの処理や、スイッチの設定およびテストをする場合は除く)。さらに実行単位には、ILE COBOL 以外の言語で作成されたプログラムをコンパイルして作成されたモジュール・オブジェクトから作成したプログラム・オブジェクトとサービス・プログラムも含めることができます。

ILE では、1 つの COBOL 実行単位は、単一の ILE 活動化グループの中で実行されるプログラム・オブジェクトおよびサービス・プログラムで構成されます。 OPM COBOL/400 と互換性のある実行単位のセマンティクスを保つためには、ILE COBOL アプリケーションは次の条件を満たさなければなりません。

- 各 ILE COBOL コンパイル単位は、コンパイルして単一プログラム・オブジェクトにバインドする必要があります。
- 実行単位の参加プログラムのすべて (ILE COBOL または他の ILE プログラム / プロシージャー) は、単一の ILE 活動化グループ内で実行する必要があります。

  注: COBOL 実行単位のセマンティクスを適切に保持するためには、名前を指定した ILE 活動化グループを使用して、そこでアプリケーションを実行する必要があります。すべての参加プログラム・オブジェクトに対して、名前付きの ILE 活動化グループを使用することにより、アプリケーションを実行する前に特定の ILE COBOL プログラムをメイン・プログラムとして指定する必要はなくなります。

  一方、アプリケーションを実行する前に特定の ILE COBOL プログラム・オブジェクトがメイン・プログラムとして認識されている場合は、ILE COBOL プログラムを UEP として使用して *PGM オブジェクトを作成する時に、

ACTGRP オプションに *NEW 属性を指定することができます。それ以外のすべての参加プログラム・オブジェクトは、ACTGRP オプションに *CALLER 属性を指定する必要があります。

- 実行単位に対応する ILE 活動化グループの最も古い呼び出しが、その ILE COBOL のものでなければなりません。これは実行単位のメイン・プログラムです。

上記の条件が満たされていない場合、STOP RUN の有効範囲をバインドする管理境界が生じて、アプリケーション全体の状態がリフレッシュされない可能性があります。

注: 上記の条件のために、*DFTACTGRP で実行される ILE COBOL プログラムは、一般的に OPM COBOL/400 実行単位とは互換性のない実行単位で実行されます。

管理境界

すべての ILE 言語 (ILE COBOL を含む) では、呼び出し先 ILE プロシージャーまたは OPM プログラム・オブジェクトとの間で制御を転送するために、**呼び出しスタック**と呼ばれる共通のメカニズムを使用します。呼び出しスタックは、呼び出し先の ILE プロシージャーまたはプログラム・オブジェクトごとにそれぞれ 1 つの項目を含む、後入れ先出しの呼び出しスタック項目のリストを構成します。各呼び出しスタック項目には、ILE プロシージャーの自動変数、およびその呼び出しスタック項目の範囲のその他のリソース (たとえば、異常事態処理ルーチンと取り消しハンドラー) に関する情報が入れられます。

ILE COBOL では、呼び出し先の ILE COBOL プログラムまたはネストされたプログラムごとに、それぞれ 1 つの呼び出しスタック項目があります。呼び出される各宣言にも、それぞれの呼び出しスタック項目があります。

呼び出しを行うと、呼び出し先の ILE プロシージャーまたは OPM プログラム・オブジェクトのスタックに新しい項目が追加され、呼び出し先オブジェクトに制御が渡されます。戻り時には、その呼び出しスタック項目は除去され、直前の呼び出しスタック項目にある呼び出された ILE プロシージャーまたはプログラム・オブジェクトに制御が渡されます。

ILE では、複数の活動化グループでプログラム・オブジェクトを実行可能なアプリケーションを作成することができます。呼び出し側プログラムの活動化グループのものとは異なる活動化グループで実行されている ILE COBOL プログラム・オブジェクトを呼び出すことができます。この場合、この呼び出し先プログラム・オブジェクトの呼び出しスタック項目は、**管理境界**と呼ばれます。管理境界とは、ILE 呼び出しスタック項目のうち、直前の呼び出しスタック項目が別の活動化グループの ILE プロシージャーまたはプログラム・オブジェクトになっているものと定義されます。直前の呼び出しスタック項目が OPM プログラム・オブジェクトのものである ILE 呼び出しスタック項目も管理境界です。

呼び出し先プログラム・オブジェクトが、特定の活動化グループで最初に活動化されるプログラム・オブジェクトである場合、その呼び出しスタック項目は、**ハード管理境界**と呼ばれます。 管理境界としての呼び出し先プログラム・オブジェクトが、活動化グループで最初に活動化されるプログラム・オブジェクトではない場

合、その呼び出しスタック項目は、**ソフト管理境界**と呼ばれます。 OPM COBOL/400 実行単位と互換性のある実行単位のメイン・プログラムは、活動化グループのハード管理境界にあります。

STOP RUN ステートメント (または ILE COBOL のメイン・プログラムの GOBACK ステートメント) が、呼び出し先の ILE COBOL プログラムの中に検出された場合、管理境界の呼び出し側に制御が渡されます。 OPM COBOL/400 実行単位と互換性のある実行単位では、STOP RUN により実行単位は停止します。

コミットメント制御が活動化グループの範囲であり、活動化が正常終了して、ファイルのクローズにエラーがなかった場合、コミットメント制御下で暗黙の COMMIT 操作がファイルに対して実行されます。活動化グループが異常終了するか、ファイルのクローズにエラーがあった場合、ROLLBACK 操作が実行されます。コミットメント制御がジョブの範囲である場合は、何も起こりません。

また、管理境界では、未処理エラーは、機能チェックに変えられます。その機能チェックも処理されなかった場合、管理境界においてその機能チェックは、一般 ILE 障害条件 CEE9901 に変更され、管理境界の呼び出し側に送られます。

メイン・プログラムとサブプログラム

活動化グループ内で最初に活動化されるプログラムは、COBOL 実行単位を開始するプログラムであり、**メイン・プログラム**と呼ばれます。メイン・プログラムは、活動化グループのハード管理境界にあります。特定のソース・ステートメントまたはオプションによって ILE COBOL プログラムをメイン・プログラムまたはサブプログラムとして識別することはできません。

サブプログラムは、呼び出しスタックでメイン・プログラムより下位にある実行単位のプログラムです。プログラム・スタックおよびプログラム間の通信に関連する他の用語の詳細については、「*CL プログラミング*」を参照してください。

ストレージの初期設定

実行単位内の ILE COBOL プログラムを最初に呼び出す時点で、そのストレージは初期設定されます。ストレージは、次の条件下で再び初期設定されます。

- ILE COBOL プログラムの PROGRAM-ID 段落が、INITIAL 文節を使用している場合。この場合、プログラムが呼び出されるたびにストレージは再初期設定されます。
- 実行単位が終了し、再初期設定される場合。
- (ILE COBOL の CANCEL ステートメントを使って) プログラムを取り消してから、再度呼び出した場合。
- (前の PERFORM ステートメントによって設定される) セクション名および段落名のブランチ・アドレスの終わりで、プログラムが呼び出されるごとに常に再初期設定される場合。

他のプログラムへの制御の移動

手続き部では、プログラムは他のプログラム (通常 COBOL 用語ではサブプログラムと呼ばれる) を呼び出し、その呼び出されたプログラムからさらに別のプログラムを呼び出すことができます。別のプログラムを呼び出すプログラムを**呼び出し側**プログラムといい、そのプログラムが呼び出すプログラムを**呼び出し先**プログラムといいます。

呼び出し先 ILE COBOL プログラムは、手続き部の非宣言部分の先頭から実行を開始します。呼び出し先 ILE COBOL プログラムに手続き部がない場合、または手続き部の非宣言部分がない場合には、単に呼び出し側 ILE COBOL プログラムに戻るだけになります。

呼び出し先プログラムの処理が終了すると、制御を呼び出し側プログラムに戻すか、または実行単位を終了することができます。 STOP RUN が出された場合、および最も近い管理境界がハード管理境界である場合には、実行単位が終了します。最も近い管理境界がソフト管理境界であった場合、制御はその管理境界の呼び出し側プログラムに戻されますが、実行単位は活動状態のままです。

呼び出し先プログラムは、直接的にも、間接的にも (たとえば、プログラム X がプログラム Y を呼び出し、プログラム Y がプログラム Z を呼び出し、さらにプログラム Z がプログラム X を呼び出す) その呼び出し側プログラムを呼び出すことはできません。呼び出し先プログラムがその呼び出し側プログラムを呼び出すことを、**再帰**呼び出しといいます。 ILE COBOL では、非再帰的なメイン・プログラムまたはサブプログラムにおいて再帰呼び出しは実行**できません**。再帰呼び出しが可能なのは、再帰呼び出しされるプログラムの PROGRAM-ID 段落で RECURSIVE 文節をコーディングしたときに限られます。PROGRAM-ID 段落に RECURSIVE 文節がコーディングされていない COBOL プログラムの再帰呼び出しを行うと、実行時エラー・メッセージが生成されます。

ILE COBOL プログラムの呼び出し

他の ILE COBOL プログラムを呼び出すには、次の方法を使用することができます。

- ネストされたプログラムの呼び出し
- 静的プロシージャー呼び出し
- 動的プログラム呼び出し

ネストされたプログラムの呼び出しを使用すると、構造化プログラミングの技法を使ってアプリケーションを作成することができます。また、PERFORM プロシージャーの代わりに使用することによって、データ項目を意図せずに変更してしまうのを防ぐことができます。ネストされたプログラムの呼び出しは、CALL *リテラル* または CALL *id* ステートメントのどちらかを使用することにより作成することができます。ネストされたプログラムの詳細については 242 ページの『ネストされたプログラムの呼び出し』を参照してください。

静的プロシージャー呼び出しは、呼び出し側 ILE COBOL プログラムと同じプログラム・オブジェクトのコピーまたは参照によってバインドされた、呼び出し先 ILE COBOL プログラムに制御を渡します。静的プロシージャー呼び出しは、CALL リ

テラル または CALL *プロシージャー・ポインター* のステートメントを使用して作成することができます。静的プロシージャー呼び出しを使用する場合、次のものを呼び出すことができます。

- 同じモジュール・オブジェクト内の ILE プロシージャー
- ネストされた ILE COBOL プログラム (CALL *リテラル* を使用)
- 呼び出し側 ILE COBOL プログラムにバインドされた別個のモジュール・オブジェクトにある ILE プロシージャー
- 別個のサービス・プログラムの ILE プロシージャー

動的プログラム呼び出しは、呼び出し側 ILE COBOL プログラムから別個のプログラム・オブジェクトにバインドされた、呼び出し先 ILE COBOL プログラムに制御を渡します。呼び出し先 ILE COBOL プログラムは、プログラム・オブジェクトの UEP でなければなりません。プログラム・オブジェクトの UEP である ILE COBOL プログラムは、別のプログラム・オブジェクトにある別の ILE COBOL プログラムから呼び出される唯一のプログラムです。 ILE COBOL プログラムは、UEP として指定されたもの以外は、プログラム・オブジェクト内からしか見えません。動的プログラム呼び出しを使うと、呼び出し先プログラム・オブジェクトは、活動化グループ内で最初に呼び出される時に活動化されます。動的プログラム呼び出しは、CALL *リテラル*、CALL *id*、または、CALL *プロシージャー・ポインター・データ項目* ステートメントを使用して作成することができます。 CALL *プロシージャー・ポインター・データ項目* ステートメントを使用する前に、SET *プロシージャー・ポインター・データ項目* TO ENTRY *プログラム・オブジェクト名* を使用して、*プロシージャー・ポインター・データ項目* を設定してください。

静的プロシージャー呼び出しと動的プログラム呼び出しの詳細については 247 ページの『静的プロシージャー呼び出しと動的プログラム呼び出しの使用』を参照してください。

呼び出し先のプログラムとプロシージャーのリンケージ・タイプの識別

呼び出し側プログラムとは異なるモジュール・オブジェクトにある他の ILE COBOL プログラムを呼び出す場合に、CALL *リテラル* のステートメントによって呼び出しを行うのであれば、呼び出し先プログラムが、ILE プログラムであるかまたは ILE プロシージャーであるかを指定する必要があります。

呼び出しのリンケージ・タイプを指定することによって、呼び出しているのがプログラム・オブジェクトかプロシージャーかを識別します。

呼び出しの LINKAGE タイプは明示的に指定したり、特定のリンケージに関連する句を指定することによって強制したりできます。たとえば、LIBRARY 句では、呼び出しが LINKAGE プログラムになるように強制されます。リンケージを強制する句がないインスタンスでは、明示的にリンケージを指定する次の 3 つの方法があります。優先度の高いもの順になっています。

1. CALL、CANCEL、または SET...ENTRY ステートメントの LINKAGE 句
   - プログラム・オブジェクトの呼び出しまたは取り消しを行うには、CALL、CANCEL、または SET...ENTRY ステートメントで、LINKAGE TYPE IS PROGRAM を指定します。

```
        PROCEDURE DIVISION.
⋮
             CALL LINKAGE TYPE IS PROGRAM literal-1
⋮
             CALL LINKAGE PROGRAM literal-2 IN LIBRARY literal-3
⋮
             CANCEL LINKAGE PROGRAM literal-2 IN LIBRARY literal-3
⋮
             CANCEL LINKAGE TYPE IS PROGRAM literal-1
```

- プロシージャーの呼び出しまたは取り消しを行うには、CALL、CANCEL ステートメントまたは SET...ENTRY ステートメントで、LINKAGE TYPE IS PROCEDURE を指定します。 IN LIBRARY 句は、LINKAGE TYPE IS PROCEDURE 句を指定した CALL、CANCEL、または SET ステートメントには指定できません。 IN LIBRARY 句は、プログラム・オブジェクト (*PGM) の IBM i ライブラリー名を指定するのに使用します。

```
        PROCEDURE DIVISION.
⋮
             CALL LINKAGE TYPE IS PROCEDURE literal-1
⋮
             CANCEL LINKAGE TYPE IS PROCEDURE literal-1
```

2. SPECIAL-NAMES 段落の LINKAGE TYPE 文節
   - プログラム・オブジェクトの呼び出しまたは取り消しを行うには、SPECIAL-NAMES 段落で LINKAGE TYPE IS PROGRAM FOR *リテラル-1* を指定します (*リテラル-1* は、呼び出し先プログラム・オブジェクトの名前)。 SPECIAL-NAMES 段落でリンケージがすでに定義されている場合は、CALL、CANCEL、または SET...ENTRY ステートメントで LINKAGE TYPE キーワードを指定する必要はありません。

```
       ENVIRONMENT DIVISION.
         CONFIGURATION SECTION.
⋮
         SPECIAL-NAMES.
             LINKAGE TYPE IS PROGRAM FOR literal-1.
⋮
         PROCEDURE DIVISION.
⋮
             CALL literal-1.
⋮
             CANCEL literal-1.
```

   - プロシージャーの呼び出しまたは取り消しを行うには、SPECIAL-NAMES 段落で LINKAGE TYPE IS PROCEDURE FOR *リテラル-1* を指定します (*リテラル-1* は、呼び出し先プロシージャーの名前)。 SPECIAL-NAMES 段落でリンケージがすでに定義されている場合は、CALL、CANCEL、または SET...ENTRY ステートメントで LINKAGE TYPE 句を指定する必要はありません。

```
       ENVIRONMENT DIVISION.
         CONFIGURATION SECTION.

  ⋮
         SPECIAL-NAMES.
              LINKAGE TYPE IS PROCEDURE FOR literal-1.

  ⋮
         PROCEDURE DIVISION.

  ⋮
              CALL literal-1.

  ⋮
              CANCEL literal-1.
```

3. CRTCBLMOD および CRTBNDCBL コマンドの LINKLIT パラメーター、または関連する PROCESS ステートメントのオプション。
   - CRTCBLMOD および CRTBNDCBL コマンドの LINKLIT パラメーターを使用すると、ILE COBOL プログラムの中のすべての外部 CALL *リテラル-1*、CANCEL *リテラル-1*、または SET *プロシージャー・ポインター・データ項目* TO ENTRY *リテラル-1* に対して、コンパイル時にリンケージ・タイプを指定することができます。 CRTCBLMOD または CRTBNDCBL の LINKLIT パラメーターでリンケージがすでに定義されている場合は、SPECIAL-NAMES 段落の LINKAGE TYPE 文節や、CALL、CANCEL、または SET...ENTRY ステートメントの LINKAGE TYPE 句を指定する必要はありません。
   - プログラム・オブジェクトを呼び出すモジュールを作成するには、次のように入力します。

     ```
     CRTCBLMOD MODULE(MYLIB/XMPLE1)
     SRCFILE(MYLIB/QCBLLESRC)  SRCMBR(XMPLE1)
     LINKLIT(*PGM)
     ```

   - プロシージャーを呼び出すモジュールを作成するには、次のように入力します。

     ```
     CRTCBLMOD MODULE(MYLIB/XMPLE1)
     SRCFILE(MYLIB/QCBLLESRC)  SRCMBR(XMPLE1)
     LINKLIT(*PRC)
     ```

   - CRTCBLMOD の LINKLIT パラメーターを使用してリンケージ・タイプを指定する場合、CALL および CANCEL ステートメントは次のようにコーディングします。

     ```
            PROCEDURE DIVISION.

     ⋮
                 CALL literal-1.

     ⋮
                 CANCEL literal-1.
     ```

ネストされたプログラムの呼び出し

ネストされたプログラムを使うと、アプリケーションのためのモジュラー関数を作成し、構造化プログラミングの技法を維持することができます。ネストされたプログラムによって、それぞれ独自の管理範囲の複数の独立した機能を 1 つのコンパイ

ル単位の中で定義することができます。それらの機能は、PERFORM プロシージャーに**ローカル・**データ項目を保護する機能を加えたものとして使用することができます。

ネストされたプログラムは、コンパイル時には呼び出し側プログラムと同じモジュールに含まれています。したがって、ネストされたプログラムは、常に呼び出し側プログラムと同じ活動化グループで実行されます。

ネストされたプログラムの構造

ILE COBOL プログラムには、他の ILE COBOL プログラムを**含める**ことができます。**含まれている**プログラムにも、他のプログラムが含まれている場合もあります。含まれているプログラムは、他のプログラムに**直接**に含まれている場合と、**間接**に含まれている場合があります。

244 ページの図 51 に、直接または間接に含まれている、ネストされたプログラムの構造を示します。

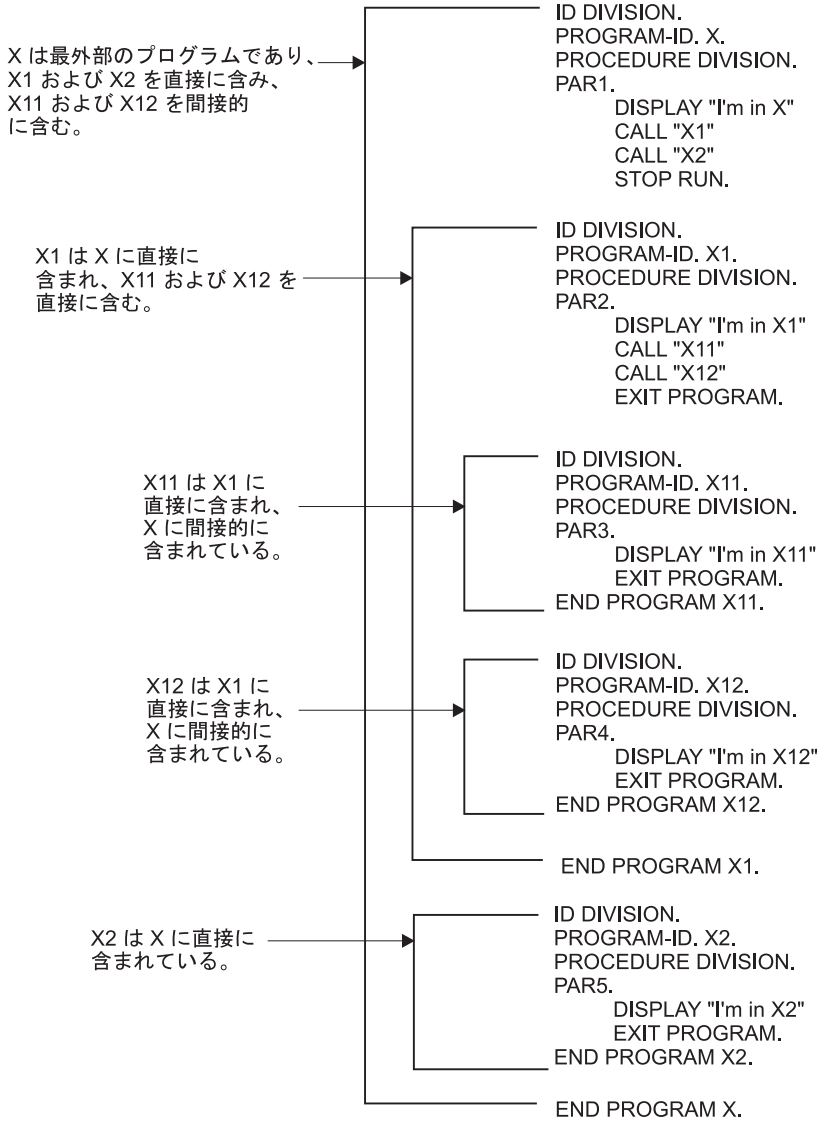


図 51. 直接または間接に含まれているプログラムのネスト構造

ネスト・プログラム構造の使用についての規則

ネスト・プログラム構造を使用するときには、適用すべきいくつかの規則があります。

1. 各プログラムには、見出し部が必要です。それ以外の部の指定は、すべてオプショナルです。
2. PROGRAM-ID 段落のプログラム名は、固有でなければなりません。
3. ネストされたプログラムの名前は、有効な COBOL 語または数値リテラルにすることができます。
4. ネストされたプログラムに構成セクションを含めることはできません。構成セクションのオプションが必要なら、それは最も外側のプログラムで指定しなければなりません。

5. ネストされたプログラムは、そのプログラムを含んでいるプログラムの中の END PROGRAM ヘッダーの直前に含められます (244 ページの図 51 を参照)。
6. 各 ILE COBOL プログラムは、END PROGRAM ヘッダーで終了しなければなりません。
7. ネストされたプログラムの呼び出しまたは取り消しは、同じモジュール・オブジェクトの ILE COBOL プログラムからしか実行できません。
8. ネストされたプログラムの呼び出しは、CALL *リテラル* または CALL *id* ステートメントのどちらかを使用することによってしか実行できません。ネストされたプログラムの呼び出しは、CALL *プロシージャー・ポインター* を使用して実行することはできません。ネストされたプログラムの呼び出しが従う規則は、静的プロシージャー呼び出しの場合と同じです。

ネストされたプログラムの呼び出し階層

ネストされたプログラムは、PROGRAM-ID 段落で COMMON として指定されているのでなければ、それを直接含むプログラムからしか呼び出せません。この場合、COMMON プログラムも、その COMMON プログラムを直接含んでいるプログラム内に (直接または間接に) 含まれているプログラムから呼び出すことができます。再帰呼び出しは、RECURSIVE 文節をもつネストされたプログラムの場合、あるいはネストされたプログラムを直接または間接的に含んでいるプログラムが RECURSIVE 文節をもっている場合にだけ可能です

図 52 に、COMMON として識別されている、いくつかの含まれたプログラムがあるネスト構造の概要を示します。

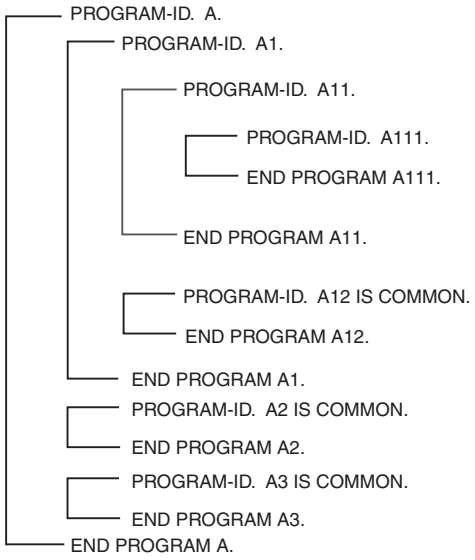


図 52. 直接または間接に含まれているプログラムのネスト構造

図 52 に示されている構造の**呼び出し階層**について、次の表で説明します。A12、A2、および A3 は COMMON として指定されており、それと関連する呼び出しの結果の相違に注目してください。

表 15. COMMON プログラムを含むネスト構造の呼び出し階層

プログラム	呼び出すことができるプログラム	呼び出される可能性のあるプログラム
A	A1、A2、A3	なし
A1	A11、A12、A2、A3	A
A11	A111、A12、A2、A3	A1
A111	A12、A2、A3	A11
A12	A2、A3	A1、A11、A111
A2	A3	A、A1、A11、A111、A12、A3
A3	A2	A、A1、A11、A111、A12、A2

次のことに注意してください。

- A1 は COMMON ではなく、A2 に直接含まれていないので、A2 から A1 を呼び出すことはできません。
- A11 または A1 または A が、その PROGRAM-ID 段落に RECURSIVE 文節をもっていない限り、 A111 が A11 を呼び出すと再帰呼び出しになるので、A111 から A11 を呼び出すことはできません。
- A2 は COMMON なので、A1 は A2 を呼び出すことができます。
- A3 は COMMON なので、A1 は A3 を呼び出すことができます。

ネスト構造内の名前の有効範囲

ネスト構造内の名前には、**ローカル**と**グローバル**という 2 つのクラスがあります。このクラスは、名前を宣言したプログラムの有効範囲以外の場所で名前が識別されるかどうかを決めるものとなります。

ローカル名: GLOBAL と宣言しない限り、名前は必ずローカルになります (プログラム名を除く)。それらのローカル名は、それを宣言したプログラム以外からは見えず、アクセスできません。このことは、含んでいるプログラムと含まれているプログラムの両方に当てはまります。

グローバル名: GLOBAL 文節を使用してグローバルとして指定された名前は、それを宣言したプログラムとそのプログラムに直接または間接に含まれているすべてのプログラムから見え、アクセスすることができます。これにより、含まれているプログラムは、それを含んでいるプログラムから項目の名前を参照するだけで、共通のデータとファイルを共用することができます。

グローバル項目に従属する項目 (条件名と指標を含む) は、自動的にグローバルになります。

異なるプログラムでそれぞれの宣言を行う場合は、GLOBAL 文節を使って同じ名前を何度も宣言することができます。ネスト構造内の名前は、同一の包含構造のうちの異なるプログラムで同じ名前を使用することによって、マスキングされてしまう (隠されてしまう) ことがあるので注意してください。

名前宣言の検索: プログラム内で名前が参照されると、その名前の宣言の検索が行われます。検索は、まず参照を含むプログラム内で行われ、次にそれを含む**外側の**プログラムに向かって、一致が検出されるまで続行されます。検索は、次の順序で行われます。

1. そのプログラム内の宣言が最初に検索されます。
2. 一致が検出されない場合、それを含む外側のプログラム内のグローバル宣言**だけ**が、外側へ向かって順に検索されます。
3. 一致する最初の名前が検出された時点で検索は終了します。一致が検出されない場合は、エラーが存在します。

静的プロシージャー呼び出しと動的プログラム呼び出しの使用

以下の説明は、個別にコンパイルされたサブプログラムにだけ適用されます。ネストされたプログラムには適用されません。ネストされたプログラム構造内での呼び出しの詳細については 242 ページの『ネストされたプログラムの呼び出し』を参照してください。

ILE COBOL プログラムで静的プロシージャー呼び出しを使用したか、動的プログラム呼び出しを使用したかにより、バインディングの処理方法は異なります。静的プロシージャー呼び出しを使用して ILE COBOL サブプログラムを呼び出した場合、まずコンパイルしてモジュール・オブジェクトにしてから、コピーまたは参照によって、呼び出し ILE COBOL プログラムと同じプログラム・オブジェクトにバインドします。動的プログラム呼び出しを使用して ILE COBOL サブプログラムを呼び出す場合、ILE COBOL サブプログラムは別個のプログラム・オブジェクトとしてコンパイルおよびバインドする必要があります。バインディング・プロセスの詳細については、「*ILE 概念*」を参照してください。

静的プロシージャー呼び出しを使うと、動的プログラム呼び出しを使う場合よりパフォーマンスが向上します。

静的プロシージャー呼び出しを使って、ILE COBOL サブプログラムを呼び出す場合、そのサブプログラムはすでに活動化されています。それは、呼び出し側プログラムと同じプログラム・オブジェクトとしてバインドされており、呼び出し側 ILE COBOL プログラムから制御を受け取ってすぐに実行されるからです。

動的プログラム呼び出しを使用して ILE COBOL サブプログラムを呼び出す場合、それ以外の多くのことを実行してからでないと、呼び出し先 ILE COBOL プログラムを実行することができない場合があります。それには、以下のことが含まれます。

- 活動化しようとして呼び出し先 ILE COBOL プログラムが含まれる活動化グループが存在しない場合は、呼び出し先 ILE COBOL プログラムを活動化する前にまずその活動化グループを作成しなければなりません。
- 呼び出し先 ILE COBOL プログラムが事前に活動化されていない場合、実行する前にまず活動化する必要があります。呼び出し先 ILE COBOL プログラムを活動化するためには、そのプログラムに (直接または間接に) バインドされたサービス・プログラムを活動化する必要があります。活動化するには、次の機能を実行しなければなりません。
  - プログラム・オブジェクトまたはサービス・プログラムが必要とする静的データを固有に割り振る。

– 使用するサービス・プログラムのシンボリック・リンクを物理アドレスのリンクに変更する。

したがって、動的プログラム呼び出しは、最初に活動化グループで実行する前にまず活動化しなければならないので、静的プロシージャー呼び出しよりも遅くなります。

さらに、動的プログラム呼び出しと静的プロシージャー呼び出しでは、呼び出し側 ILE COBOL プログラムから呼び出し先 ILE COBOL プログラムに渡せるオペランドの数も異なります。動的プログラム呼び出しを使用する場合、最高 255 個のオペランドを渡すことができます。静的プロシージャー呼び出しを使用する場合、最高 400 個のオペランドを渡すことができます。

OMITTED として指定された引き数または操作記述子と関連していると指定された引き数は、静的プロシージャー呼び出しを使用することによってしか渡すことができません。それらの引き数は、動的プログラム呼び出しを使用して渡すことはできません。

CALL リテラルを使用した静的プロシージャー呼び出しの実行

CALL *リテラル* のステートメントを使用して、静的プロシージャー呼び出しを実行することができます (*リテラル* はサブプログラムの名前)。呼び出しを静的プロシージャー呼び出しとして指定するには、次の 3 とおりの方法があります。優先度の高いもの順になっています。

注: IN LIBRARY 句は、静的プロシージャー呼び出しと互換性がありません。

1. CALL ステートメントの LINKAGE 句を使用する。
   - CALL ステートメントで LINKAGE TYPE IS PROCEDURE を指定することによって、呼び出し先プログラムが静的プロシージャー呼び出しを使用して呼び出されるようにする。

```
       PROCEDURE DIVISION.
⋮
            CALL LINKAGE TYPE IS PROCEDURE literal-1
```

2. SPECIAL-NAMES 段落の LINKAGE TYPE 文節を使用する。
   - SPECIAL-NAMES 段落で LINKAGE TYPE IS PROCEDURE FOR *リテラル-1* を指定します (*リテラル-1* は呼び出し先 ILE COBOL プログラムの名前)。SPECIAL-NAMES 段落でリンケージがすでに定義されている場合、CALL ステートメントで LINKAGE TYPE 句を指定する必要はありません。

```
     ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
⋮
       SPECIAL-NAMES.
            LINKAGE TYPE IS PROCEDURE FOR literal-1.
⋮
       PROCEDURE DIVISION.
⋮
            CALL literal-1.
```

3. CRTCBLMOD および CRTBNDCBL のコマンドの LINKLIT パラメーターを使用するか、または関連した PROCESS ステートメントのオプションを使用する。
   - コンパイル時に、CRTCBLMOD および CRTBNDCBL コマンドの LINKLIT パラメーターで *PRC を指定して、ILE COBOL プログラムのすべての外部 CALL *リテラル-1* ステートメントに対して静的プロシージャー呼び出しが実行されるようにします。 CRTCBLMOD の LINKLIT パラメーターでリンケージがすでに定義されている場合は、SPECIAL-NAMES 段落の LINKAGE TYPE 文節や、CALL または CANCEL ステートメントの LINKAGE TYPE 句を指定する必要はありません。

     ```
     CRTCBLMOD MODULE(MYLIB/XMPLE1)
     SRCFILE(MYLIB/QCBLLESRC)  SRCMBR(XMPLE1)
     LINKLIT(*PRC)
     ```

   - CRTCBLMOD の LINKLIT パラメーターを使用してリンケージ・タイプを指定する場合、CALL ステートメントは次のようにコーディングします。

     ```
           PROCEDURE DIVISION.

     ⋮
                CALL literal-1.
     ```

CALL リテラルを使用した動的プログラム呼び出しの実行

CALL *リテラル* のステートメント (*リテラル* はサブプログラムの名前) または CALL *id* のステートメントを使用して動的プログラム呼び出しを実行することができます。 CALL *id* の詳細については 250 ページの『CALL id の使用』を参照してください。 CALL *リテラル* を使って、呼び出しを動的プログラム呼び出しとして指定するには、次の 3 とおりの方法があります。優先度の高いもの順になっています。

1. CALL ステートメントの LINKAGE 句を使用する。
   - CALL ステートメントで LINKAGE TYPE IS PROGRAM を指定することによって、呼び出し先プログラムが動的プログラム呼び出しを使用して呼び出されるようにする。

     ```
           PROCEDURE DIVISION.

     ⋮
                CALL LINKAGE TYPE IS PROGRAM literal-1
     ```

2. SPECIAL-NAMES 段落の LINKAGE TYPE 文節を使用する。
   - SPECIAL-NAMES 段落で LINKAGE TYPE IS PROGRAM FOR *リテラル-1* を指定します (*リテラル-1* は呼び出す ILE COBOL プログラムの名前)。 SPECIAL-NAMES 段落でリンケージがすでに定義されている場合、CALL ステートメントで LINKAGE TYPE 句を指定する必要はありません。

     ```
          ENVIRONMENT DIVISION.
            CONFIGURATION SECTION.

     ⋮
            SPECIAL-NAMES.
                 LINKAGE TYPE IS PROGRAM FOR literal-1.

     ⋮
            PROCEDURE DIVISION.
     ```

```
⋮
                CALL literal-1.
```

3. CRTCBLMOD および CRTBNDCBL のコマンドの LINKLIT パラメーターを使用するか、または関連した PROCESS ステートメントのオプションを使用する。
   - コンパイル時に、CRTCBLMOD および CRTBNDCBL コマンドの LINKLIT パラメーターで *PGM を指定して、ILE COBOL プログラムのすべての外部 CALL *リテラル-1* ステートメントに対して動的プログラム呼び出しが実行されるようにします。 CRTCBLMOD の LINKLIT パラメーターでリンケージがすでに定義されている場合は、SPECIAL-NAMES 段落の LINKAGE TYPE 文節や、CALL または CANCEL ステートメントの LINKAGE TYPE 句を指定する必要はありません。

     ```
     CRTCBLMOD MODULE(MYLIB/XMPLE1)
     SRCFILE(MYLIB/QCBLLESRC)  SRCMBR(XMPLE1)
     LINKLIT(*PGM)
     ```

   - CRTCBLMOD の LINKLIT パラメーターを使用してリンケージ・タイプを指定する場合、CALL ステートメントは次のようにコーディングします。

     ```
           PROCEDURE DIVISION.

     ⋮
                 CALL literal-1.
     ```

動的プログラム呼び出しは、サブプログラムを実行時に活動化します。次の場合には、動的呼び出しステートメントを使用してください。

- 保守作業を簡単にし、コードを再利用したい場合。

  サブプログラムの変更時には、静的に呼び出され、コピーを使ってバインドされているすべてのモジュール・オブジェクト (サービス・プログラムを除く) を、再度バインドしなければなりません。参照によってバインドされている場合は、サブプログラムとモジュール・オブジェクト間のインターフェースが変更されていなければ、再度バインドする必要はありません。変更されたサブプログラムが動的に呼び出される場合、再バインドの必要があるのはその変更されたサブプログラムだけです。したがって、動的呼び出しを使用すると、バインディングの量を最小にすることができるので、サブプログラムのコピーが保守しやすくなります。

- CALL *リテラル* で呼び出したサブプログラムが、あまり使用されないか、または大規模な場合。

  いくつかの条件においてのみサブプログラムが呼び出される場合、動的呼び出しによって必要な時だけサブプログラムを活動化することができます。

  サブプログラムが大規模かまたは数が多い場合、静的呼び出しを使うと、主記憶域で必要な作業セットのサイズが大きくなってしまう可能性があります。

CALL id の使用

CALL *id* を使用すると、ネストされた ILE COBOL プログラムを呼び出したり、プログラム・オブジェクトを呼び出したりすることができます (*id* はプロシージャー・ポインターではありません)。 ID の内容によって、ネストされたプログラムか

プログラム・オブジェクトのどちらかを呼び出すのかが実行時に決められます。 ID の内容が可視のネストされたプログラムの名前と一致する場合、呼び出しはネストされたプログラムになります。それ以外の場合は、ID の内容で指定される名前のプログラム・オブジェクトに対して動的プログラム呼び出しが実行されます。

CALL id で指定された IN LIBRARY 句は、呼び出しがプログラム・オブジェクトになるように強制します。

CALL ステートメントで最初に ID を使用するときに、CALL id (およびそれに関連する IN LIBRARY 項目) とオブジェクトを関連付けるオープン・ポインターが設定されます。

あるプログラムを指す ID を使用して呼び出しを実行した後、そのプログラムを削除または名前変更する場合、CANCEL ステートメントを使用することによって、その ID に関連付けられているオープン・ポインターをヌルにする必要があります。これにより、次に ID を使用してプログラム・オブジェクトを呼び出した場合、関連したオープン・ポインターが再び設定されます。

次の例は、CANCEL ステートメントを ID に適用する方法を示すものです。

```
MOVE "ABCD" TO IDENT-1.
CALL IDENT-1.
CANCEL IDENT-1.
```

CANCEL ステートメントを直接リテラル ″ABCD″ に適用する場合、IDENT-1 に関連づけられるオープン・ポインターはヌルにしない でください。その代わりに、CALL ステートメントで IDENT-1 を使用することにより、プログラム ABCD の呼び出しを続行することができます。

CALL id の値を変更し、その新しい値を使って呼び出しを実行すると、オープン・ポインターの値も変更されます。 オープン・ポインターの値は、それに関連した IN LIBRARY 項目によっても影響を受けます。それ以前の IDENT-1 への呼び出しとは異なるライブラリーを CALL に指定すると、オープン・ポインターはリセットされます。

CALL プロシージャー・ポインターの使用

CALL *プロシージャー・ポインター* のステートメントを使用して、静的プロシージャー呼び出し、または動的プログラム呼び出しを実行することができます。

CALL *プロシージャー・ポインター* のステートメントを使う前に、アドレス値にプロシージャー・ポインター・データ項目を設定しなければなりません。*プロシージャー・ポインター・データ項目*は、最も外側の COBOL プログラム (ILE プロシージャー)、別のコンパイル単位の ILE プロシージャー、またはプログラム・オブジェクトに設定することができます。*プロシージャー・ポインター・データ項目*を設定するには、形式 6 の SET ステートメントを使用します。

*プロシージャー・ポインター・データ項目*を ILE プロシージャーに設定するには、SET ステートメントで LINKAGE TYPE IS PROCEDURE を指定します。

*プロシージャー・ポインター・データ項目*をプログラム・オブジェクトに設定するには、SET ステートメントで LINKAGE TYPE IS PROGRAM を指定します。

SPECIAL-NAMES 段落の LINKAGE TYPE 文節または CRTCBLMOD および CRTBNDCBL の各コマンドの LINKLIT パラメーターを使用して、プロシージャー・ポインター・データ項目を設定するオブジェクトのタイプを決めることもできます。 SPECIAL-NAMES 段落の LINKAGE TYPE 文節または CRTCBLMOD および CRTBNDCBL の各コマンドの LINKLIT パラメーターを使用したリンケージ・タイプの設定の詳細については 240 ページの『呼び出し先のプログラムとプロシージャーのリンケージ・タイプの識別』を参照してください。

CALL *プロシージャー・ポインター* を使用して静的プロシージャー呼び出しを実行するには、SET ステートメントおよび CALL ステートメントを次のようにコーディングします。

```
        PROCEDURE DIVISION.
⋮
             SET procedure-pointer
                 TO ENTRY LINKAGE TYPE IS PROCEDURE literal-1.
⋮
             CALL procedure-pointer.
```

CALL *プロシージャー・ポインター* を使用して動的プログラム呼び出しを実行するには、SET ステートメントおよび CALL ステートメントを次のようにコーディングします。

```
        PROCEDURE DIVISION.
⋮
             SET procedure-pointer
                 TO ENTRY LINKAGE TYPE IS PROGRAM literal-1.
⋮
             CALL procedure-pointer.
```

再帰呼び出しの使用

以前の呼び出しがまだアクティブなときに、プログラムへの再帰再入を可能にするため、 PROGRAM-ID 段落に RECURSIVE 文節をコーディングします。 RECURSIVE 文節を使用してプログラムを再帰的プログラムにする例、および再帰的プログラムでローカル・ストレージ・セクションのデータ項目を使用する方法を、以下に示します。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL              MYLIB/FACTORIAL          ISERIES    06/02/15 17:25:51        ページ     2
                                ソ ー ス
  STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
     1     000100 IDENTIFICATION DIVISION.
    2      000200 PROGRAM-ID. FACTORIAL RECURSIVE.
           000300
    3      000400 ENVIRONMENT DIVISION.
    4      000500 CONFIGURATION SECTION.
    5      000600    SOURCE-COMPUTER. IBM-ISERIES.
    6      000700    OBJECT-COMPUTER. IBM-ISERIES.
           000800
    7      000900 DATA DIVISION.
    8      001000 WORKING-STORAGE SECTION.
    9      001100 01 NUMB PIC 9(4) VALUE 5.
   10      001200 01 FACT PIC 9(8) VALUE 0.
           001300
   11      001400 LOCAL-STORAGE SECTION.
   12      001500 01 NUM PIC 9(4).
           001600
   13      001700 PROCEDURE DIVISION.
   14      001800     MOVE NUMB TO NUM.
   15      001900     IF NUMB = 0
   16      002000         MOVE 1 TO FACT
           002100     ELSE
   17      002200         SUBTRACT 1 FROM NUMB
   18      002300         CALL "FACTORIAL"
   19      002400         MULTIPLY NUM BY FACT
           002500     END-IF.
   20      002600     DISPLAY NUM "! = " FACT.
   21      002700     GOBACK.
   22      002800 END PROGRAM FACTORIAL.
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 53. ある数の階乗を計算する再帰呼び出しの例

ILE COBOL プログラムからの戻り

呼び出し先 ILE COBOL プログラムから制御を戻すには、STOP RUN、EXIT PROGRAM、または GOBACK ステートメントを出します。

エラーの発生時またはプログラムの終了時に、呼び出し先プログラムから制御がどのように戻るかを判別するには、ILE COBOL プログラムがメイン・プログラムなのか、またはサブプログラムなのかが分かっていなければなりません。メイン・プログラムおよびサブプログラムの説明については 238 ページの『メイン・プログラムとサブプログラム』を参照してください。

メイン・プログラムからの戻り

メイン・プログラムから制御を戻すには、CONTINUE 句で STOP RUN、GOBACK、または EXIT PROGRAM のいずれかを使用します。 STOP RUN および GOBACK ステートメントは実行単位を終了し、制御はメイン・プログラムの呼び出し側に戻されます。 CONTINUE 句のない EXIT PROGRAM を使用してメイン・プログラムから制御を戻すことはできません。メイン・プログラムで CONTINUE 句のない EXIT PROGRAM が検出されると命令は実行されず、メイン・プログラムの次のステートメントから処理が続行されます。

*NEW 活動化グループからの戻り

*NEW 活動化グループで呼び出し先 ILE COBOL メイン・プログラムから、CONTINUE 句を指定した STOP RUN、GOBACK、または EXIT PROGRAM を実

行すると、制御が呼び出し側プログラムに戻った時点で活動化グループが終了します。活動化グループはすべてのファイルをクローズし、すべてのリソースをシステムに戻します。

活動化グループが終了した結果、呼び出し先 ILE COBOL プログラムは最初の状態に戻ります。

名前付き活動化グループからの戻り

名前付き活動化グループで、呼び出し先 ILE COBOL メイン・プログラムから、CONTINUE 句を指定した EXIT PROGRAM を実行すると、その活動化グループは活動状態のままで、制御が呼び出し側プログラムに戻ります。活動化グループのすべてのファイルとリソースは、最後に使用された状態のままです。

名前付き活動化グループで、呼び出し先 ILE COBOL メイン・プログラムから STOP RUN または GOBACK ステートメントを実行すると、活動化グループは制御が呼び出し側プログラムに戻った時点で終了されます。活動化グループはすべてのファイルをクローズし、すべてのリソースをシステムに戻します。

デフォルト (*DFTACTGRP) 活動化グループからの戻り

デフォルト (*DFTACTGRP) 活動化グループで、呼び出し先 ILE COBOL メイン・プログラムから STOP RUN または GOBACK ステートメントを実行すると、活動化グループは活動状態のままになり、制御は呼び出し側プログラムに戻ります。活動化グループで使用されたすべてのファイルとリソースは、最後に使用された状態のままです。

サブプログラムからの戻り

サブプログラムから制御を戻すには、サブプログラムを EXIT PROGRAM、GOBACK、または STOP RUN ステートメントを使用して終了させることができます。サブプログラムを EXIT PROGRAM または GOBACK ステートメントを使用して終了させた場合は、実行単位は終了されず、制御はそれを呼び出したプログラムに戻されます。 呼び出し先プログラムに続く実行可能ステートメントが存在しない場合、暗黙のうちに EXIT PROGRAM ステートメントが生成されます。サブプログラムを STOP RUN ステートメントを使用して終了させた場合、実行単位のプログラムのうち最も近い管理境界までのすべてのプログラムは終了し、制御はその管理境界の前のプログラムに戻されます。

サブプログラムは、EXIT PROGRAM または GOBACK で終了させた場合、通常**最後に使用された状態**のままです。次回にサブプログラムが実行単位で呼び出されても、内部値はその状態のままです (ただし、すべての PERFORM ステートメントが完了したと見なされて、初期値にリセットされた場合は除く)。これとは対照的に、メイン・プログラムの場合は呼び出されるたびに初期設定されます。ただし、次の 2 つの例外があります。

- 動的に呼び出されてから取り消されたサブプログラムは、次に呼び出された時に初期状態になります。
- PROGRAM-ID 段落に INITIAL 文節が指定されているプログラムは、呼び出されるたびに初期状態になります。

OPM COBOL/400 実行単位で定義された STOP RUN のセマンティクスの維持

STOP RUN ステートメントを OPM COBOL/400 実行単位でも互換性があるように動作させるには、ILE COBOL アプリケーションは特定の条件を使用して作成しなければなりません。これらの条件の説明については 236 ページの『COBOL 実行単位』を参照してください。

ILE COBOL プログラムからの戻りの例

次の例は、名前付き、*NEW、および *DFTACTGP の各活動化グループを組み合わせた時に、EXIT PROGRAM、STOP RUN、および GOBACK がどのように作用するかを示すものです。

図 54 は、プログラム A、B、C、D、および E を含んでいる活動化グループを示しています。 A は B および C を呼び出し、C は D および E を呼び出します。

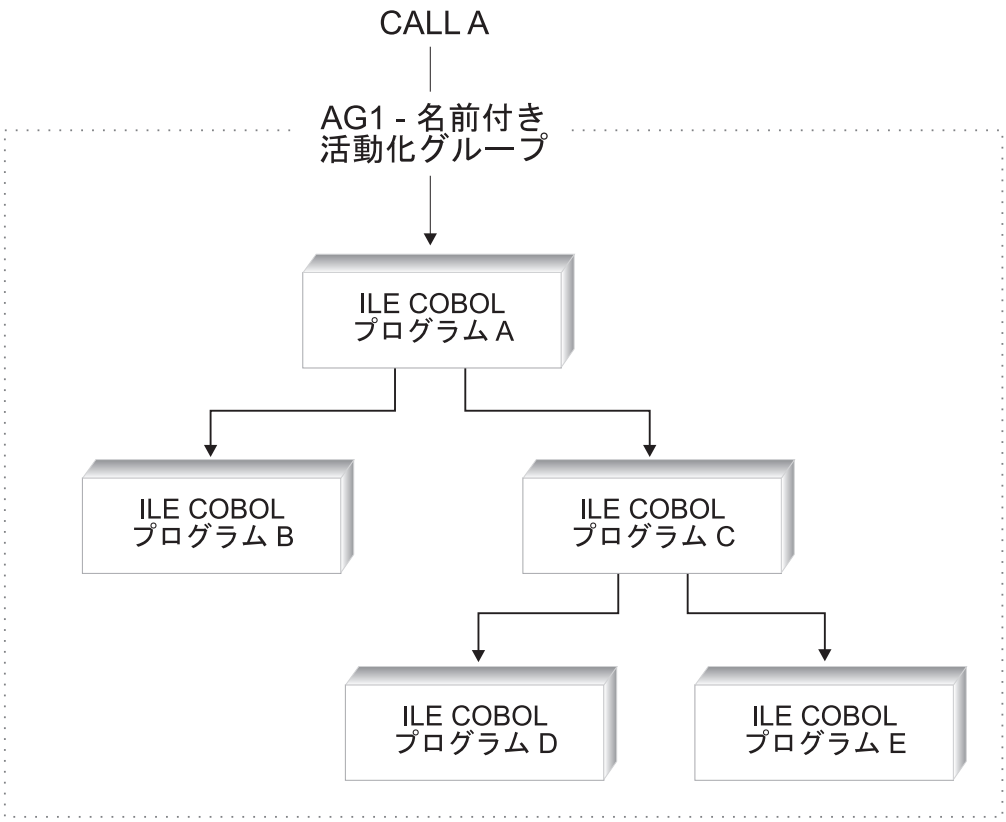


図 54. 1 つの名前付き活動化グループにおける EXIT PROGRAM、STOP RUN、および GOBACK の作用例

ステートメント	プログラム A	プログラム B	プログラム C	プログラム D	プログラム E
EXIT PROGRAM	1	4	4	2	2
STOP RUN	3	3	3	3	3
GOBACK	3	4	4	2	2

1 ステートメントはメイン・プログラム内にあるので、CONTINUE 句のない EXIT PROGRAM がコーディングされていても、処理は行われません。処理は、プログラムの次のステートメントから続行されます。活動化グループは活動状態のままで、CONTINUE 句を指定した EXIT PROGRAM は、制御をプログラム A の呼び出し側に戻します。活動化グループで使用されたすべてのファイルとリソースは、最後に使用された状態のままです。

2 活動化グループは活動状態のままで、制御はプログラム C に戻されます。活動化グループで使用されたすべてのファイルとリソースは、最後に使用された状態のままです。

3 活動化グループは終了し、制御はメイン・プログラムの呼び出し側に戻されます。活動化グループは、その活動化グループの範囲のすべてのファイルをクローズします。その活動化グループの範囲の保留中のコミット処理は、暗黙のうちにコミットされます。活動化グループに割り振られたすべてのリソースは、システムに戻されます。活動化グループが終了する結果、活動化グループ内で活動化されていたすべてのプログラムは、最初の状態に戻ります。

4 活動化グループは活動状態のままで、制御はプログラム A に戻されます。活動化グループで使用されたすべてのファイルとリソースは、最後に使用された状態のままです。

図 55 は、2 つの活動化グループを示しています。活動化グループ 1 にはプログラム A および B が含まれています。活動化グループ 2 にはプログラム C、D、および E が含まれています。A は B および C を呼び出します。 C は D および E を呼び出します。

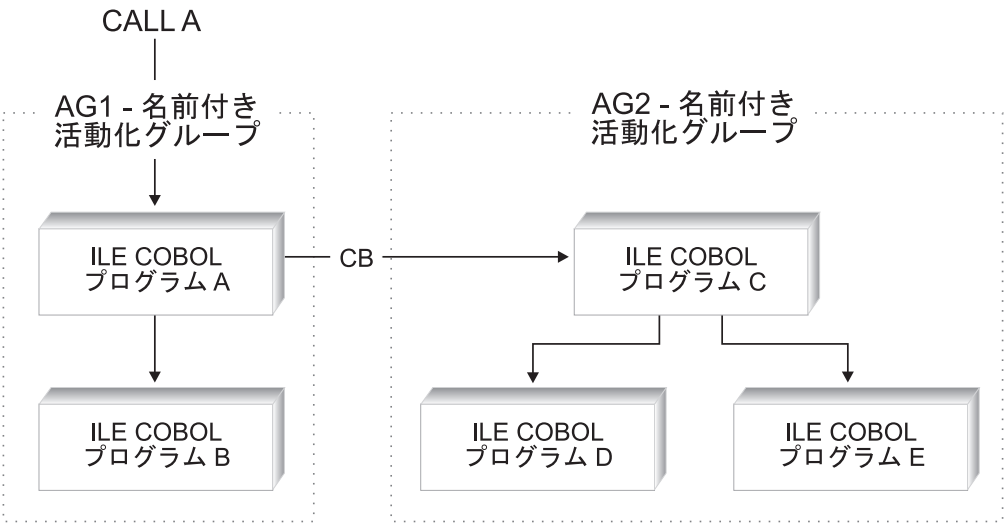


図 55. 2 つの名前付き活動化グループにおける EXIT PROGRAM、STOP RUN、および GOBACK の作用例

ステートメント	プログラム A	プログラム B	プログラム C	プログラム D	プログラム E
EXIT PROGRAM	1	5	1	2	2
STOP RUN	3	3	4	4	4
GOBACK	3	5	4	2	2

1 ステートメントはメイン・プログラムにあるので、CONTINUE 句のない EXIT PROGRAM ステートメントを使用しても、何も処理は行われません。処理は、プログラムの次のステートメントから続行されます。CONTINUE 句を指定した EXIT PROGRAM を使用すると、活動化グループは活動状態のままで、制御は呼び出し側プログラムまたはコマンドに戻ります。活動化グループで使用されたすべてのファイルとリソースは、最後に使用された状態のままです。

2 活動化グループは活動状態のままで、制御はプログラム C に戻されます。活動化グループで使用されたすべてのファイルとリソースは、最後に使用された状態のままです。

3 活動化グループは終了し、制御はメイン・プログラムの呼び出し側に戻されます。活動化グループは、その活動化グループの範囲のすべてのファイルをクローズします。その活動化グループの範囲の保留中のコミット処理は、暗黙のうちにコミットされます。活動化グループに割り振られたすべてのリソースは、システムに戻されます。活動化グループが終了する結果、活動化グループ内で活動化されていたすべてのプログラムは、最初の状態に戻ります。

4 活動化グループは終了し、制御はプログラム A に戻されます。活動化グループは、その活動化グループの範囲のすべてのファイルをクローズします。その活動化グループの範囲の保留中のコミット処理は、暗黙のうちにコミットされます。活動化グループに割り振られたすべてのリソースは、システムに戻されます。活動化グループが終了する結果、活動化グループ内で活動化されていたすべてのプログラムは、最初の状態に戻ります。

5 活動化グループは活動状態のままで、制御はプログラム A に戻されます。活動化グループで使用されたすべてのファイルとリソースは、最後に使用された状態のままです。

258 ページの図 56 は、2 つの名前付き活動化グループおよび 1 つの *NEW 活動化グループを示しています。活動化グループ 1 にはプログラム A および D が含まれています。活動化グループ 2 にはプログラム C および E が含まれています。*NEW 活動化グループにはプログラム B が含まれています。 A は B および C を呼び出します。C は D および E を呼び出します。

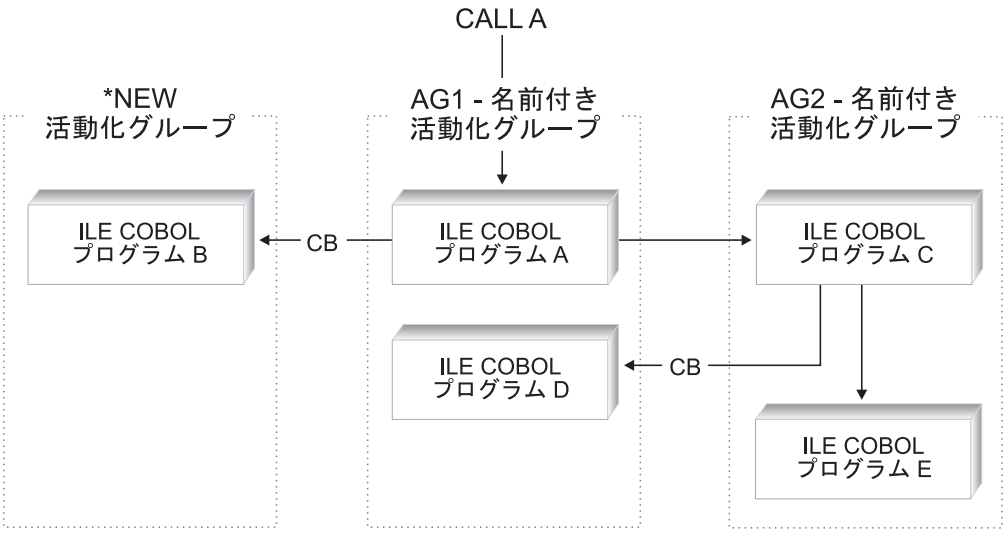


*図 56. 複数の *NEW および名前付き活動化グループにおける EXIT PROGRAM、STOP RUN、および GOBACK の作用例*

ステートメント	プログラム A	プログラム B	プログラム C	プログラム D	プログラム E
EXIT PROGRAM	1	5	1	2	2
STOP RUN	3	4	4	2	4
GOBACK	3	4	4	2	2

1 ステートメントはメイン・プログラムにあるので、CONTINUE 句のない EXIT PROGRAM ステートメントを使用しても、何も処理は行われません。処理は、プログラムの次のステートメントから続行されます。CONTINUE 句を指定した EXIT PROGRAM を使用すると、活動化グループは活動状態のままで、制御は呼び出し側プログラムまたはコマンドに戻ります。活動化グループで使用されたすべてのファイルとリソースは、最後に使用された状態のままです。

2 活動化グループは活動状態のままで、制御はプログラム C に戻されます。活動化グループで使用されたすべてのファイルとリソースは、最後に使用された状態のままです。

3 活動化グループは終了し、制御はメイン・プログラムの呼び出し側に戻されます。活動化グループは、その活動化グループの範囲のすべてのファイルをクローズします。その活動化グループの範囲の保留中のコミット処理は、暗黙のうちにコミットされます。活動化グループに割り振られたすべてのリソースは、システムに戻されます。活動化グループが終了する結果、活動化グループ内で活動化されていたすべてのプログラムは、最初の状態に戻ります。

4 活動化グループは終了し、制御はプログラム A に戻されます。活動化グループは、その活動化グループの範囲のすべてのファイルをクローズします。その活動化グループの範囲の保留中のコミット処理は、暗黙のうちにコミットされます。活動化グループに割り振られたすべてのリソースは、システム

に戻されます。活動化グループが終了する結果、活動化グループ内で活動化されていたすべてのプログラムは、最初の状態に戻ります。

5 ステートメントはメイン・プログラムにあるので、CONTINUE 句のない EXIT PROGRAM ステートメントを使用しても、何も処理は行われません。処理は、プログラムの次のステートメントから続行されます。

CONTINUE 句を指定した EXIT PROGRAM を使用すると、制御は呼び出し側プログラムまたはコマンドに戻ります。 *NEW 活動化グループでは、メイン・プログラムが呼び出し側に制御を戻す時点で活動化グループが終了します。活動化グループは、その活動化グループの範囲のすべてのファイルをクローズします。その活動化グループの範囲の保留中のコミット処理は、暗黙のうちにコミットされます。

活動化グループに割り振られたすべてのリソースは、システムに戻されます。活動化グループが終了する結果、活動化グループ内で活動化されていたすべてのプログラムは、最初の状態に戻ります。

図 57 は、名前付き活動化グループ、デフォルト活動化グループ、および *NEW 活動化グループの間の相互作用を示しています。活動化グループ 1 にはプログラム A および D が含まれています。 *DFTACTGP 活動化グループには OPM COBOL/400 プログラム C および E が含まれています。 *NEW 活動化グループにはプログラム B が含まれています。 A は B および C を呼び出します。C は D および E を呼び出します。

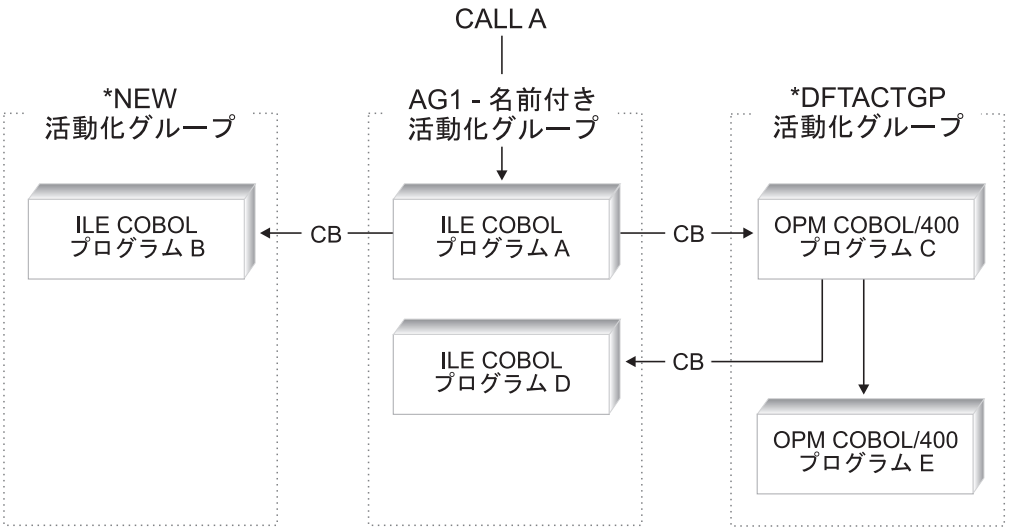


*図 57. *NEW、名前付き、および *DFTACTGP 活動化グループにおける EXIT PROGRAM、STOP RUN、および GOBACK の作用例*

ステートメント	プログラム A	プログラム B	プログラム C	プログラム D	プログラム E
EXIT PROGRAM	1	6	7	2	2
STOP RUN	3	4	5	2	5
GOBACK	3	4	5	2	2

1 ステートメントはメイン・プログラムにあるので、CONTINUE 句のない EXIT PROGRAM ステートメントを使用しても、何も処理は行われません。処理は、プログラムの次のステートメントから続行されます。CONTINUE 句を指定した EXIT PROGRAM を使用すると、活動化グループは活動状態のままで、制御は呼び出し側プログラムまたはコマンドに戻ります。活動化グループで使用されたすべてのファイルとリソースは、最後に使用された状態のままです。

2 活動化グループは活動状態のままで、制御はプログラム C に戻されます。活動化グループで使用されたすべてのファイルとリソースは、最後に使用された状態のままです。

3 活動化グループは終了し、制御はメイン・プログラムの呼び出し側に戻されます。活動化グループは、その活動化グループの範囲のすべてのファイルをクローズします。その活動化グループの範囲の保留中のコミット処理は、暗黙のうちにコミットされます。活動化グループに割り振られたすべてのリソースは、システムに戻されます。活動化グループが終了する結果、活動化グループ内で活動化されていたすべてのプログラムは、最初の状態に戻ります。

4 活動化グループは終了し、制御はプログラム A に戻されます。活動化グループは、その活動化グループの範囲のすべてのファイルをクローズします。その活動化グループの範囲の保留中のコミット処理は、暗黙のうちにコミットされます。活動化グループに割り振られたすべてのリソースは、システムに戻されます。活動化グループが終了する結果、活動化グループ内で活動化されていたすべてのプログラムは、最初の状態に戻ります。

5 活動化グループは活動状態のままで、制御はプログラム A に戻されます。プログラム C またはプログラム E によってオープンされたファイルはすべてクローズされます。プログラム C またはプログラム E によってオープンされたファイルに対する保留中のコミット処理は、暗黙のうちにコミットされます。プログラム C とプログラム E のストレージは解放されます。

6 ステートメントはメイン・プログラムにあるので、CONTINUE 句のない EXIT PROGRAM ステートメントを使用しても、何も処理は行われません。処理は、プログラムの次のステートメントから続行されます。

CONTINUE 句を指定した EXIT PROGRAM を使用すると、制御は呼び出し側プログラムまたはコマンドに戻ります。

*NEW 活動化グループでは、メイン・プログラムが呼び出し側に制御を戻す時点で活動化グループが終了します。活動化グループは、その活動化グループの範囲のすべてのファイルをクローズします。その活動化グループの範囲の保留中のコミット処理は、暗黙のうちにコミットされます。

活動化グループに割り振られたすべてのリソースは、システムに戻されます。活動化グループが終了する結果、活動化グループ内で活動化されていたすべてのプログラムは、最初の状態に戻ります。

7 ステートメントはメイン・プログラムにあるため、何も処理は行われません。処理は、プログラムの次のステートメントから続行されます。

戻りコード情報の受け渡し (RETURN-CODE 特殊レジスター)

RETURN-CODE 特殊レジスターを使用することによって、ILE COBOL プログラム間で戻りコードを受け渡しすることができます。 RETURN-CODE 特殊レジスターは、呼び出し先 ILE COBOL プログラムから戻る前に設定することができます。

ネストされたプログラムで使用する場合、RETURN-CODE 特殊レジスターは、最外部の ILE COBOL プログラムで暗黙のうちに GLOBAL として定義されます。 RETURN-CODE 特殊レジスターに加えられた変更は、モジュール・オブジェクト内のすべての ILE COBOL プログラムにグローバルに影響を与えます。

ILE COBOL プログラムからその呼び出し側に戻ると、RETURN-CODE 特殊レジスターの内容が、呼び出し側プログラムの RETURN-CODE 特殊レジスターに転送されます。

制御がメイン ILE COBOL プログラムからオペレーティング・システムに戻されると、RETURN-CODE 特殊レジスターの内容はユーザーの戻りコードとして戻されます。

プログラム間でのデータの受け渡しと共用

ILE COBOL プログラム間でデータの受け渡しと共用を行うには、以下のようないくつかの方法があります。

- データを GLOBAL として宣言して、ネストされたプログラムで使用することができます。
- CALL ステートメントの RETURNING 句を使用して、データを呼び出し側プログラムに戻すことができます。
- CALL ステートメント実行時に、BY REFERENCE、BY VALUE、または BY CONTENT により、呼び出し先プログラムにデータを渡すことができます。
- データを EXTERNAL と宣言すると、別々にコンパイルしたプログラムで共用することができます。 EXTERNAL データは、モジュール・オブジェクト内のネストされた ILE COBOL プログラムの間でも共用することができます。
- ファイルを EXTERNAL と宣言すると、別々にコンパイルされたプログラムで共用することができます。 EXTERNAL ファイルは、モジュール・オブジェクト内のネストされた ILE COBOL プログラムの間でも共用することができます。
- 動的に検索されるデータ項目のアドレスを渡したい場合は、ポインターを使用することができます。
- データは、データ域を使って受け渡しすることができます。

ローカル・データとグローバル・データの比較

ローカル・データとグローバル・データの概念が適用されるのは、ネストされたプログラムだけです。

ローカル・データとは、そのデータを宣言したプログラム内からしかアクセスできないデータです。ローカル・データは、それを宣言したプログラム以外からは見えずアクセスできません。これは、含まれたプログラムと含むプログラムの両方に適用されます。

明示的にグローバル・データとして宣言する場合以外は、すべてのデータはローカル・データと見なされます。

グローバル・データとは、そのデータを宣言したプログラムからアクセス可能であり、さらにそのデータを宣言したプログラムに直接または間接に含まれる他のネストされたプログラムからもアクセスできるデータです。

データ名、ファイル名、およびレコード名はグローバルとして宣言することができます。

データ名をグローバルとして宣言するには、データ名を宣言するのに使用したデータ記述項目またはそのデータ記述項目が従属する他の項目で GLOBAL 文節を指定します。

ファイル名をグローバルとして宣言するには、そのファイル名のファイル記述項目で GLOBAL 文節を指定します。

レコード名をグローバルとして宣言するには、レコード名を宣言するのに使用したレコード記述項目で GLOBAL 文節を指定するか、またはファイル・セクションのレコード記述項目の場合は、そのレコード記述項目が関連づけられたファイル名のファイル記述項目で GLOBAL 文節を指定します。

GLOBAL 文節の詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

CALL...BY REFERENCE、BY VALUE、または BY CONTENT を使用したデータの受け渡し

BY REFERENCE は、受け取った変数にサブプログラムが加えた変更を、呼び出し側プログラムから見ることができることを意味します。

BY CONTENT は、呼び出し側プログラムがリテラル または *id* の**内容**だけを渡すことを意味します。 CALL...BY CONTENT を使用すると、呼び出し先プログラムは、受け取ったパラメーターを変更しても、呼び出し側プログラムのリテラル または *id* の値は変更することができません。

BY VALUE は、呼び出し側プログラムが、送信項目の参照ではなくリテラル または *id* の値を受け渡すことを意味します。呼び出し先プログラムがパラメーターを変更できるのは、その呼び出し先プログラム内だけです。しかし、サブプログラムは送信項目の一時的なコピーにアクセスするだけなので、これらの変更は呼び出し側プログラムの引き数には影響を与えません。

データを渡すために BY REFERENCE、BY VALUE、または BY CONTENT のどれを使うかは、プログラムでデータをどのように処理するかによって決まります。

- 呼び出し側プログラムの CALL ステートメントの引き数の定義と、呼び出し先プログラムのパラメーターの定義が同じ記憶域を共用するようにしたい場合は、次のように指定します。

  ```
  CALL...BY REFERENCE identifier
  ```

  サブプログラムがパラメーターに対して変更を加えると、呼び出し側プログラムの引き数も変更されます。

- レコード域のアドレスを、呼び出し先プログラムに渡すには、次のように指定します。

  ```
  CALL...BY REFERENCE ADDRESS OF record-name
  ```

  サブプログラムは、指定したレコード名の ADDRESS OF 特殊レジスターを受け取ります。

  呼び出し側プログラムおよび呼び出し先プログラムのリンケージ・セクションに、レコード名をレベル 01 項目またはレベル 77 項目として定義しておかなければなりません。リンケージ・セクションのレコードごとに、それぞれ別個の ADDRESS OF 特殊レジスターが提供されます。
- データ項目のアドレスを、呼び出し先プログラムに渡すには、次のように指定します。

  ```
  CALL...BY CONTENT ADDRESS OF data-item-name
  ```
- 呼び出し側プログラムの CALL ステートメントの引き数の定義と、呼び出し先サブプログラムのパラメーターの定義が同じ記憶域を共用するようにしたくない場合は、次のように指定します。

  ```
  CALL...BY CONTENT identifier
  ```
- データを、BY VALUE パラメーターを要求する ILE プログラムに渡すには、次のように指定します。

  ```
  CALL...BY VALUE item
  ```
- さまざまな長さの整数値を渡すには、次のように指定します。

  ```
  CALL...BY VALUE integer-1 SIZE integer-2
  ```

  整数値は、長さが「整数-2」の 2 進数値として渡されます。 SIZE 句の指定はオプショナルです。指定されない場合、「整数-1」は 4 バイトの 2 進数として渡されます。
- 戻り値を伴う ILE C、C++、または RPG の関数を呼び出すには、次のように指定します。

  ```
  CALL...RETURNING identifier
  ```
- リテラルの値を呼び出し先プログラムに渡すには、次のように指定します。

  ```
  CALL...BY CONTENT literal
  ```

  呼び出し先プログラムでは、リテラルの値を変更できません。
- データ項目の長さを渡すには、次のように指定します。

  ```
  CALL...BY CONTENT LENGTH OF identifier
  ```

  呼び出し側プログラムは、その LENGTH OF 特殊レジスターに基づいて *id* の長さを渡します。
- データ項目とデータ項目の長さの両方をサブプログラムに渡すには、BY REFERENCE と BY CONTENT を組み合わせて指定します。以下に例を示します。

  ```
  CALL 'ERRPROC' USING BY REFERENCE A
       BY CONTENT LENGTH OF A.
  ```
- 呼び出し先プログラムが対応する引数を受け取らないようにする場合、または呼び出し先プログラムが引数のデフォルト値を使用するようにする場合は、

CALL...BY REFERENCE または CALL...BY CONTENT ステートメントで、省略する各データ項目の代わりに OMITTED 句を指定します。たとえば、以下のようになります。

```
CALL...BY REFERENCE OMITTED
CALL...BY CONTENT OMITTED OMITTED
```

呼び出し先プログラムでは、CEETSTA API を使用することによって、指定されたパラメーターが OMITTED かそうでないかを判別することができます。

- **操作記述子**でデータ項目を渡すには、SPECIAL-NAMES 段落に LINKAGE TYPE IS PRC...USING ALL DESCRIBED 文節を指定します。その後、CALL...BY REFERENCE、CALL...BY CONTENT、または CALL...BY VALUE ステートメントを使用してデータを渡します。呼び出し先 ILE プロシージャーで渡されるデータ項目の形式を正確に予測できない場合でも、操作記述子を使用すれば、呼び出し先 ILE プロシージャーに記述情報が提供されます。異なる ILE 言語で作成された呼び出し先 ILE プロシージャーによって必要とされる場合、および ILE バインド可能 API によって必要とされる場合には、操作記述子を使用します。操作記述子の詳細については、「*ILE 概念*」を参照してください。たとえば、以下のようになります。

```
SPECIAL-NAMES.  LINKAGE TYPE PRC FOR 'ERRPROC'
                USING ALL DESCRIBED.

⋮
CALL 'ERRPROC' USING BY REFERENCE identifier.
```

または

```
SPECIAL-NAMES.  LINKAGE TYPE PRC FOR 'ERRPROC'
                USING ALL DESCRIBED.

⋮
CALL 'ERRPROC' USING BY CONTENT identifier.
```

呼び出し側プログラムのデータ項目は、直接または間接に呼び出すすべてのプログラムのリンケージ・セクションに記述することができます。この場合、これらの項目のストレージは、最外部の呼び出し側プログラムに割り振られます。

呼び出し側プログラムの引き数の記述

呼び出し側プログラムから渡されるデータを**引き数**といいます。呼び出し側プログラムにおいて、データ部に引き数が記述される方法は、他のデータ項目がデータ部に記述される方法と同じです。引き数がリンケージ・セクションにあるのでない限り、ストレージが呼び出し側プログラムのこれらの項目に割り振られます。ファイル中のデータを参照する場合、データの参照時にそのファイルはオープンされていなければなりません。引き数を渡すには、CALL ステートメントの USING 文節をコーディングします。

呼び出し先プログラムでのパラメーターの記述

呼び出し先プログラムで受け取られるデータを**パラメーター**といいます。呼び出し先プログラムでは、パラメーターはリンケージ・セクションに記述されます。パラメーターを受け取るには、PROCEDURE-DIVISION ヘッダーの後ろに USING 文節をコーディングします。

呼び出し先プログラムのリンケージ・セクションの作成: 呼び出し側プログラムから渡される内容を理解しておき、呼び出し先プログラムがそれを受け入れることができるようリンケージ・セクションを設定することが必要です。呼び出し先プログラムには、データを渡すために CALL ステートメントのどの文節 (BY REFERENCE、BY VALUE または BY CONTENT) を使用するかは問題ではありません。 どの場合でも、呼び出し先プログラムで、受け取ったデータを記述しなければなりません。 この記述はリンケージ・セクションで行われます。

呼び出し先プログラムの *id* リストのデータ名 の数は、呼び出し側プログラムの *id* リストのデータ名 の数より大きくすることはできません。それらの位置は 1 対 1 で対応しています。すなわち、呼び出し側プログラムの最初の *id* は、呼び出し先プログラムの最初の *id* に渡され、2 番目以降も同様になります。 ILE COBOL コンパイラーでは、引き数の数とパラメーターの数についても、または引き数と対応するパラメーター間でのタイプとサイズについても、その整合性を強制することはありません。

引き数の数とパラメーターの数が矛盾している場合、実行時例外が生じる可能性があります。動的プログラム呼び出しの場合、引き数の数がパラメーターの数よりも多い時に、CALL ステートメントを実行しようとすると、呼び出し側プログラムで実行時例外が生成されます。 CALL ステートメントに ON EXCEPTION 句を指定すると、この例外を捕らえることができます。

引き数の数がパラメーターの数より小さい場合、CALL ステートメントを実行しても呼び出し側プログラムに実行時例外は生成されません。しかし、呼び出し先プログラムで、指定されていないパラメーターにアクセスしようとすると、ポインター例外が生成されます。

引き数が BY VALUE により渡された場合、サブプログラムの PROCEDURE DIVISION ヘッダーには次のことが指定されていなければなりません。

```
PROCEDURE DIVISION USING BY VALUE DATA-ITEM.
```

引き数が BY REFERENCE または BY CONTENT により渡された場合、PROCEDURE DIVISION ヘッダーでは、引き数がどのように渡されたかを指定する必要はありません。ヘッダーは以下のどちらかでもかまいません。

```
PROCEDURE DIVISION USING DATA-ITEM
```

または

```
PROCEDURE DIVISION USING BY REFERENCE DATA-ITEM
```

渡されるデータのグループ化

プログラム間で渡したいデータ項目をすべてグループ化し、それらを 1 つのレベル 01 項目にまとめることができます。これを行うと、プログラム間で単一のレベル 01 レコードを渡すことができます。この方法の例として、 266 ページの図 58 を参照してください。

レコードの不一致が発生する可能性をさらに低くするには、レベル 01 レコードをコピー・メンバーに入れて、両方のプログラムにコピーします。 (すなわち、呼び出し側プログラムの WORKING-STORAGE SECTION と、呼び出し先プログラムの LINKAGE SECTION にコピーします。)

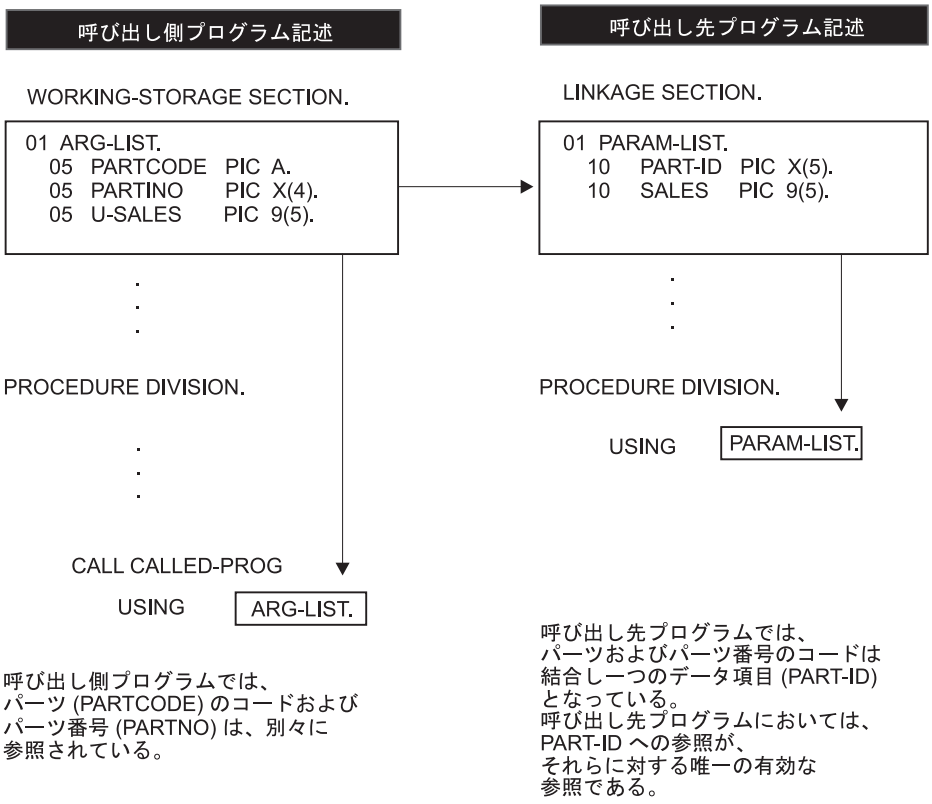


図 58. サブプログラム・リンケージにおける共通データ項目

EXTERNAL データの共用

個別にコンパイルされた ILE COBOL プログラム (一連の ILE COBOL ソース・プログラムのプログラムを含む) では、EXTERNAL 文節を使うことによってデータ項目を共用することができます。この EXTERNAL データは、弱エクスポートとして処理されます。強エクスポートと弱エクスポートについては、「*ILE 概念*」を参照してください。

EXTERNAL 文節は、ILE COBOL プログラムの WORKING-STORAGE SECTION の 01 レベルのデータ記述で指定し、次の規則が適用されます。

1. EXTERNAL グループ項目に従属する項目も EXTERNAL です。
2. データ項目に使用した名前は、同一プログラム内の他の EXTERNAL 項目の名前として使用することはできません。
3. VALUE 文節を、グループ項目、または従属項目である EXTERNAL に対して指定することはできません。
4. EXTERNAL データを初期設定することはできません。 EXTERNAL データの実行時の初期値は未定義です。ユーザーのアプリケーションで EXTERNAL データ項目の初期設定が必要ならば、メイン・プログラム中でそれらを明示的に初期設定することが推奨されます。

実行単位内の ILE COBOL プログラムのうち、項目のデータ記述が、その項目を含むプログラムと同じであるものからは、そのデータ項目にアクセスして処理をすることができます。たとえば、プログラム A に次のデータ記述がある場合、

```
01 EXT-ITEM1          PIC 99 EXTERNAL.
```

プログラム B の WORKING-STORAGE SECTION 内のデータ記述を同じにすることによって、そのデータ項目にアクセスすることができます。

同じ名前付き EXTERNAL データ項目を宣言しているすべてのモジュール・オブジェクトにおいては、その EXTERNAL データ項目のサイズはすべて同じでなければなりません。 名前が同じでサイズが異なる EXTERNAL データ項目がコンパイル単位の複数の ILE COBOL プログラム内で宣言されている場合、その中の最長のサイズがそのデータ項目の長さとして使用されます。

さらに、名前が同じでサイズが異なる EXTERNAL データ項目が、同じ活動化グループで活動化される複数のプログラム・オブジェクトまたはサービス・プログラムで宣言されており、後から活動化されたプログラム・オブジェクトまたはサービス・プログラム内に、名前が同じでサイズが大きい EXTERNAL データ項目がある場合、後から活動化されたプログラム・オブジェクトまたはサービス・プログラムの活動化は失敗します。

複数の ILE COBOL プログラムで宣言された同じ名前のデータ項目におけるタイプの整合性が ILE COBOL コンパイラーによって強制されることはありません。これらのデータ項目の使用を整合性のあるものにするのは、プログラマーの責任です。

EXTERNAL データ項目にアクセスするどのプログラムも、この項目の値を変更する可能性があることに注意してください。この文節は、保護しなければならないデータ項目には使用しないでください。

EXTERNAL ファイルの共用

ファイルに EXTERNAL 文節を使用することによって、実行単位内で個別にコンパイルされたプログラムは、共通のファイルにアクセスすることができます。これらの EXTERNAL ファイルは、弱エクスポートとして処理されます。強エクスポートと弱エクスポートについては、「*ILE 概念*」を参照してください。

EXTERNAL ファイルが複数の ILE COBOL プログラムで定義されている場合、ILE COBOL プログラムのどれか 1 つによってオープンされると、すべてのプログラムからアクセス可能になります。同様に、いずれかのプログラムが EXTERNAL ファイルをクローズすると、どのプログラムからもアクセスできないようになります。

複数のモジュール・オブジェクトの複数の ILE COBOL プログラムの場合、EXTERNAL ファイルを宣言している ILE COBOL プログラムが最初に呼び出された時点で実行時整合性検査が行われ、そのモジュール・オブジェクトの定義と、その他のモジュール・オブジェクトにおいてすでに呼び出されている ILE COBOL プログラムの定義とで整合性があるかどうかが検査されます。矛盾が検出された場合は、実行時例外メッセージが出されます。

269 ページの図 59 の例は、EXTERNAL ファイルを使用した場合の利点について示すものです。

- メイン・プログラムは、入出力ステートメントがない場合でも、ファイルのレコード域を参照することができます。
- 各サブプログラムは、OPEN または READ などの単一の入出力機能を制御することができます。
- 各プログラムはファイルにアクセスすることができます。

次の表に、 269 ページの図 59 の例におけるプログラム (またはサブプログラム) の名前と、その機能を示します。

表 16. EXTERNAL ファイルを使用した入出力例のプログラム名

NAME	機能
EF1MAIN	これはメイン・プログラムです。すべてのサブプログラムを呼び出した後、レコード域の内容を検査します。
EF1OPENO	このプログラムは出力用に外部ファイルをオープンし、ファイル状況コードを検査します。
EF1WRITE	このプログラムはレコードを外部ファイルに書き込み、ファイル状況コードを検査します。
EF1OPENI	このプログラムは入力用に外部ファイルをオープンし、ファイル状況コードを検査します。
EF1READ	このプログラムは外部ファイルからレコードを読み取り、ファイル状況コードを検査します。
EF1CLOSE	このプログラムは外部ファイルをクローズし、ファイル状況コードを検査します。

このサンプル・プログラムは、WORKING-STORAGE SECTION のデータ項目に対しても EXTERNAL 文節を使用します。この項目は、ファイル状況コードを検査するために使用されます。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/EXTLFL         ISERIES1    06/02/15 13:11:39        ページ    2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.  EF1MAIN.
          000300*
          000400* これは外部ファイル処理を制御する
          000500* メイン・プログラムです。
          000600*
          000700
    3     000800 ENVIRONMENT DIVISION.
    4     000900 INPUT-OUTPUT SECTION.
    5     001000 FILE-CONTROL.
    6     001100      SELECT EF1
    7     001200         ASSIGN TO DISK-EFILE1
    8     001300         FILE STATUS IS EFS1
    9     001400         ORGANIZATION IS SEQUENTIAL.
          001500
   10     001600 DATA DIVISION.
   11     001700 FILE SECTION.
   12     001800 FD  EF1 IS EXTERNAL
          001900           RECORD CONTAINS 80 CHARACTERS.
   13     002000 01  EF-RECORD-1.
   14     002100     05  EF-ITEM-1     PIC X(80).
          002200
   15     002300 WORKING-STORAGE SECTION.
   16     002400 01  EFS1              PIC 99 EXTERNAL.
          002500
   17     002600 PROCEDURE DIVISION.
          002700 EF1MAIN-PROGRAM SECTION.
          002800 MAINLINE.
   18     002900      CALL "EF1OPENO"
   19     003000      CALL "EF1WRITE"
   20     003100      CALL "EF1CLOSE"
   21     003200      CALL "EF1OPENI"
   22     003300      CALL "EF1READ"
   23     003400      IF EF-RECORD-1 = "First Record" THEN
   24     003500         DISPLAY "First record correct"
          003600      ELSE
   25     003700         DISPLAY "First record incorrect"
   26     003800         DISPLAY "Expected:  First Record"
   27     003900         DISPLAY "Found:    " EF-RECORD-1
          004000      END-IF
   28     004100      CALL "EF1CLOSE"
   29     004200      GOBACK.
   30     004300 END PROGRAM EF1MAIN.
          004400
          004600
                       * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 59. EXTERNAL ファイルを使用した入出力 (1/6)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/EXTLFL          ISERIES1   06/02/15 13:11:39        ページ     5
                                    ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     004700 IDENTIFICATION DIVISION.
    2     004800 PROGRAM-ID.  EF1OPENO.
          004900*
          005000* このプログラムは出力用に外部ファイルをオープンします。
          005100*
          005200
    3     005300 ENVIRONMENT DIVISION.
    4     005400 INPUT-OUTPUT SECTION.
    5     005500 FILE-CONTROL.
    6     005600      SELECT EF1
    7     005700         ASSIGN TO DISK-EFILE1
    8     005800         FILE STATUS IS EFS1
    9     005900         ORGANIZATION IS SEQUENTIAL.
          006000
   10     006100 DATA DIVISION.
   11     006200 FILE SECTION.
   12     006300 FD  EF1 IS EXTERNAL
          006400           RECORD CONTAINS 80 CHARACTERS.
   13     006500 01  EF-RECORD-1.
   14     006600     05  EF-ITEM-1     PIC X(80).
          006700
   15     006800 WORKING-STORAGE SECTION.
   16     006900 01  EFS1              PIC 99 EXTERNAL.
          007000
   17     007100 PROCEDURE DIVISION.
          007200 EF1OPENO-PROGRAM SECTION.
          007300 MAINLINE.
   18     007400      OPEN OUTPUT EF1
   19     007500      IF EFS1 NOT = 0 THEN
   20     007600         DISPLAY "File Status " EFS1 " on OPEN OUTPUT"
   21     007700         STOP RUN
          007800      END-IF
   22     007900      GOBACK.
   23     008000 END PROGRAM EF1OPENO.
          008100
          008300
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 59. EXTERNAL ファイルを使用した入出力 (2/6)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/EXTLFL          ISERIES1   06/02/15 13:11:39        ページ    8
                                    ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     008400 IDENTIFICATION DIVISION.
    2     008500 PROGRAM-ID.  EF1WRITE.
          008600*
          008700* このプログラムはレコードを外部ファイルに書き込みます。
          008800*
          008900
    3     009000 ENVIRONMENT DIVISION.
    4     009100 INPUT-OUTPUT SECTION.
    5     009200 FILE-CONTROL.
    6     009300      SELECT EF1
    7     009400         ASSIGN TO DISK-EFILE1
    8     009500         FILE STATUS IS EFS1
    9     009600         ORGANIZATION IS SEQUENTIAL.
          009700
   10     009800 DATA DIVISION.
   11     009900 FILE SECTION.
   12     010000 FD  EF1 IS EXTERNAL
          010100            RECORD CONTAINS 80 CHARACTERS.
   13     010200 01  EF-RECORD-1.
   14     010300     05  EF-ITEM-1     PIC X(80).
          010400
   15     010500 WORKING-STORAGE SECTION.
   16     010600 01  EFS1              PIC 99 EXTERNAL.
          010700
   17     010800 PROCEDURE DIVISION.
          010900 EF1WRITE-PROGRAM SECTION.
          011000 MAINLINE.
   18     011100      MOVE "First record" TO EF-RECORD-1
   19     011200      WRITE EF-RECORD-1
   20     011300      IF EFS1 NOT = 0 THEN
   21     011400         DISPLAY "File Status " EFS1 " on WRITE"
   22     011500         STOP RUN
          011600      END-IF
   23     011700      GOBACK.
   24     011800 END PROGRAM EF1WRITE.
          011900
          012100
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 59. EXTERNAL ファイルを使用した入出力 (3/6)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/EXTLFL          ISERIES1   06/02/15 13:11:39        ページ   11
                                    ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     012200 IDENTIFICATION DIVISION.
    2     012300 PROGRAM-ID.  EF1OPENI.
          012400*
          012500* このプログラムは入力用に外部ファイルをオープンします。
          012600*
          012700
    3     012800 ENVIRONMENT DIVISION.
    4     012900 INPUT-OUTPUT SECTION.
    5     013000 FILE-CONTROL.
    6     013100      SELECT EF1
    7     013200         ASSIGN TO DISK-EFILE1
    8     013300         FILE STATUS IS EFS1
    9     013400         ORGANIZATION IS SEQUENTIAL.
          013500
   10     013600 DATA DIVISION.
   11     013700 FILE SECTION.
   12     013800 FD  EF1 IS EXTERNAL
          013900           RECORD CONTAINS 80 CHARACTERS.
   13     014000 01  EF-RECORD-1.
   14     014100     05  EF-ITEM-1     PIC X(80).
          014200
   15     014300 WORKING-STORAGE SECTION.
   16     014400 01  EFS1              PIC 99 EXTERNAL.
          014500
   17     014600 PROCEDURE DIVISION.
          014700 EF1OPENI-PROGRAM SECTION.
          014800 MAINLINE.
   18     014900      OPEN INPUT EF1
   19     015000      IF EFS1 NOT = 0 THEN
   20     015100         DISPLAY "File Status " EFS1 " on OPEN INPUT"
   21     015200         STOP RUN
          015300      END-IF
   22     015400      GOBACK.
   23     015500 END PROGRAM EF1OPENI.
          015600
          015800
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 59. EXTERNAL ファイルを使用した入出力 (4/6)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/EXTLFL          ISERIES1   06/02/15 13:11:39        ページ   14
                                   ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     015900 IDENTIFICATION DIVISION.
    2     016000 PROGRAM-ID.  EF1READ.
          016100*
          016200* このプログラムは外部ファイルからレコードを読み取ります。
          016300*
          016400
    3     016500 ENVIRONMENT DIVISION.
    4     016600 INPUT-OUTPUT SECTION.
    5     016700 FILE-CONTROL.
    6     016800      SELECT EF1
    7     016900         ASSIGN TO DISK-EFILE1
    8     017000         FILE STATUS IS EFS1
    9     017100         ORGANIZATION IS SEQUENTIAL.
          017200
   10     017300 DATA DIVISION.
   11     017400 FILE SECTION.
   12     017500 FD  EF1 IS EXTERNAL
          017600            RECORD CONTAINS 80 CHARACTERS.
   13     017700 01  EF-RECORD-1.
   14     017800     05  EF-ITEM-1     PIC X(80).
          017900
   15     018000 WORKING-STORAGE SECTION.
   16     018100 01  EFS1              PIC 99 EXTERNAL.
          018200
   17     018300 PROCEDURE DIVISION.
          018400 EF1READ-PROGRAM SECTION.
          018500 MAINLINE.
   18     018600      READ EF1
   19     018700      IF EFS1 NOT = 0 THEN
   20     018800         DISPLAY "File Status " EFS1 " on READ"
   21     018900         STOP RUN
          019000      END-IF
   22     019100      GOBACK.
   23     019200 END PROGRAM EF1READ.
          019300
          019500
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 59. EXTERNAL ファイルを使用した入出力 (5/6)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/EXTLFL         ISERIES1   06/02/15 13:11:39        ページ   17
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     019600 IDENTIFICATION DIVISION.
    2     019700 PROGRAM-ID.  EF1CLOSE.
          019800*
          019900* このプログラムは外部ファイルからレコードを読み取ります。
          020000*
          020100
    3     020200 ENVIRONMENT DIVISION.
    4     020300 INPUT-OUTPUT SECTION.
    5     020400 FILE-CONTROL.
    6     020500      SELECT EF1
    7     020600         ASSIGN TO DISK-EFILE1
    8     020700         FILE STATUS IS EFS1
    9     020800         ORGANIZATION IS SEQUENTIAL.
          020900
   10     021000 DATA DIVISION.
   11     021100 FILE SECTION.
   12     021200 FD  EF1 IS EXTERNAL
          021300           RECORD CONTAINS 80 CHARACTERS.
   13     021400 01  EF-RECORD-1.
   14     021500     05  EF-ITEM-1     PIC X(80).
          021600
   15     021700 WORKING-STORAGE SECTION.
   16     021800 01  EFS1              PIC 99 EXTERNAL.
          021900
   17     022000 PROCEDURE DIVISION.
          022100 EF1CLOSE-PROGRAM SECTION.
          022200 MAINLINE.
   18     022300      CLOSE EF1
   19     022400      IF EFS1 NOT = 0 THEN
   20     022500         DISPLAY "File Status " EFS1 " on CLOSE"
   21     022600         STOP RUN
          022700      END-IF
   22     022800      GOBACK.
   23     022900 END PROGRAM EF1CLOSE.
          023000
          023100
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 59. EXTERNAL ファイルを使用した入出力 (6/6)

ポインターを使用したデータの受け渡し

動的に検索されるデータ項目のアドレスをやりとりしたい場合は、ILE COBOL プログラム内でポインターを使用することができます。

ILE COBOL プログラムにおけるポインターの使用方法については 377 ページの『第 14 章 ILE COBOL プログラムでのポインターの使用』を参照してください。

データ域を使用したデータの受け渡し

データ域とは、ジョブ内またはジョブ間のプログラム間で、変数値などのデータを渡すために使用される IBM i オブジェクトのことです。データ域は、あるプログラムまたはジョブで使用する前に、そのプログラムに対して宣言したり作成することができます。データ域の作成方法や宣言方法の詳細については、「*CL プログラミング*」を参照してください。

ローカル・データ域の使用

ローカル・データ域を使用すると、ジョブ内のプログラムの相互間で任意の情報を渡すことができます。この情報は、略式メッセージのようなフリー・フォームのデータにしたり、構造化またはフォーマット設定されたフィールドにしたりすることができます。

内部および外部浮動小数点データ項目は、ローカル・データ域を使用して渡すことができます。 DISPLAY ステートメントを使用してローカル・データ域に書き込まれた内部浮動小数点数は、外部浮動小数点数に変換されます。

システムは、各ジョブごとにローカル・データ域を自動的に作成します。ローカル・データ域は、ILE COBOL プログラムの外部に 1024 バイトの域として定義されます。

ジョブが発行されると、発行するジョブのローカル・データ域が、発行されるジョブのローカル・データ域にコピーされます。発行ジョブがない場合、ローカル・データ域はブランクに初期設定されます。

ILE COBOL プログラムは、ACCEPT および DISPLAY ステートメントによってジョブのローカル・データ域にアクセスすることができます。その際には、環境名 LOCAL-DATA に対応する簡略名を使用します。

各ジョブごとにローカル・データ域が 1 つのみ対応しています。複数のワークステーションが 1 つのジョブによって獲得されている場合でも、そのジョブのローカル・データ域は 1 つだけです。各ワークステーションごとにローカル・データ域があるわけではありません。

自分で作成したデータ域の使用

自分でデータ域を作成し、それを使用することによって、プログラム間でデータを渡すこともできます。この情報は、略式メッセージのようなフリー・フォームのデータにしたり、構造化またはフォーマット設定されたフィールドにしたりすることができます。その作成時には、ライブラリーとデータ域の名前を指定します。

ローカル・データ域の形式とは違い、それらのデータ域にアクセスするには、ACCEPT および DISPLAY ステートメントのデータ域形式を使用します。 FOR 句を使用すれば、データ域の名前を指定できます。オプショナルとして、IN LIBRARY 句を指定して、データ域が存在する IBM i ライブラリーを指定することもできます。 IN LIBRARY 句を指定しない場合、ライブラリーはデフォルトにより *LIBL になります。

作成したデータ域に DISPLAY ステートメントを使用してデータを書き込むと、データがそのデータ域に書き込まれる前に、そのデータ域はシステムによって LEAR (Lock Exclusive Allow Read) ロックでロックされます。データ域のロックがそれ以外であるなら、LEAR ロックは適用されず、データ域には書き込まれません。WITH LOCK 句を指定することによって、表示操作の完了後にデータ域をロックしておくことができます。

ACCEPT ステートメントを使用して、作成したデータ域からデータを検索すると、システムは、LSRD (Lock Shared for Read) ロックを適用して、データ域の読み取り中にデータ域が変更されないようにします。読み取りが完了すると、LSRD ロックが除去され、WITH LOCK 句が指定されている場合はデータ域が LEAR ロックでロックされます。

ACCEPT ステートメントと DISPLAY ステートメントのいずれにおいても、WITH LOCK 句が指定されていないなら、そのステートメントより前の LEAR ロックは除去されます。

ILE COBOL では、COBOL 実行単位 (活動化グループ) が活動状態の場合、LEAR ロックは 1 つのデータ域に対して 1 つだけかけられます。活動化グループの終了時に、ロックされているデータ域があるなら、そのロックは除去されます。

以下に示すような理由で ON EXCEPTION 条件が存在する場合があります。

- FOR 句で指定されたデータ域が、以下の状態である。
  - 見つからない。
  - データ域への権限がない。
  - 直前の活動化グループまたは別のジョブでデータ域がロックされている。
- AT 位置が、以下の状態である。
  - 1 より小さいか、データ域の長さよりも大きい。

内部および外部浮動小数点データ項目は、データ域を使用して渡すことができます。 DISPLAY ステートメントを使用してデータ域に書き込まれた内部浮動小数点数値は、外部浮動小数点に変換されます。

ILE COBOL では、10 進数 (*DEC)、文字 (*CHAR)、論理 (*LGL)、および DDM (*DDM) データ域がサポートされています。データ域の種類には関係なく、情報はデータ域との間で左そろえでやり取りされます。 10 進数データ域または論理データ域の参照時に AT 位置を指定する場合には、1 でなければなりません。

10 進数データ域との間では、データはパック形式でやり取りされます。指定した数の総桁数および 10 進数桁数に基づいて、10 進数データ域が作成されます。 10 進数データ域にアクセスする ILE COBOL プログラムで、これと同じ桁数を宣言する必要があります。たとえば、以下のようになります。

- データ域を作成する CL コマンドは、以下のとおりです。

```
CRTDTAARA DTAARA(QGPL/DECDATA) TYPE(*DEC) LEN(5 2)
```

- ILE COBOL プログラムのうち、データ域にアクセスする部分は、以下のとおりです。

```
WORKING-STORAGE SECTION.
01 data-value.
   05 returned-packed1  pic s9(3)v9(2) packed-decimal.
PROCEDURE DIVISION.
      move 345.67 to returned-packed1.
      DISPLAY  data-value UPON data-area
         FOR "DECDATA" LIBRARY "QGPL".
      ACCEPT   data-value FROM data-area
         FOR "DECDATA" LIBRARY "QGPL".
```

プログラム初期設定パラメーター (PIP) データ域の使用

PIP データ域は、事前開始ジョブによって使用されます。一般に、事前開始ジョブとは、ICF の下のリモート・システムからのジョブであり、始動しておき、呼び出すまで作動可能状態にしておくジョブのことです。

事前開始ジョブを使用すると、呼び出したプログラムがジョブの始動処理が行われる間に待機する必要がなくなります。プログラムが実際に開始する前に、ジョブの始動処理が実行されています。ジョブはすでに始動されているので、事前開始ジョブを使うことによって、プログラムの始動要求が受け取られるとすぐにプログラムが始動されます。

ILE COBOL プログラムは、ACCEPT ステートメントによってジョブの PIP データ域にアクセスすることができます。その際には、関数名 PIP-DATA に対応する簡略名を使用します。

PIP データ域は、2000 バイトの英数字項目で、呼び出し側プログラムから受け取ったパラメーターが入れられています。このデータ域により、プログラム初期設定パラメーター (事前開始ジョブでないなら標準 COBOL パラメーターにより提供されるもの) が提供されます。

PIP データ域にアクセスするには、形式 5 の ACCEPT ステートメントを使います。これは、ローカル・データ域から読み取るために形式 4 の ACCEPT ステートメントを使うのと同じ方法です。 ILE COBOL を使用して PIP データ域を更新することはできませんので、ご注意ください。構文の詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

事前開始ジョブおよび PIP データ域の詳細については、「*CL プログラミング*」を参照してください。

EXIT PROGRAM、STOP RUN、GOBACK、および CANCEL を内部ファイルで使用した場合の効果

次のステートメントを使用すると、ファイルの状態がそれぞれ以下のように変わります。

- EXIT PROGRAM ステートメントは、実行単位のファイルの状況を変更することはありません。ただし、次の場合は例外です。
  - EXIT PROGRAM を発行する ILE COBOL プログラムに INITIAL 属性がある場合。 INITIAL 属性がある場合、そのプログラムで定義されているすべての内部ファイルはクローズされます。
  - AND CONTINUE RUN UNIT 句の指定された EXIT PROGRAM ステートメントを使用するサブプログラムも、最後に使用した状態のままです。その場合、メイン・プログラムから呼び出し側に制御が戻り、次いで *NEW 活動化グループは終了して、活動化グループ内の全ファイルはクローズされます。
- STOP RUN ステートメントは、最も近い制御境界でプログラムの呼び出し側に制御を戻します。これがハード管理境界である場合は、活動化グループ (実行単位) が終了し、その活動化グループの範囲のすべてのファイルをクローズします。
- メイン・プログラムから出された GOBACK ステートメント (これは常にハード制御境界にある) の動作は、STOP RUN ステートメントと同じです。サブプログラムから出された GOBACK ステートメントの動作は、EXIT PROGRAM ステートメントと同じです。サブプログラムから出された GOBACK ステートメントが実行単位のファイルの状況を変更することはありません。ただし、GOBACK を出した ILE COBOL プログラムに INITIAL 属性がある場合を除きます。 INITIAL 属性がある場合、そのプログラムで定義されているすべての内部ファイルはクローズされます。
- CANCEL ステートメントにより、内部ファイルに関する情報を含むストレージはリセットされます。プログラムに CANCEL ステートメントの処理時にオープンされていた内部ファイルがある場合、それらの内部ファイルはプログラム取り消し時にクローズされます。プログラムは、再オープンしない限りそのファイルを

使用できなくなります。一度取り消したプログラムを再度呼び出した場合、プログラムはファイルをクローズしているものと見なします。プログラムがファイルをオープンすると、そのファイルに対する新しいリンケージが確立されます。

ILE COBOL プログラムの取り消し

INITIAL 属性のないサブプログラムを EXIT PROGRAM または GOBACK を使用して終了すると、最後に使用された状態のままになります。 AND CONTINUE RUN UNIT 句の指定された EXIT PROGRAM ステートメントを使用するサブプログラムも、最後に使用した状態のままです。 実行単位の中で次にサブプログラムを呼び出すと、内部値はそのままです。ただし、PERFORM ステートメントはリセットされます。

サブプログラムの内部値を再呼び出し前の初期状態にリセットするには、サブプログラムを取り消さなければなりません。サブプログラムを取り消すことにより、次の呼び出し時にサブプログラムは初期状態になります。

別の ILE COBOL プログラムからの取り消し

ILE COBOL では、CANCEL ステートメントを使用してサブプログラムを取り消します。 CANCEL ステートメントを実行するには、サブプログラムはその取り消しを行うプログラムと同じ活動化グループになければなりません。

呼び出し先サブプログラムに CANCEL ステートメントを実行することにより、このサブプログラムとプログラムとの間の論理的接続はなくなります。サブプログラムによって記述されている EXTERNAL データ・レコードと EXTERNAL ファイルのデータ項目の内容は、サブプログラムが取り消されても変更されません。同じサブプログラム名の実行単位の中で、後ほど CALL ステートメントが実行された場合、このサブプログラムは初期状態になります。

呼び出し先サブプログラムに CANCEL ステートメントを含めることができます。しかし、それを呼び出すプログラムまたは呼び出し階層で上位にあるプログラムを直接または間接に取り消す CANCEL ステートメントを含めることはできません。呼び出し先サブプログラムがそれを呼び出すプログラムを取り消そうとしても、そのサブプログラムの CANCEL ステートメントは無視されます。

CANCEL ステートメントに指定するプログラムは、呼び出されてからまだ制御を呼び出し側プログラムへ戻していないプログラムを参照することはできません。呼び出された後に制御が戻されたプログラムは、同じ活動化グループのものなら、取り消すことができます。たとえば、以下のようになります。

```
A calls B and B calls C          When A receives control,
                                 it can cancel C.
A calls B and A calls C          When C receives control,
                                 it can cancel B.
```

注: 複数の ILE COBOL プログラムを含むプログラム・オブジェクトを取り消す場合は、そのプログラム・オブジェクトの PEP に関連した ILE COBOL プログラムだけが取り消されます。

CANCEL ステートメントの詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

別の言語からの取り消し

静的プロシージャー呼び出しを使用して取り消しプロシージャーを呼び出すことにより、最外部の ILE COBOL プログラムを ILE RPG、ILE C、および ILE CL から取り消すことができます。取り消しプロシージャーの名前は、最外部の ILE COBOL プログラムの名前から取られたもので、それに `_reset` を追加したものになります。

ILE COBOL プログラムは、OPM COBOL/400 プログラムまたは OPM RPG/400 プログラムからは取り消せません。

ILE COBOL プログラムを取り消すのに、リソース・レクラメーション (RCLSRC) CL コマンドは使用しないでください。 RCLRSC を ILE 活動化グループ内で出すと、例外が発生します。 RCLRSC コマンドの詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「プログラミング」カテゴリーの中の『*CL および API*』セクションを参照してください。

第 10 章 COBOL および eBusiness ワールド

この章では、eBusiness ソリューションの一部として ILE COBOL を使用する方法について説明します。この章に含まれる情報は、次のとおりです。

- 『COBOL および XML』
- 『COBOL および MQSeries』
- 282 ページの『COBOL および Java プログラム』

COBOL および XML

Extensible Markup Language (XML) は、World Wide Web Consortium (W3C) によって開発された SGML のサブセットです。XML の目的は、現在の HTML で可能な方法で、汎用 SGML による Web 上におけるサービスの提供、受け取り、処理を可能にすることです。XML の設計目的は、インプリメンテーションの容易さだけでなく、SGML および HTML との相互運用性です。XML は、データ・ストアと入出力の両方の機構として使用することができます。

XML 整形式パーサーが ILE COBOL に追加されました。詳しくは、311 ページの『第 11 章 XML 入力の処理』を参照してください。

XML の詳細については、`http://www.w3.org/XML` を参照してください。

IBM では、XML と COBOL のプログラムを統合するために使用できる、2 つのツールを開発しました。 IBM の XML for C++ パーサー (XML4C) は、C++ の移植可能なサブセットで書かれた XML パーサーの妥当性検査を行います。これは、3 つの共用ライブラリー (2 つがコード用で、1 つがデータ用) からなり、XML 文書の解析、生成、操作、および妥当性検査のためのクラスを提供します。

C、COBOL、および RPG などの手順型言語でパーサーを使用するためには、RPG に対する XML インターフェースも必要です。このラッパー・インターフェースによって、i5/OS 上の ILE C、RPG、および COBOL プログラムは、XML パーサーとのインターフェースが可能となります。

これらの製品は、どちらも常に進化しています。これらは alphaWorks を介して入手でき、これによって、IBM の新しい『アルファ・コード』テクノロジーへの直接アクセスと早期の採用が可能になります。アルファ・コードをダウンロードして、IBM の研究者および開発者とのオンライン・ディスカッションに参加することができます。これらのアルファ・テクノロジーに関する最新情報は、ハードウェアとソフトウェアの要件を含めて、`http://www.alphaWorks.ibm.com/` を参照してください。

COBOL および MQSeries

IBM MQSeries メッセージング・プロダクトは、データをメッセージとして送受信することにより、ビジネス・アプリケーションが異なるプラットフォーム間で情報を交換するのを援助することにより、アプリケーションの統合を可能にします。このプロダクトは、ネットワーク・インターフェースを管理し、メッセージの送達が

「1 回のみ」になるよう保証し、通信プロトコルを処理し、使用可能なリソースに動的に作業負荷を配分し、システム問題発生後のリカバリーを処理し、プログラムを移植可能にするための援助を行います。 MQSeries は、35 を超えるプラットフォームで使用可能です。

MQSeries for iSeries V5.2 を使用すると、ILE COBOL プログラムは同じメッセージング製品を使用して、同じプラットフォームまたは異なるプラットフォームにある他のプログラムと通信することができます。ハードウェア要件およびソフトウェア要件を含め、MQSeries for iSeries V5.2 についての詳細は、IBM Systems Software Information Center を参照してください。

MQSeries API を使用する i5/OS COBOL アプリケーションのいくつかの例については、MQSeries の文書を調べてください。 Information Center で、「**プログラミング**」>「**WebSphere MQ**」をクリックします。

COBOL および Java プログラム

Java ネイティブ・インターフェース (JNI) によって、Java 仮想マシン (JVM) 内の Java コードは、COBOL、RPG、C、C++、およびアセンブラーなど、他のプログラム言語で作成されたアプリケーションおよびライブラリーとの相互操作が可能になります。この章では、COBOL と Java プログラムが、JNI を使用して一緒に作動する方法について説明します。

COBOL プログラムと Java プログラムを結合するために IBM Toolbox for Java の以下のコンポーネントも使用することできます。

- IBM Toolbox for Java 内の ProgramCall クラスは、IBM i ホスト・システム・プログラム呼び出しドライバーを使用して、IBM i プログラム・オブジェクトを呼び出します。
- PCML (Program Call Markup Language) は、XML (Extensible Markup Language) に基づくタグ言語です。 COBOL コマンドで PGMINFO と INFOSTMF のパラメーターを指定することによって、 Java アプリケーションから呼び出す ILE COBOL プログラムの入力と出力のパラメーターを完全に記述するタグを生成できます。IBM Toolbox for Java には CML を解釈し、プログラムを呼び出し、さらに System i マシンから戻されるデータの検索を単純化するアプリケーション・プログラミング・インターフェース (API) が含まれています。

上記の方法の詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「プログラミング」カテゴリーの中の『*Java*』セクションを参照してください。

システム要件

COBOL と Java のプログラムを統合するには、以下の要件を考慮してください。

- IBM Qshell インタープリターは、IBM i の無料のオプションです (5722-SS1、オプション 30)。
- IBM Developer Kit for Java (5722-JV1) は WebSphere Development Studio for i5/OS のコンポーネントで、インストール時に指定できます。
- IBM の Java 2 ソフトウェア開発キット (J2SDK) の標準版 v1.2.2 は、WebSphere Development Studio for i5/OS と一緒に出荷されます。 これと異なる

Java 開発キットを選択する場合、Java コードのすべての部分の機能を保証するには、バージョン 1.2 以上でなければなりません。

- IBM Toolbox for Java のクラスを使用する場合には、PCML も含めて、IBM Toolbox for Java (5722-JC1) が必要です。これは、WebSphere Development Studio for i5/OS コンポーネントと一緒に提供され、インストール時に指定することができます。

COBOL および PCML

COBOL プログラムの入力と出力のパラメーターを記述する、プログラム呼び出しマークアップ言語 (PCML) ソース・ファイルを使用して、 COBOL プログラムを Java アプリケーションから呼び出すことができます。 Java アプリケーションは、PCML ソース・ファイルを参照して、 ProgramCallDocument オブジェクトを作成することによって、PCML を使用できます。 Java で PCML を使用する方法の詳細については、 http://www.ibm.com/systems/i/infocenter/ の i5/OS Information Center で、**『プログラミング』->『Java』->『IBM Toolbox for Java』->『プログラム呼び出しマークアップ言語』**を参照してください。 PCML は、COBOL 形式と Java 形式間のデータ・タイプ変換を行います。

PGMINFO コマンド・パラメーターまたは PGMINFO PROCESS オプションを指定すると、ILE COBOL コンパイラーはご使用の COBOL プログラム用に PCML を生成します。 PCML は、ストリーム・ファイルに生成するか、ご使用の COBOL モジュールの一部にすることができます。 PCML がご使用の COBOL モジュールの一部である場合、QBNRPII API を使用して、ご使用のモジュールを含むプログラムまたはサービス・プログラムから検索できます。

PCML をストリーム・ファイルに生成するには、INFOSTMF コンパイラー・パラメーターと一緒に PGMINFO(*PCML) コマンド・パラメーターを指定することで、生成されたファイルを受け取る統合ファイル・システムの出力ファイルの名前を指定します。

PCML を直接 COBOL モジュールに生成するには、PGMINFO(*PCML *MODULE) コマンド・パラメーターを指定するか、またはPGMINFO(PCML MODULE) PROCESS オプションを指定します。

INFOSTMF パラメーターと一緒に PGMINFO(*PCML *ALL) コマンド・パラメーターを指定することにより、または PGMINFO(*PCML) および INFOSTMF コマンド・パラメーターを指定して、PGMINFO(PCML MODULE) PROCESS オプションを指定することにより、PCML をストリーム・ファイルおよび COBOL モジュールの両方で生成できます。

PROCESS オプションの PGMINFO(NOPGMINFO) を指定すると、PGMINFO(*PCML) コマンド・パラメーターを指定したとしても PCML は生成されません。

284 ページの表 17 は、PCML での COBOL データ・タイプに対するサポートを示しています。

表 17. *COBOL* データ・タイプおよび対応する *PCML* サポート

COBOL データ・タイプ	COBOL の形式	PCML でサポートされている	PCML データ・タイプ	長さ	精度	カウント
文字	X(n)	はい	文字	n		
	A(n)	はい	文字	n		
	X(n) OCCURS DEPENDING ON M	はい	構造体			m
	A(n) OCCURS DEPENDING ON m	はい	構造体			m
数字	9(n) DISPLAY	はい	ゾーン 10 進数	n	0	
	S9(n-p)V9(p) DISPLAY	はい	ゾーン 10 進数	n	p	
	9(n-p)V9(p) PACKED-DECIMAL 注 3 を参照	はい	パック 10 進数	n	p	
	S9(n-p)V9(p) PACKED-DECIMAL 注 3 を参照	はい	パック 10 進数	n	p	
	9(4) BINARY 注 1、2 を参照	はい	整数	2	15	
	9(4) COMP-5	はい	整数	2	16	
	S9(4) BINARY S9(4) COMP-5 注 1、2 を参照	はい	整数	2	15	
	9(9) BINARY 注 1、2 を参照	はい	整数	4	31	
	9(9) COMP-5	はい	整数	4	32	
	S9(9) BINARY S9(9) COMP-5 注 1、2 を参照	はい	整数	4	31	
	S9(18) BINARY S9(18) COMP-5 注 1、2 を参照	はい	整数	8	63	
	9(18) BINARY 注 1、2 を参照	はい	整数	8	63	
	9(18) COMP-5	サポートなし。	整数	8	64	
	9(n)V9(p) BINARY S9(n)V9(p) BINARY 9(n)V9(p) COMP-5 S9(n)V9(p) COMP-5	サポートなし。				
	USAGE COMP-1	はい	float	4		
	USAGE COMP-2	はい	float	8		
UCS2	N(n) 注 4 を参照	はい	UCS-2/グラフィック	n		
	N(n) OCCURS DEPENDING ON m 注 4 を参照	はい	構造体			m

表 17. COBOL データ・タイプおよび対応する PCML サポート (続き)

グラフィック	G(n)	はい	UCS-2/グラフィック	n		
	G(n) OCCURS DEPENDING ON M	はい	構造体			m
指標	USAGE INDEX	はい	整数	4	31	
ブール	1	サポートなし。				
日付	FORMAT DATE	サポートなし。				
時刻	FORMAT TIME	サポートなし。				
タイム・スタンプ	FORMAT TIMESTAMP	サポートなし。				
ポインター	USAGE POINTER	サポートなし。				
プロシージャー・ポインター	PROCEDURE POINTER	サポートなし。				

注:

1. BINARY データ項目の切り捨てを減らすには、PROCESS ステートメントで NOSTDTRUNC を指定します。BINARY データ項目を渡すときには、必ず NOSTDTRUNC を指定する必要があります。
2. PCML では、BINARY、COMP-4、COMPUTATIONAL-4 は整数にマップされます。
3. PACKED-DECIMAL、COMP-3、COMPUTATIONAL-3、COMP、および COMPUTATIONAL は同等のものであり、同じ PCML にマップされます (COMPASBIN PROCESS オプションを指定しない限り。詳しくは 68 ページの『PROCESS ステートメントのオプション』を参照)。
4. USAGE NATIONAL を指定した場合、あるいは USAGE を指定せずに、NATIONAL コンパイラー・オプションを指定した場合は、PIC N は国別 (UCS2) 項目になります。それ以外の場合は、USAGE DISPLAY-1 (DBCS) が指定されたことになります。

PCML は、COBOL プログラムの、Procedure Division の USING と GIVING/RETURNING の句の内容、および LINKAGE セクションの内容に基づいて、生成されます。PROCEDURE DIVISION ヘッダーの USING 句で指定するすべてのパラメーターに対して、PCML が生成されます。このヘッダーの GIVING/RETURNING 句で指定するパラメーターに対して、PCML が生成されます。GIVING/RETURNING 項目が 4 バイトの符号付き 2 進整数でない場合は、エラーが出ます。 生成された PCML で「inputoutput」として定義された、USING 句で指定した項目は、情報を呼び出し側プログラムに戻すために使用できます。TYPE 文節で定義されている項目は、PCML エラーを受け取ります。 呼び出しプログラム (JAVA プログラムなど) が、RETURN-CODE 特殊レジスターの内容を見るには、PROCEDURE DIVISION ヘッダーの USING 句で、RETURN-CODE 特殊レジスターを指定する必要があります。OCCURS DEPENDING ON (ODO) のサブジェクト・データ項目に対して PCML が正しく生成されるためには、 ODO のオブジェクト・データ項目が linkage section で定義され、 PROCEDURE DIVISION ヘッダーの USING 句でパラメーターとして指定されていなければなりません。

名前変更または再定義された項目には PCML は生成されません。

CRTCBLMOD を使用し、サービス・プログラムを作成する場合は、ProgramCallDocument クラスの setPath(String) メソッドを使用して、Java コードでサービス・プログラムを指定してください。例を以下に示します。

```
AS400 as400;
ProgramCallDocument pcd;
String path = "/QSYS.LIB/MYLIB.LIB/MYSRVPGM.SRVPGM";
as400 = new AS400 ();
pcd = new ProgramCallDocument (as400, "myModule");
pcd.setPath ("MYFUNCTION", path);
pcd.setValue ("MYFUNCTION.PARM1", "abc");
rc = pcd.callProgram("MYFUNCTION");
```

CRTCBLMOD を使用し、サービス・プログラム以外のプログラムを作成する場合は、PCML からエントリー・ポイント属性を除去する必要があります。この属性は、サービス・プログラムを呼び出す場合にのみ必要だからです。

例:

以下に、COBOL ソース・プログラムの例、およびこのプログラムに対して生成された対応する PCML を示します。

```
 5722WDS V5R4M0  060210 LN  IBM ILE COBOL               TESTLIB/MYPCML          ISERIES1   06/02/15 12:09:25       ページ    2
                                 ソース
STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
   1     000100 IDENTIFICATION DIVISION.
   2     000200 PROGRAM-ID.     MYPGM4.
         000300
   3     000400 DATA DIVISION.
   4     000500 WORKING-STORAGE SECTION.
   5     000600 01 RETN-VAL PIC 9(8) USAGE COMP-4.
         000700
   6     000800 LINKAGE SECTION.
   7     000900 01 PARM-LIST.
   8     001000    05 EMPL OCCURS 5 TIMES.
   9     001100      10 NAMES      PIC A(20).
  10     001200      10 ADDRESSES  PIC X(60).
  11     001300      10 PHN-NUM    PIC 9(11) DISPLAY.
  12     001400    05 NUM-1A     PIC S9(5)V9(3)  PACKED-DECIMAL.
  13     001500    05 NUM-2A     PIC 9(5)V9(3)   COMP.
  14     001600    05 TAB-NUM-3A PIC S9(5)V9(3)  COMP OCCURS 10 TIMES.
  15     001700    05 NUM-4A     PIC 9(5)V9(3)   COMP-3.
  16     001800    05 NUM-5A     PIC S9(5)V9(3)  COMP-3.
  17     001900    05 NUM-6A     PIC 9(4)        BINARY.
  18     002000    05 NUM-7A                     COMP-1.
  19     002100    05 NUM-8A                     COMP-2.
  20     002200    05 INTLNAME   PIC N(10)       NATIONAL.
         002300
         002400*****************************************************************
         002500* Test PCML for arrays of basic supported types.
         002600*****************************************************************
  21     002700 PROCEDURE DIVISION USING BY REFERENCE PARM-LIST
         002800     GIVING    RETN-VAL.
         002900 MAIN-LINE.
  22     003000     MOVE 1 TO RETN-VAL.
  23     003100     DISPLAY "THIS PGM TO BE CALLED BY A JAVA PGM".
  24     003200     STOP RUN.
                      *****  ソース仕様の終わり  *****
```

図 60. PCML ソース・プログラム

以下に、CRTBNDCBL コマンドに、PGMINFO(*PCML) と INFOSTMF('/dirname/mypgm4.pcml') のオプションを指定してプログラムをコンパイルしたときに生成される PCML の例を示します。

```
<pcml version="4.0">

   <!-- COBOL program: MYPCML  -->
   <!-- created: 02/03/21 12:09:25 -->
   <!-- source: TESTLIB/QCBLLESRC(MYPCML) -->
   <programname="MYPCML" path="/QSYS.LIB/TESTLIB.LIB/MYPCML.PGM" returnvalue="integer">
```

```
      <struct name="PARM-LIST" usage="inputoutput">

         <struct name="EMPL" usage="inherit" count="5">

            <data name="NAMES" type="char" length="20" usage="inherit">
            <data name="ADDRESSES" type="char" length="60" usage="inherit">
            <data name="PHN-NUM" type="zoned" length="11" precision="0" usage="inherit">
         </struct>
         <data name="NUM-1A" type="packed" length="8" precision="3" usage="inherit">
         <data name="NUM-2A" type="packed" length="8" precision="3" usage="inherit">
         <data name="TAB-NUM-3A" type="packed" length="8" precision="3" count="10"
usage="inherit">
         <data name="NUM-4A" type="packed" length="8" precision="3" usage="inherit">
         <data name="NUM-5A" type="packed" length="8" precision="3" usage="inherit">
         <data name="NUM-6A" type="int" length="2" precision="16" usage="inherit">
         <data name="NUM-7A" type="float" length="4" usage="inherit">
         <data name="NUM-8A" type="float" length="8" usage="inherit">
         <data name="INTLNAME" type="char" length="10" chartype="twobyte" ccsid="13488" usage="inherit">
      </struct>
      <data name="RETN-VAL" type="int" length="4" precision="32" passby="value"
usage="output">
   </program></pcml>
```

COBOL および JNI

Java プログラムからの COBOL プログラムの呼び出し

Java プログラムから COBOL プログラムを呼び出すには、以下のステップを実行してください。

- 『COBOL プログラムのコーディング』
- 292 ページの『COBOL モジュールの作成』
- 293 ページの『サービス・プログラムの作成』
- 294 ページの『Java プログラムのコーディング』
- 295 ページの『Java プログラムのコンパイル』.

COBOL プログラムのコーディング: このセクションでは、Java プログラムによって呼び出される COBOL プログラムをコーディングする方法について説明します。ガイドラインを 2 つの COBOL プログラム例で示します。後のセクションで、これらの COBOL プログラムと対話する 2 つの Java プログラムを示します。

COBOL プログラムを Java プログラムで呼び出す場合には、以下のようにしてください。

1. PROCESS ステートメントの NOMONOPRC (大文字小文字を区別する名前に備えて) およびオプション THREAD(SERIALIZE) を使用します。 COBOL プログラムは、Java プログラムから呼び出されると、Java スレッド内で実行します。バイナリー・データ項目の内容を保存するには、 NOSTDTRUNC Process オプションを指定します。
2. 以下から構成される名前を使用して COBOL プログラムを識別してください。
   - 接頭部 Java_
   - マングル済み完全修飾クラス名
   - 下線 (_) 区切り文字
   - マングル済みメソッド名
   - 多重定義ネイティブ・メソッドの場合には、2 つの下線 (__) に続くマングル済み引き数シグニチャー

3. 事前定義インターフェース関数テーブルをプログラムにコピーします。事前定義インターフェース関数テーブルのリストについては 304 ページの『メンバー JNI』を参照してください。
4. COBOL プログラムから Java プログラムに変数を渡すために、CALL で BY VALUE 句を指定します。以下の引き数を以下の順序で受け取ります。
   a. JNI インターフェース・ポインター
   b. Java クラス (静的ネイティブ・メソッドの場合) またはオブジェクト (非静的ネイティブ・メソッドの場合) への参照
   c. 必要なすべての追加の引き数。これらの引き数は、通常の Java メソッドの引き数に対応します。

   COBOL と Java のデータ・タイプは完全に等価ではないことに注意してください。 301 ページの『COBOL および Java のデータ・タイプ』を参照してください。

```
  PROCESS NOMONOPRC NOSTDTRUNC OPTIONS THREAD(SERIALIZE).      1

*** COBOL native program called from Java
*** static method

  IDENTIFICATION DIVISION.
  PROGRAM-ID.    "Java_Hello_displayHello".     2
  Author.
  INSTALLATION.   IBM Toronto Lab.
  DATE-WRITTEN.
  DATE-COMPILED.

  ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
  SOURCE-COMPUTER.    IBM-ISERIES
  OBJECT-COMPUTER.    IBM-ISERIES

  INPUT-OUTPUT SECTION.
  FILE-CONTROL.

  DATA DIVISION.
  FILE SECTION.
  WORKING-STORAGE SECTION.

  01 IS-COPY     PIC 1.
  01 NAME-PTR            USAGE POINTER.
  01 NAME-LENGTH PIC 9(4) BINARY.
  01 I           PIC 9(4) BINARY.

  01 NAME-X.
       05 CHAR-X OCCURS 20 TIMES PIC X.

  LINKAGE SECTION.

*** JNI interface function table

  COPY JNI.      3

  01 NAME.
       05 CHAR OCCURS 20 TIMES PIC N USAGE NATIONAL.

  01 ENV-PTR              USAGE POINTER.
  01 CLASS-REF   PIC S9(9) BINARY.
  01 TITLE-CODE  PIC S9(9) BINARY.
  01 NAME-REF    PIC S9(9) BINARY.

  01 INTERFACE-PTR        USAGE POINTER.
```

図 61. COBOL プログラム HELLO (1/2)

```
  PROCEDURE DIVISION USING BY VALUE ENV-PTR      4a
                                    CLASS-REF    4b
                                    TITLE-CODE   4c
                                    NAME-REF.    4c

 MAIN-LINE SECTION.
 MAIN-PROGRAM-LOGIC.

             SET ADDRESS OF INTERFACE-PTR TO ENV-PTR.
             SET ADDRESS OF JNI-NATIVE-INTERFACE TO INTERFACE-PTR.

*** Callback JNI interface function GET-STRING-LENGTH to
*** retrieve the name length

             CALL GET-STRING-LENGTH USING BY VALUE ENV-PTR      4
                                                   NAME-REF
                                    RETURNING INTO NAME-LENGTH.

*** Callback JNI interface function GET-STRING-CHARS to
*** retrieve the name characters

              CALL GET-STRING-CHARS USING BY VALUE ENV-PTR      4
                                                   NAME-REF
                                                   IS-COPY
                                    RETURNING INTO NAME-PTR.

             SET ADDRESS OF NAME TO NAME-PTR.
             INITIALIZE NAME-X.

             PERFORM VARYING I FROM 1 BY 1 UNTIL (I > NAME-LENGTH)
                   MOVE CHAR(I) TO CHAR-X(I)
             END-PERFORM.

             EVALUATE TITLE-CODE
                   WHEN  1
                                 DISPLAY "Hello, Mr. ", NAME-X
                   WHEN  2
                                 DISPLAY "Hello, Ms. ", NAME-X
                   WHEN OTHER
                                 DISPLAY "Hello, ", NAME-X
             END-EVALUATE.

             GOBACK.
```

図 61. COBOL プログラム HELLO (2/2)

```
  PROCESS NOMONOPRC NOSTDTRUNC OPTIONS THREAD(SERIALIZE).      1

*** COBOL native program called from Java
*** instance method

  IDENTIFICATION DIVISION.
  PROGRAM-ID.    "Java_Bye_displayBye".      2
  Author.
  INSTALLATION.   IBM Toronto Lab.
  DATE-WRITTEN.
  DATE-COMPILED.

  ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
  SOURCE-COMPUTER.    IBM-ISERIES
  OBJECT-COMPUTER.    IBM-ISERIES

  INPUT-OUTPUT SECTION.
  FILE-CONTROL.

  DATA DIVISION.
  FILE SECTION.

  WORKING-STORAGE SECTION.

  01 IS-COPY     PIC 1.
  01 NAME-PTR            USAGE POINTER.
  01 NAME-LENGTH PIC 9(4) BINARY.
  01 I           PIC 9(4) BINARY.

  01 NAME-X.
       05 CHAR-X OCCURS 20 TIMES PIC X.

  LINKAGE SECTION.

*** JNI interface function table

  COPY JNI.      3

  01 NAME.
       05 CHAR OCCURS 20 TIMES PIC N USAGE NATIONAL.

  01 ENV-PTR              USAGE POINTER.
  01 OBJECT-REF  PIC S9(9) BINARY.
  01 TITLE-CODE  PIC S9(9) BINARY.
  01 NAME-REF    PIC S9(9) BINARY.

  01 INTERFACE-PTR        USAGE POINTER.
```

図 62. COBOL プログラム BYE (1/2)

```
  PROCEDURE DIVISION USING BY VALUE ENV-PTR      4
                                    OBJECT-REF
                                    TITLE-CODE
                                    NAME-REF.

  MAIN-LINE SECTION.
  MAIN-PROGRAM-LOGIC.

               SET ADDRESS OF INTERFACE-PTR TO ENV-PTR.
               SET ADDRESS OF JNI-NATIVE-INTERFACE TO INTERFACE-PTR.

*** Callback JNI interface function GET-STRING-LENGTH to
*** retrieve the name length

               CALL GET-STRING-LENGTH USING BY VALUE ENV-PTR      4
                                                     NAME-REF
                                  RETURNING INTO NAME-LENGTH.

*** Callback JNI interface function GET-STRING-CHARS to
*** retrieve the name characters

              CALL GET-STRING-CHARS USING BY VALUE ENV-PTR      4
                                                   NAME-REF
                                                   IS-COPY
                                    RETURNING INTO NAME-PTR.

               SET ADDRESS OF NAME TO NAME-PTR.
               INITIALIZE NAME-X.

               PERFORM VARYING I FROM 1 BY 1 UNTIL (I > NAME-LENGTH)
                       MOVE CHAR(I) TO CHAR-X(I)
               END-PERFORM.

               EVALUATE TITLE-CODE
                      WHEN  1
                                  DISPLAY "Bye, Mr. ", NAME-X
                      WHEN  2
                                  DISPLAY "Bye, Ms. ", NAME-X
                      WHEN  OTHER
                                  DISPLAY "Bye, ", NAME-X
               END-EVALUATE.

               GOBACK.
```

図 62. COBOL プログラム BYE (2/2)

COBOL モジュールの作成: COBOL モジュールを作成するには、以下の 2 つの画面の例に示すように、 CRTCBLMOD コマンドを使用します。

```
                    COBOL モジュールの作成 (CRTCBLMOD)

選択項目を入力して、実行キーを押してください。

モジュール . . . . . .    *BYE          名前 , *PGMID
 ライブラリー. . . . . .     *CURLIB     名前 , *CURLIB
ソース・ファイル . . . . . .   QCBLLESRC     名前
  ライブラリー . . . . . .     *LIBL       名前 , *LIBL, *CURLIB
ソース・メンバー . . . . . .   BYE           名前 , *MODULE
ソース・ストリーム・ファイル

生成重大度レベル . . . . . .   30           0-30
テキスト記述 . . . . . .    *SRCMBRTXT


                          追加のパラメーター

 出力  . . . . . .    *PRINT       *PRINT, *NONE
 コンパイラー・オプション  . . .               *SOURCE, *NOSOURCE, *SRC...
               値の続きは+

                                                                  続く ...
F3= 終了  F4= プロンプト   F5= 最新表示    F12= 取り消し
F13= この画面の使用法     F24= キーの続き
```

```
                    COBOL モジュールの作成 (CRTCBLMOD)

選択項目を入力して、実行キーを押してください。

モジュール . . . . . .  > HELLO        名前 , *PGMID
 ライブラリー. . . . . .     *CURLIB     名前 , *CURLIB
ソース・ファイル . . . . . .   QCBLLESRC     名前
  ライブラリー . . . . . .     *LIBL       名前 , *LIBL, *CURLIB
ソース・メンバー . . . . . .  > HELLO        名前 , *MODULE
ソース・ストリーム・ファイル

生成重大度レベル . . . . . .   30           0-30
テキスト記述 . . . . . .    *SRCMBRTXT


                          追加のパラメーター

出力  . . . . . .    *PRINT       *PRINT, *NONE
 コンパイラー・オプション . . .               *SOURCE, *NOSOURCE, *SRC...
               値の続きは+

                                                                  続く ...
F3= 終了  F4= プロンプト   F5= 最新表示    F12= 取り消し
F13= この画面の使用法     F24= キーの続き
```

サービス・プログラムの作成: 以下に示したように、CRTSRVPGM コマンドを使用して、1 つまたは複数のモジュールを 1 つのサービス・プログラムにバインドします。 EXPORT オプションを指定してください。

```
                    サービス・プログラムの作成 (CRTSRVPGM)

選択項目を入力して、実行キーを押してください。

サービス・プログラム . . . . . SRVPGM       > HELLOBYE
  ﾗｲﾌﾞﾗﾘｰ  . . . . . .                        *CURLIB
モジュール . . . . . . . . . . MODULE       > HELLO
  ﾗｲﾌﾞﾗﾘｰ  . . . . . .                      >   *CURLIB
                       値の続きは +         > BYE
                                           >   *CURLIB
エクスポート . . . . . . . . . EXPORT       > *ALL
ソース・ファイルのエクスポート . SRCFILE      QSRVSRC
  ライブラリー   . . . . . .                  *LIBL
ソース・メンバーのエクスポート . SRCMBR       *SRVPGM
テキスト '記述'  . . . . . . . TEXT         *BLANK




                                                          続く...
F3= 終了    F4= プロンプト    F5= 最新表示    F12= 取り消し
F13= この画面の使用法        F24= キーの続き
```

Java プログラムのコーディング: このセクションでは、COBOL プログラムを呼び出す Java プログラムをコーディングする方法について説明します。上記の COBOL プログラムを呼び出す 2 つの Java プログラム例で、ガイドラインを示します。

Java ソース・ファイルは統合ファイル・システム (IFS) に保管されています。これらのファイルは、ストリーム・ファイル・エディターである EDTF を使用して編集することができます。

Java プログラムで COBOL プログラムを呼び出すには、以下のステップを実行してください。

1. システム・メソッド `System.loadLibrary` に対して静的初期設定呼び出しを行って、前のステップで作成した COBOL サービス・プログラムをロードします。(この例では、サービス・プログラムの名前は HELLOBYE です。)
2. キーワード `native` を使用して COBOL メソッドを宣言します。ネイティブ・メソッドの本文として、セミコロンのみを指定します。これは、インプリメンテーションが省略されていることを示します。

   短い名前 (引き数シグニチャーがない名前) を指定することができます。 JVM は、この名前を持つメソッドをネイティブ・ライブラリーで探します。見つからない場合、JVM はロング・ネームを探します。別のネイティブ・メソッドを多重定義したい場合には、ロング・ネームを使用してください。ネイティブ・メソッドが Java メソッドと同じ名前を持っている場合には、Java メソッドはネイティブ・ライブラリーに存在していないので、ロング・ネームを指定する必要はありません。

```
class Hello {
        static {
                System.loadLibrary("HELLOBYE");      1
        }

        static native void displayHello(int parm1, String parm2);      2

        public static void main(String[ ] args) {
              int titleCode;
              String name;

              switch (args.length) {
              case 1:
                           titleCode = Integer.parseInt(args[0]);
                           name = "Someone";
                           break;
              case 2:
                           titleCode = Integer.parseInt(args[0]);
                           name = args[1];
                           break;
              default:
                           titleCode = 0;
                           name = "Someone";
                           break;
              }
              displayHello(titleCode, name);
              Bye bye = new Bye( );
              bye.displayBye(titleCode, name);
        }
}
```

図 63. Java プログラム Hello.java

```
class Bye {
         static {
                 System.loadLibrary("HELLOBYE");      1
         }
         static native void displayBye(int parm1, String parm2);      2
}
```

図 64. Java プログラム Bye.java

Java プログラムのコンパイル: Java ソース・プログラムをコンパイルするために、Qshell インタープリター (QSH) を開始して、以下のコマンドを実行することができます。

```
javac Hello.java

javac Bye.java
```

Java プログラムの起動: Java ソース・プログラムを起動するために、Qshell インタープリター (QSH) を開始して、以下のコマンドを実行することができます。

```
>java Hello
Hello, Someone
Bye, Someone
>java Hello 1
Hello, Mr. Someone
Bye, Mr. Someone
>java Hello 2 USA
Hello, Ms. USA
Bye, Ms. USA
```

javah ツールを使用すれば、Java プログラムのヘッダー・ファイルを生成することができます。これらのヘッダー・ファイルは、COBOL コンパイラーではなく、C および C++ コンパイラーによって使用されますが、ネイティブ・プログラムの命名を検査する場合に役立ちます。

```
javah -jni Hello

javah -jni Bye
```

COBOL プログラムからの Java メソッドの呼び出し

COBOL プログラムから Java メソッドを呼び出すには、以下のステップを実行してください。

- 『COBOL プログラムのコーディング』
- 300 ページの『COBOL プログラムの作成』
- 301 ページの『Java プログラムのコーディング』
- 301 ページの『Java プログラムのコンパイル』

COBOL プログラムのコーディング: このセクションでは、Java メソッドを呼び出す COBOL プログラムをコーディングする方法について説明します。ガイドラインを、COBOL プログラム例と Java プログラム例で示します。

COBOL プログラムで Java メソッドを呼び出す場合には、以下のようにしてください。

1. PROCESS ステートメントの NOMONOPRC (大文字小文字を区別する名前に備えて) およびオプション THREAD(SERIALIZE) を使用します。
2. JDKINIT および JNI メンバーをコピーします。これらのメンバーのリストについては 304 ページの『メンバー JNI』および 309 ページの『メンバー JDK11INIT』を参照してください。
3. 適切な Java 呼び出し API 関数を呼び出します。以下の API 関数が使用可能です。
   - JNI_GetDefaultJavaVMInitArgs()
   - JNI_GetCreatedJavaVMs()
   - JNI_CreateJavaVM()
   - AttachCurrentThread()
   - DetachCurrentThread()

   このような呼び出し API 関数に関連するパラメーターの詳細については、「*Java Native Interface Specification Release 1.1 (Revised May, 1997)*」を参照してください。

   以下の例で、
   a. JVM 用のクラスパスを指定します。"- Djava.class.path=/home/myclass" の "/home/myclass" を、javac コマンドを使用して Java プログラムをコンパイルした時に作成されたクラス・ファイル HelloWorld.class を含む実際のディレクトリーに変更する必要があります。
   b. JVM が起動されます。

COBOL プログラム HELLOWORLD

```
 PROCESS MAP NOMONOPRC NOSTDTRUNC OPTIONS THREAD(SERIALIZE).     1

 IDENTIFICATION DIVISION.
 PROGRAM-ID. HELLOWORLD.
 Author.
 INSTALLATION.   IBM Toronto Lab.
 DATE-WRITTEN.
 DATE-COMPILED.

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER.    IBM-ISERIES.
 OBJECT-COMPUTER.    IBM-ISERIES.

 INPUT-OUTPUT SECTION.
 FILE-CONTROL.

 DATA DIVISION.
 FILE SECTION.

 WORKING-STORAGE SECTION.

*** JDK 1.2 VM initialization arguments

 01 VM-INIT-ARGS.
    05  VERSION             PIC S9(9) BINARY VALUE 65538.
    05  NUMBER-OF-OPTIONS   PIC S9(9) BINARY.
    05  OPTIONS-PTR                          USAGE POINTER.
    05  FILLER              PIC  X(1).

 01 VM-OPTIONS.
    05  OPTIONS-STRING-PTR                   USAGE POINTER.
    05  EXTRA-INFO-PTR                       USAGE POINTER.

***

 01 JVM-PTR                           USAGE POINTER.
 01 ENV-PTR                           USAGE POINTER.

 01 RC1                     PIC S9(9) BINARY VALUE 1.
 01 RC2                     PIC S9(9) BINARY VALUE 1.
 01 RC3                     PIC S9(9) BINARY VALUE 1.

 01 CLASS-NAME              PIC X(30).
 01 CLASS-NAME-PTR                    USAGE POINTER.

 01 METHOD-NAME             PIC X(30).
 01 METHOD-NAME-PTR                   USAGE POINTER.

 01 SIGNATURE-NAME          PIC X(30).
 01 SIGNATURE-NAME-PTR                USAGE POINTER.


*** CLASSPATH Parameters
 01 CLASSPATH               PIC X(500).

*** Object Reference Variables
 01 MY-CLASS-REF            PIC S9(9) BINARY.
 01 STRING-CLASS-REF        PIC S9(9) BINARY.
 01 METHOD-ID               PIC S9(9) BINARY.
 01 INIT-METHOD-ID          PIC S9(9) BINARY.
 01 STATIC-METHOD-ID        PIC S9(9) BINARY.
 01 OBJECT-REF              PIC S9(9) BINARY.
 01 ARG-REF                 PIC S9(9) BINARY.
 01 STRING-REF              PIC S9(9) BINARY.

*** Parameter Array for calling METHODA
```

```
                       01 PARM-ARRAY.
                          05 PARM-ARRAY-ELEMENT OCCURS 10 TIMES.
                             10 PARM-ARRAY-ELEMENT-VALUE  PIC S9(9) BINARY.
                             10 FILLER                    PIC X(4).

                       01 PARM-ARRAY-PTR                   USAGE POINTER.


                       LINKAGE SECTION.

                      *** JNI interface function table

                       COPY JNI.    2


                       01 INTERFACE-PTR                    USAGE POINTER.
                       01 JVM                    PIC S9(9) BINARY.


                       PROCEDURE DIVISION.

                       MAIN-LINE SECTION.
                       MAIN-PROGRAM-LOGIC.


                             STRING FUNCTION UTF8STRING("-Djava.class.path=/home/myclass")
3a                                   DELIMITED BY SIZE
                                X"00" DELIMITED BY SIZE
                               INTO CLASSPATH

                             SET OPTIONS-STRING-PTR TO ADDRESS OF CLASSPATH.
                             MOVE 1 TO NUMBER-OF-OPTIONS.
                             SET OPTIONS-PTR TO ADDRESS OF VM-OPTIONS.

                      *** Load and initializes the Java VM
3b
                             CALL PROCEDURE "JNI_CreateJavaVM"
                                  USING JVM-PTR ENV-PTR VM-INIT-ARGS
                                  RETURNING INTO RC2.

                             DISPLAY RC2.

                             SET ADDRESS OF INTERFACE-PTR TO ENV-PTR.
                             SET ADDRESS OF JNI-NATIVE-INTERFACE TO INTERFACE-PTR.

                      *** Callback JNI interface function FIND-CLASS "HelloWorld"

                             STRING FUNCTION UTF8STRING("HelloWorld") DELIMITED BY SIZE
                                    X"00" DELIMITED BY SIZE
                               INTO CLASS-NAME.

                             SET CLASS-NAME-PTR TO ADDRESS OF CLASS-NAME.

                             CALL FIND-CLASS USING BY VALUE ENV-PTR
                                                           CLASS-NAME-PTR
                                           RETURNING INTO MY-CLASS-REF.

                             DISPLAY MY-CLASS-REF.

                      *** Callback JNI interface function FIND-CLASS "java/lang/String"

                             STRING FUNCTION UTF8STRING("java/lang/String")
                                          DELIMITED BY SIZE
                                    X"00" DELIMITED BY SIZE
                               INTO CLASS-NAME.

                             SET CLASS-NAME-PTR TO ADDRESS OF CLASS-NAME.
```

```
           CALL FIND-CLASS USING BY VALUE ENV-PTR
                                          CLASS-NAME-PTR
                           RETURNING INTO STRING-CLASS-REF.

           DISPLAY STRING-CLASS-REF.

      *** Callback JNI interface function GET-METHOD-ID "<init>"
      *** to retrieve constructor method ID

           STRING FUNCTION UTF8STRING("<init>") DELIMITED BY SIZE
                  X"00" DELIMITED BY SIZE
             INTO METHOD-NAME.

           STRING FUNCTION UTF8STRING("()V") DELIMITED BY SIZE
                  X"00" DELIMITED BY SIZE
             INTO SIGNATURE-NAME.

           SET METHOD-NAME-PTR TO ADDRESS OF METHOD-NAME.
           SET SIGNATURE-NAME-PTR TO ADDRESS OF SIGNATURE-NAME.

           CALL GET-METHOD-ID USING BY VALUE ENV-PTR
                                             MY-CLASS-REF
                                             METHOD-NAME-PTR
                                             SIGNATURE-NAME-PTR
                              RETURNING INTO INIT-METHOD-ID.

           DISPLAY INIT-METHOD-ID.

      *** Callback JNI interface function NEW-OBJECT "HelloWorld"

           CALL NEW-OBJECT USING BY VALUE ENV-PTR
                                          MY-CLASS-REF
                                          INIT-METHOD-ID
                           RETURNING INTO OBJECT-REF.

           DISPLAY OBJECT-REF.

      *** Callback JNI interface function GET-STATIC-METHOD-ID "main"

           STRING FUNCTION UTF8STRING("main") DELIMITED BY SIZE
                  X"00" DELIMITED BY SIZE
             INTO METHOD-NAME.

           STRING FUNCTION UTF8STRING("([Ljava/lang/String;)V")
                        DELIMITED BY SIZE
                  X"00" DELIMITED BY SIZE
             INTO SIGNATURE-NAME.

           SET METHOD-NAME-PTR TO ADDRESS OF METHOD-NAME.
           SET SIGNATURE-NAME-PTR TO ADDRESS OF SIGNATURE-NAME.

           CALL GET-STATIC-METHOD-ID USING BY VALUE ENV-PTR
                                                    MY-CLASS-REF
                                                    METHOD-NAME-PTR
                                                    SIGNATURE-NAME-PTR
                                       RETURNING INTO STATIC-METHOD-ID.

           DISPLAY STATIC-METHOD-ID.

      *** Callback JNI interface function NEW-OBJECT-ARRAY

           CALL NEW-OBJECT-ARRAY USING BY VALUE ENV-PTR
                                                0
                                                STRING-CLASS-REF
                                                0
                                  RETURNING INTO ARG-REF.
```

```
           DISPLAY ARG-REF.

      *** Callback JNI interface function CALL-STATIC-VOID-METHODA

           SET PARM-ARRAY-PTR TO ADDRESS OF PARM-ARRAY.

           INITIALIZE PARM-ARRAY.

           MOVE ARG-REF TO PARM-ARRAY-ELEMENT-VALUE(1).

           CALL CALL-STATIC-VOID-METHODA USING BY VALUE ENV-PTR
                                                        MY-CLASS-REF
                                                        STATIC-METHOD-ID
                                                        PARM-ARRAY-PTR.

      *** Callback JNI interface function GET-METHOD-ID "display"

           STRING FUNCTION UTF8STRING("display") DELIMITED BY SIZE
                  X"00" DELIMITED BY SIZE
             INTO METHOD-NAME.

           STRING FUNCTION UTF8STRING("([II)V") DELIMITED BY SIZE
                  X"00" DELIMITED BY SIZE
             INTO SIGNATURE-NAME.

           SET METHOD-NAME-PTR TO ADDRESS OF METHOD-NAME.
           SET SIGNATURE-NAME-PTR TO ADDRESS OF SIGNATURE-NAME.

           CALL GET-METHOD-ID USING BY VALUE ENV-PTR
                                             MY-CLASS-REF
                                             METHOD-NAME-PTR
                                             SIGNATURE-NAME-PTR
                              RETURNING INTO METHOD-ID.

           DISPLAY METHOD-ID.

      *** Callback JNI interface function NEW-INT-ARRAY

           CALL NEW-INT-ARRAY USING BY VALUE ENV-PTR
                                             10
                              RETURNING INTO ARG-REF.

           DISPLAY ARG-REF.

      *** Callback JNI interface function CALL-VOID-METHODA

           SET PARM-ARRAY-PTR TO ADDRESS OF PARM-ARRAY.

           INITIALIZE PARM-ARRAY.

           MOVE ARG-REF TO PARM-ARRAY-ELEMENT-VALUE(1).
           MOVE 2       TO PARM-ARRAY-ELEMENT-VALUE(2).

           CALL CALL-VOID-METHODA USING BY VALUE ENV-PTR
                                                 OBJECT-REF
                                                 METHOD-ID
                                                 PARM-ARRAY-PTR.

           GOBACK.
```

COBOL プログラムの作成: COBOL モジュールを作成するには、以下に示したように CRTBNDCBL コマンドを使用します。

```
                バインド COBOL PGM の作成 (CRTBNDCBL)

選択項目を入力して、実行キーを押してください。

プログラム . . . . . .  HELLOWORLD    名前, *PGMID
  ライブラリー . . . . . .    *CURLIB     名前, *CURLIB
ソース・ファイル . . . . . .  QCBLLESRC     名前
  ライブラリー . . . . . .    *CURLIB     名前, *LIBL, *CURLIB
ソース・メンバー . . . . . .  HELLOWORLD    名前, *PGM
ソース・ストリーム・ファイル

生成重大度レベル . . . . . .   30           0-30
テキスト記述 . . . . . .   *SRCMBRTXT


                        追加のパラメーター

 出力  . . . . . .   *PRINT        *PRINT, *NONE
 コンパイラー・オプション  . . .                *SOURCE, *NOSOURCE, *SRC...
              値の続きは＋
                                                                  続く ...

F3= 終了  F4= プロンプト   F5= 最新表示   F12= 取り消し
F13= この画面の使用法      F24= キーの続き
```

Java プログラムのコーディング:

```
class HelloWorld {

     public static void main(String[] args) {

          System.out.println("Hello World");

     }

     void display(int[] args, int i) {

          System.out.println("Length of integer array is " + args.length);
          System.out.println("Value of integer variable is " + i);
          System.out.println("Bye World");

     }

}
```

図 65. Java プログラム HelloWorld.java

Java プログラムのコンパイル: Java ソース・プログラムをコンパイルするために、Qshell インタープリター (QSH) を開始して、以下のコマンドを実行することができます。

```
javac HelloWorld.java
```

COBOL および Java のデータ・タイプ

以下の表は、各 Java プリミティブ・タイプに対応する COBOL データ・タイプを示しています。

表 18. COBOL および Java のデータ・タイプの比較

Java プリミティブ・タイプ	**説明**	**Java のデータ範囲**	**COBOL データ・タイプ**	**COBOL のデータ範囲**
boolean	符号なし 8 ビット	0 (偽) または 1 (真)	PIC 9(4) BINARY	0 〜 255
byte	符号付き 8 ビット	-128 〜 127	PIC X	-128 〜 127
char	符号なし 16 ビット	0 ('¥u0000') 〜 65535 ('¥uffff')	PIC N USAGE NATIONAL	0 ('¥u0000') 〜 65535 ('¥uffff')
short	符号付き 16 ビット	-32768 〜 32767	PIC S9(4) BINARY [1]	-32768 〜 32767
int	符号付き 32 ビット	-2147483648 〜 2147483647	PIC S9(9) BINARY [1]	-2147483648 〜 2147483647
long	符号付き 64 ビット	-9223372036854775808 〜 9223372036854775807	PIC S9(18) BINARY [1]	-9223372036854775808 〜 9223372036854775807
float	32 ビット	1.40239846e-45f 〜 3.40282347e+38f	USAGE COMP-1	0.14012985e-44 〜 0.34028235e39
double	64 ビット	4.94065645841246544e-324 〜 1.79769313486231570e+308	USAGE COMP-2	.11125369292536009e-307 〜 .17976931348623155e+309
void	該当せず	該当せず	該当せず	該当せず

注:

1. short、int、および long プリミティブ・タイプの場合に切り捨てを保存するには、PROCESS ステートメントで NOSTDTRUNC を指定する必要があります。

COBOL および Java のデータ範囲は似ています。

- boolean、byte、char、short、および int に関しては、COBOL の範囲は、Java の範囲と同じか、それより大きくなります。
- float および double に関しては、COBOL のデータ範囲は、マシンのインプリメンテーションに依存します。
- void に対応する COBOL のデータ・タイプはありません。

Java の参照タイプは、クラス、インターフェースおよび配列からなります。参照タイプは Java の int 型引き数として渡されます。

```
01 JBOOLEAN          TYPEDEF PIC 9(4) BINARY.
01 JBYTE             TYPEDEF PIC X.
01 JCHAR             TYPEDEF PIC N USAGE NATIONAL.
01 JSHORT            TYPEDEF PIC S9(4) BINARY.      (and NOSTDTRUNC on PROCESS statement)
01 JINT              TYPEDEF PIC S9(9) BINARY.      (and NOSTDTRUNC on PROCESS statement)
01 JLONG             TYPEDEF PIC S9(18) BINARY.     (and NOSTDTRUNC on PROCESS statement)
01 JFLOAT            TYPEDEF USAGE COMP-1.
01 JDOUBLE           TYPEDEF USAGE COMP-2.
```

図 66. Java のデータ・タイプの定義

COBOL 用の JNI コピー・メンバー

以下のレイアウトは、JNI インターフェース関数テーブルの COBOL でのインプリメンテーションです。これらは、ライブラリー QSYSINC の中にあります。各 JNI ファンクションに関連するパラメーターの詳細については、「*Java Native Interface Specification Release 1.1 (Revised May, 1997)*」を参照してください。

- 304 ページの『メンバー JNI』
- 309 ページの『メンバー JDK11INIT』.

メンバー JNI

```
*** COBOL copybook for JNI native interface
*** based on Java Native Interface Specification Release 1.1
*** (Revised May, 1997)

01 JNI-NATIVE-INTERFACE.

     05 FILLER                          USAGE PROCEDURE-POINTER.
     05 FILLER                          USAGE PROCEDURE-POINTER.
     05 FILLER                          USAGE PROCEDURE-POINTER.
     05 FILLER                          USAGE PROCEDURE-POINTER.
     05 GET-VERSION                     USAGE PROCEDURE-POINTER.

     05 DEFINE-CLASS                    USAGE PROCEDURE-POINTER.
     05 FIND-CLASS                      USAGE PROCEDURE-POINTER.
     05 FILLER                          USAGE PROCEDURE-POINTER.
     05 FILLER                          USAGE PROCEDURE-POINTER.
     05 FILLER                          USAGE PROCEDURE-POINTER.
     05 GET-SUPERCLASS                  USAGE PROCEDURE-POINTER.
     05 IS-ASSIGNABLE-FROM              USAGE PROCEDURE-POINTER.
     05 FILLER                          USAGE PROCEDURE-POINTER.

     05 THROW                           USAGE PROCEDURE-POINTER.
     05 THROW-NEW                       USAGE PROCEDURE-POINTER.
     05 EXCEPTION-OCCURRED              USAGE PROCEDURE-POINTER.
     05 EXCEPTION-DESCRIBE              USAGE PROCEDURE-POINTER.
     05 EXCEPTION-CLEAR                 USAGE PROCEDURE-POINTER.
     05 FATAL-ERROR                     USAGE PROCEDURE-POINTER.
     05 FILLER                          USAGE PROCEDURE-POINTER.
     05 FILLER                          USAGE PROCEDURE-POINTER.

     05 NEW-GLOBAL-REF                  USAGE PROCEDURE-POINTER.
     05 DELETE-GLOBAL-REF               USAGE PROCEDURE-POINTER.
     05 DELETE-LOCAL-REF                USAGE PROCEDURE-POINTER.
     05 IS-SAME-OBJECT                  USAGE PROCEDURE-POINTER.
     05 FILLER                          USAGE PROCEDURE-POINTER.
     05 FILLER                          USAGE PROCEDURE-POINTER.

     05 ALLOC-OBJECT                    USAGE PROCEDURE-POINTER.
     05 NEW-OBJECT                      USAGE PROCEDURE-POINTER.
     05 NEW-OBJECTV                     USAGE PROCEDURE-POINTER.
     05 NEW-OBJECTA                     USAGE PROCEDURE-POINTER.

     05 GET-OBJECT-CLASS                USAGE PROCEDURE-POINTER.
     05 IS-INSTANCE-OF                  USAGE PROCEDURE-POINTER.

     05 GET-METHOD-ID                   USAGE PROCEDURE-POINTER.

     05 CALL-OBJECT-METHOD              USAGE PROCEDURE-POINTER.
     05 CALL-OBJECT-METHODV             USAGE PROCEDURE-POINTER.
     05 CALL-OBJECT-METHODA             USAGE PROCEDURE-POINTER.
```

図 67. メンバー JNI (1/5)

```
    05 CALL-BOOLEAN-METHOD             USAGE PROCEDURE-POINTER.
    05 CALL-BOOLEAN-METHODV            USAGE PROCEDURE-POINTER.
    05 CALL-BOOLEAN-METHODA            USAGE PROCEDURE-POINTER.
    05 CALL-BYTE-METHOD                USAGE PROCEDURE-POINTER.
    05 CALL-BYTE-METHODV               USAGE PROCEDURE-POINTER.
    05 CALL-BYTE-METHODA               USAGE PROCEDURE-POINTER.
    05 CALL-CHAR-METHOD                USAGE PROCEDURE-POINTER.
    05 CALL-CHAR-METHODV               USAGE PROCEDURE-POINTER.
    05 CALL-CHAR-METHODA               USAGE PROCEDURE-POINTER.
    05 CALL-SHORT-METHOD               USAGE PROCEDURE-POINTER.
    05 CALL-SHORT-METHODV              USAGE PROCEDURE-POINTER.
    05 CALL-SHORT-METHODA              USAGE PROCEDURE-POINTER.
    05 CALL-INT-METHOD                 USAGE PROCEDURE-POINTER.
    05 CALL-INT-METHODV                USAGE PROCEDURE-POINTER.
    05 CALL-INT-METHODA                USAGE PROCEDURE-POINTER.
    05 CALL-LONG-METHOD                USAGE PROCEDURE-POINTER.
    05 CALL-LONG-METHODV               USAGE PROCEDURE-POINTER.
    05 CALL-LONG-METHODA               USAGE PROCEDURE-POINTER.
    05 CALL-FLOAT-METHOD               USAGE PROCEDURE-POINTER.
    05 CALL-FLOAT-METHODV              USAGE PROCEDURE-POINTER.
    05 CALL-FLOAT-METHODA              USAGE PROCEDURE-POINTER.
    05 CALL-DOUBLE-METHOD              USAGE PROCEDURE-POINTER.
    05 CALL-DOUBLE-METHODV             USAGE PROCEDURE-POINTER.
    05 CALL-DOUBLE-METHODA             USAGE PROCEDURE-POINTER.
    05 CALL-VOID-METHOD                USAGE PROCEDURE-POINTER.
    05 CALL-VOID-METHODV               USAGE PROCEDURE-POINTER.
    05 CALL-VOID-METHODA               USAGE PROCEDURE-POINTER.

    05 CALL-NONVIRTUAL-OBJECT-METHOD   USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-OBJECT-METHODV  USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-OBJECT-METHODA  USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-BOOLEAN-METHOD  USAGE PROCEDURE-POINTER.

*** Note that the naming of the following 2 procedures deviates
*** slightly from the others due to the 30 character field
*** name limitation.
    05 CALL-NONVIRTUAL-BOOLEAN-MTHDV   USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-BOOLEAN-MTHDA   USAGE PROCEDURE-POINTER.

    05 CALL-NONVIRTUAL-BYTE-METHOD     USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-BYTE-METHODV    USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-BYTE-METHODA    USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-CHAR-METHOD     USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-CHAR-METHODV    USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-CHAR-METHODA    USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-SHORT-METHOD    USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-SHORT-METHODV   USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-SHORT-METHODA   USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-INT-METHOD      USAGE PROCEDURE-POINTER.
    05 CALL-NONVIRTUAL-INT-METHODV     USAGE PROCEDURE-POINTER.
```

図 67. メンバー JNI (2/5)

```
05 CALL-NONVIRTUAL-INT-METHODA     USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-LONG-METHOD     USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-LONG-METHODV    USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-LONG-METHODA    USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-FLOAT-METHOD    USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-FLOAT-METHODV   USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-FLOAT-METHODA   USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-DOUBLE-METHOD   USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-BOOLEAN-MTHDA   USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-BYTE-METHOD     USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-BYTE-METHODV    USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-BYTE-METHODA    USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-CHAR-METHOD     USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-CHAR-METHODV    USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-CHAR-METHODA    USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-SHORT-METHOD    USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-SHORT-METHODV   USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-SHORT-METHODA   USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-INT-METHOD      USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-INT-METHODV     USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-INT-METHODA     USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-LONG-METHOD     USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-LONG-METHODV    USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-LONG-METHODA    USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-FLOAT-METHOD    USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-FLOAT-METHODV   USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-FLOAT-METHODA   USAGE PROCEDURE-POINTER.
05 CALL-NONVIRTUAL-DOUBLE-METHOD   USAGE PROCEDURE-POINTER.
05 SET-CHAR-FIELD                  USAGE PROCEDURE-POINTER.
05 SET-SHORT-FIELD                 USAGE PROCEDURE-POINTER.
05 SET-INT-FIELD                   USAGE PROCEDURE-POINTER.
05 SET-LONG-FIELD                  USAGE PROCEDURE-POINTER.
05 SET-FLOAT-FIELD                 USAGE PROCEDURE-POINTER.
05 SET-DOUBLE-FIELD                USAGE PROCEDURE-POINTER.

05 GET-STATIC-METHOD-ID            USAGE PROCEDURE-POINTER.

05 CALL-STATIC-OBJECT-METHOD       USAGE PROCEDURE-POINTER.
05 CALL-STATIC-OBJECT-METHODV      USAGE PROCEDURE-POINTER.
05 CALL-STATIC-OBJECT-METHODA      USAGE PROCEDURE-POINTER.
05 CALL-STATIC-BOOLEAN-METHOD      USAGE PROCEDURE-POINTER.
05 CALL-STATIC-BOOLEAN-METHODV     USAGE PROCEDURE-POINTER.
05 CALL-STATIC-BOOLEAN-METHODA     USAGE PROCEDURE-POINTER.
05 CALL-STATIC-BYTE-METHOD         USAGE PROCEDURE-POINTER.
05 CALL-STATIC-BYTE-METHODV        USAGE PROCEDURE-POINTER.
05 CALL-STATIC-BYTE-METHODA        USAGE PROCEDURE-POINTER.
05 CALL-STATIC-CHAR-METHOD         USAGE PROCEDURE-POINTER.
05 CALL-STATIC-CHAR-METHODV        USAGE PROCEDURE-POINTER.
05 CALL-STATIC-CHAR-METHODA        USAGE PROCEDURE-POINTER.
05 CALL-STATIC-SHORT-METHOD        USAGE PROCEDURE-POINTER.
```

図 67. メンバー JNI (3/5)

```
    05 CALL-STATIC-SHORT-METHODV       USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-SHORT-METHODA       USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-INT-METHOD          USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-INT-METHODV         USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-INT-METHODA         USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-LONG-METHOD         USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-LONG-METHODV        USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-LONG-METHODA        USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-FLOAT-METHOD        USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-FLOAT-METHODV       USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-FLOAT-METHODA       USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-DOUBLE-METHOD       USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-DOUBLE-METHODV      USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-DOUBLE-METHODA      USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-VOID-METHOD         USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-VOID-METHODV        USAGE PROCEDURE-POINTER.
    05 CALL-STATIC-VOID-METHODA        USAGE PROCEDURE-POINTER.
    05 GET-STATIC-FILED-ID             USAGE PROCEDURE-POINTER.

    05 GET-STATIC-OBJECT-FIELD         USAGE PROCEDURE-POINTER.
    05 GET-STATIC-OBJECT-BOOLEAN-FIELD USAGE PROCEDURE-POINTER.
    05 GET-STATIC-OBJECT-BYTE-FIELD    USAGE PROCEDURE-POINTER.
    05 GET-STATIC-OBJECT-CHAR-FIELD    USAGE PROCEDURE-POINTER.
    05 GET-STATIC-OBJECT-SHORT-FIELD   USAGE PROCEDURE-POINTER.
    05 GET-STATIC-OBJECT-INT-FIELD     USAGE PROCEDURE-POINTER.
    05 GET-STATIC-OBJECT-LONG-FIELD    USAGE PROCEDURE-POINTER.
    05 GET-STATIC-OBJECT-FLOAT-FIELD   USAGE PROCEDURE-POINTER.
    05 GET-STATIC-OBJECT-DOUBLE-FIELD  USAGE PROCEDURE-POINTER.

    05 SET-STATIC-OBJECT-FIELD         USAGE PROCEDURE-POINTER.
    05 SET-STATIC-OBJECT-BOOLEAN-FIELD USAGE PROCEDURE-POINTER.
    05 SET-STATIC-OBJECT-BYTE-FIELD    USAGE PROCEDURE-POINTER.
    05 SET-STATIC-OBJECT-CHAR-FIELD    USAGE PROCEDURE-POINTER.
    05 SET-STATIC-OBJECT-SHORT-FIELD   USAGE PROCEDURE-POINTER.
    05 SET-STATIC-OBJECT-INT-FIELD     USAGE PROCEDURE-POINTER.
    05 SET-STATIC-OBJECT-LONG-FIELD    USAGE PROCEDURE-POINTER.
    05 SET-STATIC-OBJECT-FLOAT-FIELD   USAGE PROCEDURE-POINTER.
    05 SET-STATIC-OBJECT-DOUBLE-FIELD  USAGE PROCEDURE-POINTER.

    05 NEW-STRING                      USAGE PROCEDURE-POINTER.
    05 GET-STRING-LENGTH               USAGE PROCEDURE-POINTER.
    05 GET-STRING-CHARS                USAGE PROCEDURE-POINTER.
    05 RELEASE-STRING-CHARS            USAGE PROCEDURE-POINTER.

    05 NEW-STRING-UTF                  USAGE PROCEDURE-POINTER.
    05 GET-STRING-UTF-LENGTH           USAGE PROCEDURE-POINTER.
    05 GET-STRING-UTF-CHARS            USAGE PROCEDURE-POINTER.
    05 RELEASE-STRING-UTF-CHARS        USAGE PROCEDURE-POINTER.

    05 GET-ARRAY-LENGTH                USAGE PROCEDURE-POINTER.
```

図 67. メンバー JNI (4/5)

```
05 NEW-OBJECT-ARRAY                USAGE PROCEDURE-POINTER.
05 GET-OBJECT-ARRAY-ELEMENT        USAGE PROCEDURE-POINTER.
05 SET-OBJECT-ARRAY-ELEMENT        USAGE PROCEDURE-POINTER.

05 NEW-BOOLEAN-ARRAY               USAGE PROCEDURE-POINTER.
05 NEW-BYTE-ARRAY                  USAGE PROCEDURE-POINTER.
05 NEW-CHAR-ARRAY                  USAGE PROCEDURE-POINTER.
05 NEW-SHORT-ARRAY                 USAGE PROCEDURE-POINTER.
05 NEW-INT-ARRAY                   USAGE PROCEDURE-POINTER.
05 NEW-LONG-ARRAY                  USAGE PROCEDURE-POINTER.
05 NEW-FLOAT-ARRAY                 USAGE PROCEDURE-POINTER.
05 NEW-DOUBLE-ARRAY                USAGE PROCEDURE-POINTER.

05 GET-BOOLEAN-ARRAY-ELEMENTS      USAGE PROCEDURE-POINTER.
05 GET-BYTE-ARRAY-ELEMENTS         USAGE PROCEDURE-POINTER.
05 GET-CHAR-ARRAY-ELEMENTS         USAGE PROCEDURE-POINTER.
05 GET-SHORT-ARRAY-ELEMENTS        USAGE PROCEDURE-POINTER.
05 GET-INT-ARRAY-ELEMENTS          USAGE PROCEDURE-POINTER.
05 GET-LONG-ARRAY-ELEMENTS         USAGE PROCEDURE-POINTER.
05 GET-FLOAT-ARRAY-ELEMENTS        USAGE PROCEDURE-POINTER.
05 GET-DOUBLE-ARRAY-ELEMENTS       USAGE PROCEDURE-POINTER.

05 RELEASE-BOOLEAN-ARRAY-ELEMENTS  USAGE PROCEDURE-POINTER.
05 RELEASE-BYTE-ARRAY-ELEMENTS     USAGE PROCEDURE-POINTER.
05 RELEASE-CHAR-ARRAY-ELEMENTS     USAGE PROCEDURE-POINTER.
05 RELEASE-SHORT-ARRAY-ELEMENTS    USAGE PROCEDURE-POINTER.
05 RELEASE-INT-ARRAY-ELEMENTS      USAGE PROCEDURE-POINTER.
05 RELEASE-LONG-ARRAY-ELEMENTS     USAGE PROCEDURE-POINTER.
05 RELEASE-FLOAT-ARRAY-ELEMENTS    USAGE PROCEDURE-POINTER.
05 RELEASE-DOUBLE-ARRAY-ELEMENTS   USAGE PROCEDURE-POINTER.

05 GET-BOOLEAN-ARRAY-REGION        USAGE PROCEDURE-POINTER.
05 GET-BYTE-ARRAY-REGION           USAGE PROCEDURE-POINTER.
05 GET-CHAR-ARRAY-REGION           USAGE PROCEDURE-POINTER.
05 GET-SHORT-ARRAY-REGION          USAGE PROCEDURE-POINTER.
05 GET-INT-ARRAY-REGION            USAGE PROCEDURE-POINTER.
05 GET-LONG-ARRAY-REGION           USAGE PROCEDURE-POINTER.
05 GET-FLOAT-ARRAY-REGION          USAGE PROCEDURE-POINTER.
05 GET-DOUBLE-ARRAY-REGION         USAGE PROCEDURE-POINTER.
05 SET-BOOLEAN-ARRAY-REGION        USAGE PROCEDURE-POINTER.
05 SET-BYTE-ARRAY-REGION           USAGE PROCEDURE-POINTER.
05 SET-CHAR-ARRAY-REGION           USAGE PROCEDURE-POINTER.
05 SET-SHORT-ARRAY-REGION          USAGE PROCEDURE-POINTER.
05 SET-INT-ARRAY-REGION            USAGE PROCEDURE-POINTER.
05 SET-LONG-ARRAY-REGION           USAGE PROCEDURE-POINTER.
05 SET-FLOAT-ARRAY-REGION          USAGE PROCEDURE-POINTER.
05 SET-DOUBLE-ARRAY-REGION         USAGE PROCEDURE-POINTER.

05 REGISTER-NATIVES                USAGE PROCEDURE-POINTER.
05 UNREGISTER-NATIVES              USAGE PROCEDURE-POINTER.

05 MONITOR-ENTER                   USAGE PROCEDURE-POINTER.
05 MONITOR-EXIT                    USAGE PROCEDURE-POINTER.

05 GET-JAVA-VM                     USAGE PROCEDURE-POINTER.
```

図 67. メンバー JNI (5/5)

メンバー JDK11INIT

```
*** COBOL copybook for JDK 1.1 VM initialization arguments
*** based on Java Native Interface Specification Release 1.1
*** (Revised May, 1997)

 01 VM-INIT-ARGS.
       05  VERSION           PIC S9(9) BINARY VALUE 65537.
       05  FILLER            PIC S9(9) BINARY.
       05  FILLER            PIC S9(9) BINARY.
       05  FILLER            PIC S9(9) BINARY.
       05  PROPERTIES                USAGE PROCEDURE-POINTER.
       05  CHECK-SOURCE      PIC S9(9) BINARY.
       05  NATIVE-STACK-SIZE PIC S9(9) BINARY.
       05  JAVA-STACK-SIZE   PIC S9(9) BINARY.
       05  MIN-HEAP-SIZE     PIC S9(9) BINARY.
       05  MAX-HEAP-SIZE     PIC S9(9) BINARY.
       05  VERIFY-MODE       PIC S9(9) BINARY.
       05  FILLER            PIC S9(9) BINARY.
       05  FILLER            PIC S9(9) BINARY.
       05  CLASSPATH                 USAGE POINTER.
       05  MESSAGE-HOOK              USAGE PROCEDURE-POINTER.
       05  EXIT-HOOK                 USAGE PROCEDURE-POINTER.
       05  ABORT-HOOK                USAGE PROCEDURE-POINTER.
       05  ENABLE-CLASSIC-GC PIC S9(9) BINARY.
       05  ENABLE-VERBOSE-GC PIC S9(9) BINARY.
       05  DISABLE-ASYNC-GC  PIC S9(9) BINARY.
       05  FILLER            PIC S9(9) BINARY.
       05  FILLER            PIC S9(9) BINARY.
       05  FILLER            PIC S9(9) BINARY.
```

図 68. メンバー JDK11INIT

第 11 章 XML 入力の処理

XML PARSE ステートメントを使用して、ILE COBOL プログラムから XML 文書を処理することができます。 XML PARSE ステートメントは、COBOL ランタイムの一部である高速 XML パーサーに対する、COBOL 言語のインターフェースです。XML 文書の処理には、XML パーサーとの制御の受け渡しも含まれます。この制御のやり取りは、XML PARSE ステートメントで開始します。このステートメントでは、XML パーサーから制御を受け取りパーサー・イベントを処理する、処理プロシージャーを指定します。パーサーとの間の情報のやりとりには、処理プロシージャー内の特殊レジスターを使用します。

XML 文書を処理するには、以下の COBOL 機能を使用します。

- XML PARSE ステートメントにより、 XML 構文解析を開始し、文書と処理プロシージャーを識別する。
- 構文解析を制御するための処理プロシージャーが XML イベントおよび関連した文書断片を受け取り、処理し、またオプションとして例外を処理する。
- 特殊レジスターにより、情報を受け渡す。
  - XML-CODE により、XML 構文解析の状況を判別する。
  - XML-EVENT により、各 XML イベントの名前を受け取る。
  - XML-TEXT により、英数字文書から XML 文書の断片を受け取る。
  - XML-NTEXT 国別文書から XML 文書の断片を受け取る。

関連概念
『COBOL の XML パーサー』

関連作業
313 ページの『XML 文書へのアクセス』
313 ページの『XML 文書の構文解析』
314 ページの『XML イベントの処理』
330 ページの『XML 文書中のエラーの処理』
328 ページの『XML 文書のエンコード方式の理解』

関連リファレンス
699 ページの『付録 F. XML 参照資料』
「*XML specification*」(www.w3c.org/XML/)

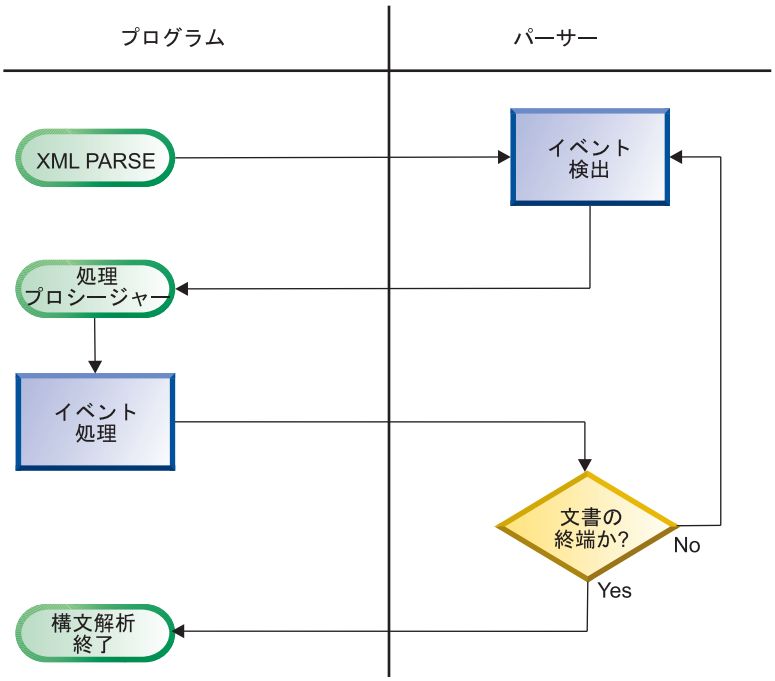
COBOL の XML パーサー

ILE COBOL は、XML 文書を構文解析し COBOL のデータ構造に変換することのできる、イベント・ベースのインターフェースを提供します。 XML パーサーは、文書中の断片 (XML イベントと関連するもの) を検出し、処理プロシージャーはその断片を元に処理します。ユーザーは、各 XML イベントを処理するように、独自のプロシージャーをコーディングします。この操作の全体を通して、制御は、パーサーとユーザーのプロシージャーとの間で渡されます。

このパーサーとのやりとりを開始するには XML PARSE ステートメントを使用し、ステートメント中でユーザーの処理プロシージャーを指定します。この XML PARSE ステートメントの実行により、構文解析が開始され、またユーザーの処理プロシージャーをパーサーに対して設定します。プロシージャーが実行されると、その度に XML パーサーは XML 文書の分析を継続し、次のイベントを報告し、検出した断片 (新しいエレメントの開始部分など) をユーザーのプロシージャーに戻します。また、XML PARSE ステートメントで、構文解析の最後に制御を渡す命令ステートメントを 2 つ指定できます。1 つは通常の終了時のためで、1 つは例外条件が存在するときのためです。

この図は、パーサーとユーザーのプログラムとの間の基本的な制御のやり取りの概要です。

XML 構文解析フロー概要



通常、構文解析は、XML 文書全体が解析されるまで続きます。

XML パーサーは、XML 文書を構文解析する際、XML 仕様で定義されている整形式の性質の大部分を検査します。文書が XML の構文に従い、その他にも終了タグが正しく使用されていることや、属性名が固有名であることなどの規則に従っているなら、その文書は整形式です。

関連作業

313 ページの『XML 文書へのアクセス』
313 ページの『XML 文書の構文解析』
320 ページの『XML を処理するプロシージャーの記述』
330 ページの『XML 文書中のエラーの処理』
328 ページの『XML 文書のエンコード方式の理解』

関連リファレンス
「*XML specification*」(www.w3c.org/XML/)
707 ページの『XML 順応』

XML 文書へのアクセス

XML 文書を XML PARSE ステートメントで構文解析する前に、文書をプログラムが使用できるようにしなければなりません。文書を獲得する方法として最も可能性が高いのは、MQSeries メッセージ、CICS 一時的待ち行列、または連絡域からの検索です。

構文解析したい XML 文書がファイルに収められているのなら、通常の COBOL 機能を使用し、文書をプログラム中のデータ項目に入れます。

- FILE-CONTROL 項目により、プログラムに対してファイルを定義
- OPEN ステートメントにより、ファイルを開く
- READ ステートメントにより、ファイルのすべてのレコードを、プログラムの WORKING-STORAGE SECTION または LOCAL-STORAGE SECTION で定義された、英数字または国別データ項目に読み込む
- オプションとして、STRING ステートメントにより、別々のレコードをすべて繋ぎあわせて 1 つの連続したストリームにし、関係のないブランクを除去し、可変長レコードを処理する

別の方法として、XML 文書を IFS ストリーム・ファイルにコピーし、 XML PARSE ステートメントの形式 2 を使用して文書にアクセスして構文解析することができます。

関連作業
CICS for i5/OS Application Programming Guide。 i5/OS Information Center http://www.ibm.com/systems/infocenter/ を参照してください。

XML 文書の構文解析

XML 文書を構文解析するには、XML 文書がデータ項目に入っている場合は、次の例のように XML PARSE ステートメントを使用します。

```
XML PARSE XMLDOCUMENT
    PROCESSING PROCEDURE XMLEVENT-HANDLER
  ON EXCEPTION
     DISPLAY 'XML document error ' XML-CODE
     STOP RUN
  NOT ON EXCEPTION
     DISPLAY 'XML document was successfully parsed.'
END-XML
```

XML PARSE ステートメントでは、最初に、 XML 文書の文字ストリームを含むデータ項目 (この例では、XMLDOCUMENT) を識別します。 DATA DIVISION では、ID を英数字データ項目または国別データ項目として宣言できます。英数字の場合は、その内容は、サポートされている EBCDIC または ASCII の 1 バイト文字セットの 1 つでエンコードされていなければなりません。国別データ項目の場合は、その内容は National CCSID コンパイラー・オプション、または NTLCCSID PROCESS オプションで指定した Unicode UCS-2 CCSID でエンコードされていなければなりませ

ん。エンコード宣言を含まない英数字の XML 文書は、COBOL ソース・メンバーの CCSID で構文解析されるか、 COBOL ソースが IFS ストリーム・ファイルにある場合は、ストリーム・ファイルの CCSID が使用されます。

次に、文書からの XML イベントを処理するプロシージャーの名前 (例では、XMLEVENT-HANDLER) を指定します。

加えて、以下の命令ステートメントの片方または両方が、構文解析の終わりに制御を受け取るように指定できます。

- ON EXCEPTION は、未処理の例外が発生したときに制御を受け取ります。
- NOT ON EXCEPTION は、そうでない場合に制御を受け取ります。

XML PARSE ステートメントを、END-XML で終了することができます。 XML PARSE ステートメントを条件ステートメントまたは他の XML PARSE ステートメント内にネストするには、この範囲終了符号を使用します。

以下のいずれかが発生するまで、XML パーサーと処理プロシージャーの間の制御のやり取りは続きます。

- END-OF-DOCUMENT イベントにより、XML 文書全体が構文解析されたことが示された。
- パーサーが文書中のエラーを発見し、 EXCEPTION イベントを通知する。処理プロシージャーは、特殊レジスター XML-CODE を 0 にリセットせずにパーサーに戻る。
- パーサーに戻る前に特殊レジスター XML-CODE を -1 に設定することで、意図的に構文解析プロセスを強制終了させる。

関連作業
328 ページの『XML 文書のエンコード方式の理解』

関連リファレンス
『XML PARSE ステートメント』(「*ILE COBOL 言語解説書*」)
『制御フロー』(「*ILE COBOL 言語解説書*」)

XML イベントの処理

パーサーが処理プロシージャーに渡すイベントを判別するには、XML-EVENT 特殊レジスターを使用します。 XML-EVENT には、'START-OF-ELEMENT' 等のイベント名が含まれています。パーサーは、XML PARSE ステートメント内の XML の ID のタイプに応じて、特殊レジスター XML-TEXT または XML-NTEXT にイベントの内容を入れて渡します。

イベントは、基本的に、このサンプル XML 文書で発生する順番に示されています。「サンプル XML テキスト」の下に示されているテキストは、このサンプルから来るものです。正確なテキストは、区切り文字「<<>>」に囲まれています。

```
<?xml version="1.0" encoding="ibm-1140" standalone="yes" ?>
<!--This document is just an example-->
<sandwich>
  <bread type="baker&apos;s best" />
  <?spread please use real mayonnaise ?>
```

```
  <meat>Ham &amp;amp; turkey</meat>
  <filling>Cheese, lettuce, tomato, etc.</filling>
  <![CDATA[We should add a <relish> element in future!]]>
</sandwich>junk
```

START-OF-DOCUMENT

説明 文書の構文解析の最初に、1 回発生します。ここでは XML テキストは文書全体であり、それは LF (ライン・フィード) や NL (改行) 等の、行の制御文字を含みます。

サンプル XML テキスト
このサンプルの場合のテキストは、長さ 336 文字です。

VERSION-INFORMATION

説明 オプションの XML 宣言の中で、バージョン情報に対して発生します。 XML テキストは、バージョンの値を含みます。*XML 宣言* とは、使用されている XML のバージョンと文書のエンコード方式を指定する XML テキストです。

サンプル XML テキスト
<<1.0>>

ENCODING-DECLARATION

説明 XML 宣言の中で、オプションのエンコード宣言に対して発生します。 XML テキストは、エンコード方式の値を含みます。

サンプル XML テキスト
<<ibm-1140>>

STANDALONE-DECLARATION

説明 XML 宣言の中で、オプションのスタンドアロン宣言に対して発生します。 XML テキストは、スタンドアロン値を含みます。

サンプル XML テキスト
<<yes>>

DOCUMENT-TYPE-DECLARATION

説明 パーサーが、文書タイプ宣言 (DTD) を検出すると発生します。文書タイプ宣言は、文字シーケンス「<!DOCTYPE」で始まり、「>」の文字で終わります。その間の中身については、かなり複雑な文法規則によって記述されています。 (「*XML specification*」を参照してください。) このイベントに対しては、XML テキストは、開始文字シーケンスと終了文字シーケンスを含む宣言文全体を含みます。このイベントは、XML テキストが区切り文字を含む、唯一のイベントです。

サンプル XML テキスト
サンプルには文書タイプ宣言がありません。

COMMENT

説明 XML 文書中のコメントに対して発生します。XML テキストは、コメント開始区切り (「<!--」) およびコメント終了区切り (「-->」) の間のデータを含みます。

サンプル XML テキスト

```
<<This document is just an example>>
```

START-OF-ELEMENT

説明 各エレメント開始タグまたは空エレメント・タグに対して 1 回ずつ発生します。XML テキストには、エレメント名が設定されます。

サンプル XML テキスト

START-OF-ELEMENT イベントとして発生する順番に挙げると次のようになります。

1. <<sandwich>>
2. <<bread>>
3. <<meat>>
4. <<filling>>

ATTRIBUTE-NAME

説明 エレメント開始タグまたは空エレメント・タグ中の各属性に対して、有効な属性名を認識する度に 1 回ずつ発生します。 XML テキストは、属性名を含みます。

サンプル XML テキスト

```
<<type>>
```

ATTRIBUTE-CHARACTERS

説明 属性値の各断片に対して発生します。XML テキストは、断片を含みます。属性値は、たとえ複数行にまたがって分かれている場合でも、通常は単一のストリングのみで構成されます。しかし、属性値が複数のイベントで構成されることはあります。

サンプル XML テキスト

ATTRIBUTE-CHARACTERS イベントとして発生する順番に挙げると次のようになります。

1. <<baker>>
2. <<s best>>

サンプルの「type」属性の値が、3 つの断片から成り立っていることに注目してください。ストリング「baker」単一の文字「'」、およびストリング「s best」です。単一の文字「'」の断片は、ATTRIBUTE-CHARACTER イベントとして、別に渡されます。

ATTRIBUTE-CHARACTER

説明 属性値の中で、事前定義エンティティー参照である「&」、「'」、「>」、「<」、および「"」に対して発生します。事前定義エンティティーについては、「*XML specification*」を参照してください。

サンプル XML テキスト

```
<<'>>
```

ATTRIBUTE-NATIONAL-CHARACTER

説明 属性値の中で、「&#dd..;」または「&#hh..;」という形式の数字参照 (Unicode のコード・ポイントまたは「スカラー値」) に対して発生します。ただし、ここでは「d」および「h」はそれぞれ 10 進数および 16 進数の桁を表しています。

サンプル XML テキスト

このサンプルには、数字参照は含まれていません。

PROCESSING-INSTRUCTION-TARGET

説明 パーサーが、処理命令 (PI) の開始文字シーケンスである、「<?」に続く名前を認識すると発生します。PI により、XML 文書はアプリケーション用の特殊な命令を含むことができます。

サンプル XML テキスト

```
<<spread>>
```

PROCESSING-INSTRUCTION-DATA

説明 PI のターゲットに続くデータに対して発生します。データは、PI の終了文字シーケンス (「?>」) までのものですが、終了文字シーケンスは含みません。 XML テキストに含まれる PI データは、後続のスペース文字を含みますが、先行のスペース文字は含みません。

サンプル XML テキスト

```
<<please use real mayonnaise >>
```

CONTENT-CHARACTERS

説明 このイベントは、XML 文書の基本部分、つまりエレメントの開始タグと終了タグの間の文字データを表します。 XML テキストはこのデータを含み、それはたとえ複数行にまたがって分かれている場合でも、通常は単一のストリングのみで構成されます。エレメントの内容が参照または他のエレメントを含む場合は、内容全体は複数のエレメントから構成されます。パーサーはまた、プログラムの CDATA セクションのテキストを渡すために、 CONTENT-CHARACTERS イベントを使用します。

サンプル XML テキスト

CONTENT-CHARACTERS イベントとして発生する順番に挙げると次のようになります。

1. <<Ham >>
2. << turkey>>
3. <<Cheese, lettuce, tomato, etc.>>
4. <<We should add a <relish> element in future!>>

サンプルの「meat」エレメントの内容はストリング「Ham」、文字「&」、およびストリング「 turkey」から成り立っていることに注目してください。単一の文字「&」の断片は、CONTENT-CHARACTER イベントとして、別に渡されます。また、この 2 つのストリング断片のそれぞれの先行のスペースおよび後続のスペースに注目してください。

CONTENT-CHARACTER

説明 エレメント・コンテントの中で、事前定義エンティティー参照である「&」、「'」、「>」、「<」、および「"」に対して発生します。事前定義エンティティーについては、「*XML specification*」を参照してください。

サンプル XML テキスト

<<&>>

CONTENT-NATIONAL-CHARACTER

説明 エレメント・コンテントの中で、「&#dd..;」または「&#hh..;」という形式の数字参照 (Unicode のコード・ポイントまたは「スカラー値」) に対して発生します。ただし、ここでは「d」および「h」はそれぞれ 10 進数および 16 進数の桁を表しています。

サンプル XML テキスト

このサンプルには、数字参照は含まれていません。

END-OF-ELEMENT

説明 各エレメント終了タグまたは空エレメント・タグに対して、パーサーがタグの終了を示す不等号括弧を認識したときに 1 回ずつ発生します。 XML テキストは、エレメント名を含みます。

サンプル XML テキスト

END-OF-ELEMENT イベントとして発生する順番に挙げると次のようになります。

1. <<bread>>
2. <<meat>>
3. <<filling>>
4. <<sandwich>>

START-OF-CDATA-SECTION

説明 CDATA セクションの先頭で発生します。CDATA セクションは、ストリング 「<![CDATA[」で開始し、ストリング「]]>」で終了します。これらのセクションは、XML マークアップとして認識される可能性がある文字を含むテキスト・ブロックを「エスケープ」するために使用されます。 XML テキストは、開始文字シーケンスの「<![CDATA[」を必ず含みます。パーサーは、これらの区切り文字の間の、 CDATA セクションの内容を、単一の CONTENT-CHARACTERS イベントとして渡します。

サンプル XML テキスト

<<<![CDATA[>>

END-OF-CDATA-SECTION

説明 パーサーが CDATA セクションの終端を認識すると発生します。

サンプル XML テキスト

<<]]>>>

UNKNOWN-REFERENCE-IN-ATTRIBUTE

説明 属性値の中で、上記の ATTRIBUTE-CHARACTER で示されている 5 つの事前定義エンティティー参照以外のエンティティー参照ついて発生します。

サンプル XML テキスト

サンプルは、不明エンティティー参照を含みません。

UNKNOWN-REFERENCE-IN-CONTENT

説明 エレメント・コンテント中で、上記の CONTENT-CHARACTER で示されている事前定義エンティティー参照以外のエンティティー参照について発生します。

サンプル XML テキスト

サンプルは、不明エンティティー参照を含みません。

END-OF-DOCUMENT

説明 文書の構文解析が完了したときに発生します。

サンプル XML テキスト

END-OF-DOCUMENT イベントの XML テキストは空です。

EXCEPTION

説明 XML 文書処理中のエラーが検出されたときに発生します。構文解

析が開始する前に通知される、エンコード方式の矛盾による例外に関しては、XML-TEXT は長さ 0 であるか、文書のエンコード宣言の値のみを含みます。

サンプル XML テキスト
例外が発生した位置 (<sandwich> エレメントの終了タグの後の余分な「junk」) を含む、その個所までに構文解析した文書の部分です。

関連リファレンス
『XML-EVENT』(「*ILE COBOL 言語解説書*」)
『4.6 Predefined entities』(「*XML specification*」www.w3.org/TR/REC-xml#sec-predefined-ent)
『2.8 Prolog and document type declaration』(www.w3.org/TR/REC-xml#sec-prolog-dtd の「*XML specification*」)

XML を処理するプロシージャーの記述

独自の処理プロシージャーでは、XML イベントを処理するステートメントをコーディングしてください。

パーサーは、次の表に示すように、検出するイベントごとに情報をいくつかの特殊レジスターに入れて処理プロシージャーに渡します。これらのレジスターを使用して、データ構造を取り込み、また処理を制御します。

表 19. **XML パーサーの使用する特殊レジスター**

特殊レジスター	内容	暗黙的な定義および使用
XML-EVENT[1]	XML イベントの名前	PICTURE X(30) USAGE DISPLAY VALUE SPACE
XML-CODE	各 XML イベントに対し、例外コードまたは 0	PICTURE S9(9) USAGE BINARY VALUE ZERO
XML-TEXT[1]	XML PARSE ID で英数字データ項目を指定した場合、パーサーが見つけたイベントに対応する XML 文書のテキスト。	可変長の英数字データ項目。サイズ制限 16,000,000 バイト
XML-NTEXT[1]	XML PARSE ID で国別データ項目を指定した場合、パーサーが見つけたイベントに対応する XML 文書のテキスト。	可変長の国別データ項目。サイズ制限 16,000,000 バイト
1. この特殊レジスターを、受け取りデータ項目として使用できません。		

ネストされたプログラムで使用する場合、これらの特殊レジスターは、最外部のプログラムで暗黙のうちに GLOBAL として定義されます。

XML-CODE の内容の理解

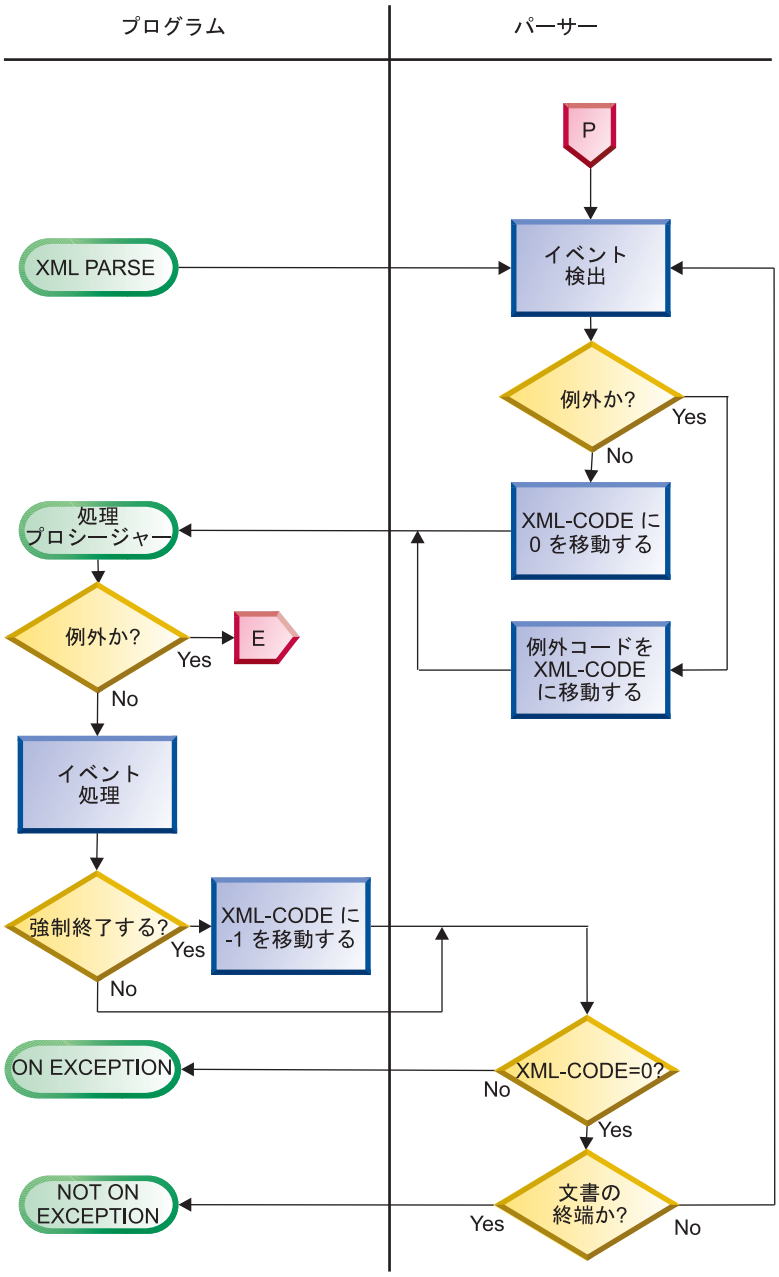
パーサーが XML PARSE ステートメントに制御を戻すとき、 XML-CODE はパーサーまたは処理プロシージャーが設定した最新の値を含みます。

EXCEPTION イベントを除くすべてのイベントでは、特殊レジスター XML-CODE の値は 0 です。 EXCEPTION 以外のイベントで、 XML パーサーに制御を戻す前に特殊レジスター XML-CODE を -1 に設定すると、戻された XML-CODE の -1 の値が示すユーザー開始の例外により、処理は停止されます。イベントから戻る前に XML-CODE を非ゼロ値に変更することによる結果は、定義されていません。

EXCEPTION イベントでは、特殊レジスター XML-CODE は例外コードを含みます。

次の図は、パーサーと処理プロシージャーの間の制御のフローと、両者の間の情報の受け渡しに XML-CODE がどのように使用されているかを示しています。 P のような他ページへの接続記号は、この章の複数の図表を接続しています。特に、次の図の E は、図表「XML 例外の制御フロー」に接続し、 P は、「XML CCSID 例外フロー制御」から接続されます。

XML-CODE の使用方法を示す、XML パーサーとプログラムの間の制御フロー



XML-TEXT および XML-NTEXT の使用

特殊レジスター XML-TEXT および XML-NTEXT は、相互に排他的です。ユーザーの指定する XML ID のタイプにより、どちらの特殊レジスターが設定されるかが決まります。ただし、ATTRIBUTE-NATIONAL-CHARACTER および CONTENT-NATIONAL-CHARACTER イベントは例外です。これらのイベントでは、XML PARSE の ID に指定したデータ項目に関わらず XML-NTEXT が設定されます。

パーサーが XML-TEXT を設定するときは、 XML-NTEXT は未定義 (長さ 0) です。パーサーが XML-NTEXT を設定するときは、 XML-TEXT は未定義 (長さ 0) です。

XML-NTEXT が含む国別文字の数を判別するには、 LENGTH 関数を使用します。 LENGTH OF XML-NTEXT は、XML-NTEXT で使用されている文字数ではなくバイト数を含みます。

XML テキストの COBOL データ項目への変換

XML データは固定長でなく、また形式も固定されていないため、 XML データを COBOL データ項目に移すには特殊な手法を使用する必要があります。

英数字項目の場合は、XML データが COBOL 項目に左寄せ (デフォルト) で入るか、右寄せで入るかを決定します。右寄せで入る場合、COBOL 項目の宣言で JUSTIFIED RIGHT 文節を指定します。

数値の XML 値に関しては、特別な考慮が必要です。特に、「$1,234.00」や「$1234」のような「修飾」された金額値の場合がそうです。これらは、XML では同じ意味ですが、COBOL の送り側フィールドとしての宣言は全く違います。次の手法を使用してください。

- シンプルではるかに柔軟性な方法として、英数字 XML データに対しては以下のものを使用してください。
  - 関数 NUMVAL を使用し、普通の数を表す XML データから、単純な数値を抽出およびデコードします。
  - 関数 NUMVAL-C を使用し、金額を示す XML データから数値を抽出およびデコードします。

  ただし、これらの関数を使用するとパフォーマンスに影響が出ることに注意してください。

処理プロシージャーの制約事項

ユーザーの処理プロシージャーは、直接 XML PARSE ステートメントを実行してはなりません。ただし、処理プロシージャーが CALL ステートメントを使用して最外部のプログラムに制御を渡す場合、そのターゲットのメソッドまたはプログラムは、同じ XML PARSE ステートメントや別の XML PARSE ステートメントを実行することができます。また、複数のスレッドで実行しているプログラムから、同じ XML ステートメントまたは別の XML ステートメントを同時に実行することができます。

処理プロシージャーの終了

コンパイラーは、処理プロシージャーの最後のステートメントの後に、戻るためのメカニズムを挿入します。処理プロシージャー中に STOP RUN ステートメントをコーディングすることで、実行単位を強制終了させることができます。ただし、GOBACK または EXIT PROGRAM ステートメントは、パーサーに制御を戻しません。処理プロシージャー中にいずれのステートメントを使用しても、重大エラーが発生します。

324 ページの『例: XML 構文解析』

関連リファレンス

699 ページの『継続可能な XML 例外』
703 ページの『継続不可能な XML 例外』
『XML-CODE』(「*ILE COBOL 言語解説書*」)
『XML-EVENT』(「*ILE COBOL 言語解説書*」)

『XML-NTEXT』(「*ILE COBOL 言語解説書*」)
『XML-TEXT』(「*ILE COBOL 言語解説書*」)

例: XML 構文解析

この例は、XML PARSE ステートメントおよび処理プロシージャーの基本的な編成を示します。ここでは、XML 文書は COBOL データ項目に入っています。構文解析のフローを追うことができるように、XML 文書はソースで挙げられています。プログラムの出力は下に挙げられています。文書とプログラムの出力を比較し、パーサーと処理プロシージャーの相互作用を追い、イベントと文書断片を突き合わせてください。

例: データ項目からの XML 構文解析

```
 Process APOST
 Identification division.
   Program-id. xmlsampl1.

 Data division.
  Working-storage section.
*****************************************************************
* XML document, encoded as initial values of data items.         *
*****************************************************************
   1 xml-document.
    2 pic x(39) value '<?xml version="1.0" encoding="ibm-37"'.
    2 pic x(19) value ' standalone="yes"?>'.
    2 pic x(39) value '<!--This document is just an example-->'.
    2 pic x(10) value '<sandwich>'.
    2 pic x(35) value '  <bread type="baker&apos;s best"/>'.
    2 pic x(41) value '  <?spread please use real mayonnaise  ?>'.
    2 pic x(31) value '  <meat>Ham &amp; turkey</meat>'.
    2 pic x(40) value '  <filling>Cheese, lettuce, tomato, etc.'.
    2 pic x(10) value '</filling>'.
    2 pic x(35) value '  <![CDATA[We should add a <relish>'.
    2 pic x(22) value ' element in future!]]>'.
    2 pic x(31) value '  <listprice>$4.99 </listprice>'.
    2 pic x(27) value '  <discount>0.10</discount>'.
    2 pic x(11) value '</sandwich>'.
   1 xml-document-length computational pic 999.

*****************************************************************
* Sample data definitions for processing numeric XML content.    *
*****************************************************************
   1 current-element pic x(30).
   1 list-price computational pic 9v99 value 0.
   1 discount computational pic 9v99 value 0.
   1 display-price pic $$9.99.

 Procedure division.
  mainline section.

     XML PARSE xml-document PROCESSING PROCEDURE xml-handler
       ON EXCEPTION
         display 'XML document error ' XML-CODE
       NOT ON EXCEPTION
         display 'XML document successfully parsed'
     END-XML

*****************************************************************
*    Process the transformed content and calculate promo price.  *
*****************************************************************
     display ' '
     display '-----++++++++++ Using information from XML '
         '++++++++++-----'
```

```
     display ' '
     move list-price to display-price
     display '  Sandwich list price: ' display-price
     compute display-price = list-price * (1 - discount)
     display '  Promotional price:   ' display-price
     display '  Get one today!'

     goback.

  xml-handler section.
     evaluate XML-EVENT
* ==> Order XML events most frequent first
       when 'START-OF-ELEMENT'
         display 'Start element tag: <' XML-TEXT '>'
         move XML-TEXT to current-element
       when 'CONTENT-CHARACTERS'
         display 'Content characters: <' XML-TEXT '>'
* ==> Transform XML content to operational COBOL data item...
         evaluate current-element
           when 'listprice'
* ==> Using function NUMVAL-C...
             compute list-price = function numval-c(XML-TEXT)
           when 'discount'
             compute discount = function numval-c(XML-TEXT)
         end-evaluate
       when 'END-OF-ELEMENT'
         display 'End element tag: <' XML-TEXT '>'
         move spaces to current-element
       when 'START-OF-DOCUMENT'
         compute xml-document-length = function length(XML-TEXT)
         display 'Start of document: length=' xml-document-length
             ' characters.'
       when 'END-OF-DOCUMENT'
         display 'End of document.'
       when 'VERSION-INFORMATION'
         display 'Version: <' XML-TEXT '>'
       when 'ENCODING-DECLARATION'
         display 'Encoding: <' XML-TEXT '>'
       when 'STANDALONE-DECLARATION'
         display 'Standalone: <' XML-TEXT '>'
       when 'ATTRIBUTE-NAME'
         display 'Attribute name: <' XML-TEXT '>'
       when 'ATTRIBUTE-CHARACTERS'
         display 'Attribute value characters: <' XML-TEXT '>'
       when 'ATTRIBUTE-CHARACTER'
         display 'Attribute value character: <' XML-TEXT '>'
       when 'START-OF-CDATA-SECTION'
         display 'Start of CData: <' XML-TEXT '>'
       when 'END-OF-CDATA-SECTION'
         display 'End of CData: <' XML-TEXT '>'
       when 'CONTENT-CHARACTER'
         display 'Content character: <' XML-TEXT '>'
       when 'PROCESSING-INSTRUCTION-TARGET'
         display 'PI target: <' XML-TEXT '>'
       when 'PROCESSING-INSTRUCTION-DATA'
         display 'PI data: <' XML-TEXT '>'
       when 'COMMENT'
         display 'Comment: <' XML-TEXT '>'
       when 'EXCEPTION'
         compute xml-document-length = function length (XML-TEXT)
         display 'Exception ' XML-CODE ' at offset '
             xml-document-length '.'
       when other
         display 'Unexpected XML event: ' XML-EVENT '.'
     end-evaluate
     .
 End program xmlsampl1.
```

構文解析例の出力: 次の出力から、構文解析のどのイベントが、文書のどの断片から起こるかを知ることができます。

```
Start of document: length=390 characters.
Version: <1.0>
Encoding: <ibm-37>
Standalone: <yes>
Comment: <This document is just an example>
Start element tag: <sandwich>
Content characters: <  >
Start element tag: <bread>
Attribute name: <type>
Attribute value characters: <baker>
Attribute value character: <'>
Attribute value characters: <s best>
End element tag: <bread>
Content characters: <  >
PI target: <spread>
PI data: <please use real mayonnaise  >
Content characters: <  >
Start element tag: <meat>
Content characters: <Ham >
Content character: <&>
Content characters: < turkey>
End element tag: <meat>
Content characters: <  >
Start element tag: <filling>
Content characters: <Cheese, lettuce, tomato, etc.>
End element tag: <filling>
Content characters: <  >
Start of CData: <<![CDATA[>
Content characters: <We should add a <relish> element in future!>
End of CData: <]]>>
Content characters: <  >
Start element tag: <listprice>
Content characters: <$4.99 >
End element tag: <listprice>
Content characters: <  >
Start element tag: <discount>
Content characters: <0.10>
End element tag: <discount>
End element tag: <sandwich>
End of document.
XML document successfully parsed

-----++++++++++ Using information from XML **********-----

  Sandwich list price:  $4.99
  Promotional price:    $4.49
  Get one today!
```

例: IFS ファイルからの XML 構文解析

この例は、IFS ファイル中にある XML 文書を構文解析する XML PARSE ステートメントを示します。プログラムの出力は、前の例と同じです。 IFS ファイルには、有効な CCSID が必要です。 IFS ファイルの各行の終わりには CR (復帰) のみが必要であり、LF (改行) があってはなりません。

```
 Process APOST
 Identification division.
   Program-id. xmlsampl2.

 Data division.
  Working-storage section.
*****************************************************************
```

```
* XML document, encoded as initial values of data items.         *
*****************************************************************
   1 xml-id pic x(27) value '/home/user1/xmlsampldoc.xml'.
   1 xml-document-length computational pic 999.

*****************************************************************
* Sample data definitions for processing numeric XML content.    *
*****************************************************************
   1 current-element pic x(30).
   1 list-price computational pic 9v99 value 0.
   1 discount computational pic 9v99 value 0.
   1 display-price pic $$9.99.

 Procedure division.
  mainline section.

     XML PARSE FILE-STREAM xml-id PROCESSING PROCEDURE xml-handler
       ON EXCEPTION
         display 'XML document error ' XML-CODE
       NOT ON EXCEPTION
         display 'XML document successfully parsed'
     END-XML

*****************************************************************
*    Process the transformed content and calculate promo price.  *
*****************************************************************
     display ' '
     display '-----++++++++++ Using information from XML '
         '++++++++++-----'
     display ' '
     move list-price to display-price
     display '  Sandwich list price: ' display-price
     compute display-price = list-price * (1 - discount)
     display '  Promotional price:   ' display-price
     display '  Get one today!'

     goback.

  xml-handler section.
     evaluate XML-EVENT
* ==> Order XML events most frequent first
       when 'START-OF-ELEMENT'
         display 'Start element tag: <' XML-TEXT '>'
         move XML-TEXT to current-element
       when 'CONTENT-CHARACTERS'
         display 'Content characters: <' XML-TEXT '>'
* ==> Transform XML content to operational COBOL data item...
         evaluate current-element
           when 'listprice'
* ==> Using function NUMVAL-C...
              compute list-price = function numval-c(XML-TEXT)
           when 'discount'
             compute discount = function numval-c(XML-TEXT)
         end-evaluate
       when 'END-OF-ELEMENT'
         display 'End element tag: <' XML-TEXT '>'
         move spaces to current-element
       when 'START-OF-DOCUMENT'
         compute xml-document-length = function length(XML-TEXT)
         display 'Start of document: length=' xml-document-length
             ' characters.'
       when 'END-OF-DOCUMENT'
         display 'End of document.'
       when 'VERSION-INFORMATION'
         display 'Version: <' XML-TEXT '>'
       when 'ENCODING-DECLARATION'
         display 'Encoding: <' XML-TEXT '>'
```

```
        when 'STANDALONE-DECLARATION'
          display 'Standalone: <' XML-TEXT '>'
        when 'ATTRIBUTE-NAME'
          display 'Attribute name: <' XML-TEXT '>'
        when 'ATTRIBUTE-CHARACTERS'
          display 'Attribute value characters: <' XML-TEXT '>'
        when 'ATTRIBUTE-CHARACTER'
          display 'Attribute value character: <' XML-TEXT '>'
        when 'START-OF-CDATA-SECTION'
          display 'Start of CData: <' XML-TEXT '>'
        when 'END-OF-CDATA-SECTION'
          display 'End of CData: <' XML-TEXT '>'
        when 'CONTENT-CHARACTER'
          display 'Content character: <' XML-TEXT '>'
        when 'PROCESSING-INSTRUCTION-TARGET'
          display 'PI target: <' XML-TEXT '>'
        when 'PROCESSING-INSTRUCTION-DATA'
          display 'PI data: <' XML-TEXT '>'
        when 'COMMENT'
          display 'Comment: <' XML-TEXT '>'
        when 'EXCEPTION'
          compute xml-document-length = function length (XML-TEXT)
          display 'Exception ' XML-CODE ' at offset '
              xml-document-length '.'
        when other
          display 'Unexpected XML event: ' XML-EVENT '.'
      end-evaluate
      .
 End program xmlsampl2.
```

以下が /home/user1/xmlsampldoc.xml にある、この例の IFS ファイルです。

```
<?xml version="1.0" encoding="ibm-37"
  standalone="yes"?>
<!--This document is just an example-->
<sandwich>
 <bread type="baker&apos;s best"/>
 <?spread please use real mayonnaise  ?>
 <meat>Ham &amp; turkey</meat>
 <filling>Cheese, lettuce, tomato, etc.
</filling>
 <![CDATA[We should add a <relish> element in future!]]>
 <listprice>$4.99 </listprice>
 <discount>0.10</discount>
</sandwich>
```

XML 文書のエンコード方式の理解

XML PARSE ステートメントは、以下のデータ項目タイプの 1 つを含む XML 文書のみをサポートします。

- Unicode UCS-2 でエンコードされた国別データ項目
- サポートされている EBCDIC または ASCII の 1 バイト文字セットの 1 つでエンコードされた英数字データ項目

XML 文書がエンコード宣言を含む場合は、それが XML PARSE ステートメントで提供されているエンコード情報および文書の基本的なエンコード方式と一致させてください。 パーサーは、最大 3 つの情報源を以下の順番で使用し、文書のエンコード方式を判断します。

1. 文書の最初のいくつかの文字
2. XML PARSE ステートメントが提供するエンコード情報

3. ステップ 2 が成功した場合、文書中のエンコード宣言

つまり、XML 文書が、サポートされているコード・ページの 1 つを指定したエンコード宣言を含んだ XML 宣言で始まる場合は、パーサーは、エンコード宣言が文書の基本エンコード方式または XML PARSE ステートメントのエンコード情報と矛盾しない限り、それを考慮します。

XML 文書が XML 宣言を持たない場合、または XML 宣言がエンコード宣言を省略している場合は、パーサーは XML PARSE ステートメントのエンコード情報が文書の基本エンコード方式と矛盾しない限り、それを使用して文書を処理します。

パーサーは、これらの情報源の間に矛盾を検出すると、XML 例外イベントを通知します。

コード・ページの指定

大部分の XML 宣言が開始する XML 宣言において、文書のエンコード情報を指定することができます。これは、エンコード宣言を含む XML 宣言の例です。

```
<?xml version="1.0" encoding="ibm-1140" ?>
```

以下の方法のどちらかで、エンコード宣言を指定します。

- 以下のいずれか (大文字、小文字またはその任意の組み合わせで) を接頭部に付けた CCSID 番号 (先行ゼロなしでまたは任意の数の先行ゼロを付けて) を指定します。
  - IBM-
  - IBM_
  - CCSID-
  - CCSID_
- 以下の、サポートされた別名 (大文字、小文字、またはその任意の組み合わせで) を使用します。

表 20. **XML エンコード宣言の別名**

コード・ページ	サポートされる別名
037	EBCDIC-CP-US、EBCDIC-CP-CA、EBCDIC-CP-WT、EBCDIC-CP-NL
500	EBCDIC-CP-BE、EBCDIC-CP-CH
813	iso-8859-7、iso_8859-7
819	iso-8859-1、iso_8859-1
920	iso-8859-9、iso_8859-9

他のコード・ページの文書の構文解析

明示的にサポートされている 1 バイト・コード・ページ以外のコード・ページでエンコードされている XML 文書を構文解析することができます。そのためには、対象の文書が COBOL データ項目にある際に、MOVE ステートメントを使用してその文書を Unicode UCS-2 に変換し、国別データ項目に入れます。 XML 文書が IFS ファイルに含まれている場合は、オブジェクトのコピー (CPY) コマンドを使用してその XML 文書をコピーし、National CCSID コンパイラー・オプション、または

NTLCCSID PROCESS オプションで指定した UCS-2 CCSID に変換します。次に、必要に応じて、特殊レジスター XML-NTEXT で処理プロシージャーに渡される文書テキストの各断片を、MOVE ステートメントで元のコード・ページに変換することができます。

関連リファレンス
『XML 文書用コード化文字セット』(「*ILE COBOL 言語解説書*」)

XML 文書中のエラーの処理

XML 文書中のエラーを処理するには、以下の機能を使用してください。

- ユーザーの処理プロシージャー
- XML PARSE ステートメントの ON EXCEPTION 句
- 特殊レジスター XML-CODE

XML パーサーが XML 文書中にエラーを検出すると、XML 例外イベントを生成します。パーサーは、処理プロシージャーに、以下の情報とともに制御を渡すことで、この例外イベントを戻します。

- 特殊レジスター XML-EVENT は「EXCEPTION」を含みます。
- 特殊レジスター XML-CODE は、数値の例外コードを含みます。
- XML-TEXT は、例外が検出されたポイントを含む、そのポイントまでの文書テキストを含みます。

エラー・コードの値が以下の範囲のいずれかのうちにある場合は、

- 1 ～ 99
- 100,001 ～ 165,535
- 200,001 ～ 265,535

例外をユーザーの処理プロシージャー中で処理し、構文解析を継続することが可能なことがあります。エラー・コードが他の非ゼロ値を持っている場合は、構文解析を継続することはできません。エンコード方式の矛盾に関する例外 (50 ～ 99 および 300 ～ 499) は、文書の構文解析が始まる前に通知されます。これの例外に関しては、XML-TEXT は長さ 0 であるか、文書のエンコード宣言の値のみを含みます。

1 ～ 49 の範囲の例外は、XML 仕様に規定された致命的エラーなので、ユーザーが例外を処理した場合でもパーサーは構文解析を継続しません。ただし、パーサーは、文書の終端に達するか、継続不能なエラーを検出するまで、追加のエラーの走査は続けます。これらの例外に関しては、パーサーはそれ以降の通常イベントは、END-OF-DOCUMENT イベントを除いて通知しません。

次の図は、パーサーと処理プロシージャーの間の制御のフローを理解するためのものです。特定の例外を処理する方法と、 XML-CODE を使用して例外を識別する方法を示しています。 Ⓟ のような他ページへの接続記号は、この章の複数の図表を接続しています。特に、次の図の Ⓒ は、図表「XML CCSID 例外フロー制御」に接続しています。この図では、 Ⓔ / Ⓔ は、他ページへの接続記号と、同ページ中の接続記号の両方の働きを持ちます。

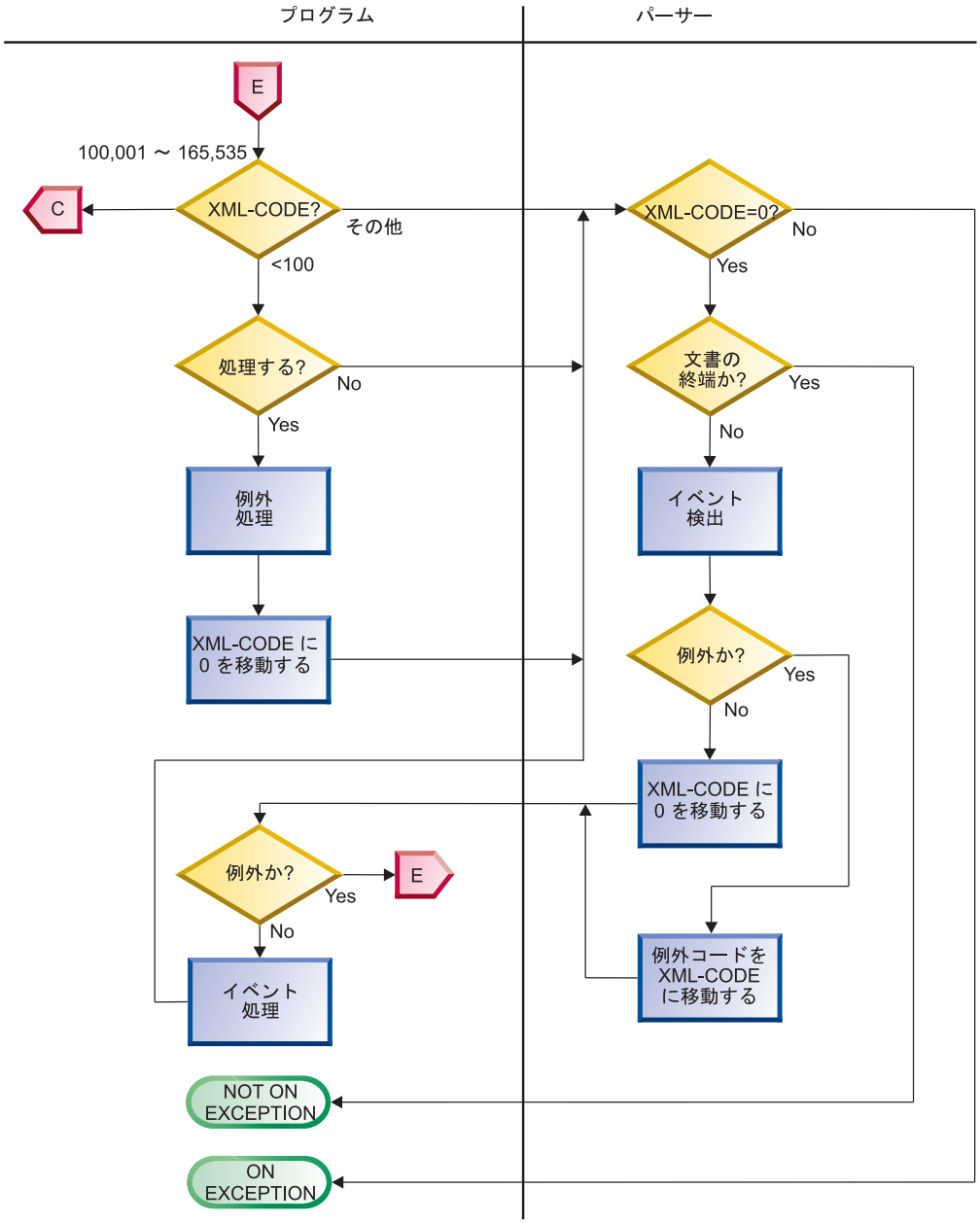


図 69. **XML 例外の制御フロー**

未処理の例外

例外を処理したくない場合は、 XML-CODE の値を変更せずにパーサーに制御を戻します。パーサーは、ユーザーが ON EXCEPTION 句で指定したステートメントに制御を転送します。 ON EXCEPTION 句をコーディングしない場合は、制御は XML PARSE ステートメントの終わりに転送されます。

例外の処理

ユーザーの処理プロシージャーで例外イベントを処理するには、以下のステップに従います。

1. XML-CODE の中身を使用して、アクションを決定します。

2. 例外を処理したことを示すには、XML-CODE を 0 に設定します。
3. パーサーに制御を戻します。これにより例外条件はもう存在しません。

構文解析終了までの間に未処理の例外が発生しなかった場合は、 NOT ON EXCEPTION 句 (通常の構文解析終了) で指定したステートメントに制御が渡されます。 NOT ON EXCEPTION 句をコーディングしない場合は、制御は XML PARSE ステートメントの終わりに渡されます。特殊レジスター XML-CODE は 0 を含みます。

XML-CODE で渡された例外コードが以下のいずれかの範囲内である場合にのみ、このような方法で例外を処理することができます。

- 1 ～ 99
- 100,001 ～ 165,535
- 200,001 ～ 265,535

そうでない場合は、パーサーはそれ以上のイベントを通知せず、 ON EXCEPTION 句で指定したステートメントに制御を渡します。この場合は、処理プロシージャーでパーサーに制御を戻す前に XML-CODE を 0 に設定したときも、XML-CODE は元の例外番号を含みます。

XML-CODE を元の例外コードとは異なる非ゼロ値に設定して、パーサーに制御を戻した場合の結果は定義されていません。

構文解析の強制終了

通常の XML イベント (EXCEPTION イベントでないイベント) からパーサーに戻る前に、ユーザーの処理プロシージャーで XML-CODE を -1 に設定することで、構文解析を意図的に強制終了させることができます。この手法は、ユーザーの目的上十分な文書の量を見終わったか、それ以上の処理が無意味となるような文書の不規則性を検出した場合に使うことができます。

その場合、例外条件が存在しますが、パーサーはそれ以降のイベントを通知しません。そのため、制御は ON EXCEPTION 句が指定されている場合、そこに戻ります。その個所で XML-CODE が -1 であるかを調べることで、構文解析を意図的に強制終了したかをテストできます。 ON EXCEPTION 句を指定していない場合は、制御は XML PARSE ステートメントの最後に戻ります。

また、どの例外 XML イベントのあとでも、XML-CODE を変更せずにパーサーに戻ることで構文解析を強制終了することができます。その結果は意図的な強制終了と類似していますが、パーサーが XML-CODE に例外番号を含んだまま XML PARSE ステートメントに戻る点が違います。

CCSID 矛盾例外

XML-CODE 中の例外コードが 100,001 ～ 165,535 または 200,001 ～ 265,535 の範囲内である例外イベントは、特殊なケースに当てはまります。 この範囲の例外コードは、文書の CCSID (エンコード宣言を含む、文書の開始部を調べることで判別する) は XML PARSE ステートメントの CCSID と矛盾していることを示します。

この場合は、XML-CODE の値から 100,000 または 200,000 を減算する (それぞれ、EBCDIC CCSID の場合と ASCII CCSID の場合) ことで文書の CCSID を判別できます。例えば、XML-CODE が 101,140 を含む場合、文書の CCSID は 1140 です。

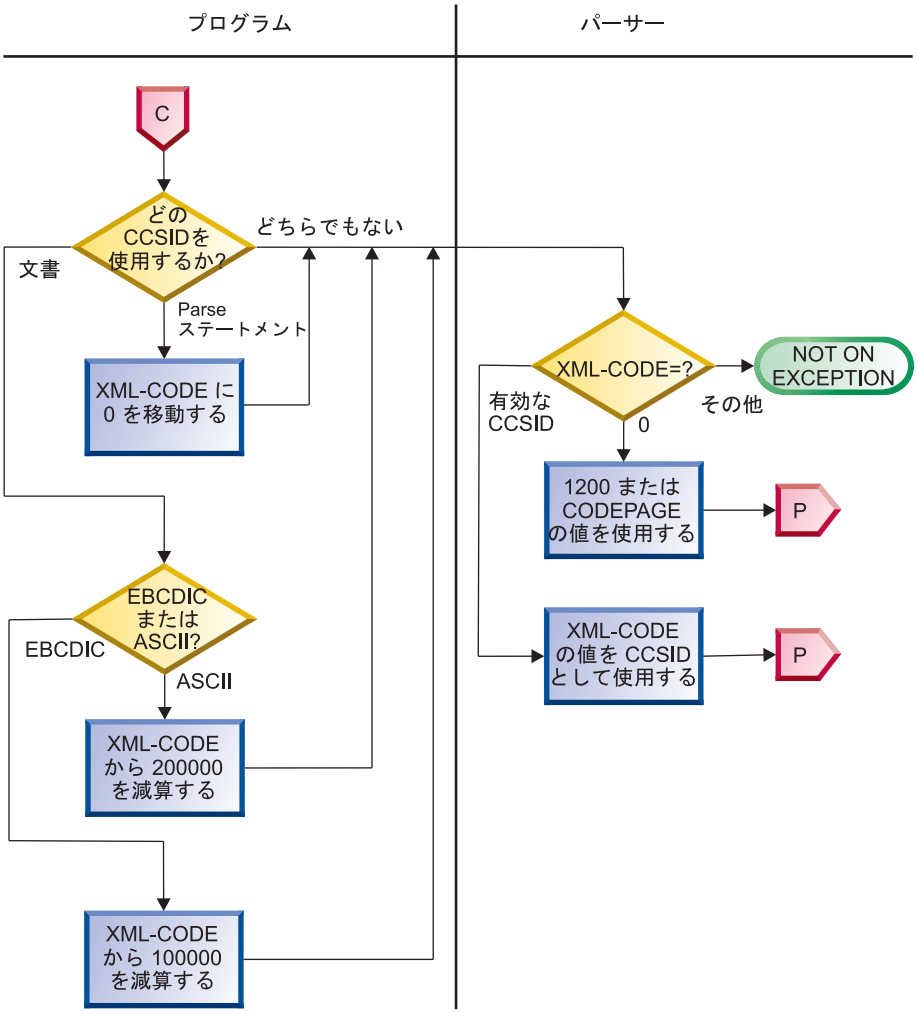
XML PARSE ステートメントの CCSID は、 XML PARSE ID のタイプにより異なります。 ID が国別データ項目の場合、CCSID は National CCSID コンパイラー・オプション、または NTLCCSID PROCESS オプションで指定され、それは Unicode を示します。 XML PARSE ID が英数字の場合、 CCSID は COBOL ソース・メンバーの CCSID です。

パーサーは、CCSID 矛盾例外イベントのために処理プロシージャーから戻った後、以下の 3 つのアクションの 1 つを取ります。

- XML-CODE を 0 に設定すると、パーサーは XML PARSE ステートメントの CCSID (つまり国別項目の場合は National CCSID コンパイラー・オプション、または NTLCCSID PROCESS オプションで指定された CCSID、そうでない場合は COBOL ソース・メンバーの CCSID) を使用します。
- XML-CODE を文書の CCSID に設定すると (つまり、元の XML-CODE 値から 100,000 または 200,000 の適切な方を引いたもの)、パーサーは文書の CCSID を使用します。これは、ユーザーの処理プロシージャーから戻るときに XML-CODE が非ゼロ値を持っていてもパーサーが継続する、唯一のケースです。
- それ以外の場合は、パーサーは文書の処理を停止し、例外条件と共に XML PARSE ステートメントに制御を戻します。 XML-CODE は、元々例外イベントに渡された例外コードを含みます。

次の図は、これらのアクションを示しています。 Ⓟ のような他ページへの接続記号は、この章の複数の図表を接続しています。特に、次の図の Ⓟ は、「XML-CODE の使用方法を示す、XML パーサーとプログラムの間の制御フロー」に接続し、 Ⓒ は、XML 例外の制御フロー」から接続されます。

XML CCSID 例外のフロー制御



関連リファレンス

699 ページの『継続可能な XML 例外』
703 ページの『継続不可能な XML 例外』
699 ページの『付録 F. XML 参照資料』

『XML-CODE』(「*ILE COBOL 言語解説書*」)

第 12 章 XML 出力の作成

XML 出力は、XML GENERATE ステートメントを使用して、COBOL プログラムから作成することができます。XML GENERATE ステートメントでは、生成された XML 出力の文字数のカウントを受け取るフィールド、および例外が発生した場合に制御を受け取るステートメントを示すこともできます。

XML 出力を作成するために、以下を行うことができます。

- XML GENERATE ステートメントを使用して、ソースおよびターゲットのデータ項目、カウント・フィールド、および ON EXCEPTION ステートメントを示す。
- 特殊レジスター XML-CODE を使用して、XML 生成の状況を決定する。
- 別の方法として、XML 文書を IFS ストリーム・ファイルに作成し、XML GENERATE ステートメントの形式 2 を使用して XML 文書を生成する。

COBOL データ項目を XML に変換した後に、作成された XML 出力をさまざまな方法 (例えば、XML 出力を WebSphere MQ にメッセージとして渡す、または後で変換を行うために、XML 出力を CICS 通信域に送信する、など) で使用することができます。

関連作業
『XML 出力の生成』
341 ページの『XML 出力の拡張』
346 ページの『生成された XML 出力のエンコードの制御』
346 ページの『XML 出力の生成中のエラーの処理』

XML 出力の生成

COBOL データを XML に変換するには、以下の例のように、XML GENERATE ステートメントを使用します。

```
XML GENERATE XML-OUTPUT FROM SOURCE-REC
      COUNT IN XML-CHAR-COUNT
  ON EXCEPTION
    DISPLAY 'XML generation error ' XML-CODE
    STOP RUN
  NOT ON EXCEPTION
    DISPLAY 'XML document was successfully generated.'
END-XML
```

XML GENERATE ステートメントで、XML 出力を受け取るデータ項目 (この例では XML-OUTPUT) を最初に示します。生成された XML 出力を含めるのに十分な大きさになるようにデータ項目を定義します。一般には、COBOL ソース・データのデータ名の長さに応じて、そのデータ・サイズの 5 倍から 8 倍にします。

DATA DIVISION では、受取 ID を英数字 (英数字グループ項目またはカテゴリー英数字の基本項目のいずれか)、または国別 (カテゴリー国別の基本項目) として宣言することができます。

XML 出力に 以下のいずれかの特性を持つ COBOL ソース・レコードのデータを含む場合は、受取 ID を国別にする必要があります。

- 国別クラスまたは DBCS クラスである。
- DBCS 名を持っている (すなわち、DBCS 文字を含む名前を持つデータ項目)。

次に、XML 形式に変換されるソース・データ項目 (この例では SOURCE-REC) を示します。このソース・データ項目は、英数字グループ項目、または英数字クラスあるいは国別クラスの基本データ項目のいずれかにすることができます。 そのデータ項目のデータ記述では RENAMES 文節を指定しないでください。

ソース・データ項目が英数字グループ項目の場合、そのソース・データ項目は、基本項目ではなく、グループ項目として処理されます。ソース・データ項目に従属するグループはすべて、グループ項目としても処理されます。

一部の COBOL データ項目は XML に変換されずに、無視されます。無視されるのは、XML に変換する英数字グループ項目の従属データ項目が以下の場合です。

- REDEFINES 文節を指定している、またはそのような再定義項目に従属している。
- RENAMES 文節を指定している。

以下の XML を生成する場合は、ソース・データ項目内のこれらの項目も無視されます。

- 基本 FILLER (または無名) データ項目
- SYNCHRONIZED データ項目用に挿入された遊びバイト

XML を生成する場合は、無視されない基本データ項目が少なくとも 1 つ必要です。無視されないデータ項目の場合は、XML に変換される ID を DATA DIVISION で宣言するときに、その ID が以下の条件を満たしていることを確認してください。

- 各基本データ項目が、索引データ項目であるか、以下のクラスに属している。
  - 英字
  - 英数字
  - DBCS
  - 数字
  - 国別

  すなわち、どの基本データ項目も USAGE POINTER 句または USAGE PROCEDURE-POINTER 句では記述されません。
- FILLER 以外の各データ名は、直接それを含んでいるグループがあれば、その中では固有である。
- Unicode に変換された DBCS データ名が、XML 仕様 バージョン 1.0 の名前として妥当である。

XML 宣言は生成されません。生成された XML を読みやすくするための空白文字 (例えば、改行やインデント) は挿入されません。

必要であれば、COUNT IN 句をコーディングして、XML 出力の生成中に充てんされた XML 文字位置の数を取得することができます。カウント・フィールドを、

PICTURE ストリング内の記号 P を持たない整数データ項目として宣言します。カウント・フィールドと参照変更を使用すると、受取データ項目のうちの、生成された XML 出力を含む部分だけを取得することができます。例えば、XML-OUTPUT(1:XML-CHAR-COUNT) は、XML-OUTPUT の最初の XML-CHAR-COUNT 文字位置を参照しています。

さらに、以下の句のいずれかまたは両方を指定して、XML 文書の生成後に制御を受け取ることもできます。

- ON EXCEPTION。XML の生成中にエラーが発生した場合に制御を受け取るための句です。
- NOT ON EXCEPTION。エラーが発生しなかった場合に制御を受け取る句です。

XML GENERATE ステートメントを明示範囲終了符号 END-XML を使用して終了することができます。条件ステートメントで ON EXCEPTION 句または NOT ON EXCEPTION 句を持つ XML GENERATE ステートメントをネストする場合は、END-XML をコーディングします。

XML 生成は、COBOL ソース・レコードが XML に変換されてしまうか、エラーが発生するまで続きます。エラーが発生した場合、結果は以下のようになります。

- 特殊レジスター XML-CODE にゼロ以外の例外コードが含まれる。
- ON EXCEPTION 句に制御が渡されるか (そのように指定されていた場合)、そのように指定されていなかった場合は、XML GENERATE ステートメントの終わりに渡される。

XML の生成中にエラーが発生しなかった場合は、特殊レジスター XML-CODE にゼロが含まれ、制御は NOT ON EXCEPTION 句に渡されるか (そのように指定されている場合)、そのように指定されていない場合は、XML GENERATE ステートメントの終わりに渡されます。

『例: XML の生成』

関連作業
346 ページの『生成された XML 出力のエンコードの制御』
346 ページの『XML 出力の生成中のエラーの処理』

関連リファレンス
『データのクラスおよびカテゴリー』(「*ILE COBOL 言語解説書*」)
『XML GENERATE ステートメント』(「*ILE COBOL 言語解説書*」)
『XML GENERATE の操作』(「*ILE COBOL 言語解説書*」)

例: XML の生成

次の例では、グループ・データ項目内の購買注文の作成をシミュレートし、その購買注文の XML バージョンを生成します。

プログラム XGFX は、XML GENERATE を使用して、ソース・レコードのグループ・データ項目 purchaseOrder から基本データ項目 xmlPO に XML 出力を生成します。ソース・レコード内の基本データ項目は、必要に応じて文字フォーマットに変換され、ソース・レコード内のデータ名から派生した名前を持つ XML 要素に挿入されます。

XGFX はプログラム Pretty を呼び出します。このプログラムは、XML PARSE ステートメントを処理プロシージャー p で使用して、XML の内容をより簡単に検証できるように、改行とインデントを使用して XML 出力をフォーマットします。

プログラム XGFX

```
PROCESS NOMONOPRC.
Identification division.
  Program-id. XGFX.
Data division.
 Working-storage section.
   01 numItems pic 99 global.
   01 purchaseOrder global.
     05 orderDate pic x(10).
     05 shipTo.
       10 country pic xx value 'US'.
       10 name pic x(30).
       10 street pic x(30).
       10 city pic x(30).
       10 state pic xx.
       10 zip pic x(10).
     05 billTo.
       10 country pic xx value 'US'.
       10 name pic x(30).
       10 street pic x(30).
       10 city pic x(30).
       10 state pic xx.
       10 zip pic x(10).
     05 orderComment pic x(80).
     05 items occurs 0 to 20 times depending on numItems.
       10 item.
         15 partNum pic x(6).
         15 productName pic x(50).
         15 quantity pic 99.
         15 USPrice pic 999v99.
         15 shipDate pic x(10).
         15 itemComment pic x(40).
   01 numChars comp pic 9(9).
   01 xmlPO pic x(999).
Procedure division.
  m.
    Move 20 to numItems
    Move spaces to purchaseOrder

    Move '1999-10-20' to orderDate

    Move 'US' to country of shipTo
    Move 'Alice Smith' to name of shipTo
    Move '123 Maple Street' to street of shipTo
    Move 'Mill Valley' to city of shipTo
    Move 'CA' to state of shipTo
    Move '90952' to zip of shipTo

    Move 'US' to country of billTo
    Move 'Robert Smith' to name of billTo
    Move '8 Oak Avenue' to street of billTo
    Move 'Old Town' to city of billTo
    Move 'PA' to state of billTo
    Move '95819' to zip of billTo
    Move 'Hurry, my lawn is going wild!' to orderComment

    Move 0 to numItems
    Call 'addFirstItem'
    Call 'addSecondItem'
    Move space to xmlPO
    Xml generate xmlPO from purchaseOrder count in numChars
```

```
    Call 'PRETTY' using xmlPO numChars
    Goback
    .

Identification division.
  Program-id. 'addFirstItem'.
Procedure division.
    Add 1 to numItems
    Move '872-AA' to partNum(numItems)
    Move 'Lawnmower' to productName(numItems)
    Move 1 to quantity(numItems)
    Move 148.95 to USPrice(numItems)
    Move 'Confirm this is electric' to itemComment(numItems)
    Goback.
End program 'addFirstItem'.

Identification division.
  Program-id. 'addSecondItem'.
Procedure division.
    Add 1 to numItems
    Move '926-AA' to partNum(numItems)
    Move 'Baby Monitor' to productName(numItems)
    Move 1 to quantity(numItems)
    Move 39.98 to USPrice(numItems)
    Move '1999-05-21' to shipDate(numItems)
    Goback.
End program 'addSecondItem'.

End program XGFX.
```

プログラム Pretty

```
Identification division.
  Program-id. Pretty.
Data division.
 Working-storage section.
   01 prettyPrint.
     05 pose pic 999.
     05 posd pic 999.
     05 depth pic 99.
     05 element pic x(30).
     05 indent pic x(20).
     05 buffer pic x(100).
 Linkage section.
  1 doc.
   2 pic x occurs 1 to 16384 times depending on len.
  1 len comp pic 9(9).
Procedure division using doc len.
  m.
    Move space to prettyPrint
    Move 0 to depth posd
    Move 1 to pose
    Xml parse doc processing procedure p
    Goback.
  p.
    Evaluate xml-event
      When 'START-OF-ELEMENT'
        If element not = space
          If depth > 1
            Display indent(1:2 * depth - 2) buffer(1:pose - 1)
          Else
            Display buffer(1:pose - 1)
          End-if
        End-if
        Move xml-text to element
        Add 1 to depth
        Move 1 to pose
```

```
                String '<' xml-text '>' delimited by size into buffer
                    with pointer pose
                Move pose to posd
              When 'CONTENT-CHARACTERS'
                String xml-text delimited by size into buffer
                    with pointer posd
              When 'CONTENT-CHARACTER'
                String xml-text delimited by size into buffer
                    with pointer posd
              When 'END-OF-ELEMENT'
                Move space to element
                String '</' xml-text '>' delimited by size into buffer
                    with pointer posd
                If depth > 1
                  Display indent(1:2 * depth - 2) buffer(1:posd - 1)
                Else
                  Display buffer(1:posd - 1)
                End-if
                Subtract 1 from depth
                Move 1 to posd
              When other
                Continue
            End-evaluate
            .
End program Pretty.
```

プログラム XGFX の出力

```
<purchaseOrder>
  <orderDate>1999-10-20</orderDate>
  <shipTo>
    <country>US</country>
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  <billTo>
    <country>US</country>
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
    <zip>95819</zip>
  </billTo>
  <orderComment>Hurry, my lawn is going wild!</orderComment>
  <items>
    <item>
      <partNum>872-AA</partNum>
      <productName>Lawnmower</productName>
      <quantity>1</quantity>
      <USPrice>148.95</USPrice>
      <shipDate> </shipDate>
      <itemComment>Confirm this is electric</itemComment>
    </item>
  </items>
  <items>
    <item>
      <partNum>926-AA</partNum>
      <productName>Baby Monitor</productName>
      <quantity>1</quantity>
      <USPrice>39.98</USPrice>
      <shipDate>1999-05-21</shipDate>
```

```
      <itemComment> </itemComment>
    </item>
  </items>
</purchaseOrder>
```

関連リファレンス
『XML GENERATE の操作』(「*ILE COBOL 言語解説書*」)

XML 出力の拡張

XML 形式で表現しようとしている情報が DATA DIVISION 内のグループ項目にすでに存在しているのに、1 つ以上の要因のために、XML 文書の作成にその項目を直接使用することができない場合があります。

例を以下に示します。

- 必要なデータの他に、XML 出力文書には関係のない値を含む従属データ項目がこの項目にある。
- 必要なデータ項目の名前が外部表示として適切でなく、プログラマーに対してしか意味がない。
- データの定義が、要求されているデータ・タイプではない。適切なフォーマットを持っているのは、(XML GENERATE ステートメントによって無視される) 再定義だけである可能性があります。
- 無関係な従属グループにおいて、必要なデータ項目のネストが深すぎる。XML 出力は、階層構造にするのではなく (デフォルトでは階層構造)、「フラット」にする必要があります。
- 必要なデータ項目が非常に多くのコンポーネントに分割されているので、それらのデータ項目は、それらを収容するグループの内容として出力する必要がある。
- 必要な情報がグループ項目に含まれているけれども、その順序が間違っている。

以上のような状態を処理するには、さまざまな方法があります。考えられる 1 つの手法は、適切な特性を持つ新規のデータ項目を定義し、必要なデータをこの新規のデータ項目の該当するフィールドに移動することです。ただし、このアプローチは少し面倒で、オリジナルのデータ項目と新規のデータ項目が常に同期しているように注意深く保守する必要があります。

それよりももう少し利点のある代替方法として、オリジナルのグループ・データ項目を再定義し、その再定義から XML 出力を生成する方法があります。そのためには、オリジナルのデータ記述セットを出発点にして、以下の変更を行います。

- 基本データ項目の名前を FILLER に変更するか、それらの名前を削除して、生成された XML からそれらの基本データ項目を除外します。
- 選択された基本項目およびそれらを含むグループ項目に、より意味のある適切な名前を付けます。
- 不必要な中間のグループ項目を除去し、階層をフラットにします。
- 別のデータ型を指定して、必要なトリミング動作を取得します。
- 一連の XML GENERATE ステートメントを使用して、出力に別の順序を選択します。

これらの変更を行う最も安全な方法は、1 つ以上の REPLACE コンパイラー指示ステートメントを伴うオリジナルの宣言をもう 1 つコピーして、それを使用することです。

『例: XML 出力の拡張』

XML 文書の作成時に、エレメント名または要素の値の一部にハイフンが含まれていることが分かったため、要素の値に含まれているハイフンは変換せずに、エレメント名のハイフンを下線に変更したい場合もあります。以下で言及している例は、そのための方法を示しています。

345 ページの『例: エレメント名に含まれるハイフンを下線に変換する』

関連リファレンス
『XML GENERATE の操作』(「*ILE COBOL 言語解説書*」)

例: XML 出力の拡張

以下のデータ構造の例を考えてみましょう。この構造から生成される XML にはいくつかの問題がありますが、これらの問題は訂正可能です。

```
01  CDR-LIFE-BASE-VALUES-BOX.
    15  CDR-LIFE-BASE-VAL-DATE    PIC X(08).
    15  CDR-LIFE-BASE-VALUE-LINE  OCCURS  2 TIMES.
        20  CDR-LIFE-BASE-DESC.
            25 CDR-LIFE-BASE-DESC1 PIC X(15).
            25  FILLER             PIC X(01).
            25  CDR-LIFE-BASE-LIT  PIC X(08).
            25  CDR-LIFE-BASE-DTE  PIC X(08).
        20  CDR-LIFE-BASE-PRICE.
            25  CDR-LIFE-BP-SPACE  PIC X(02).
            25  CDR-LIFE-BP-DASH   PIC X(02).
            25  CDR-LIFE-BP-SPACE1 PIC X(02).
        20  CDR-LIFE-BASE-PRICE-ED  REDEFINES
             CDR-LIFE-BASE-PRICE  PIC $$$.$$.
        20  CDR-LIFE-BASE-QTY.
            25  CDR-LIFE-QTY-SPACE PIC X(08).
            25  CDR-LIFE-QTY-DASH PIC X(02).
            25  CDR-LIFE-QTY-SPACE1 PIC X(02).
            25  FILLER               PIC X(02) VALUE "00".
        20  CDR-LIFE-BASE-QTY-ED    REDEFINES
                CDR-LIFE-BASE-QTY PIC ZZ,ZZZ,ZZZ.ZZZ.
        20  CDR-LIFE-BASE-VALUE   PIC X(15).
        20  CDR-LIFE-BASE-VALUE-ED  REDEFINES
                CDR-LIFE-BASE-VALUE
                                  PIC $(4),$$$,$$9.99.
    15  CDR-LIFE-BASE-TOT-VALUE-LINE.
        20  CDR-LIFE-BASE-TOT-VALUE   PIC X(15).
```

このデータ構造にいくつかのサンプル値を取り込んで、XML をこのデータ構造から直接生成し、プログラム Pretty (337 ページの『例: XML の生成』で示しています) を使用してフォーマットすると、その結果は以下のようになります。

```
<CDR-LIFE-BASE-VALUES-BOX>
  <CDR-LIFE-BASE-VAL-DATE>01/02/03</CDR-LIFE-BASE-VAL-DATE>
  <CDR-LIFE-BASE-VALUE-LINE>
    <CDR-LIFE-BASE-DESC>
      <CDR-LIFE-BASE-DESC1>First</CDR-LIFE-BASE-DESC1>
      <CDR-LIFE-BASE-LIT> </CDR-LIFE-BASE-LIT>
      <CDR-LIFE-BASE-DTE>01/01/01</CDR-LIFE-BASE-DTE>
    </CDR-LIFE-BASE-DESC>
    <CDR-LIFE-BASE-PRICE>
```

```
      <CDR-LIFE-BP-SPACE>$2</CDR-LIFE-BP-SPACE>
      <CDR-LIFE-BP-DASH>3.</CDR-LIFE-BP-DASH>
      <CDR-LIFE-BP-SPACE1>00</CDR-LIFE-BP-SPACE1>
    </CDR-LIFE-BASE-PRICE>
    <CDR-LIFE-BASE-QTY>
      <CDR-LIFE-QTY-SPACE>       1</CDR-LIFE-QTY-SPACE>
      <CDR-LIFE-QTY-DASH>23</CDR-LIFE-QTY-DASH>
      <CDR-LIFE-QTY-SPACE1>.0</CDR-LIFE-QTY-SPACE1>
    </CDR-LIFE-BASE-QTY>
    <CDR-LIFE-BASE-VALUE>        $765.00</CDR-LIFE-BASE-VALUE>
  </CDR-LIFE-BASE-VALUE-LINE>
  <CDR-LIFE-BASE-VALUE-LINE>
    <CDR-LIFE-BASE-DESC>
      <CDR-LIFE-BASE-DESC1>Second</CDR-LIFE-BASE-DESC1>
      <CDR-LIFE-BASE-LIT> </CDR-LIFE-BASE-LIT>
      <CDR-LIFE-BASE-DTE>02/02/02</CDR-LIFE-BASE-DTE>
    </CDR-LIFE-BASE-DESC>
    <CDR-LIFE-BASE-PRICE>
      <CDR-LIFE-BP-SPACE>$3</CDR-LIFE-BP-SPACE>
      <CDR-LIFE-BP-DASH>4.</CDR-LIFE-BP-DASH>
      <CDR-LIFE-BP-SPACE1>00</CDR-LIFE-BP-SPACE1>
    </CDR-LIFE-BASE-PRICE>
    <CDR-LIFE-BASE-QTY>
      <CDR-LIFE-QTY-SPACE>       2</CDR-LIFE-QTY-SPACE>
      <CDR-LIFE-QTY-DASH>34</CDR-LIFE-QTY-DASH>
      <CDR-LIFE-QTY-SPACE1>.0</CDR-LIFE-QTY-SPACE1>
    </CDR-LIFE-BASE-QTY>
    <CDR-LIFE-BASE-VALUE>        $654.00</CDR-LIFE-BASE-VALUE>
  </CDR-LIFE-BASE-VALUE-LINE>
  <CDR-LIFE-BASE-TOT-VALUE-LINE>
    <CDR-LIFE-BASE-TOT-VALUE>Very high!</CDR-LIFE-BASE-TOT-VALUE>
  </CDR-LIFE-BASE-TOT-VALUE-LINE>
</CDR-LIFE-BASE-VALUES-BOX>
```

このようにして生成された XML には、以下のようないくつかの問題があります。

- エレメント名が長く、それほど意味のあるものではない。
- 不要なデータがある (例えば、CDR-LIFE-BASE-LIT および CDR-LIFE-BASE-DTE)。
- 必要なデータに不必要な親がある。例えば、CDR-LIFE-BASE-DESC1 には CDR-LIFE-BASE-DESC という親があります。
- サブコンポーネントに分割された他の必須フィールドの数が多すぎる。例えば、CDR-LIFE-BASE-PRICE には、1 つの金額に対して 3 つのサブコンポーネントがあります。

XML 出力のこのような特性およびその他の特性は、以下のように記憶域を再定義することにより修正することができます。

```
1 BaseValues redefines CDR-LIFE-BASE-VALUES-BOX.
 2 BaseValueDate pic x(8).
 2 BaseValueLine occurs 2 times.
  3 Description pic x(15).
  3 pic x(9).
  3 BaseDate pic x(8).
  3 BasePrice pic x(6) justified.
  3 BaseQuantity pic x(14) justified.
  3 BaseValue pic x(15) justified.
 2 TotalValue pic x(15).
```

上で示したデータ値の定義セットから XML を生成しフォーマットすると、その結果がより使いやすいものになります。

```
<BaseValues>
  <BaseValueDate>01/02/03</BaseValueDate>
  <BaseValueLine>
    <Description>First</Description>
    <BaseDate>01/01/01</BaseDate>
    <BasePrice>$23.00</BasePrice>
    <BaseQuantity>123.000</BaseQuantity>
    <BaseValue>$765.00</BaseValue>
  </BaseValueLine>
  <BaseValueLine>
    <Description>Second</Description>
    <BaseDate>02/02/02</BaseDate>
    <BasePrice>$34.00</BasePrice>
    <BaseQuantity>234.000</BaseQuantity>
    <BaseValue>$654.00</BaseValue>
  </BaseValueLine>
  <TotalValue>Very high!</TotalValue>
</BaseValues>
```

上に示したように、オリジナルのデータ定義を直接再定義することができます。ただし、一般的には、オリジナルの定義を使用する場合であっても、コンパイラーのテキスト操作機能を使用してそれを適切に変更する方が安全です。その一例を以下の REPLACE コンパイラー指示ステートメントで示します。この REPLACE ステートメントは複雑に見えるかもしれませんが、オリジナルのデータ定義が変更された場合でも、それ自体は維持されるという利点があります。

```
replace ==CDR-LIFE-BASE-VALUES-BOX== by
           ==BaseValues redefines CDR-LIFE-BASE-VALUES-BOX==
        ==CDR-LIFE-BASE-VAL-DATE== by ==BaseValueDate==
        ==CDR-LIFE-BASE-VALUE-LINE== by ==BaseValueLine==
        ==20  CDR-LIFE-BASE-DESC.== by ====
        ==CDR-LIFE-BASE-DESC1== by ==Description==
        ==CDR-LIFE-BASE-LIT== by ====
        ==CDR-LIFE-BASE-DTE== by ==BaseDate==
        ==20  CDR-LIFE-BASE-PRICE.== by ====
        ==25  CDR-LIFE-BP-SPACE PIC X(02).== by ====
        ==25  CDR-LIFE-BP-DASH PIC X(02).== by ====
        ==25  CDR-LIFE-BP-SPACE1 PIC X(02).== by ====
        ==CDR-LIFE-BASE-PRICE-ED== by ==BasePrice==
        ==REDEFINES CDR-LIFE-BASE-PRICE PIC $$$.$$.== by
            ==pic x(6) justified.==
        ==20  CDR-LIFE-BASE-QTY.
            25  CDR-LIFE-QTY-SPACE PIC X(08).
            25  CDR-LIFE-QTY-DASH PIC X(02).
            25  CDR-LIFE-QTY-SPACE1 PIC X(02).
            25  FILLER PIC X(02).== by ====
        ==CDR-LIFE-BASE-QTY-ED== by ==BaseQuantity==
        ==REDEFINES CDR-LIFE-BASE-QTY PIC ZZ,ZZZ,ZZZ.ZZZ.== by
            ==pic x(14) justified.==
        ==CDR-LIFE-BASE-VALUE-ED== by ==BaseValue==
        ==20  CDR-LIFE-BASE-VALUE PIC X(15).== by ====
        ==REDEFINES CDR-LIFE-BASE-VALUE PIC $(4),$$$,$$9.99.==
            by ==pic x(15) justified.==
        ==CDR-LIFE-BASE-TOT-VALUE-LINE. 20== by ====
        ==CDR-LIFE-BASE-TOT-VALUE== by ==TotalValue==.
```

この REPLACE ステートメントの後にオリジナルの定義セットの 2 つ目のインスタンスが続いたときの結果は、前に提示した、上記グループ項目 BaseValues の再定義に似ています。この REPLACE ステートメントは、不必要な定義を除去し、保持しておく必要のある定義を変更するためのさまざまな手法を示しています。ユーザーの状態に適した手法を使用してください。

関連リファレンス
『XML GENERATE の操作』(「*ILE COBOL 言語解説書*」)
『REPLACE ステートメント』 (「*ILE COBOL 言語解説書*」)

例: エレメント名に含まれるハイフンを下線に変換する

ハイフンを含むデータ名を持つ項目を含むデータ構造から XML 文書を生成すると、生成された XML には、ハイフンを含むエレメント名が含まれます。次の例は、要素の値に含まれるハイフンを変更せずに、エレメント名に含まれるハイフンを下線に変換する方法を示しています。

以下のデータ構造を考えます。

```
1 Customer-Record.
 2 Customer-Number pic 9(9).
 2 First-Name      pic x(10).
 2 Last-Name       pic x(20).
```

このデータ構造にいくつかのサンプル値を取り込んで、XML をこのデータ構造から直接生成し、プログラム Pretty (337 ページの『例: XML の生成』で示しています) を使用してフォーマットすると、その結果は以下のようになります。

```
<Customer-Record>
  <Customer-Number>12345</Customer-Number>
  <First-Name>John</First-Name>
  <Last-Name>Smith-Jones</Last-Name>
</Customer-Record>
```

エレメント名にハイフンが含まれており、要素 Last-Name の内容にもハイフンが含まれています。

この XML 文書がデータ項目 xmldoc の内容であり、charcnt がこの XML 文書の長さに設定されていると仮定すると、エレメント名内のすべてのハイフンを下線に変更できますが、以下のコードを使用すると、要素の値は変更せずにそのままにしておくことができます。

```
1 xmldoc          pic x(16384).
1 charcnt  comp pic 9(5).
1 pos      comp pic 9(5).
1 tagstate comp pic 9  value zero.
. . .
dash-to-underscore.
  perform varying pos from 1 by 1
          until pos > charcnt
    if xmldoc(pos:1) = '<'
      move 1 to tagstate
    end-if
    if tagstate = 1 and xmldoc(pos:1) = '-'
      move '_' to xmldoc(pos:1)
    else
      if xmldoc(pos:1) = '>'
        move 0 to tagstate
      end-if
    end-if
  end-perform.
```

データ項目 xmldoc 内の修正された XML 文書には、以下に示しているように、エレメント名にはハイフンの代わりに下線が含まれます。

```
<Customer_Record>
  <Customer_Number>12345</Customer_Number>
  <First_Name>John</First_Name>
  <Last_Name>Smith-Jones</Last_Name>
</Customer_Record>
```

生成された XML 出力のエンコードの制御

XML GENERATE ステートメントを使用して XML 出力を生成する場合は、XML 出力を受け取るデータ項目のカテゴリーによって、出力のエンコード方式を制御することができます。以下の表に、可能な出力形式を示します。

表 21. **生成された XML 出力のエンコード方式**

定義する受取 XML ID	生成された XML 出力のエンコード方式
英数字	PROCESS ステートメントの CCSID オプション d で指定された CCSID - ソースがコンパイルされたときに有効であった XML GENERATE 1 バイト・データ CCSID。有効な CCSID が 65535 の場合、実行時のジョブのデフォルト CCSID が使用されます。
国別	Unicode (National CCSID コンパイラー・オプション、または NTLCCSID PROCESS オプションで指定された UCS-2、CCSID)[1]
1. バイト・オーダー・マーク は生成されません。	

データ項目を XML に変換する方法、および XML エレメント名を COBOL データ名から形成する方法については、XML GENERATE ステートメントの操作に関する以下の関連リファレンスを参照してください。

関連リファレンス
『XML GENERATE の操作』(「*ILE COBOL 言語解説書*」)

XML 出力の生成中のエラーの処理

XML 出力の生成中にエラーが検出された場合は、例外状態が存在しています。エラー・タイプを示す数値の例外コードを含んでいる特殊レジスター XML-CODE をチェックするコードを作成することができます。

エラーを処理するには、XML GENERATE ステートメントの以下の句のいずれか一方または両方を使用します。

- ON EXCEPTION
- COUNT IN

XML GENERATE ステートメントに ON EXCEPTION 句をコーディングした場合、制御は指定した命令ステートメントに移動します。例えば、XML-CODE 値を表示する命令ステートメントをコーディングできます。 ON EXCEPTION 句をコーディングしなかった場合、制御は XML GENERATE ステートメントの終わりに移ります。

エラーが発生した場合、XML 出力を受け取るデータ項目の大きさが十分ではないことが問題になる場合があります。その場合は、XML 出力は完了せず、特殊レジスター XML-CODE にエラー・コード 400 が含まれます。

以下のステップを実行すると、生成された XML 出力を調べることができます。

1. XML GENERATE ステートメントに COUNT IN 句をコーディングします。

   指定したカウント・フィールドには、XML 生成中に充てんされた XML 文字位置のカウントが保持されます。XML 出力を国別として定義した場合は、そのカウントは国別文字位置 (UCS-2 文字エンコード単位) で表され、それ以外はバイトで表されます。
2. カウント・フィールドを参照変更とともに使用して、生成された XML 出力を含む受取データ項目のサブストリングを参照します。

   例えば、XML-OUTPUT が XML 出力を受け取るデータ項目で、XML-CHAR-COUNT がカウント・フィールドの場合、XML-OUTPUT(1:XML-CHAR-COUNT) は XML 出力を参照します。

XML-CODE の内容を使用して、どのような訂正処置を取る必要があるのかを決定します。XML 生成中に発生する可能性のある例外のリストについては、以下の関連リファレンスを参照してください。

関連リファレンス

710 ページの『XML 生成例外』

第 13 章 別の言語によるデータの呼び出しと共用

ILE COBOL は、他の ILE、OPM、および EPM 言語を呼び出したり、それらによって呼び出されたりすることができます。

この章では次のことについて説明します。

- ILE COBOL から他の言語への呼び出しとデータ引き渡しの方法
- 他の言語から ILE COBOL へ制御を戻す方法
- ILE COBOL プログラムから CL コマンドを出す方法
- ILE COBOL プログラムへの構造化照会言語 (SQL) ステートメントの組み込み方法

一般的に、次のことが言えます。

- ILE COBOL プログラムが他の言語のプログラムの**呼び出し側**である場合、パラメーター・リストを構成する項目を指す USING 句を指定した CALL ステートメントを使用してください。制御は、呼び出し側プログラムの CALL ステートメントの次のステートメントに戻されます (ただし、呼び出し先プログラムまたは呼び出し先プログラムによって呼び出されたプログラムが実行単位を終了した場合を除く)。
- ILE COBOL プログラムが他の言語によってパラメーターを指定して**呼び出されて**いる場合、PROCEDURE DIVISION ステートメントに USING 句が、また受け取るパラメーターを記述しているリンケージ・セクションが必要です。 ILE COBOL プログラムは、GOBACK ステートメントまたは EXIT PROGRAM ステートメントを使用して制御を呼び出し側プログラムに戻すことができます。
- 最も近い管理境界がハード管理境界の場合、ILE COBOL プログラムは、STOP RUN ステートメントまたは GOBACK ステートメントを使用して実行単位を停止することができます。最も近い管理境界がソフト管理境界の場合は、実行単位を停止することができません。

  他の言語からの ILE COBOL プログラムの呼び出しについては、その言語のプログラミングの手引きを参照してください。

ILE COBOL 以外のプログラムとデータの受け渡しをするときに考慮しなければならない点の 1 つは、パラメーター・リストの一致です。作成する ILE COBOL プログラムから ILE COBOL 以外のプログラムを呼び出す場合は、入力時に必要な手順を理解し、それに応じてデータ項目をセットアップしなければなりません。反対に呼び出される場合は、渡されるデータの内容を理解し、それを受け入れることができるようにリンケージ・セクションをセットアップする必要があります。

別の考慮事項として、RETURN-CODE 特殊レジスターの取り扱いがあります。RETURN-CODE 特殊レジスターは、ILE COBOL 以外のプログラムでは設定できません。呼び出し先プログラムから制御が戻された後に、RETURN-CODE 特殊レジスターに正しくない値が含まれていた場合、ILE COBOL プログラムが制御を呼び出

し側プログラムに戻す前に RETURN-CODE 特殊レジスターの値を正しい値に設定しなければなりません。そうしない場合は、正しくない戻りコードが呼び出し側プログラムに戻されます。

ILE C および VisualAge C++ プログラムおよびプロシージャーの呼び出し

注: このセクションにおいて ILE C を参照している個所は、すべて VisualAge C++ にも適用されます。

ILE COBOL プログラムでは、動的プログラム呼び出しまたは静的プロシージャー呼び出しを使用して、ILE C プログラムおよびプロシージャーを呼び出すことができます。

動的プログラム呼び出しを使用して ILE C プログラムを呼び出す場合、ILE C プログラムは別個のプログラム・オブジェクトとしてコンパイルおよびバインドしなければなりません。静的プロシージャー呼び出しを使用して ILE C プロシージャーを呼び出す場合、ILE C プロシージャーでは、まずコンパイルしてモジュール・オブジェクトにして、それから ILE COBOL 呼び出し側プログラムにバインドしなければなりません。 ILE C において、ILE プロシージャーは ILE C 関数に対応しています。 ILE C プログラムおよびプロシージャーのコンパイルとバインディングについては、「*IBM Rational Development Studio for i: ILE C/C++ Programmer's Guide*」を参照してください。

ILE C プログラムまたはプロシージャーを ILE COBOL プログラムから呼び出すには、CALL *リテラル* のステートメントを使用します (*リテラル* は ILE C プログラムまたはプロシージャーの名前)。 ILE C プログラムまたはプロシージャーを呼び出すには、別の ILE COBOL サブプログラムを呼び出す場合と同様に、CALL *リテラル* のステートメントを入力します。 ILE COBOL プログラムに CALL ステートメントを書いて、動的プログラム呼び出しまたは静的プロシージャー呼び出しを使用して ILE C プログラムを呼び出す方法については 247 ページの『静的プロシージャー呼び出しと動的プログラム呼び出しの使用』を参照してください。

CALL *id* を使用することによって、ILE C プログラムを ILE COBOL プログラムから呼び出すこともできます。 CALL *id* については 250 ページの『CALL id の使用』を参照してください。

あるいは、ILE C プログラムまたはプロシージャーを ILE COBOL プログラムから呼び出すのに、CALL *プロシージャー・ポインター* を使用することもできます。CALL *プロシージャー・ポインター* については 251 ページの『CALL プロシージャー・ポインターの使用』を参照してください。 ILE COBOL のプロシージャー・ポインターは、ILE C の関数ポインターに相当します。プロシージャー・ポインターは、CALL ステートメント上で ILE COBOL から ILE C 関数へ引き数として渡すことができます。また ILE C 関数によりパラメーターを関数ポインターとして定義させることができます。

ILE COBOL CALL ステートメントの RETURNING 句が指定されていた場合は、値を戻す ILE C 関数しか呼び出すことができません。

同じ活動化グループにある 2 つ以上の ILE C プログラムが、互いの実行時リソースとやりとりすることができます。その方法の詳細については、「*IBM Rational Development Studio for i: ILE C/C++ Programmer's Guide*」を参照してください。したがって、ILE COBOL から呼び出した ILE C プログラムが、同じ活動化グループで他のプログラムと一緒に作動するように設計されているかどうかを確認する必要があります。同じ活動化グループにある 2 つの ILE C プログラムは、errno、シグナル・ベクトル、およびストレージ・プールなどを共用することができます。ILE COBOL プログラムで 2 つ以上の ILE C プログラムを呼び出す必要があり、それらのプログラムが同じランタイムを共用するように設計されていない場合、ILE C プログラムを実行する活動化グループに別の名前を指定してください。

デフォルトのプログラム・タイプの場合、 ILE C では再帰可能ですが、ILE COBOL では再帰は不可能です。COBOL プログラムを再帰的プログラムにするには、PROGRAM-ID 段落で RECURSIVE 文節を使用する必要があります。 ILE C 関数が ILE COBOL の非再帰的プログラムを再帰的に呼び出すと、 ILE COBOL プログラムから実行時エラー・メッセージが生成されます。

ILE C 関数を ILE COBOL プログラムから呼び出すとき、呼び出す ILE C 関数名は、大文字小文字の区別が必要であったり、10 文字より長い (最高 256 文字) 必要があったり、いくつかの特殊文字が入っている必要があったりする場合があります。この場合、静的プロシージャー呼び出しを使用し、CRTCBLMOD または CRTBNDCBL の各コマンドの OPTION パラメーターに *NOMONOPRC 値を指定して ILE COBOL プログラムをコンパイルしてください。

ILE C++ コンパイラー・プロシージャーが ILE COBOL から呼び出されるとき、ILE C++ コンパイラーの関数名のマングリングを防止するために、ILE C++ コンパイラー用関数プロトタイプにキーワード extern ″COBOL″ または extern ″C″ がなければなりません。 ILE COBOL が BY VALUE の引き数を ILE C++ コンパイラーに渡す場合は、extern ″C″ を使用してください。

ILE C プログラムまたはプロシージャーへのデータの引き渡し

CALL...BY REFERENCE、CALL...BY VALUE、または CALL...BY CONTENT を使用することにより、呼び出し先の ILE C プログラムまたはプロシージャーにデータを渡すことができます。 CALL...BY REFERENCE、CALL...BY VALUE、または CALL...BY CONTENT の使用の詳細については 262 ページの『CALL...BY REFERENCE、BY VALUE、または BY CONTENT を使用したデータの受け渡し』を参照してください。

CALL...BY REFERENCE を使用してデータを ILE C プログラムに渡すと、ILE C プログラムに受け入れられた引き数リストの中にデータ項目へのポインターが入れられます。

CALL...BY CONTENT を使用してデータを ILE C プログラムに渡すと、データ項目の値が一時ロケーションにコピーされ、コピーの一時ロケーションのアドレスを含むポインターは、ILE C プログラムに受け入れられた引き数リストに置かれます。

CALL...BY VALUE では、ILE C プログラムに受け入れられた引き数リストに、項目の値が入れられます。 CALL...BY VALUE は、ILE C プロシージャーを呼び出すのに使用できます。しかし、ILE C プログラム・オブジェクトの呼び出しには使用できません。

ILE COBOL プログラムにおいて、ILE C プログラムまたはプロシージャーに渡したい引き数をデータ部に記述する方法は、他のデータ項目をデータ部に記述するときの方法と同じです。渡したい引き数の記述方法の詳細については 261 ページの『プログラム間でのデータの受け渡しと共用』を参照してください。

呼び出し先 ILE C プログラム・オブジェクトの実行が始まると、関数 `main` が自動的に呼び出されます。各 ILE C プログラム・オブジェクトには、`main` という名前の関数が必ず 1 つ存在しなければなりません。パラメーターを ILE C プログラム・オブジェクトに渡す場合、関数 `main` に 2 つのパラメーターを宣言する必要があります。これらのパラメーターには任意の名前を付けることができますが、通常は *argc* と *argv* を使います。最初のパラメーター *argc* (引き数カウント) のタイプは `int` で、これは ILE C プログラム・オブジェクトを呼び出した CALL ステートメントに引き数を何個指定したかを示すものです。 2 番目のパラメーター *argv* (引き数ベクトル) のデータ・タイプは、`char` 配列オブジェクトへのポインターの配列です。

argc の値は、配列 *argv* 内のポインターの数を示しています。プログラム名が使用可能なら、*argv* の最初のエレメントは、実行中の ILE C プログラムの呼び出し名のプログラム名を含む文字配列を指します。 *argv* の残りのエレメントには、呼び出し先 ILE C プログラムに渡されるパラメーターを指すポインターが入れられます。最後のエレメント *argv[argc]* は、常に `NULL` です。

呼び出し先 ILE C プログラムまたはプロシージャーにパラメーターを記述する方法の詳細については、「*IBM Rational Development Studio for i: ILE C/C++ Programmer's Guide*」を参照してください。

ILE C と ILE COBOL との間のデータ・タイプの互換性

ILE C と ILE COBOL のデータ・タイプは異なります。 ILE C で作成されたプログラムと ILE COBOL で作成されたプログラムとの間でデータを渡したい場合には、このデータ・タイプの違いに注意する必要があります。 ILE C と ILE COBOL のデータ・タイプによっては、相手の言語に直接対応するものがない場合があります。

ILE C プログラムは常に、文字ストリングはヌル文字で終了していることを想定しています。ILE C プログラムに渡されるストリング・データがヌル終了されていることを確認してください。詳しくは、 233 ページの『ヌル終了ストリングの操作』を参照してください。

353 ページの表 22 に ILE COBOL データ・タイプと ILE C データ・タイプの互換性を示します。

表 22. *ILE COBOL と ILE C との間のデータ・タイプの互換性*

ILE COBOL	プロトタイプでの ILE C 宣言	長さ	説明
PIC X(n).	char[n] または char *	n	n=1 〜 16,711,568 の文字フィールド。
FORMAT DATE リテラル	char[6]	6	日付フィールド。
FORMAT TIME リテラル	char[8]	8	時刻フィールド。
FORMAT TIMESTAMP	char[n]	26	タイム・スタンプ・フィールド。
PIC G(n)	char[2n]	2n	グラフィック・フィールド。
PIC 1 INDIC ..	char	1	標識。
PIC S9(n) DISPLAY	char[n]	n	ゾーン 10 進数。
PIC S9(n-p)V9(p) COMP-3	decimal(n,p)	n/2+1	パック 10 進数。
PIC S9(n-p)V9(p) PACKED-DECIMAL.	decimal(n,p)	n/2+1	パック 10 進数。
PIC S9(4) COMP-4 BINARY	short int	2	-9999〜+9999 の 2 バイト符号付き整数。
PIC S9(4) COMP-4 と *NOSTDTRUNC PIC S9(4) BINARY と *NOSTDTRUNC PIC S9(4) COMP-5	short int	2	-32768〜+32767 の 2 バイト符号付き整数。
PIC S9(4) COMP-5	unsigned short int	2	0〜65535 の 2 バイト符号なし整数。
PIC S9(9) COMP-4 PIC S9(9) BINARY	int long int	4	-999999999〜+999999999 の 4 バイト符号付き整数。
PIC S9(9) COMP-4 と *NOSTDTRUNC PIC S9(9) BINARY と *NOSTDTRUNC PIC S9(9) COMP-5	int long int	4	-2147483648〜+2147483647 の 4 バイト符号付き整数。
PIC S9(9) COMP-5	unsigned int	4	0〜4294967295 の 4 バイト符号なし整数。
PIC S9(18) COMP-4 PIC S9(18) BINARY	long long	8	8 バイトの整数。
PIC S9(18) COMP-4 と *NOSTDTRUNC PIC S9(18) BINARY と *NOSTDTRUNC PIC S9(18) COMP-5	long long	8	8 バイトの整数。
PIC S9(18) COMP-5	unsigned int	8	8 バイトの符号なし整数。
05 VL-FIELD. 10 i PIC S9(4) COMP-4. 10 data PIC X(n).	_Packed struct {short i; char[n]}	n+2	可変長フィールド (i は指定した長さ、n は最大長)。
05 n PIC 9(9) COMP-4. 05 x redefines n PIC X(4).	struct {unsigned int : n}x;	4	ビット・フィールドは 16 進数リテラルを使用して操作可能。

表 22. ILE COBOL と ILE C との間のデータ・タイプの互換性 (続き)

ILE COBOL	プロトタイプでの ILE C 宣言	長さ	説明
01 record 05 field1... 05 field2...	struct	n	構造体。 struct に _Packed 修飾子を使用。構造体の内容を変更する場合、渡す構造は、構造体を指すポインターとして渡すことが必要。
USAGE IS POINTER	*	16	ポインター。
PROCEDURE-POINTER	関数を指すポインター。	16	プロシージャーを指す 16 バイトのポインター。
USAGE IS INDEX	int	4	4 バイトの整数。
REDEFINES	union.element	n	共用体のエレメント。
OCCURS	data_type[n]	n*(data_type の長さ)	配列。
USAGE IS COMP-1	float	4	4 バイトの浮動小数点数。
USAGE IS COMP-2	double	8	8 バイトの浮動小数点数。
サポートなし。	long double	8	8 バイトの long double。
サポートなし。	enum	1、2、4	列挙型

ILE C プログラムまたはプロシージャーとの外部データの共用

ILE COBOL プログラムと ILE C プログラムとの間で外部データを共用することができます。データ項目を共用するためには、その項目を ILE COBOL プログラムと ILE C プログラム内で、同じ名前と記述を使用して定義する必要があります。 ILE C プログラムと ILE COBOL プログラムで共用する外部データが、異なるプログラムのサイズで定義してある場合、外部データ項目のサイズは、ILE C プログラムで `extern` キーワードを使用して定義された外部データのサイズになります。

外部データ項目を共用するためには、ILE COBOL プログラムと ILE C プログラムは静的にバインドされている必要があります。

ILE COBOL プログラムでは、データ項目は、WORKING-STORAGE SECTION に EXTERNAL 文節を使用して記述する必要があります。 ILE COBOL プログラムでの外部データの使用の詳細については 266 ページの『EXTERNAL データの共用』または ILE C の EXTERNAL 文節の部分を参照してください。

ILE C プログラムでは、データ項目は `extern` キーワードを使用して宣言する必要があります。 `extern` キーワードの詳細については、「*IBM Rational Development Studio for i: ILE C/C++ Programmer's Guide*」を参照してください。

ILE C プログラムまたはプロシージャーからの制御の戻り

ILE C で `return` キーワードを使用すると、制御が呼び出し側プログラムに戻されます。 ILE C `return` キーワードが void 以外のものを戻す場合、ILE COBOL ステートメントに RETURNING 句が入っていなければなりません。さらに、ILE C から戻される項目のデータ・タイプおよび長さは、COBOL 呼び出しステートメントの RETURNING 句の ID のデータ・タイプおよび長さと一致していなければなりません。

ILE C プログラムから return を出すと、呼び出し先 ILE C プログラムが実行されている ILE 活動化グループは終了します。 ILE C プログラムが *NEW 活動化グループで実行するように定義されていた場合、return がハード管理境界で出されると、ILE C プログラムが実行されていた活動化グループは終了します。 ILE C プログラムが *CALLER 活動化グループまたは名前付き活動化グループで実行されるように定義されていた場合は、return が出されても、ILE C プログラムが実行されていた活動化グループは活動状態のままです。その後、この活動化グループで ILE C プログラムを呼び出すと、その ILE C プログラムは最後に使用されたときの状態になっています。

exit(n) 関数により、制御を最も近い管理境界に戻すことができます。例外条件が発生すると、例外ハンドラーが呼び出されるか、または最も近い管理境界に制御が戻されることがあります。

ILE C プログラムの実行されている名前付き活動化グループが呼び出し側 ILE COBOL プログラムとは違う場合、exit(n) または未処理の例外によって、次のことが生じる場合があります。 exit(n) または未処理の例外がハード管理境界の近くで生じると、ILE C が実行されていた活動化グループは停止します。それがソフト管理境界の近くで生じた場合には、活動化グループは活動状態のままです。未処理の例外によって ILE C プログラムが実行されていた活動化グループが停止した場合、呼び出し側 ILE COBOL プログラムの活動化グループにおいて CEE9901 エスケープ・メッセージが出されます。

ILE C プログラムと呼び出し ILE COBOL プログラムが同じ活動化グループで実行されている場合は、exit(n) または未処理の例外によって次のことが生じる場合があります。 exit(n) または未処理の例外がハード制御境界の近くで生じると、ILE COBOL プログラムを含め、活動化グループは停止します。 ILE C プログラムと ILE COBOL プログラムの両方が実行されていた活動化グループが未処理の例外によって停止されると、ハード管理境界の前のプログラムに対して CEE9901 エスケープ・メッセージが出されます。 exit(n) または未処理の例外がソフト管理境界の近くで生じた場合、exit(n) が実行された ILE C プログラムからソフト管理境界のプログラムまでのすべてのプログラムとプロシージャー (ILE COBOL プログラムを含む) が停止されます。

呼び出し先プログラムが例外なしで終了した場合、ILE COBOL プログラムの CALL ステートメントの次のステートメントに制御が戻されます。呼び出し先プログラムが例外で終了した場合には、ILE COBOL プログラムで指定されている例外処理プロシージャーが呼び出されます。例外処理プロシージャーに制御を渡す方法については 415 ページの『第 16 章 ILE COBOL のエラーおよび例外の処理』を参照してください。

呼び出し先プログラムでは、ILE COBOL 呼び出し側プログラムをすべてスキップするエスケープ・メッセージを出すこともできます。この場合、ILE COBOL プログラムの呼び出しは取り消されます。呼び出しの取り消しは、ILE COBOL プログラムから戻る場合と同様です。

ILE COBOL プログラムからの ILE C プロシージャー呼び出しの例

各例は、ILE C プロシージャーを呼び出す ILE COBOL プログラムから構成されています。

ILE C プロシージャー呼び出し例 1 のサンプル・コード

例 1 には、以下の 2 つのサンプル・コードがあります。

C1 QCSRC
ILE COBOL プログラムにバインドされている ILE C プロシージャー。

CBL1 QCBLLESRC
バインド済み ILE C プロシージャーを呼び出す ILE COBOL プロシージャー。

C1 QCSRC のサンプル・コードは、図 70 に示されています。

```
/* C1 QCSRC --- ILE C プロシージャー */
#include &lt;stdio.h&gt;
#include &lt;stdlib.h&gt;
void C1(char *result)
{
    *(result+9) = '*';
    *(result+10) = '#';
    return;
}
```

図 70. C1 QCSRC のソース・コード

CBL1 QCBLLESRC のサンプル・コードは、 357 ページの図 71 に示されています。

```
*************************************************************************
* cbl1 qcbllesrc
*
* Description:
*
*   COBOL source with ILE C procedure call.
*
*************************************************************************
 Identification Division.
  Program-Id.    cbl1.
  Author.        Author's Name.
  Installation.  IBM Toronto Lab
  Date-Written.  July 14, 1998.
  Date-Compiled. Will be replaced by compile date.
 Environment Division.
  Configuration Section.
   Source-Computer.   IBM-ISERIES.
   Object-Computer.   IBM-ISERIES.
   Special-Names.
 INPUT-OUTPUT SECTION.

 File-Control.
 Data Division.
  Working-Storage Section.
   01  RESULT-STRING    PIC X(20)       VALUE ALL "X".

 Procedure Division.

 TEST1-INIT.
     DISPLAY RESULT-STRING.
     CALL PROCEDURE "C1" USING RESULT-STRING.
     DISPLAY RESULT-STRING.
     STOP RUN.
*------------------------------------------------------------------------
* Output before call
* XXXXXXXXXXXXXXXXXXXX
* Output after call
* XXXXXXXXX*#XXXXXXXXX
```

図 71. CBL1 QCBLLESRC のソース・コード

ILE C プロシージャー呼び出し例 2 のサンプル・コード

例 2 には、以下の 2 つのサンプル・コードがあります。

CPROC1 QCSRC
ILE COBOL プログラムにバインドされている ILE C プロシージャー。

VARG1 QCBLLESRC
バインド済み ILE C プロシージャーを呼び出す ILE COBOL プロシージャー。

CPROC1 QSRC のサンプル・コードは、358 ページの図 72 に示されています。

```
/* CPROC1 QCSRC --- ILE C Procedure */
#include &lt;stdio.h&gt;

int inner(va_list);

int CPROC1(int p0, ...)
{
    int rc;
    va_list args;
    va_start(args,p0);
    rc = inner(args);
    va_end(args);
    return rc;
}

int inner(va_list v) {
    int p1,p2,p3=0;
    int p4,p5,p6=0;
    int p7,p8,p9=0;
    p1 = va_arg(v,int);
    p2 = va_arg(v,int);
    p3 = va_arg(v,int);
    p4 = va_arg(v,int);
    p5 = va_arg(v,int);
    p6 = va_arg(v,int);
    p7 = va_arg(v,int);
    p8 = va_arg(v,int);
    p9 = va_arg(v,int);
    printf("In inner with p1=%d p2=%d p3=%d&yen;n",
                              p1,   p2,   p3);
    printf("In inner with p4=%d p5=%d p6=%d&yen;n",
                              p4,   p5,   p6);
    printf("In inner with p7=%d p8=%d p9=%d&yen;n",
                              p7,   p8,   p9);
    return(p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8 + p9);
}
```

図 72. CPROC1 QSRC のソース・コード

VARG1 QCBLLESRC のサンプル・コードは、359 ページの図 73 に示されています。

```
*************************************************************************
* cbl1 qcbllesrc
*
* Description:
*
*   COBOL source with ILE C procedure call.
*
*************************************************************************
       IDENTIFICATION DIVISION.

 PROGRAM-ID.     VARG1.

*** This program demonstrates how to call a C procedure
*** using variable-length argument list.

 AUTHOR.
 INSTALLATION.   IBM Toronto Lab.
 DATE-WRITTEN.
 DATE-COMPILED.

 ENVIRONMENT DIVISION.

 CONFIGURATION SECTION.

 SOURCE-COMPUTER. IBM-ISERIES.
 OBJECT-COMPUTER. IBM-ISERIES.
   SPECIAL-NAMES. LINKAGE PROCEDURE FOR "CPROC1"
                          USING ALL DESCRIBED.

 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
 DATA DIVISION.
 FILE SECTION.
 WORKING-STORAGE SECTION.

 01 PARM0                   PIC S9(9) BINARY VALUE 0.
 01 PARM1                   PIC S9(9) BINARY VALUE 1.
 01 PARM2                   PIC S9(9) BINARY VALUE 2.
 01 PARM3                   PIC S9(9) BINARY VALUE 3.
 01 PARM4                   PIC S9(9) BINARY VALUE 4.
 01 PARM5                   PIC S9(9) BINARY VALUE 5.
 01 PARM6                   PIC S9(9) BINARY VALUE 6.
 01 PARM7                   PIC S9(9) BINARY VALUE 7.
 01 PARM8                   PIC S9(9) BINARY VALUE 8.
 01 PARM9                   PIC S9(9) BINARY VALUE 9.
 01 RC1                     PIC S9(9) BINARY VALUE 0.

 PROCEDURE DIVISION.

 MAIN.

       CALL PROCEDURE "CPROC1" USING BY VALUE
                                PARM0
                                PARM1
                                PARM2
                                PARM3
                                PARM4
                                PARM5
                                PARM6
                                PARM7
                                PARM8
                                PARM9
            RETURNING INTO RC1.
       GOBACK.
```

図 73. VARG1 QCBLLESRC のソース・コード

ILE C プロシージャー呼び出し例の作成および実行

ILE C プロシージャー呼び出し例 1 を作成し、実行するには、以下のステップに従ってください。

1. 1 つの ILE COBOL モジュールおよび 1 つの ILE C モジュールを作成します。
   - ILE COBOL モジュール CBL1 を作成するには、以下のように入力します。

     ```
     CRTCBLMOD  MODULE(CBL1) SRCFILE(*CURLIB/QCBLLESRC)
     ```

   - ILE C モジュール C1 を作成するには、以下のように入力します。

     ```
     CRTCMOD    MODULE(C1) SRCFILE(*CURLIB/QCSRC)
     ```

2. 2 つのモジュールを使用してプログラムを作成します。

   ```
   CRTPGM     PGM(CBL1) MODULE(*CURLIB/CBL1 *CURLIB/C1)
   ```

3. プログラムを呼び出します。

   ```
   CALL       PGM(*CURLIB/CBL1)
   ```

ILE C プロシージャー呼び出し例 2 を作成し、実行するには、以下のステップに従ってください。

1. 1 つの ILE COBOL モジュールおよび 1 つの ILE C モジュールを以下のようにして作成します。
   - 以下のように入力し、ILE COBOL モジュール VARG1 を作成します。

     ```
     CRTCBLMOD  MODULE(VARG1) SRCFILE(*CURLIB/QCBLLESRC)
     ```

   - 以下のように入力し、ILE C モジュール CPROC1 を作成します。

     ```
     CRTCMOD    MODULE(CPROC1) SRCFILE(*CURLIB/QCSRC)
     ```

2. 2 つのモジュールを使用してプログラムを作成します。

   ```
   CRTPGM     PGM(VARG1) MODULE(*CURLIB/VARG1 *CURLIB/CPROC1)
   ```

3. プログラムを呼び出します。

   ```
   CALL       PGM(*CURLIB/VARG1)
   ```

ILE COBOL プログラムからの ILE C プログラム呼び出しの例

ILE C プログラムを呼び出す ILE COBOL プログラムの例を以下に示します。

ILE C プログラム呼び出し例のサンプル・コード

2 つのコード・サンプルの例を示します。

C2 QCSRC
ILE C プログラム。

CBL2 QCBLLESRC
ILE C プログラム呼び出しを使用する ILE COBOL プログラム。

C2 QCSRC のサンプル・コードは、361 ページの図 74 に示されています。

```
/* C2 QCSRC --- ILE C プログラム */
#include &lt;stdio.h&gt;
#include &lt;stdlib.h&gt;
void main(int argc, char *argv[])
{
    *(argv[1]+9) = '*';
    *(argv[1]+10) = '#';
    return;
}
```

図 74. C2 QCSRC のソース・コード

CBL2 QCBLLESRC のサンプル・コードは、図 75 に示されています。

```
      ****************************************************************
      * cbl2 qcbllesrc
      *
      * Description:
      *
      *   COBOL source with ILE C program call.
      *
      ****************************************************************
       Identification Division.
        Program-Id.    cbl2.
        Author.        Author's Name.
        Installation.  IBM Toronto Lab
        Date-Written.  July 14, 1998.
        Date-Compiled. Will be replaced by compile date.
       Environment Division.
        Configuration Section.
         Source-Computer.   IBM-ISERIES.
         Object-Computer.   IBM-ISERIES.
         Special-Names.
       INPUT-OUTPUT SECTION.

       File-Control.
       Data Division.
        Working-Storage Section.
         01  RESULT-STRING    PIC X(20)       VALUE ALL "X".

       Procedure Division.

       TEST1-INIT.
           DISPLAY RESULT-STRING.
           CALL "C2" USING BY REFERENCE RESULT-STRING.
           DISPLAY RESULT-STRING.
           STOP run.
      *----------------------------------------------------------------
      * Output before call
      * XXXXXXXXXXXXXXXXXXXX
      * Output after call
      * XXXXXXXXX*#XXXXXXXXX
```

図 75. CBL2 QCBLLESRC のソース・コード

ILE C プログラム呼び出し例の作成および実行

ILE C プログラム呼び出し例を作成し、実行するには、以下のステップに従ってください。

1. 1 つの ILE COBOL プログラムおよび 1 つの ILE C プログラムを作成します。
   - ILE COBOL プログラム CBL2 を作成するには、以下のように入力します。

     ```
     CRTBNDCBL  PGM(CBL2) SRCFILE(*CURLIB/QCBLLESRC)
     ```

   - ILE C プログラム C2 を作成するには、以下のように入力します。

     ```
     CRTBNDC    PGM(C2) SRCFILE(*CURLIB/QCSRC)
     ```

2. ILE COBOL プログラムを呼び出します。

   ```
   CALL       PGM(*CURLIB/CBL2)
   ```

ILE RPG プログラムおよびプロシージャーの呼び出し

ILE COBOL プログラムでは、動的プログラム呼び出しまたは静的プロシージャー呼び出しを使用して、ILE RPG プログラムおよびプロシージャーを呼び出すことができます。

動的プログラム呼び出しを使用して ILE RPG プログラムを呼び出す場合、ILE RPG プログラムは別個のプログラム・オブジェクトとしてコンパイルおよびバインドしなければなりません。静的プロシージャー呼び出しを使用して ILE RPG プロシージャーを呼び出す場合、ILE RPG プロシージャーでは、まずコンパイルしてモジュール・オブジェクトにして、それから ILE COBOL 呼び出し側プログラムにバインドしなければなりません。 ILE RPG プログラムおよびプロシージャーのコンパイルとバインディングについては、「*IBM Rational Development Studio for i: ILE RPG プログラマーの手引き*」を参照してください。

ILE RPG プログラムまたはプロシージャーを ILE COBOL プログラムから呼び出すには、CALL *リテラル* のステートメントを使用します (*リテラル* は ILE RPG プログラムまたはプロシージャーの名前)。 ILE RPG プログラムまたはプロシージャーを呼び出すには、別の ILE COBOL サブプログラムを呼び出す場合と同様に、CALL *リテラル* のステートメントを入力します。 ILE COBOL プログラムに CALL ステートメントを書いて、動的プログラム呼び出しまたは静的プロシージャー呼び出しを使用して ILE RPG プログラムを呼び出す方法については 247 ページの『静的プロシージャー呼び出しと動的プログラム呼び出しの使用』を参照してください。

CALL *id* を使用することによって、ILE RPG プログラムを ILE COBOL プログラムから呼び出すこともできます。 CALL *id* については 250 ページの『CALL id の使用』を参照してください。

ILE RPG プログラムまたはプロシージャーへのデータの引き渡し

CALL...BY REFERENCE、CALL...BY VALUE、または CALL...BY CONTENT を使用することにより、呼び出し先の ILE RPG プログラムまたはプロシージャーにデータを渡すことができます。 CALL...BY REFERENCE、CALL...BY VALUE、また

は CALL...BY CONTENT の使用の詳細については 262 ページの『CALL...BY REFERENCE、BY VALUE、または BY CONTENT を使用したデータの受け渡し』を参照してください。

CALL...BY REFERENCE を使用してデータを ILE RPG プログラムに渡すと、ILE RPG プログラムに受け入れられた引き数リストの中にデータ項目へのポインターが入れられます。 CALL...BY CONTENT を使用してデータを ILE RPG プログラムに渡すと、データ項目の値が一時ロケーションにコピーされ、コピーの一時ロケーションのアドレスを含むポインターは、ILE RPG プログラムに受け入れられた引き数リストに置かれます。 CALL...BY VALUE では、ILE RPG プログラムに受け入れられた引き数リストに、項目の値が入れられます。 CALL...BY VALUE は、ILE RPG プロシージャーを呼び出すのに使用できます。しかし、ILE RPG プログラム・オブジェクトの呼び出しには使用できません。

ILE COBOL プログラムにおいて、ILE RPG プログラムまたはプロシージャーに渡したい引き数をデータ部に記述する方法は、他のデータ項目をデータ部に記述するときの方法と同じです。渡したい引き数の記述方法の詳細については 261 ページの『プログラム間でのデータの受け渡しと共用』を参照してください。

呼び出し先 ILE RPG プログラムにおいて、ILE COBOL プログラムから受け取るパラメーターは、PARM 操作を使用して記述します。受け取る各パラメーターは、別個の PARM 操作として定義されます。パラメーターの名前は結果フィールドに指定します。演算項目 1 および演算項目 2 の指定はオプショナルであり、変数またはリテラルを示します。演算項目 1 のフィールドの値は、呼び出し時に結果フィールド項目から移されます。演算項目 2 のフィールドの値は、戻り時に結果フィールドに入れられます。

ILE RPG プログラムでパラメーターを定義するもう 1 つの方法は、それらをプロトタイプに指定することです。各パラメーターごとに、別個の定義仕様で定義されます。 BY REFERENCE で渡されるパラメーターでは、特別なキーワードは必要ありません。 BY VALUE により渡されるパラメーターでは、VALUE キーワードが使用されます。 ILE RPG プログラムに引き数を記述する方法については、「*IBM Rational Development Studio for i: ILE RPG プログラマーの手引き*」を参照してください。

ILE RPG と ILE COBOL との間のデータ・タイプの互換性

ILE RPG と ILE COBOL のデータ・タイプは異なります。 ILE RPG で作成されたプログラムと ILE COBOL で作成されたプログラムとの間でデータを渡したい場合には、このデータ・タイプの違いに注意する必要があります。 ILE RPG と ILE COBOL のデータ・タイプによっては、相手の言語に直接対応するものがない場合があります。

364 ページの表 23 に ILE COBOL データ・タイプと ILE RPG データ・タイプの互換性を示します。

表 23. ILE COBOL と ILE RPG との間のデータ・タイプの互換性

ILE COBOL	ILE RPG の I 仕様書、 D 仕様書、または C 仕様書	長さ	説明
PIC X(n).	データ・タイプ欄がブランクまたは A、長さ欄が n、小数点以下桁数欄がブランク	n	n=1 ～ 32,767 の文字フィールド。
PIC 1 INDIC ..	*INxxxx	1	標識。
PIC S9(n) DISPLAY	データ・タイプ欄が S、長さ欄が n、小数点以下桁数欄が 0	n	ゾーン 10 進数。
PIC S9(n-p)V9(p) COMP-3	データ・タイプ欄が P、長さ欄が n、小数点以下桁数欄が p	n/2 + 1	パック 10 進数。
PIC S9(n-p)V9(p) PACKED-DECIMAL.	データ・タイプ欄が P、長さ欄が n、小数点以下桁数欄が p	n/2 + 1	パック 10 進数。
サポートなし。	データ・タイプ欄が I、長さ欄が 3、小数点以下桁数欄が 0	1	-128～127 の 1 バイト符号付き整数。
サポートなし。	データ・タイプ欄が U、長さ欄が 3、小数点以下桁数欄が 0	1	0～255 の 1 バイト符号なし整数。
PIC S9(4) COMP-4 BINARY	データ・タイプ欄が B、長さ欄が 4、小数点以下桁数欄が 0	2	-9999～+9999 の 2 バイト符号付き整数。
PIC S9(4) BINARY と *NOSTDTRUNC PIC S9(4) COMP-5	データ・タイプ欄が I、長さ欄が 5、小数点以下桁数欄が 0	2	-32768～32767 の 2 バイト符号付き整数。
PIC 9(4) COMP-5	データ・タイプ欄が U、長さ欄が 5、小数点以下桁数欄が 0	2	0～65535 の 2 バイト符号なし整数。
PIC S9(9) COMP-4 PIC S9(9) BINARY	データ・タイプ欄が B、長さ欄が 9、小数点以下桁数欄が 0	4	-999999999～+999999999 の 4 バイト符号付き整数。
PIC S9(9) BINARY と *NOSTDTRUNC PIC S9(9) COMP-5	データ・タイプ欄が I、長さ欄が 10、小数点以下桁数欄が 0	4	-2147483648～2147483647 の 4 バイト符号付き整数。
PIC 9(9) COMP-5	データ・タイプ欄が U、長さ欄が 10、小数点以下桁数欄が 0	4	0～4294967295 の 4 バイト符号なし整数。
PIC S9(18) COMP-4 PIC S9(18) BINARY PIC S9(18) COMP-5	データ・タイプ欄が I、長さ欄が 20、小数点以下桁数欄が 0	8	-9223372036854775808 ～ 9223372036854775807 の 8 バイト符号付き整数。
PIC 9(18) COMP-5	データ・タイプ欄が U、長さ欄が 20、小数点以下桁数欄が 0	8	0～18446744073709551615 の 8 バイト符号なし整数。
USAGE IS COMP-1	データ・タイプ欄が F、長さ欄が 4	4	4 バイトの内部浮動小数点フィールド。
USAGE IS COMP-2	データ・タイプ欄が F、長さ欄が 8	8	8 バイトの内部浮動小数点フィールド。
05 VL-FIELD. 10 i PIC S9(4) COMP-4. 10 data PIC X(n).	データ・タイプ欄が A、長さ欄が n。 キーワードは VARYING。	n+2	可変長フィールド (i は指定した長さ、n は最大長)。

表 23. ILE COBOL と ILE RPG との間のデータ・タイプの互換性 (続き)

ILE COBOL	ILE RPG の I 仕様書、 D 仕様書、または C 仕様書	長さ	説明
05 n PIC 9(9) COMP-4. 05 x redefines n PIC X(4).	データ・タイプ欄が U、長さ欄が 4。文字配列によってオーバーレイされたデータ構造内の符号のないフィールドへの移動を操作し、各バイトに対してビット操作を使用するため。	4	ビット・フィールドは 16 進数リテラルを使用して操作可能。
01 record 05 field1... 05 field2...	データ構造	n	構造体。構造体の内容を変更する場合、渡す構造は、構造体を指すポインターとして渡すことが必要。
USAGE IS POINTER	データ・タイプ欄が *	16	ポインター。
PROCEDURE-POINTER	データ・タイプ欄が *、キーワードが PROCPTR	16	プロシージャーを指す 16 バイトのポインター。
USAGE IS INDEX	データ・タイプ欄が I、長さ欄が 10、小数点以下桁数欄が 0	4	4 バイトの整数。
REDEFINES	データ構造サブフィールド	n	共用体のエレメント。
OCCURS	キーワード OCCURS またはキーワード DIM	n*(data_type の長さ)	配列。
FORMAT DATE	データ・タイプ欄が D	n	日付データ・タイプ。
FORMAT TIME	データ・タイプ欄が T	n	時刻データ・タイプ。
FORMAT TIMESTAMP	データ・タイプ欄が Z	n	タイム・スタンプ・データ・タイプ。
PIC G(n)	データ・タイプ欄が G	n*2	グラフィック (2 バイト) データ・タイプ。
サポートなし。	データ・タイプ欄が C	n*2	UCS-2 (汎用文字セット) データ・タイプ。

ILE RPG プログラムまたはプロシージャーからの制御の戻り

ILE RPG メイン・プロシージャーから戻ると、RETURN 命令コードによって制御が呼び出し側プログラムに戻されます。 RETURN 命令コードを実行する前に、SETON 命令コードを使用して LR 標識を設定した場合、呼び出し先 ILE RPG プログラムは、呼び出し側プログラムへの戻り時にその初期状態にリセットされます。それ以外の場合には、呼び出し先 ILE RPG プログラムは最後に使用された状態のままになります。

ILE RPG サブプロシージャーから戻ると、RETURN 命令コードによって制御が呼び出し側プログラムに戻されます。プロシージャーが値を戻すと、戻された値は RETURN 命令の拡張演算項目 2 に指定されます。サブプロシージャーが値を戻す場合、COBOL CALL ステートメントには RETURNING 句がなければなりません。

注: LR 標識は、サブプロシージャーから戻った場合は、何の意味もありません。

呼び出し先プログラムが例外なしで終了した場合、ILE COBOL プログラムの CALL ステートメントの次のステートメントに制御が戻されます。呼び出し先プロ

グラムが例外で終了した場合には、制御は ILE COBOL プログラムで指定された例外処理プロシージャーに戻されます。例外処理プロシージャーに制御を渡す方法については 415 ページの『第 16 章 ILE COBOL のエラーおよび例外の処理』を参照してください。

呼び出し先プログラムでは、ILE COBOL 呼び出し側プログラムをすべてスキップするエスケープ・メッセージを出すこともできます。この場合、ILE COBOL プログラムの呼び出しは取り消されます。呼び出しの取り消しは、ILE COBOL プログラムから戻る場合と同様です。

ILE CL プログラムおよびプロシージャーの呼び出し

ILE COBOL プログラムでは、動的プログラム呼び出しまたは静的プロシージャー呼び出しを使用して、ILE CL プログラムおよびプロシージャーを呼び出すことができます。

動的プログラム呼び出しを使用して ILE CL プログラムを呼び出す場合、ILE CL プログラムは別個のプログラム・オブジェクトとしてコンパイルおよびバインドしなければなりません。静的プロシージャー呼び出しを使用して ILE CL プロシージャーを呼び出す場合、ILE CL プロシージャーでは、まずコンパイルしてモジュール・オブジェクトにして、それから ILE COBOL 呼び出し側プログラムにバインドしなければなりません。 ILE CL プログラムおよびプロシージャーのコンパイルとバインディングについては、「*CL プログラミング*」を参照してください。

ILE CL プログラムまたはプロシージャーを ILE COBOL プログラムから呼び出すには、CALL *リテラル* のステートメントを使用します (*リテラル* は ILE CL プログラムまたはプロシージャーの名前)。 ILE CL プログラムまたはプロシージャーを呼び出すには、別の ILE COBOL サブプログラムを呼び出す場合と同様に、CALL *リテラル* のステートメントを入力します。 ILE COBOL プログラムに CALL ステートメントを書いて、動的プログラム呼び出しまたは静的プロシージャー呼び出しを使用して ILE CL プログラムを呼び出す方法については 247 ページの『静的プロシージャー呼び出しと動的プログラム呼び出しの使用』を参照してください。

CALL *id* を使用することによって、ILE CL プログラムを ILE COBOL プログラムから呼び出すこともできます。 CALL *id* については 250 ページの『CALL id の使用』を参照してください。

ILE CL プログラムまたはプロシージャーへのデータの引き渡し

CALL...BY REFERENCE または CALL...BY CONTENT を使用することにより、呼び出し先の ILE CL プログラムまたはプロシージャーにデータを渡すことができます。 CALL...BY REFERENCE または CALL...BY CONTENT の使用の詳細については 262 ページの『CALL...BY REFERENCE、BY VALUE、または BY CONTENT を使用したデータの受け渡し』を参照してください。

CALL...BY REFERENCE を使用してデータを ILE CL プログラムに渡すと、ILE CL プログラムに受け入れられた引き数リストの中にデータ項目へのポインターが入れられます。 CALL...BY CONTENT を使用してデータを ILE CL プログラムに渡

すと、データ項目の値が一時ロケーションにコピーされ、コピーの一時ロケーションのアドレスを含むポインターは、ILE CL プログラムに受け入れられた引き数リストに置かれます。

ILE COBOL プログラムにおいて、ILE CL プログラムまたはプロシージャーに渡したい引き数をデータ部に記述する方法は、他のデータ項目をデータ部に記述するときの方法と同じです。渡したい引き数の記述方法の詳細については 261 ページの『プログラム間でのデータの受け渡しと共用』を参照してください。

呼び出し先 ILE CL プログラムにおいて、ILE COBOL プログラムから受け取るパラメーターは、PGM ステートメントの PARM パラメーターに記述します。受け取るパラメーターの PARM パラメーターに指定する順序は、ILE COBOL プログラムの CALL ステートメントでの順序と同じでなければなりません。パラメーターの位置のほかに、パラメーターの長さとタイプにも注意してください。呼び出し先 ILE CL プログラムで指定されているパラメーターの長さとタイプは、呼び出し側 ILE COBOL プログラムで指定されているパラメーターの長さとタイプと同じに宣言されていなければなりません。

受け取るパラメーターを呼び出し先 ILE CL プログラムに記述するには、DCL ステートメントを使用します。 DCL ステートメントの順序は重要ではありません。どの変数を受け取るかを決定するのは、PGM ステートメントにパラメーターを指定する際の順序だけです。次の例は、ILE CL プログラムでパラメーターを記述する方法を示すものです。

```
PGM PARM(&P1 &P2);
DCL VAR(&P1); TYPE(*CHAR) LEN(32)
DCL VAR(&P2); TYPE(*DEC) LEN(15 5)
.
.
.
RETURN
ENDPGM
```

ILE CL プログラムにおけるパラメーターの記述方法については、「*CL プログラミング*」を参照してください。

ILE CL と ILE COBOL との間のデータ・タイプの互換性

ILE CL と ILE COBOL のデータ・タイプは異なります。 ILE CL で作成されたプログラムと ILE COBOL で作成されたプログラムとの間でデータを渡したい場合には、このデータ・タイプの違いに注意する必要があります。 ILE CL と ILE COBOL のデータ・タイプによっては、相手の言語に直接対応するものがない場合があります。

表 24 に ILE COBOL データ・タイプと ILE CL データ・タイプの互換性を示します。

表 24. ILE COBOL と ILE CL との間のデータ・タイプの互換性

ILE COBOL	ILE CL	長さ	説明
PIC X(n).	TYPE(*CHAR) LEN(n)	n	n=1～32,766 の文字フィールド。

表 24. ILE COBOL と ILE CL との間のデータ・タイプの互換性 (続き)

ILE COBOL	ILE CL	長さ	説明
01 flag PIC 1. 88 flag-on VALUE B'1'. 88 flag-off VALUE B'0'.	TYPE(*LGL)	1	'1' または '0' を入れる。
PIC S9(n-p)V9(p) COMP-3.	TYPE(*DEC) LEN(n p)	n/2+1	パック 10 進数。 n の最大値 =15。 p の最大値 =9。
PIC S9(n-p)V9(p) PACKED-DECIMAL.	TYPE(*DEC) LEN(n p)	n/2+1	パック 10 進数。 n の最大値 =15。 p の最大値 =9。
USAGE IS COMP-1	サポートなし。	4	4 バイトの内部浮動小数点数。
USAGE IS COMP-2	サポートなし。	8	8 バイトの内部浮動小数点数。

ILE CL プログラムまたはプロシージャーからの制御の戻り

ILE CL で RETURN コマンドを使用すると、制御が呼び出し側プログラムに戻されます。

呼び出し先プログラムが例外なしで終了した場合、ILE COBOL プログラムの CALL ステートメントの次のステートメントに制御が戻されます。呼び出し先プログラムが例外で終了した場合には、制御は ILE COBOL プログラムで指定された例外処理プロシージャーに戻されます。例外処理プロシージャーに制御を渡す方法については 415 ページの『第 16 章 ILE COBOL のエラーおよび例外の処理』を参照してください。

呼び出し先プログラムでは、ILE COBOL 呼び出し側プログラムをすべてスキップするエスケープ・メッセージを出すこともできます。この場合、ILE COBOL プログラムの呼び出しは取り消されます。呼び出しの取り消しは、ILE COBOL プログラムから戻る場合と同様です。

OPM 言語の呼び出し

OPM COBOL/400 または OPM RPG/400® などの OPM 言語で作成されたプログラムを ILE COBOL から呼び出す場合、動的プログラム呼び出ししか使用できません。 OPM プログラムを静的に ILE COBOL プログラムにバインドすることはできません。静的プロシージャー呼び出しを使用して OPM プログラムを呼び出そうとすると、エラー・メッセージが出されます。バインド時に、静的プロシージャー呼び出しに未解決参照をしたことに対する警告メッセージが、バインダーから出されます。この警告メッセージを無視して ILE プログラム・オブジェクトを作成すると、実行時に静的プロシージャー呼び出しを行った時点で例外が発生します。

OPM プログラムを ILE COBOL プログラムから呼び出すには、CALL *リテラル* のステートメントを使用します (*リテラル* は OPM プログラムの名前)。 OPM プログラムを呼び出すには、別の ILE COBOL サブプログラムを動的プログラム呼び出しを使用して呼び出す場合と同じようにして、CALL *リテラル* のステートメントを入力します。 ILE COBOL プログラムに CALL ステートメントを書いて、動的プログラム呼び出しを使用して OPM プログラムを呼び出す方法については 249 ページの『CALL リテラルを使用した動的プログラム呼び出しの実行』を参照してください。

CALL *id* を使用することによって、OPM プログラムを ILE COBOL プログラムから呼び出すこともできます。 CALL *id* については 250 ページの『CALL id の使用』を参照してください。

OPM 言語で作成されたプログラムは、デフォルト活動化グループ (*DFTACTGRP) でしか実行できません。

OPM プログラムから ILE COBOL プログラムを呼び出すには、別の OPM プログラムを呼び出すときと同じ呼び出しのセマンティクスを使用して行うことができます。

OPM プログラムと ILE COBOL プログラムの間で外部データを共用することはできません。

OPM COBOL/400 プログラムの呼び出し

OPM COBOL/400 プログラムは、デフォルト活動化グループ (*DFTACTGRP) でしか実行できません。 ILE COBOL プログラムは、デフォルト活動化グループ (*DFTACTGRP)、*NEW ILE 活動化グループ、および名前付き ILE 活動化グループで実行することができます。

注: CRTPGM では *DFTACTGRP を ACTGRP パラメーターに明示的に指定することはできませんが、*CALLER を ACTGRP パラメーターに指定することはできます。 *CALLER を ACTGRP パラメーターに指定すると、OPM COBOL/400 プログラム (または任意の OPM プログラム) から呼び出された ILE COBOL プログラムをデフォルト活動化グループで実行することができます。 ILE COBOL プログラムをデフォルト活動化グループで実行するのに、これ以外の方法はありません。 ILE COBOL プログラムを、デフォルト活動化グループのハード管理境界にすることはできません。

OPM COBOL/400 プログラムと ILE COBOL プログラムの複数言語混在アプリケーションを実行する場合は、次の方法に従って、できるだけ OPM COBOL/400 実行単位に近い形で実行する必要があります。

- 実行するすべてのプログラムは、デフォルト活動化グループ (*DFTACTGRP) で実行しなければなりません。
- 活動化グループで最初に活動化される COBOL プログラムは、OPM COBOL/400 プログラムでなければなりません。
- 実行単位を終了するには、STOP RUN を OPM COBOL/400 プログラムから出すか、または GOBACK を OPM COBOL/400 メイン・プログラムから出す必要があります。
- 暗黙のうちに STOP RUN を生じさせる例外がある場合は、暗黙のうちに STOP RUN が OPM COBOL/400 によって起動される場合と同じように処理する必要があります。

OPM COBOL/400 プログラムと ILE COBOL プログラムの複数言語混在アプリケーションをデフォルト活動化グループで実行する場合、各 ILE COBOL プログラムは、OPM COBOL/400 プログラムにより非 COBOL プログラムと見なされます。また、各 OPM COBOL/400 プログラムは、ILE COBOL プログラムにより非 COBOL

プログラムと見なされます。さらに、OPM COBOL/400 により呼び出される各 ILE COBOL プログラムは、ILE COBOL プログラムが出した STOP RUN の有効範囲を決めるソフト管理境界を生成します。

STOP RUN が ILE COBOL プログラムにより出されると、制御が OPM COBOL/400 プログラムに戻され、ILE COBOL プログラムの状態は更新されず、OPM COBOL/400 実行単位は終了しません。 STOP RUN が OPM COBOL/400 プログラムから出されると、制御は現行のメイン OPM COBOL/400 プログラムを呼び出したプログラムに戻され、OPM COBOL/400 実行単位は終了します。

OPM COBOL/400 プログラムと ILE COBOL プログラムの複数言語混在アプリケーションでは、ILE COBOL プログラムが *NEW または名前付きの ILE 活動化グループで実行され、OPM COBOL/400 プログラムがデフォルト活動化グループで実行されている場合、ILE COBOL プログラムから出される STOP RUN による効果は、最も近い管理境界がハード管理境界であるかソフト管理境界であるかによって異なります。最も近い管理境界がハード管理境界である場合には、制御はハード管理境界の呼び出し側に戻され、*NEW または名前付き ILE 活動化グループは終了します。最も近い管理境界がソフト管理境界である場合は、制御はソフト管理境界の呼び出し側に戻されますが、ILE COBOL プログラムの *NEW または名前付き ILE 活動化グループは活動状態のままです。

注: 上記の内容は、OPM COBOL/400 実行単位に準じたものではありません。

EPM 言語の呼び出し

EPM C/400®、Pascal、および FORTRAN などの EPM 言語で作成されたプログラムは、QPXXCALL への CALL を使用して、ILE COBOL プログラムから呼び出すことができます。

次の例では、ILE COBOL プログラムが QPXXCALL を使用して、Pascal のプロシージャーを呼び出しています。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/COBTOPAS       ISERIES1   06/02/15 13:38:36       ページ    2
                                ソース
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID. COBTOPAS.
    3     000300 ENVIRONMENT DIVISION.
    4     000400 CONFIGURATION SECTION.
    5     000500   SOURCE-COMPUTER. IBM-ISERIES.
    6     000600   OBJECT-COMPUTER. IBM-ISERIES.
    7     000700 DATA DIVISION.
    8     000800 WORKING-STORAGE SECTION.
    9     000900 01  PARAMETER-LIST.
   10     001000     05  ENTRY-NAME   PIC X(100)         VALUE "SQUARE".
   11     001100     05  ENTRY-ID     PIC X(10)          VALUE "*MAIN".
   12     001200     05  PROG-NAME    PIC X(20)          VALUE "MATH".
   13     001300     05  A-REAL       PIC S9(9)  COMP-4  VALUE 0.
   14     001400     05  CLEAN        PIC S9(9)  COMP-4  VALUE 0.
   15     001500     05  INPT         PIC S99            VALUE 0.
   16     001600 PROCEDURE DIVISION.
          001700 MAINLINE.
   17     001800     DISPLAY "ENTER AREA NUMBER:".
   18     001900     ACCEPT INPT.
   19     002000     MOVE INPT TO A-REAL.
   20     002100     CALL "QPXXCALL" USING ENTRY-NAME
          002200                           ENTRY-ID
          002300                           PROG-NAME
          002400                           A-REAL.
   21     002500     DISPLAY A-REAL.
   22     002600     CALL "QPXXDLTE" USING CLEAN.
   23     002700     STOP RUN.
          002800
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 76. ILE COBOL プログラムからの Pascal プロシージャーの呼び出し

```
segment MATH;
procedure SQUARE ( var X : integer) ; external ;
procedure SQUARE;
   begin
   X := X * X
end; .
```

図 77. ILE COBOL プログラムから呼び出される Pascal 入り口点

Pascal を使用すると、ILE COBOL プログラムは、Pascal プロシージャーをサブプログラムとして呼び出すことができます。そのためには、Pascal プロシージャーに EXTERNAL ディレクティブを指定します (図 77 を参照)。上記の例では、QPXXCALL の ENTRY-ID パラメーターを最初に呼び出した時点で、Pascal メイン環境が確立されます。 QPXXDLTE を使用することにより、Pascal 再入可能 (リエントラント) およびメイン環境を消去することができます。 CLEAN パラメーターにゼロを指定して QPXXDLTE に渡すと、現行の Pascal のメイン環境が削除されます。

EPM プログラムから ILE COBOL プログラムを呼び出すには、別の EPM プログラムを呼び出すときと同じ呼び出しのセマンティクスを使用して行うことができます。

EPM プログラムと ILE COBOL プログラムの間で外部データを共用することはできません。

ILE COBOL プログラムからの CL コマンドの発行

QCMDEXC の動的プログラム呼び出しにより、ILE COBOL プログラムから CL コマンドを出すことができます。

次のプログラム例では、QCMDEXC の CALL (シーケンス番号 000160) により、ライブラリー・リスト項目の追加 (ADDLIBLE) CL コマンド (シーケンス番号 000110) が処理されます。この CL コマンドが正常に終了すると、ライブラリー COBOLTEST がライブラリー・リストに追加されます。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/CMDXMPLE       ISERIES1   06/02/15 13:40:28    ページ  2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID. CMDXMPLE.
    3     000300 ENVIRONMENT DIVISION.
    4     000400 CONFIGURATION SECTION.
    5     000500   SOURCE-COMPUTER. IBM-ISERIES.
    6     000600   OBJECT-COMPUTER. IBM-ISERIES.
    7     000700 DATA DIVISION.
    8     000800 WORKING-STORAGE SECTION.
    9     000900 01  PROGRAM-VARIABLES.
   10     001000     05  CL-CMD       PIC X(33)
          001100                      VALUE "ADDLIBLE COBOLTEST".
   11     001200     05  PACK-VAL     PIC 9(10)V9(5) COMP-3
          001300                      VALUE 18.
   12     001400 PROCEDURE DIVISION.
          001500 MAINLINE.
   13     001600     CALL "QCMDEXC" USING CL-CMD PACK-VAL.
   14     001700     STOP RUN.
          001800
                          * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

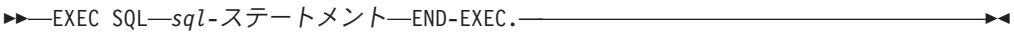
図 78. ILE COBOL プログラムからの CL コマンドの発行

QCMDEXC の詳細については、「*CL プログラミング*」を参照してください。

ILE COBOL プログラムへの構造化照会言語 (SQL) ステートメントの組み込み

ILE COBOL ソース・プログラムに組み込まれる SQL ステートメントの構文は、次のとおりです。

SQL ステートメントの組み込み



ソース・プログラムのメンバー・タイプが SQLCBLLE の場合、COBOL 構文検査プログラムが SQL ステートメントを検出すると、そのステートメントは SQL 構文検査プログラムに渡されます。エラーが検出されると、メッセージが戻されます。

SQL ステートメントが検出され、メンバー・タイプが SQLCBLLE でなかった場合、ILE COBOL ステートメントにエラーがあることを示すメッセージが戻されます。

組み込まれた SQL ステートメントとそれよりも前の ILE COBOL ステートメントの両方にエラーがある場合、SQL エラー・メッセージは、前の COBOL エラーが訂正されるまで表示されません。

SQL プログラムを作成して、それを ILE COBOL プログラムで実行することができます。 ILE COBOL プログラムで使用する SQL カーソルは、モジュール・オブ

ジェクトまたは活動化グループのどちらかに範囲指定されます。 SQL カーソルの範囲指定は、SQL プログラム作成コマンド (CRTSQLxxx) の CLOSQLCSR パラメーターで行います。

SQL ステートメントと SQL カーソルの詳細については、 Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベース」カテゴリーの中の『*DB2 Universal Database for AS/400*』を参照してください。

現在の西暦を取り出す ILE API の呼び出し

以下の図 79 の例は、ILE バインド可能 API、ローカル現在時刻取得 (CEELOCT) を使用して、4 桁年を取り出す方法を示すものです。この API は、3 つの形式で現在のローカル時刻を取り出します。 3 つ目の形式はグレゴリオ暦日付であり、最初の 4 文字が年です。

次のセクション 374 ページの『現在の西暦を取り出すための組み込み関数または ACCEPT ステートメントの使用法』では、いくつかの組み込み関数と、ACCEPT ステートメントを使用して同じことを行う方法について説明します。

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. DATE1.
* Example program to get the 4 digit year in ILE COBOL for ISERIES
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
   SOURCE-COMPUTER. IBM-ISERIES
   OBJECT-COMPUTER. IBM-ISERIES
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 date-vars.
    05 lilian            pic 9(9) usage binary.
    05 lilian-time-stamp usage comp-2.
    05 gregorian-date.
       10 greg-year      pic x(4).
       10 greg-month     pic x(2).
       10 greg-day       pic x(2).
       10 greg-time      pic x(9).
       10 filler         pic x(6).
 PROCEDURE DIVISION.
 TEST-PARA.
     call procedure "CEELOCT" using
         lilian lilian-time-stamp
         gregorian-date.
     display "date is " gregorian-date.
     display "year " greg-year.
     display "month " greg-month.
     STOP RUN.
```

図 79. 現在の西暦を取り出す例

現在の西暦を取り出すための組み込み関数または ACCEPT ステートメントの使用法

次の組み込み関数のいずれかを使用しても、現在の西暦すなわち 4 桁年を取り出すことができます。

CURRENT-DATE
現在の 4 桁年のほか、現在日付と時刻に関する他の情報も戻します。

DATE-OF-INTEGER
整数日付、すなわち 1600 年 12 月 31 日以降の日数をとり、形式 YYYYMMDD の 4 桁年のグレゴリオ暦日付を戻します。

DAY-OF-INTEGER
整数日付、すなわち 1600 年 12 月 31 日以降の日数をとり、形式 YYYYDDD の 4 桁年の年間通算日を戻します。

DATE-TO-YYYYMMDD
2 桁年のグレゴリオ暦日付を 4 桁年のグレゴリオ暦日付に変換します。

DAY-TO-YYYYDDD
2 桁年の年間通算日を 4 桁年の年間通算日に変換します。

EXTRACT-DATE-TIME
日付、時刻、またはタイム・スタンプ項目に含まれている日付または時刻情報の一部を抽出します。年は、4 桁の年または 1 桁の世紀として抽出されます。

YEAR-TO-YYYY
2 桁の年を 4 桁の年に変換します。

ACCEPT ステートメントの FROM DATE YYYYMMDD 句を使用すると、4 桁年のグレゴリオ暦日付をシステムから取り出すことができます。 ACCEPT ステートメントの FROM DAY YYYYDDD 句を使用すると、4 桁年の年間通算日をシステムから取り出すことができます。

IFS API の呼び出し

ILE COBOL プログラムから IFS API を呼び出すことができます。 IFS API では、COBOL プログラムの中で操作可能な記述子が必要であるかどうかを、確実にチェックする必要があります。必要であれば、操作可能な記述子を SPECIAL-NAMES 段落に指定する必要があります。

図 80 は、IFS API の呼び出しの例です。

```
SPECIAL NAMES.
 LINKAGE TYPE PROCEDURE FOR "open" USING ALL DESCRIBED.


 ... more declaration and procedure statements

 CALL "open" USING ...
```

図 80. IFS API の呼び出し

第 14 章 ILE COBOL プログラムでのポインターの使用

データ項目、ILE プロシージャー、またはプログラム・オブジェクトのアドレスの受け渡しをしたいときは、ILE COBOL プログラム内で**ポインター** (アドレス値を保管できるデータ項目) を使用することができます。

この章では次のことについて説明します。
- ポインターを定義および再定義する方法
- ポインターを初期設定する方法
- ポインターを読み書きする方法
- ポインターを使用してデータを操作する方法

ポインターの定義

ポインターは、次の 2 とおりの方法で定義できます。
- データ項目を指すポインター。このポインターは、USAGE POINTER 文節を使用して定義します。定義結果のデータ項目は、**ポインター・データ項目**と呼ばれます。
- ILE COBOL プログラム、ILE プロシージャー、またはプログラム・オブジェクトを指すポインター。このポインターは、USAGE PROCEDURE-POINTER 文節を使用して定義します。定義結果のデータ項目は、**プロシージャー・ポインター・データ項目**と呼ばれます。

AS/400 システムでのポインターの長さは 16 バイトです。

ILE COBOL ポインター・データ項目は、システム・スペース・オブジェクトを指します。ポインターの一部分には、どの AS/400 スペース・オブジェクトを指しているかなどの属性が記述されています。ポインターの別の部分には、AS/400 システム・スペース・オブジェクトのオフセットが入れられます。

ILE COBOL プロシージャー・ポインター・データ項目は、AS/400 **オープン・ポインター**です。オープン・ポインターには、別タイプの AS/400 ポインターとして使用される機能があります。特に、ILE COBOL プロシージャー・ポインター・データ項目がプログラム・オブジェクトに設定されている場合は、オープン・ポインターには AS/400 システム・ポインターが含まれます。 ILE COBOL プロシージャー・ポインター・データ項目が ILE プロシージャーに設定されている場合、オープン・ポインターには AS/400 プロシージャー・ポインターが含まれます。

ポインターは、等価性の比較を行ったり、他のポインター項目の値を設定するために使用できる 16 バイトの基本項目です。

ポインター・データ項目は次のところでしか使用できません。
- SET ステートメント (形式 5 と 7 のみ)
- 関係条件
- CALL ステートメントまたは手続き部ヘッダーの USING 句。
- LENGTH OF および ADDRESS OF 特殊レジスターのオペランド。

プロシージャー・ポインター・データ項目は、次のところでしか使用できません。
- SET ステートメント (形式 6 のみ)
- 関係条件
- CALL ステートメントまたは手続き部ヘッダーの USING 句。
- LENGTH OF および ADDRESS OF 特殊レジスターのオペランド。
- ターゲットとしての CALL ステートメント。

ポインターを関係条件で使用する場合、有効なのは等号演算子または非等号演算子だけです。

AS/400 システムにおいて、ポインター・データ項目は単なる 2 進数ではないので、ポインターを整数として処理することはできません。

ポインター・データ項目は、USAGE IS POINTER 文節を使用して明示的に定義され、ADDRESS OF 特殊レジスターまたは ADDRESS OF 項目を使用して暗黙のうちに定義されます。

グループ項目が USAGE IS POINTER 文節で記述されている場合、そのグループ項目内の基本項目はポインター・データ項目です。そのグループ自体はポインター・データ項目ではなく、ポインター・データ項目を使用できる構文中では使用できません。グループ項目が USAGE PROCEDURE-POINTER 文節で記述されている場合は、同じ規則が適用されます。基本項目の USAGE 文節が、その項目が属するグループの USAGE 文節に矛盾するものであってはなりません。

ポインターは、MOVE ステートメントまたは入出力ステートメント中で参照されるグループの一部にすることができます。しかし、ポインターがグループの一部である場合、そのステートメントの実行時にポインターの値から別の内部表記形式への変換は実行されません。

ILE C および他の言語を使用して、ポインターをテラスペース記憶域に宣言することができます。 ILE C は、このタイプのストレージをアドレッシングするために特殊なコンパイル時オプションが必要ですが、ILE COBOL では、V4R4M0 のターゲット・リリースまたはそれ以上でコンパイルすれば、このストレージを常にアドレッシングすることができます。テラスペースのポインターについての詳細は、「*ILE 概念*」を参照してください。

ポインターの位置合わせ

ポインターは、任意のレベル (88 を除く) のプログラムの FILE SECTION、WORKING-STORAGE SECTION、または LINKAGE SECTION に定義できます。

ポインターが i5/OS で参照される場合、16 バイトのストレージ境界になければなりません。 **ポインターの位置合わせ**は、ILE COBOL コンパイラーによって、グループ項目中のポインター項目の位置を、レコードの先頭から 16 バイトの倍数のオフセットにする処理のことです。ポインター項目が 16 バイトの境界にない場合は、ポインター位置合わせ例外が ILE COBOL プログラムに送られます。通常、ポインター位置合わせ例外は LINKAGE セクションで発生します。そこでの位置合わせは、ユーザーが行うことになります。

FILE SECTION と WORKING-STORAGE SECTION では、コンパイラーは暗黙の FILLER 項目を追加することにより、この例外が起こらないようにします。コンパイラーが暗黙の FILLER 項目を追加すると、警告が出されます。 LINKAGE SECTION では、コンパイラーが暗黙の FILLER 項目を追加することはありません。しかし、FILE SECTION または WORKING-STORAGE SECTION にグループ項目が検出されたとしたら何バイトの FILLER が追加されることになるかを示す警告が出されます。

以下の例のように、USAGE IS POINTER 文節または USAGE IS PROCEDURE-POINTER 文節を指定することで、データ項目をポインターとして定義することができます。

```
ID DIVISION.
PROGRAM-ID. PROGA.
WORKING-STORAGE SECTION.
 77  APTR USAGE POINTER.
 77  APROC-PTR USAGE PROCEDURE-POINTER.
 01  AB.
     05 BPTR  USAGE POINTER.
     05 BVAR  PIC S9(3) PACKED-DECIMAL.
LINKAGE SECTION.
01  AVAR.
    05 CVAR  PIC X(30).
PROCEDURE DIVISION.
     SET APTR TO ADDRESS OF AVAR.
     SET APROC-PTR TO ENTRY "PROGA".
```

上記の例で、AVAR は 01 レベルのデータ項目なので、ADDRESS OF AVAR は ADDRESS OF 特殊レジスターです。特殊レジスターは実際のストレージ域なので、SET ステートメントにより ADDRESS OF AVAR の内容がポインター・データ項目 APTR に転送されます。

上記の例で、SET ステートメントで ADDRESS OF CVAR が使用される場合には、特殊レジスターは存在しません。その代わりに、計算された CVAR のアドレスがポインター・データ項目 APTR に割り当てられます。

上記の例の 2 番目の SET ステートメントは、プロシージャー・ポインター・データ項目 APROC-PTR を最外部の ILE COBOL プログラム ″PROGA″ に設定します。

ポインター位置合わせを行うための FILE SECTION および WORKING-STORAGE SECTION の書き方

FILE SECTION および WORKING-STORAGE SECTION では、すべての 01 レベルの項目と 77 レベルの項目 (および一部の 66 レベルの項目) は、16 バイト境界に置かれます。

グループ構造の中では、ポインターも 16 バイトの境界になければなりません。そのために、ILE COBOL コンパイラーはポインターの直前に FILLER 項目を追加します。 FILLER 項目を追加しないようにするには、ポインターをグループ項目の先頭に置いてください。

ポインターがテーブルの一部である場合、そのテーブルの最初の項目が 16 バイト境界に置かれます。それ以降に出現するポインターの位置を 16 バイト境界にするため、必要に応じて FILLER 項目がテーブルの末尾に追加されます。

ポインター位置合わせの例を次に示します。

```
WORKING-STORAGE SECTION.
 77  APTR USAGE POINTER.
 01  AB.
     05 ALPHA-NUM PIC X(10).
     05 BPTR  USAGE PROCEDURE-POINTER.
 01  EF.
     05 ARRAY-1 OCCURS 3 TIMES.
        10 ALPHA-NUM-TWO PIC X(14).
        10 CPTR  USAGE POINTER.
        10 ALPHA-NUM-THREE PIC X(5).
```

この例で、APTR はポインター・データ項目です。したがって、77 レベルの項目の位置は 16 バイト境界になります。グループ項目 AB は 01 レベルの項目であり、自動的に 16 バイト境界になります。グループ項目 AB の中で、BPTR は 16 バイト境界にはなりません。位置合わせを適切に行うために、コンパイラーは 6 バイトの FILLER 項目を ALPHA-NUM の後に挿入します。最後に、最初に出現する CPTR の位置合わせのために、2 バイトの FILLER が必要です。
ALPHA-NUM-THREE の長さは 5 バイトしかないので、これ以降に出現する CPTR をすべて位置合わせするには、別に 11 バイトの FILLER を ARRAY-1 の末尾に追加しなければなりません。

FILE SECTION でポインターが定義されていて、かつファイルのブロック化が有効でない場合、各 01 項目は 16 バイト境界になります。ファイルのブロック化が有効な場合は、ブロックの最初のレコードが 16 バイト境界になることしか保証されません。したがって、ブロック化が有効なファイルには、ポインターを定義しないでください。ブロック化については 466 ページの『入力レコードの非ブロック化および出力レコードのブロック化』を参照してください。

ポインターの再定義

ポインター・データ項目やプロシージャー・ポインター・データ項目は、REDEFINES 文節の主語 (サブジェクト) にすることも目的語 (オブジェクト) にすることもできます。

ポインターが REDEFINES 文節のサブジェクトである場合は、そのオブジェクト・データ項目は 16 バイト境界になければなりません。たとえば、以下の場合、

```
WORKING-STORAGE SECTION.
 01  AB.
     05 ALPHA-NUM PIC X(16).
     05 APTR REDEFINES ALPHA-NUM USAGE POINTER.
     05 BPTR  USAGE POINTER.
     05 CPTR REDEFINES BPTR USAGE POINTER.
```

上記の例では、APTR と CPTR は、いずれも 16 バイト位置合わせがなされている項目を再定義するポインター・データ項目です。次の例では、再定義された項目によって重大なコンパイラー・エラーになります。

```
WORKING-STORAGE SECTION.
 01  EF.
     05 ALPHA-NUM PIC X(5).
     05 HI.
        10 ALPHA-NUM-TWO PIC X(11).
        10 APTR  USAGE POINTER.
     05 BPTR REDEFINES HI USAGE POINTER.
```

上記の例では、APTR が 16 バイトの境界に位置合わせされます。したがって、ILE COBOL コンパイラーが FILLER 項目を追加して APTR を位置合わせする必要はありません。グループ項目 HI は 16 バイト境界にないので、ポインター・データ項目 BPTR も 16 バイト境界にはなりません。 ILE COBOL コンパイラーが FILLER 項目を追加して BPTR を 16 バイトの境界にすることができないため、重大エラーになります。

次の例は上記の例と似ていますが、ILE COBOL コンパイラーはポインター・データ項目を 16 バイト境界にすることができます。

```
WORKING-STORAGE SECTION.
 01  EF.
     05 ALPHA-NUM PIC X(5).
     05 HI.
        10 ALPHA-NUM-TWO PIC X(11).
        10 APTR  USAGE POINTER.
        10 ALPHA-NUM-THREE PIC X(5).
     05 KL REDEFINES HI.
        10 BPTR USAGE POINTER.
```

この例では、グループ項目 KL は 16 バイト境界になりませんが、コンパイラーは 11 バイトの FILLER をポインター・データ項目 BPTR の前に追加することによって、BPTR が 16 バイト境界になるようにします。

NULL 形象定数を用いてポインターを初期設定する

NULL 形象定数は、USAGE IS POINTER、USAGE IS PROCEDURE-POINTER、ADDRESS OF、または ADDRESS OF 特殊レジスターに有効なアドレスがないことを示すために使用される値を表します。たとえば、次のとおりです。

```
WORKING-STORAGE SECTION.
 77 APTR  USAGE POINTER VALUE NULL.
PROCEDURE DIVISION.
    IF APTR = NULL THEN
       DISPLAY 'APTR IS NULL'
    END-IF.
```

この例の WORKING-STORAGE SECTION で、ポインター APTR は NULL に設定されています。手続き部の比較は真になり、DISPLAY ステートメントが実行されます。

AS/400 システムでは、ポインター・データ項目やプロシージャー・ポインター・データ項目の初期値は、VALUE 文節が NULL であるかどうかに関係なく NULL になります。

ポインターの読み取りと書き込み

FILE SECTION にポインターを定義して、WORKING-STORAGE のその他のポインター・データ項目と同じように設定したり使用したりすることができます。ただし、次の制約事項があります。

- ファイルのブロック化が有効な場合は、ブロックの最初のレコードが 16 バイト境界になることしか保証されません。したがって、ブロック化が有効なファイルには、ポインターを定義しないでください。
- ポインターを含むレコードは、ファイルに書き込むことができます。しかし、以後そのレコードを読み取ると、ポインター・データ項目とプロシージャー・ポインター・データ項目は NULL になります。

ポインターと LENGTH OF 特殊レジスターの併用

LENGTH OF 特殊レジスターには、ID によって使用されるバイト数が入れられます。このレジスターは、ポインター・データ項目やプロシージャー・ポインター・データ項目の値 16 を戻します。

手続き部で、LENGTH OF 特殊レジスターの暗黙定義と同じ定義の数値データ項目が使用される場所ならどこでも LENGTH OF を使用できます。ただし、LENGTH OF を添え字または受け取りデータ項目として使用することはできません。
LENGTH OF には、次の暗黙の定義があります。

```
USAGE IS BINARY, PICTURE 9(9)
```

次の例で、ポインターと LENGTH OF を併用する方法を示します。

```
 WORKING-STORAGE SECTION.
  77  APTR  USAGE POINTER.
  01  AB.
      05  BPTR  USAGE PROCEDURE-POINTER.
      05  BVAR  PIC S9(3) PACKED-DECIMAL.
      05  CVAR  PIC S9(3) PACKED-DECIMAL.
 PROCEDURE DIVISION.
      MOVE LENGTH OF AB TO BVAR.
      MOVE LENGTH OF BPTR TO CVAR.
```

この例では、グループ項目 AB の長さが変数 BVAR に移動されます。 BPTR の長さは 16 バイトで、BVAR 変数と CVAR 変数の長さは両方とも 2 バイトなので、BVAR の値は 20 です。 CVAR は値 16 を受け取ります。

LENGTH OF 特殊レジスターを使用するなら、ユーザー・スペース中のデータ構造をセットアップしたり、別のプログラムから受け取るアドレスを増やしたりすることができます。 LENGTH OF 特殊レジスターを使用してユーザー・スペース中のデータ構造を定義するプログラムの例については 387 ページの図 82 を参照してください。

LINKAGE SECTION 項目のアドレスの設定

通常、ある ILE COBOL プログラムが別のプログラムを呼び出すと、次の方法で、呼び出し側 ILE COBOL プログラムから呼び出し先 ILE COBOL プログラムにオペランドが渡されます。

- 呼び出し側プログラムは、CALL USING ステートメントを使用して、呼び出し先プログラムにオペランドを渡します。
- 呼び出し先プログラムは、PROCEDURE DIVISION ヘッダーに USING 句を指定します。

ILE 呼び出し側プログラムの CALL USING ステートメントにリストされている各オペランドごとに、対応するオペランドを、呼び出し先プログラムの PROCEDURE DIVISION の USING 句に指定しなければなりません。

ADDRESS OF 特殊レジスターを使用するなら、2 つのプログラムの USING 句の間で 1 対 1 マッピングを行う必要はありません。 LINKAGE SECTION のデータ項目が、PROCEDURE DIVISION ヘッダーの USING 句に指定されていない場合は、SET ステートメントを使用してそのデータ構造の開始アドレスを指定できます。 SET ステートメントが実行されると、このデータ項目のアドレスは設定済みなので、データ項目を自由に参照できます。この方法で SET ステートメントを使用する例については 387 ページの図 83 を参照してください。 387 ページの図 83 の **15** と **16** には、SET ステートメントを使用して、*ls-header-record* データ構造と *ls-user-space* データ構造の開始アドレスをユーザー・スペースの先頭に設定する方法が示されています。

ADDRESS OF および ADDRESS OF 特殊レジスターの使用

ILE COBOL プログラムの中で ADDRESS OF を指定すると、ADDRESS OF または ADDRESS OF 特殊レジスターとして参照される計算済みのアドレスを使用するかどうかをコンパイラーが判別します。 ADDRESS OF 特殊レジスターは、データ構造の開始アドレスで、すべての計算アドレスはそのアドレスから決定されます。ADDRESS OF 特殊レジスターはその構造の開始アドレスなので、01 レベルか 77 レベルのデータ項目でなければなりません。このデータ項目を変更するような参照を行うと、特殊レジスターはもはやデータ構造の開始アドレスではなくなります。それは計算アドレスすなわち ADDRESS OF になります。 ADDRESS OF 基本項目を使用していて、かつ ADDRESS OF 01 レベル項目が NULL に設定されているなら、ポインター例外 (MCH3601) になります。

計算された ADDRESS OF は、項目が変更される可能性のある場所では使用できません。変更できるのは ADDRESS OF 特殊レジスターだけです。たとえば 387 ページの図 83 の **17** では、ADDRESS OF 特殊レジスター項目のレベルは 01 なので、SET ステートメントはこのレジスターを使用します。 **18** では、レベル 01 の項目ですが、参照変更されているため、ADDRESS OF が使用されています。

MOVE ステートメントでのポインターの使用

MOVE ステートメントを使用して基本ポインターを移動することはできません。この場合は、SET ステートメントを使用しなければなりません。しかし、ポインターがグループ項目の一部のときは、暗黙のうちに移動されます。

MOVE ステートメントのコンパイル時に、ILE COBOL コンパイラーは、グループ項目にポインターを保持するコード (ポインター MOVE) か、保持しないコード (非ポインター MOVE) を生成します。

次のすべての条件を満たす場合、ポインター MOVE が実行されます。

1. MOVE ステートメントのソースまたは受け取り側にポインターが含まれている。
2. 両方の項目の長さが 16 バイト以上である。
3. データ項目が適切に位置合わせされている。
4. データ項目が英数字かグループ項目である。

上記の条件下では、2 つのデータ項目が適切に位置合わせされているかどうかを判別することが、非常に難しくなることがあります。

注: SET ステートメントを使用してポインターを転送するより、ポインター MOVE を使用する方が速度は遅くなります。

ポインター MOVE を行うには、項目は、16 バイト境界に対するオフセットが同じでなければなりません。 (これが真でない場合は警告が出されます。)

次の例は、3 つのデータ構造があって、MOVE ステートメントが出された場合の結果を示すものです。

```
WORKING-STORAGE SECTION.
  01  A.
      05  B      PIC X(10).
      05  C.
          10  D      PIC X(6).
          10  E      POINTER.
  01  A2.
      05  B2     PIC X(6).
      05  C2.
          10  D2     PIC X(10).
          10  E2     POINTER.
 01  A3.
     05  B3     PIC X(22).
     05  C3.
         10  D3     PIC X(10).
         10  E3     POINTER.
PROCEDURE DIVISION.
MOVE  A  to A2. 1
MOVE  A  to A3. 1
MOVE  C  to C2. 2
MOVE  C2 to C3. 3
```

図 81. MOVE ステートメントでのポインターの使用

1 移動される各グループ項目のオフセットはゼロなので、この結果はポインター MOVE になります。ポインターの整合性は維持されます。

2 オフセットが一致していないので、この結果は非ポインター MOVE になります。グループ項目 C のオフセットは 10、グループ項目 C2 のオフセットは 6 です。ポインターの整合性は維持されません。

3 グループ項目 C2 のオフセットが 6 で、16 バイト境界に関する C3 のオフセットも 6 なので、この結果はポインター MOVE になります。(オフセットが 16 より大きい場合は、16 バイト境界に対するオフセットは、オフセットを 16 で割って計算します。その余りが相対オフセットになります。この場合はオフセットが 22 なので、16 で割った余りの 6 が相対オフセットになります。) ポインターの整合性は維持されます。

グループ項目中にポインターが含まれていて、ILE COBOL コンパイラーが 16 バイト境界に対するオフセットを判別できない場合は、ILE COBOL コンパイラーは警告メッセージを出し、ポインターの移動が試行されます。しかし、ポインターの整合性は維持されないことがあります。 LINKAGE SECTION の中で項目が定義されている場合、または認識されていない開始位置で項目が参照変更されている場合は、ILE COBOL コンパイラーはオフセットを判別できません。ポインターの位置合わせは必ず維持しなければなりません。そうしないと、マシン・チェック・エラーが起こることがあります。

ILE COBOL コンパイラーは、01 レベルの項目と 77 レベルの項目のすべてを、ポインターの有無に関係なく 16 バイト境界にします。

CALL ステートメントにおけるポインターの使用

CALL ステートメントの中でポインター・データ項目またはプロシージャー・ポインター・データ項目が渡されると、その項目は他のすべての USING 項目と同じように扱われます。すなわち、ポインター・データ項目が BY REFERENCE (または BY CONTENT) によって渡されると、そのポインター・データ項目を指すポインター (またはポインター・データ項目のコピー) が、呼び出し先プログラムに渡されます。ポインター・データ項目が BY VALUE によって渡されると、ポインター・データ項目の内容は呼び出しスタックに入れられます。プロシージャー・ポインター・データ項目も同じ方法で渡されます。

BY CONTENT 句の指定された CALL ステートメントを使用してポインターとポインターを含むグループ項目を渡す場合は、特別な考慮が必要です。これは、MOVE ステートメントの場合も同じです。 CALL...BY CONTENT の場合は、項目の暗黙の MOVE が実行されてその項目が一時域に作成されます。このポインター MOVE に関するポインター位置合わせを行うため、ILE COBOL コンパイラーまたはランタイムは、16 バイト境界に関する BY CONTENT 項目のオフセットを判別しなければなりません。最高のパフォーマンスを得るためには、ILE COBOL コンパイラーがこのオフセットを判別するのと同じ方法で BY CONTENT 項目をコーディングする必要があります。

BY CONTENT 項目が次のようになっている場合は、ILE COBOL ランタイムが 16 バイト境界に関する項目のオフセットを判別する必要があります。
- 認識されていない開始位置で参照変更されている場合、または
- LINKAGE SECTION 中に定義されている場合。

オペランドが参照変更されている場合、そのオフセットは、参照変更の開始位置から 1 を引いて、データ構造中のオペランドのオフセットを足したものになります。オペランドが LINKAGE SECTION 中にある場合、そのオフセットは呼び出し側プログラムにより判別されます。

ポインターの位置合わせに関する問題を避けるため、BY REFERENCE を使用して項目を渡してください。

ポインターの値の調整

次の例は、整数値を UP BY で増やしたり、DOWN BY で減らすことによってポインターの値を調整する方法を示しています。このような方法でポインターの値を変更することは、ポインター・データ項目で参照されるテーブル中の項目にアクセスする場合に便利です。

```
WORKING-STORAGE SECTION.
 01 A.
    05 ARRAY-USER-INFO OCCURS 300 TIMES.
       10  USER-NAME PIC X(10).
       10  USER-ID   PIC 9(7).
    01 ARRAY-PTR USAGE IS POINTER.
LINKAGE SECTION.
01  USER-INFO.
    05 USER-NAME LIKE USER-NAME OF ARRAY-USER-INFO.
    05 USER-ID LIKE USER-ID OF ARRAY-USER-INFO.
PROCEDURE DIVISION.
      SET ARRAY-PTR TO ADDRESS OF ARRAY-USER-INFO(200). [1]
      SET ADDRESS OF USER-INFO TO ARRAY-PTR. [2]
      SET ARRAY-PTR UP BY LENGTH OF USER-INFO. [3]
      SET ADDRESS OF USER-INFO TO ARRAY-PTR. [4]
      MOVE "NEW NAME" TO USER-NAME OF USER-INFO.[5]
```

注:

1. 最初の SET ステートメントは、ARRAY-USER-INFO 配列の 200 番目のエレメントのアドレスをポインター ARRAY-PTR の中に入れます。
2. 2 番目の SET ステートメントによって、データ項目 USER-INFO に ARRAY-USER-INFO 配列の 200 番目のエレメントと同じアドレスが与えられます。
3. 3 番目の SET ステートメントは、配列の 1 つのエレメントの長さずつポインター ARRAY-PTR 内に含まれるアドレスを増やします。
4. 4 番目の SET ステートメントによって、データ項目 USER-INFO に ARRAY-USER-INFO 配列の 201 番目のエレメントと同じアドレスが与えられます。
5. この移動は次の移動と同じです。

   ```
   MOVE "NEW NAME" to USER-NAME OF ARRAY-USER-INFO (201).
   ```

SET ステートメントの詳しい定義については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

ポインターと API を使用してユーザー・スペースにアクセスする

次の例は、ポインターを使用してユーザー・スペースにアクセスし、レコードのチェーンを作成する方法を示します。

POINTA は、ユーザー・スペース中の得意先の名前 (CUSTOMER NAME) と所在地 (CUSTOMER ADDRESS) を読み取ってから、情報をリスト表示するプログラムです。このプログラムでは、POINTACU というファイルに得意先情報が入っているものとしています。

得意先アドレス・フィールドは可変長フィールドであり、長いアドレスが可能です。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* THIS IS THE CUSTOMER INFORMATION FILE  - POINTACUST
     A
     A
     A          R FSCUST                    TEXT('CUSTOMER MASTER RECORD')
     A            FS_CUST_NO     8S00       TEXT('CUSTOMER NUMBER')
     A                                       ALIAS(FS_CUST_NUMBER)
     A            FS_CUST_NM     20         TEXT('CUSTOMER NAME')
     A                                       ALIAS(FS_CUST_NAME)
     A            FS_CUST_AD     100        TEXT('CUSTOMER ADDRESS')
     A                                       ALIAS(FS_CUST_ADDRESS)
     A                                       VARLEN
```

図 82. ポインターを使用してユーザー・スペースにアクセスする例 - DDS

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL              CBLGUIDE/POINTA          ISERIES1   06/02/15 13:43:25       ページ    2
                           ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
         000100 PROCESS varchar  1
    1    000200 ID DIVISION.
         000300* このプログラムは、可変長レコードをユーザー・スペース
         000400*  に読み取ります。次に、ディスプレイ装置にレコードを
         000500*  表示します。
    2    000600 PROGRAM-ID. pointa.
    3    000700 ENVIRONMENT DIVISION.
    4    000800 CONFIGURATION SECTION.
    5    000900 SPECIAL-NAMES. CONSOLE IS CRT,
    7    001000                CRT STATUS IS ws-crt-status.  2
    8    001100 INPUT-OUTPUT SECTION.
    9    001200 FILE-CONTROL.
   10    001300     SELECT cust-file ASSIGN TO DATABASE-pointacu
   12    001400            ORGANIZATION IS SEQUENTIAL
   13    001500            FILE STATUS IS ws-file-status.
   14    001600 DATA DIVISION.
   15    001700 FILE SECTION.
   16    001800 FD  cust-file.
   17    001900 01  fs-cust-record.
         002000* 下線をダッシュに変更し、別名を使用して、
         002100*  フィールド名にコピーします。
         002200 COPY DDR-ALL-FORMATS-I OF pointacu.
   18    +000001       05  POINTACU-RECORD PIC X(130).                                <-ALL-FMTS
         +000002*    I-O FORMAT:FSCUST     FROM FILE POINTACU   OF LIBRARY CBLGUIDE  <-ALL-FMTS
         +000003*                          CUSTOMER MASTER RECORD                    <-ALL-FMTS
   19    +000004       05  FSCUST        REDEFINES POINTACU-RECORD.                  <-ALL-FMTS
   20    +000005           06 FS-CUST-NUMBER        PIC S9(8).                       <-ALL-FMTS
         +000006*                 CUSTOMER NUMBER                                   <-ALL-FMTS
   21    +000007           06 FS-CUST-NAME          PIC X(20).                       <-ALL-FMTS
         +000008*                 CUSTOMER NAME                                     <-ALL-FMTS
   22    +000009           06 FS-CUST-ADDRESS.  3                                    <-ALL-FMTS
         +000010*                 (可変長フィールド)                                <-ALL-FMTS
   23    +000011               49 FS-CUST-ADDRESS-LENGTH                            <-ALL-FMTS
         +000012                                  PIC S9(4) COMP-4.                 <-ALL-FMTS
   24    +000013               49 FS-CUST-ADDRESS-DATA                              <-ALL-FMTS
         +000014                                  PIC X(100).                       <-ALL-FMTS
         +000015*                 CUSTOMER ADDRESS                                  <-ALL-FMTS
   25    002300 WORKING-STORAGE SECTION.
   26    002400 01  ws-file-status.
   27    002500     05 ws-file-status-1 PIC X.
   28    002600        88  ws-file-stat-good   VALUE "0".
   29    002700        88  ws-file-stat-at-end VALUE "1".
   30    002800     05 ws-file-status-2 PIC X.
   31    002900 01  ws-crt-status.  4
   32    003000     05 ws-status-1           PIC 9(2).
   33    003100        88 ws-status-1-ok       VALUE 0.
   34    003200        88 ws-status-1-func-key VALUE 1.
   35    003300        88 ws-status-1-error    VALUE 9.
   36    003400     05 ws-status-2           PIC 9(2).
   37    003500        88 ws-func-03           VALUE 3.
   38    003600        88 ws-func-07           VALUE 7.
   39    003700        88 ws-func-08           VALUE 8.
   40    003800     05 ws-status-3           PIC 9(2).
```

図 83. ポインターを使用してユーザー・スペースにアクセスする例 (1/8)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/POINTA         ISERIES1   06/02/15 13:43:25        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   41     003900 01  ws-params.  5
   42     004000     05  ws-space-ptr         POINTER.  6
   43     004100     05  ws-space.
   44     004200       10  ws-space-name      PIC X(10) VALUE "MYSPACE".
   45     004300       10  ws-space-lib       PIC X(10) VALUE "QTEMP".
   46     004400     05  ws-attr              PIC X(10) VALUE "PF".
   47     004500     05  ws-init-size         PIC S9(5) VALUE 32000 BINARY.
   48     004600     05  ws-init-char         PIC X     VALUE SPACE.
   49     004700     05  ws-auth              PIC X(10) VALUE "*ALL".
   50     004800     05  ws-text              PIC X(50) VALUE
          004900                               "Customer Information Records".
   51     005000     05  ws-replace           PIC X(10) VALUE "*YES".
   52     005100     05  ws-err-data.  7
   53     005200         10 ws-input-l        PIC S9(6) BINARY VALUE 16.
   54     005300         10 ws-output-l       PIC S9(6) BINARY.
   55     005400         10 ws-exception-id   PIC X(7).
   56     005500         10 ws-reserved       PIC X(1).
          005600
   57     005700 77  ws-accept-data           PIC X   VALUE SPACE.
   58     005800   88 ws-acc-blank             VALUE SPACE.
   59     005900   88 ws-acc-create-space      VALUE "Y", "y".
   60     006000   88 ws-acc-use-prv-space     VALUE "N", "n".
   61     006100   88 ws-acc-delete-space      VALUE "Y", "y".
   62     006200   88 ws-acc-save-space        VALUE "N", "n".
          006300
   63     006400 77  ws-prog-indicator        PIC X   VALUE "G".
   64     006500   88  ws-prog-continue       VALUE "G".
   65     006600   88  ws-prog-end            VALUE "C".
   66     006700   88  ws-prog-loop           VALUE "L".
          006800
   67     006900 77  ws-line                  PIC 99.
          007000* エラー・メッセージ行
   68     007100 77  ws-error-msg             PIC X(50) VALUE SPACES.
          007200* それ以上のアドレス情報標識
   69     007300 77  ws-plus                  PIC X.
          007400* 表示するアドレス情報の長さ
   70     007500 77  ws-temp-size             PIC 9(2).
          007600
   71     007700 77  ws-current-rec           PIC S9(4) VALUE 1.
   72     007800 77  ws-old-rec               PIC S9(4) VALUE 1.
   73     007900 77  ws-old-space-ptr         POINTER.
          008000* 表示する最大行数
   74     008100 77  ws-displayed-lines       PIC S99 VALUE 20.
          008200* レコードの表示を開始する行
   75     008300 77  ws-start-line            PIC S99 VALUE 5.
          008400* スペースに新しいレコードを作成するための変数
   76     008500 77  ws-addr-inc              PIC S9(4) PACKED-DECIMAL.
   77     008600 77  ws-temp                  PIC S9(4) PACKED-DECIMAL.
   78     008700 77  ws-temp-2                PIC S9(4) PACKED-DECIMAL.
          008800* 直前のレコードへのポインター
   79     008900 77  ws-cust-prev-ptr         POINTER VALUE NULL.
   80     009000 LINKAGE SECTION.
   81     009100 01 ls-header-record.  8
   82     009200    05  ls-hdr-cust-ptr            USAGE POINTER.
          009300* FROM ファイルから読み取るレコードの数
```

図 83. ポインターを使用してユーザー・スペースにアクセスする例 (2/8)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/POINTA          ISERIES1   06/02/15 13:43:25        ページ     4
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   83     009400    05  ls-record-counter           PIC S9(3) BINARY.
   84     009500    05  FILLER                      PIC X(14).   9
   85     009600 01 ls-user-space.  10
   86     009700     05  ls-customer-rec.
          009800* 直前の得意先レコードへのポインター
   87     009900         10 ls-cust-prev-ptr        USAGE POINTER.
   88     010000         10 ls-cust-rec-length      PIC S9(4) BINARY.
   89     010100         10 ls-cust-name            PIC X(20).
   90     010200         10 ls-cust-number          PIC S9(8).
          010300* 次レコードが 16 バイト境界になることを確実にするための、
          010400*  充てん文字を含めた、このレコードの全長
   91     010500         10 ls-cust-address-length  PIC S9(4) BINARY.
   92     010600     05  ls-cust-address-data       PIC X(116).
          010700
          010800* ls-user-space のサイズは実際に必要であるより 16 だけ大きい。
          010900* これにより、次レコードの開始アドレスが、
          011000* 宣言されたサイズを超えずに確立されるようになる。
          011100* サイズは、ポインター調整を可能にするため 16 だけ大きい。
          011200
   93     011300 PROCEDURE DIVISION.
          011400* PROC... DIV に "USING" 項目が必要でないことに注意。
   94     011500 DECLARATIVES.
          011600 cust-file-para SECTION.
          011700     USE AFTER ERROR PROCEDURE ON cust-file.
          011800 cust-file-para-2.
   95     011900     MOVE "Error XX on file pointacu" TO ws-error-msg.
   96     012000     MOVE ws-file-status TO ws-error-msg(7:2).
          012100 END DECLARATIVES.
          012200
          012300 main-program section.
          012400 mainline.
          012500* 入力データが訂正されるまで、初期画面の読み取りを続ける。
   97     012600     SET ws-prog-loop TO TRUE.
   98     012700     PERFORM initial-display THRU read-initial-display
          012800         UNTIL NOT ws-prog-loop.
          012900* プログラムを続行し、得意先情報域を作成したい
          013000* 場合は、スペースに、得意先ファイルからのレコード
          013100* を充てんする。
   99     013200     IF ws-prog-continue AND
          013300        ws-acc-create-space THEN
  100     013400       PERFORM read-customer-file
  101     013500       MOVE 1 TO ws-current-rec
          013600* ポインターをヘッダー・レコードに設定
  102     013700       SET ADDRESS OF ls-header-record TO ws-space-ptr
          013800* 最初の得意先レコードをスペースに設定
  103     013900       SET ADDRESS OF ls-user-space TO ls-hdr-cust-ptr
          014000     END-IF.
  104     014100     IF ws-prog-continue THEN
  105     014200       PERFORM main-loop UNTIL ws-prog-end
          014300     END-IF.
          014400 end-program.
  106     014500     PERFORM clean-up.
  107     014600     STOP RUN.
          014700
          014800 initial-display.  11
```

図 83. ポインターを使用してユーザー・スペースにアクセスする例 (3/8)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/POINTA          ISERIES1   06/02/15 13:43:25        ページ     5
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
  108     014900     DISPLAY "Create Customer Information Area" AT 0118 WITH
          015000                BLANK SCREEN REVERSE-VIDEO
          015100             "Create customer information area (Y/N)=>   <="
          015200                AT 1015
          015300             "F3=Exit" AT 2202.
  109     015400     IF ws-error-msg NOT = SPACES THEN
  110     015500       DISPLAY ws-error-msg at 2302 with beep highlight
  111     015600       MOVE SPACES TO ws-error-msg
          015700     END-IF.
          015800
          015800
  112     016000     ACCEPT ws-accept-data AT 1056 WITH REVERSE-VIDEO
          016100       ON EXCEPTION
  113     016200         IF ws-status-1-func-key THEN
  114     016300           IF ws-func-03 THEN
  115     016400             SET ws-prog-end TO TRUE
          016500           ELSE
  116     016600             MOVE "Invalid Function Key" TO ws-error-msg
          016700           END-IF
          016800         ELSE
  117     016900           MOVE "Unknown Error" TO ws-error-msg
          017000         END-IF
          017100       NOT ON EXCEPTION
  118     017200         IF ws-acc-create-space THEN
  119     017300           PERFORM create-space THRU set-space-ptrs
  120     017400           SET ws-prog-continue TO TRUE
          017500         ELSE
  121     017600           IF ws-acc-use-prv-space THEN
  122     017700             PERFORM get-space
  123     017800             IF ws-space-ptr = NULL
  124     017900               MOVE "No Customer Information Area" TO ws-error-msg
          018000             ELSE
  125     018100               PERFORM set-space-ptrs
  126     018200               SET ws-prog-continue TO TRUE
          018300             END-IF
          018400           ELSE
  127     018500             MOVE "Invalid Character Entered" TO ws-error-msg
          018600           END-IF
          018700         END-IF
          018800     END-ACCEPT.
          018900
          019000 create-space.
  128     019100     CALL "QUSCRTUS" USING ws-space, ws-attr, ws-init-size,   13
          019200                           ws-init-char, ws-auth, ws-text,
          019300                           ws-replace, ws-err-data.
          019400
          019500* スペースを作成する際のエラーの検査をここで追加できる。
          019600
          019700 get-space.
  129     019800     CALL "QUSPTRUS" USING ws-space, ws-space-ptr, ws-err-data.   14
          019900
          020000 set-space-ptrs.
          020100* スペースの先頭にヘッダー・レコードを設定
  130     020200     SET ADDRESS OF ls-header-record   15
          020300         ADDRESS OF ls-user-space   16
```

図 83. ポインターを使用してユーザー・スペースにアクセスする例 (4/8)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/POINTA          ISERIES1   06/02/15 13:43:25        ページ    6
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
         020400         TO ws-space-ptr.
         020500* ヘッダー・レコードの後の最初の得意先レコードを設定
  131    020600     SET ADDRESS OF ls-user-space TO   17
         020700         ADDRESS OF ls-user-space(LENGTH OF ls-header-record   18
         020800                                  + 1:1).
         020900* 最初のレコードへのポインターをヘッダー・レコードに保管
  132    021000     SET ls-hdr-cust-ptr TO ADDRESS OF ls-user-space.
         021100
         021200 delete-space.
  133    021300     CALL "QUSDLTUS" USING ws-space, ws-err-data.   19
         021400
         021500 read-customer-file.
         021600* 得意先ファイルから全レコードを読み取り、スペースに移動
  134    021700     OPEN INPUT cust-file.
  135    021800     IF ws-file-stat-good THEN
  136    021900       READ cust-file AT END CONTINUE
         022000       END-READ
  138    022100       PERFORM VARYING ls-record-counter FROM 1 BY 1
         022200            UNTIL not ws-file-stat-good
  139    022300         SET  ls-cust-prev-ptr  TO ws-cust-prev-ptr
         022400*  ファイルからの情報をスペースに移動
  140    022500         MOVE fs-cust-name      TO ls-cust-name
  141    022600         MOVE fs-cust-number    TO ls-cust-number
  142    022700         MOVE fs-cust-address-length  TO ls-cust-address-length
  143    022800         MOVE fs-cust-address-data(1:fs-cust-address-length)
         022900              TO ls-cust-address-data(1:ls-cust-address-length)
         023000*  現行レコードへのポインターを保管
  144    023100         SET  ws-cust-prev-ptr  TO ADDRESS OF ls-user-space
         023200*  次レコードが 16 バイト境界になるよう確認
  145    023300         ADD LENGTH OF ls-customer-rec   20
         023400             ls-cust-address-length TO 1 GIVING ws-addr-inc
  146    023500         DIVIDE ws-addr-inc BY 16 GIVING ws-temp
         023600             REMAINDER ws-temp-2
  147    023700         SUBTRACT ws-temp-2 FROM 16 GIVING ws-temp
         023800*  合計レコード長をユーザー・スペースに保管
  148    023900         ADD ws-addr-inc TO ws-temp GIVING ls-cust-rec-length
  149    024000         SET ADDRESS OF ls-user-space
         024100          TO ADDRESS OF ls-user-space(ls-cust-rec-length + 1:1)
         024200*  ファイルから次レコードを読み取る           024100          TO ADDRESS OF ls-user-space(ls-cust-rec-length + 1:1)
  150    024300         READ cust-file AT END CONTINUE
         024400         END-READ
         024500       END-PERFORM
         024600*  ループの終わりでは、実際よりも 1 つだけレコード
         024700*  が多い
  152    024800       SUBTRACT 1 FROM ls-record-counter
         024900     END-IF.
  153    025000     CLOSE cust-file.
         025100
         025200 main-loop.   21
         025300* F3 が入力されるまで、ディスプレイにレコードを表示する   153     025000     CLOSE cust-file.
  154    025400     DISPLAY "Customer Information" AT 0124 WITH
         025500                BLANK SCREEN REVERSE-VIDEO
         025600             "Cust     Customer Name        Customer"
         025700                AT 0305
         025800             " Address"
```

図 83. ポインターを使用してユーザー・スペースにアクセスする例 (5/8)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/POINTA          ISERIES1   06/02/15 13:43:25        ページ     7
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
         025900             "Number"   AT 0405
         026000             "F3=Exit" AT 2202.
         026100* 保留中のエラーがあれば、ディスプレイに表示する
  155    026200     IF ws-error-msg NOT = SPACES THEN
  156    026300       DISPLAY ws-error-msg at 2302 with beep highlight
  157    026400       MOVE SPACES TO ws-error-msg
         026500     END-IF.
         026600* リストの中央であれば、F7 をディスプレイに表示する
  158    026700     IF ws-current-rec > 1 THEN   22
  159    026800       DISPLAY "F7=Back" AT 2240
         026900     END-IF.
         027000* 現行レコードを保管
  160    027100     MOVE ws-current-rec TO ws-old-rec.
  161    027200     SET  ws-old-space-ptr TO ADDRESS OF ls-user-space.   23
         027300* 各レコードをディスプレイに移動
  162    027400     PERFORM VARYING ws-line FROM  ws-start-line BY 1
         027500         UNTIL ws-line > ws-displayed-lines or
         027600               ws-current-rec > ls-record-counter
         027700* アドレスが表示幅より大きければ、"+" を表示する
  163    027800         IF ls-cust-address-length > 40 THEN
  164    027900           MOVE "+" TO ws-plus
  165    028000           MOVE 40 TO ws-temp-size
         028100         ELSE
  166    028200           MOVE ls-cust-address-length TO ws-temp-size
  167    028300           MOVE SPACE TO ws-plus
         028400         END-IF
  168    028500         DISPLAY ls-cust-number at line ws-line column 5
         028600                 ls-cust-name ls-cust-address-data with
         028700                   size ws-temp-size ws-plus at line
         028800                   ws-line column 78
         028900* 次レコードをスペースに読み取る
  169    029000         ADD 1 TO ws-current-rec
  170    029100         SET ADDRESS OF ls-user-space
         029200          TO ADDRESS OF ls-user-space
         029300            (ls-cust-rec-length + 1:1)
         029400     END-PERFORM.
         029500* 順方向に進むことができれば、F8 をディスプレイに表示する
  171    029600     IF ws-current-rec < ls-record-counter THEN   22
  172    029700       DISPLAY "F8=Forward" AT 2250
         029800     END-IF.
         029900* 続行か、終了か、または次レコードか前レコード
         030000* を読み取るかを調べる。
  173    030100     SET ws-acc-blank to TRUE.
  174    030200     ACCEPT ws-accept-data WITH SECURE   24
         030300       ON EXCEPTION
  175    030400         IF ws-status-1-func-key THEN
  176    030500           IF ws-func-03 THEN
  177    030600             SET ws-prog-end TO TRUE
         030700           ELSE
  178    030800            IF ws-func-07 THEN
  179    030900             PERFORM back-screen
         031000            ELSE
  180    031100            IF ws-func-08 THEN
  181    031200             PERFORM forward-screen
         031300            ELSE
```

図 83. ポインターを使用してユーザー・スペースにアクセスする例 (6/8)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/POINTA          ISERIES1   06/02/15 13:43:25        ページ    8
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
  182     031400             MOVE "Invalid Function Key" TO ws-error-msg
  183     031500             MOVE ws-old-rec TO ws-current-rec
  184     031600             SET ADDRESS OF ls-user-space TO ws-old-space-ptr
          031700           END-IF
          031800         END-IF
          031900       ELSE
  185     032000         MOVE "Unknown Error" TO ws-error-msg
  186     032100         MOVE ws-old-rec TO ws-current-rec
  187     032200         SET ADDRESS OF ls-user-space TO ws-old-space-ptr
          032300       END-IF
          032400     NOT ON EXCEPTION
  188     032500       MOVE ws-old-rec TO ws-current-rec
  189     032600       SET ADDRESS OF ls-user-space TO ws-old-space-ptr
          032700   END-ACCEPT.
          032800
          032900 clean-up.
          033000* プログラムに対する終結処理を実行。
          033100* 入力データが訂正されるまで、初期画面の読み取りを続ける。
  190     033200   SET ws-prog-loop to TRUE.
  191     033300   SET ws-acc-blank to TRUE.
  192     033400   PERFORM final-display THRU read-final-display   25
          033500       UNTIL NOT ws-prog-loop.
          033600
          033700 final-display.
  193     033800   DISPLAY "Delete Customer Information Area" AT 0118 WITH   26
          033900              BLANK SCREEN REVERSE-VIDEO
          034000            "Delete customer information area (Y/N)=>   <="
          034100              AT 1015
          034200            "F3=Exit" AT 2202.
  194     034300   IF ws-error-msg NOT = SPACES THEN
  195     034400     DISPLAY ws-error-msg at 2302 with beep highlight
  196     034500     MOVE SPACES TO ws-error-msg
          034600   END-IF.
          034700
          034800 read-final-display.
  197     034900   ACCEPT ws-accept-data AT 1056 WITH REVERSE-VIDEO
          035000     ON EXCEPTION
  198     035100       IF ws-status-1-func-key THEN
  199     035200         IF ws-func-03 THEN
  200     035300           SET ws-prog-end TO TRUE
          035400         ELSE
  201     035500           MOVE "Invalid Function Key" TO ws-error-msg
          035600         END-IF
          035700       ELSE
  202     035800         MOVE "Unknown Error" TO ws-error-msg
          035900       END-IF
          036000     NOT ON EXCEPTION
  203     036100       IF ws-acc-delete-space THEN
  204     036200         PERFORM delete-space
  205     036300         SET ws-prog-continue TO TRUE
          036400       ELSE
  206     036500         IF ws-acc-save-space THEN
  207     036600           SET ws-prog-continue TO TRUE
          036700         ELSE
  208     036800           MOVE "Invalid Character Entered" TO ws-error-msg
```

図 83. ポインターを使用してユーザー・スペースにアクセスする例 (7/8)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/POINTA         ISERIES1   06/02/15 13:43:25        ページ     9
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
          036900           END-IF
          037000         END-IF
          037100     END-ACCEPT.
          037200
          037300 back-screen.  27
   209    037400     IF ws-old-rec <= 1 THEN
   210    037500       MOVE "Top of customer records" TO ws-error-msg
   211    037600       MOVE ws-old-rec TO ws-current-rec  28
   212    037700       SET ADDRESS OF ls-user-space TO ws-old-space-ptr
          037800     ELSE
   213    037900       MOVE ws-old-rec TO ws-current-rec  28
   214    038000       SET ADDRESS OF ls-user-space TO ws-old-space-ptr
   215    038100       PERFORM VARYING ws-line FROM ws-start-line BY 1
          038200         UNTIL ws-line > ws-displayed-lines or
          038300               ws-current-rec <= 1
          038400* 一度に 1 レコードずつ逆方向に移動
   216    038500         SET ws-cust-prev-ptr TO ls-cust-prev-ptr  29
   217    038600         SET ADDRESS OF ls-user-space TO ws-cust-prev-ptr
   218    038700         SUBTRACT 1 FROM ws-current-rec
          038800       END-PERFORM
          038900     END-IF.
          039000
          039100 forward-screen.  30
          039200* 現行レコードが最大レコード数以上であれば、
          039300* 最大レコード数に達した旨のエラーを印刷する。
   219    039400     IF ws-current-rec >= ls-record-counter
   220    039500       MOVE "No more customer records" TO ws-error-msg
   221    039600       MOVE ws-old-rec TO ws-current-rec
   222    039700       SET ADDRESS OF ls-user-space TO ws-old-space-ptr
          039800     ELSE
   223    039900       MOVE ws-current-rec TO ws-old-rec
   224    040000       SET ws-old-space-ptr TO ADDRESS OF ls-user-space
          040100     END-IF.
          040200
                          * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 83. ポインターを使用してユーザー・スペースにアクセスする例 (8/8)

2 CRT STATUS IS は、拡張 ACCEPT ステートメントの終了後に状況値を入れる場所のデータ名を指定します。この例では、STATUS キー値を使用して、押されたファンクション・キーを判別します。

3 *fs-cust-address* は可変長フィールドです。ここで FILLER 以外の意味がある名前を参照するには、 **1** のように、CRTCBLMOD コマンドまたは CRTBNDCBL コマンドの CVTOPT パラメーターに *VARCHAR を指定するか、または PROCESS ステートメントに VARCHAR を指定してください。可変長フィールドの詳細については 490 ページの『SAA データ・タイプを使用したデータ項目の宣言』を参照してください。

4 **2** で言及されている CRT STATUS は、ここで定義されます。

5 *ws-params* 構造には、API を呼び出してユーザー・スペースにアクセスする際に使用するパラメーターが入れられます。

6 *ws-space-ptr* は、API QUSPTRUS で設定されるポインター・データ項目を定義します。これはユーザー・スペースの先頭を指し、LINKAGE SECTION の中で項目のアドレスを設定するために使用します。

7 *ws-err-data* は、ユーザー・スペース API のエラー・パラメーターの構造です。 *ws-input-l* はゼロですが、これによってすべての例外はプログラムに通知され、エラー・コード・パラメーターでは渡されないことに注意してください。 エラー・コード・パラメーターの詳細については、 Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「プログラミング」カテゴリーの中の『*CL* および *API*』セクションを参照してください。

8 ユーザー・スペース中に定義される最初のデータ構造 (*ls-header-record*)。

9 FILLER は、*Is-header-record* の長さは 16 バイトの倍数にすることによってポインター位置合わせを維持するのに使用されています。

10 ユーザー・スペース中に定義される 2 番目のデータ構造 (*ls-user-space*)。

11 *initial-display* によって、「Create Customer Information Area」画面が表示されます。

12 *read-initial-display* は最初の表示画面を読み取って、ユーザーが継続を選択したかそれともプログラム終了を選択したかを判別します。実行キーを押してプログラムを継続すると、プログラムは *ws-accept-data* を調べて、得意先情報域 (Customer Information Area) を作成するのかどうかを調べます。

13 QUSCRTUS はユーザー・スペースを作成するために使用する API です。

14 QUSPTRUS は、ユーザー・スペースの先頭へのポインターを戻すために使用する API です。

15 最初のデータ構造 (*ls-header-record*) をユーザー・スペースの先頭にマッピングします。

16 2 番目のデータ構造 (*ls-user-space*) をユーザー・スペースの先頭にマッピングします。

17 ADDRESS OF 特殊レジスターを使用します。

18 ADDRESS OF が参照変更されているので、ADDRESS OF 特殊レジスターではなく ADDRESS OF を使用します。

19 QUSDLTUS はユーザー・スペースを削除するために使用する API です。

20 この後にある 4 つの算術ステートメントは、各レコードの全長を計算し、各レコードの長さが必ず 16 バイトの倍数になるようにします。

21 *main-loop* は、「Customer Information」画面を表示します。

22 これらのステートメントは、プログラムがファンクション・キー F7 および F8 を表示する必要があるかどうかを判別するためのものです。

23 画面上の最初の得意先レコードへのポインターを保管します。

24 この ACCEPT ステートメントは、「Customer Information」画面からの入力を待ちます。押されるファンクション・キーに基づいて、該当する段落を呼び出して次のレコード集合 (*forward-screen*) または直前のレコード集合 (*back-screen*) を表示するか、または F3 が押された場合はルーチンを終了するための標識を設定します。

25 終結ルーチンは、該当するキーが押されるまで、「Delete Customer Information Area」画面を表示します。

26 このステートメントは、「Delete Customer Information Area」画面を表示します。

27 各レコードには、直前の得意先レコードを指すポインターが入れられます。ADDRESS OF 特殊レジスターは、現行の得意先レコードを指します。ADDRESS OF 特殊レジスターを変更すると、現行の得意先レコードが変更されます。

back-screen は、現行のレコード・ポインターを一度に 1 レコードずつ逆方向に移動します。 **29** これは、現行の得意先レコード (ADDRESS OF) を指すポインターに直前の得意先レコードへのポインターを移動することによって行います。一度に 1 レコードずつ逆方向に移動する前に、まずプログラムは現行の得意先レコードを現在表示されている最初のレコードに設定します。 **28**

30 *forward-screen* は *ws-old-space-ptr* (表示画面上の最初のレコードを指している) を、現行レコード (表示されている最後のレコードの後) を指すように設定します。

ユーザー・スペースは常に 16 バイト境界で始まるので、ここに示されている方法では、**すべての**レコードが必ず位置合わせされます。また、レコードのチェーンの作成に *ls-cust-rec-length* も使用されます。

POINTA を実行すると、次の画面が表示されます。

```
CMDSTR                          開始コマンド

次の 1 つを選んでください。

   コマンド
      1. NSLOOKUP 照会の開始                                  NSLOOKUP
      2. QSH の開始                                           QSH
      3. RPC BIND プログラム・デーモン の開始                 RPCBIND
      4. ACCESS FOR WEB の開始                                STRACCWEB2
      5. AFP ユーティリティーの開始                           STRAFPU
      6. エージェント・サービスの開始                        STRAGTSRV
      7. 拡張印刷機能開始                                    STRAPF
      8. BRM を使用したアーカイブ開始                         STRARCBRM
      9. アーカイブ・ストレージ管理の開始                     STRASMOND
     10. ASP バランス化開始                                   STRASPBAL
     11. ASP セッションの開始                                 STRASPSSN
     12. BRM の媒体バランシングの開始                         STRBALBRM
                                                                 続く ...
選択またはコマンド
===> CALL POINTA

F3= 終了    F4= プロンプト    F9=コマンドの複写    F12= 取り消し
F16= メジャー・メニュー
(C) COPYRIGHT IBM CORP. 1980, 2009.
出力ファイル POINTSCREE がライブラリー HORNER に作成された。
```

```
                  Create Customer Information Area


              Create customer information area (Y/N)=> y <=









F3=Exit
```

```
                    Customer Information
  Cust     Customer Name        Customer  Address
  Number
  00000001 Bakery Unlimited     30 Bake Way, North York
  00000002 Window World         150 Eglinton Ave E., North York, Ontario
  00000003 Jons Clothes         101 Park St, North Bay, Ontario, Canada
  00000004 Pizza World          254 Main Street, Toronto, Ontario         +
  00000005 Marv's Auto Body     9 George St, Peterborough, Ontario, Cana  +
  00000006 Jack's Snacks        23 North St, Timmins, Ontario, Canada
  00000007 Video World          14 Robson St, Vancouver, B.C, Canada
  00000008 Pat's Daycare        8 Kingston Rd, Pickering, Ontario, Canad  +
  00000009 Mary's Pies          3 Front St, Toronto, Ontario, Canada
  00000010 Carol's Fashions     19 Spark St, Ottawa, Ontario, Canada
  00000011 Grey Optical         5 Lundy's Lane, Niagara Falls, Ont. Cana  +
  00000012 Fred's Forage        33 Dufferin St, Toronto, Ontario, Canada  +
  00000013 Dave's Trucking      15 Water St, Guelph, Ontario, Canada
  00000014 Doug's Music         101 Queen St. Toronto, Ontario, Canada    +
  00000015 Anytime Copiers      300 Warden Ave, Scarborough, Ontario, Ca  +
  00000016 Rosa's Ribs          440 Avenue Rd, Toronto, Ontario, Canada
F3=Exit                                        F8=Forward
```

```
                    Customer Information
  Cust     Customer Name        Customer  Address
  Number
  00000017 Picture It           33 Kingston Rd, Ajax, Ontario, Canada
  00000018 Paula's Flowers      144 Pape Ave, Toronto, Ontario, Canada
  00000019 Mom's Diapers        101 Ford St, Toronto, Ontario, Canada
  00000020 Chez Francois        1202 Rue Ste Anne, Montreal, PQ, Canada
  00000021 Vetements de Louise  892 Rue Sherbrooke, Montreal E, PQ, Cana  +
  00000022 Good Eats            355 Lake St, Port Hope, Ontario, Canada
F3=Exit                                 F7=Back
```

```
                    Customer Information
  Cust     Customer Name        Customer  Address
  Number
  00000001 Bakery Unlimited     30 Bake Way, North York
  00000002 Window World         150 Eglinton Ave E., North York, Ontario
  00000003 Jons Clothes         101 Park St, North Bay, Ontario, Canada
  00000004 Pizza World          254 Main Street, Toronto, Ontario         +
  00000005 Marv's Auto Body     9 George St, Peterborough, Ontario, Cana  +
  00000006 Jack's Snacks        23 North St, Timmins, Ontario, Canada
  00000007 Video World          14 Robson St, Vancouver, B.C, Canada
  00000008 Pat's Daycare        8 Kingston Rd, Pickering, Ontario, Canad  +
  00000009 Mary's Pies          3 Front St, Toronto, Ontario, Canada
  00000010 Carol's Fashions     19 Spark St, Ottawa, Ontario, Canada
  00000011 Grey Optical         5 Lundy's Lane, Niagara Falls, Ont. Cana  +
  00000012 Fred's Forage        33 Dufferin St, Toronto, Ontario, Canada  +
  00000013 Dave's Trucking      15 Water St, Guelph, Ontario, Canada
  00000014 Doug's Music         101 Queen St. Toronto, Ontario, Canada    +
  00000015 Anytime Copiers      300 Warden Ave, Scarborough, Ontario, Ca  +
  00000016 Rosa's Ribs          440 Avenue Rd, Toronto, Ontario, Canada
F3=Exit                                        F8=Forward
```

```
                    Delete Customer Information Area


                Delete customer information area (Y/N)=> y <=









F3=Exit
```

```
CMDSTR                          開始コマンド

次の 1 つを選んでください。

  コマンド
     1. NSLOOKUP 照会の開始                                  NSLOOKUP
     2. QSH の開始                                           QSH
     3. RPC BIND プログラム・デーモン の開始                 RPCBIND
     4. ACCESS FOR WEB の開始                                STRACCWEB2
     5. AFP ユーティリティーの開始                           STRAFPU
     6. エージェント・サービスの開始                        STRAGTSRV
     7. 拡張印刷機能開始                                    STRAPF
     8. BRM を使用したアーカイブ開始                         STRARCBRM
     9. アーカイブ・ストレージ管理の開始                     STRASMOND
    10. ASP バランス化開始                                   STRASPBAL
    11. ASP セッションの開始                                 STRASPSSN
    12. BRM の媒体バランシングの開始                         STRBALBRM
                                                               続く ...
選択またはコマンド
===> ENDCPYSCN

F3= 終了    F4= プロンプト    F9=コマンドの複写    F12= 取り消し
F16= メジャー・メニュー
(C) COPYRIGHT IBM CORP. 1980, 2009.
```

ポインターを使用してチェーン・リストを処理する

ポインター・データ項目を使用するアプリケーションの典型的なものとして、チェーン・リスト (次のレコードを指す一連のレコード) の処理があります。

次の例は、個人の給与 (SALARY) レコードで構成されるデータのチェーン・リストです。 399 ページの図 84 は、これらのレコードがストレージでリンクされて表示される 1 つの方法を示すものです。

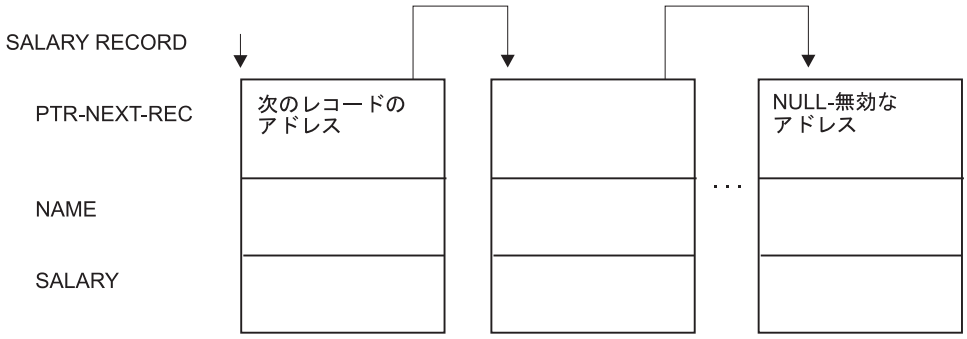


図 84. NULL で終わるチェーン・リストの表示

各レコード (最後のレコードを除く) の最初の項目は、次のレコードを指すポインターです。最後のレコードの最初の項目にはアドレスではなくヌル値が入っています。これはこのレコードが最後のレコードであることを示します。

これらのレコードを処理するアプリケーションの上位レベルのロジックは、次のようになります。

```
OBTAIN ADDRESS OF FIRST RECORD IN CHAINED LIST FROM ROUTINE
CHECK FOR END OF THE CHAINED LIST
DO UNTIL END OF THE CHAINED LIST
   PROCESS RECORD
   GO ON TO THE NEXT RECORD
END
```

400 ページの図 85 に、チェーン・リスト処理のこの例で使用される処理プログラム CHAINLST の概略を示します。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/CHAINLST       ISERIES1   06/02/15 13:45:02       ページ    2
                                ソース
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID. CHAINLST.
    3     000300 ENVIRONMENT DIVISION.
    4     000400 DATA DIVISION.
          000500*
    5     000600 WORKING-STORAGE SECTION.
    6     000700 77  PTR-FIRST         POINTER  VALUE IS NULL.
    7     000800 77  DEPT-TOTAL        PIC 9(4) VALUE IS 0.
          000900*
    8     001000 LINKAGE SECTION.
    9     001100 01  SALARY-REC.
   10     001200   05  PTR-NEXT-REC    POINTER.
   11     001300   05  NAME            PIC X(20).
   12     001400   05  DEPT            PIC 9(4).
   13     001500   05  SALARY          PIC 9(6).
   14     001600 01  DEPT-X            PIC 9(4).
          001700*
   15     001800 PROCEDURE DIVISION USING DEPT-X.
          001900 CHAINLST-PROGRAM SECTION.
          002000 MAINLINE.
          002100*
          002200* FOR EVERYONE IN THE DEPARTMENT RECEIVED AS DEPT-X,
          002300* GO THROUGH ALL OF THE RECORDS IN THE CHAINED LIST BASED ON THE
          002400* ADDRESS OBTAINED FROM THE PROGRAM CHAINANC
          002500* AND ACCUMULATE THE SALARIES.
          002600* IN EACH RECORD, PTR-NEXT-REC IS A POINTER TO THE NEXT RECORD
          002700* IN THE LIST; IN THE LAST RECORD, PTR-NEXT-REC IS NULL.
          002800* DISPLAY THE TOTAL.
          002900*
   16     003000     CALL "CHAINANC" USING PTR-FIRST
   17     003100     SET ADDRESS OF SALARY-REC TO PTR-FIRST
          003200*
   18     003300     PERFORM WITH TEST BEFORE UNTIL ADDRESS OF SALARY-REC = NULL
   19     003400        IF DEPT = DEPT-X THEN
   20     003500           ADD SALARY TO DEPT-TOTAL
          003600        END-IF
   21     003700        SET ADDRESS OF SALARY-REC TO PTR-NEXT-REC
          003800     END-PERFORM
          003900*
   22     004000     DISPLAY DEPT-TOTAL
   23     004100     GOBACK.
          004200
                         *****  ソース仕様の終わり  *****
```

図 85. チェーン・リストを処理する ILE COBOL プログラム

プログラムとプロシージャーとの間でのポインターの受け渡し

CHAINLST プログラムは、最初の SALARY-REC レコード域のアドレスを取得する際に、次のようにしてプログラム CHAINANC を呼び出します。

```
CALL "CHAINANC" USING PTR-FIRST
```

PTR-FIRST は、呼び出し側プログラム (CHAINLST) の WORKING-STORAGE にポインター・データ項目として定義されています。

```
WORKING-STORAGE SECTION.
77  PTR-FIRST         POINTER  VALUE IS NULL.
```

CHAINANC に対する呼び出しから戻る時点で、PTR-FIRST にはチェーン・リスト中の最初のレコードのアドレスが入れられています。

PTR-FIRST は、最初は論理チェックとしてのヌル値に定義されています。呼び出しエラーが起こり、かつ PTR-FIRST がチェーン中の最初のレコードのアドレス値を受け取ることがない場合、PTR-FIRST はヌル値のままになり、プログラムのロジックによって、レコードの処理は行われなくなります。

NULL は、無効なアドレス値をポインター項目に代入するのに使用される形象定数です。 NULL は VALUE IS NULL 文節、SET ステートメント、およびポインターとの関係条件オペランドとして使用されます。

呼び出し側プログラムの LINKAGE SECTION には、チェーン・リスト中の最初のレコードに関する記述が入れられます。また、CALL ステートメントの USING 句を使用して渡される部門コードに関する記述も含まれています。

```
LINKAGE SECTION.
01  SALARY-REC.
  05  PTR-NEXT-REC    POINTER.
  05  NAME            PIC X(20).
  05  DEPT            PIC 9(4).
  05  SALARY          PIC 9(6).
01  DEPT-X            PIC 9(4).
```

レコード記述 SALARY-REC を PTR-FIRST 中のアドレスに基づくものにするには、次のように SET ステートメントを使用してください。

```
CALL "CHAINANC" USING PTR-FIRST
SET ADDRESS OF SALARY-REC TO PTR-FIRST
```

チェーン・リストの終わりの検査

この例のチェーン・リストはセットアップ済みなので、最後のレコードには無効なアドレスが入っています。そのためには、最後のレコード中のポインター・データ項目に値 NULL を代入します。

ポインター・データ項目に値 NULL を代入する方法として、次の 3 とおりの方法があります。

- データ定義中に VALUE IS NULL 文節を指定して、ポインター・データ項目を定義する。
- NULL を SET ステートメント中の送出フィールドにする。
- ポインター・データ項目の初期値は、VALUE 文節が NULL であるかどうかに関係なく NULL になります。

チェーン・リスト中の最後のレコードのポインターにヌル値が入っている場合、そのリストの終わりを検査するためのコードは次のようになります。

```
IF PTR-NEXT-REC = NULL
⋮
(logic for end of chain)
```

リストの終わりに達していない場合は、レコードを処理して次のレコードに移動します。

プログラム CHAINLST では、次のような「DO WHILE」構造を使用して、チェーン・リストの終わりに関するテストを行っています。

```
PERFORM WITH TEST BEFORE UNTIL ADDRESS OF SALARY-REC = NULL
 IF DEPT = DEPT-X
   THEN ADD SALARY TO DEPT-TOTAL
   ELSE CONTINUE
 END-IF
 SET ADDRESS OF SALARY-REC TO PTR-NEXT-REC
END-PERFORM
```

次のレコードの処理

次のレコードに移動するには、LINKAGE SECTION 中のレコードのアドレスを、次のレコードのアドレスの値に設定してください。この処理は、SALARY-REC 中の最初のフィールドとして送られるポインター・データ項目によって行われます。

```
SET ADDRESS OF SALARY-REC TO PTR-NEXT-REC
```

次に、レコード処理ルーチンを繰り返すことにより、チェーン・リスト中の次のレコードが処理されます。

別のプログラムから受け取るアドレスの増分

呼び出し側プログラムから渡されるデータに、無視したいヘッダー情報 (たとえば、コマンド・レベルにマイグレーションされない CICS/400® アプリケーションから受け取るデータの) が含まれている場合があります。

ポインター・データ項目は数値ではないので、この項目に関する演算を直接行うことはできません。しかし、SET verb を使用することによって、渡されるアドレスを増分し、ヘッダー情報をう回することができます。

LINKAGE SECTION を次のようにセットアップすることができます。

```
LINKAGE SECTION.
01  RECORD-A.
  05  HEADER          PIC X(16).
  05  REAL-SALARY-REC PIC X(30).

⋮
01  SALARY-REC.
  05  PTR-NEXT-REC    POINTER.
  05  NAME            PIC X(20).
  05  DEPT            PIC 9(4).
  05  SALARY          PIC 9(6).
```

PROCEDURE DIVISION では、次のようにして SALARY-REC のアドレスを REAL-SALARY-REC のアドレスに基づくものにしてください。

```
SET ADDRESS OF SALARY-REC TO ADDRESS OF REAL-SALARY-REC
```

これで、SALARY-REC は RECORD-A のアドレス + 16 に基づくものになります。

プロシージャー・ポインターによる入り口点アドレスの受け渡し

プロシージャー・ポインター・データ項目 (USAGE IS PROCEDURE-POINTER 文節で定義されるもの) を使用することによって、プログラムの入り口アドレスを、特定の ILE 呼び出し可能サービスで必要な形式で渡すことができます。

たとえば、プログラム実行時に例外条件が発生した時点で、ユーザー作成のエラー処理ルーチンに制御を与えるには、まず ILE プロシージャー (最外部の ILE COBOL プログラムなど) の入り口アドレスを CEEHDLR (条件を管理する ILE 呼び出し可能サービス) に渡して、このアドレスを登録しなければなりません。

プロシージャー・ポインター・データ項目は、次のようなタイプのプログラムの入り口アドレスを入れるように設定できます。

- 最外部の ILE COBOL プログラム

- 別の ILE 言語で作成されている ILE プロシージャー
- ILE プログラム・オブジェクトまたは OPM プログラム・オブジェクト

注: プロシージャー・ポインター・データ項目を、ネストされている ILE COBOL プログラムのアドレスに設定することはできません。

プロシージャー・ポインター・データ項目は、SET ステートメントの形式 6 を使用する場合に限り設定できます。

USAGE IS PROCEDURE-POINTER 文節と SET ステートメントの詳細な定義については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

第 15 章 ILE COBOL プログラムをマルチスレッド化するための準備

i5/OS 環境では、プログラムは、プロセスの**スレッド**内で実行することができます。 ILE COBOL は、THREAD PROCESS ステートメント・オプション (69 ページの『THREAD オプション』を参照) を使用して、マルチスレッド化の実行をサポートします。マルチスレッド化のための ILE COBOL サポートについての本章の説明を理解するためには、以下の用語の理解が必要です。

ジョブ i5/OS では、ジョブは 1 つのプロセスを表します。 オペレーティング・システムおよびマルチスレッド化アプリケーションは、ジョブ内の実行の流れを取り扱うことができます。複数のジョブを同時に実行することができ、ジョブ内で実行しているプログラムはリソースを共用することができます。ジョブは、プログラムの記憶域およびリソースのコンテナーです。

スレッド
ジョブ内で、アプリケーションは、1 つまたはそれ以上のスレッドを開始することができます。 1 つのスレッド内では、制御が、実行プログラム間で転送されます。

実行単位
i5/OS では、実行単位はプログラム活動化グループを表します。 1 つの実行単位には、複数のスレッドを入れることができます。マルチスレッド環境で COBOL 実行単位が終了する場合、ジョブも終了します。実行単位内で、ILE COBOL プログラムは、非 ILE COBOL プログラムを呼び出すことができます (また、その逆も可能です)。

プログラム呼び出しインスタンス
1 つのスレッド内で、制御は、別個の ILE COBOL プログラムと非 ILE COBOL プログラム間で転送されます。たとえば、ILE COBOL プログラムは、他の ILE COBOL プログラムまたは ILE C プログラムを CALL することができます。それぞれ別々に呼び出された (つまり、CALL された) プログラムが、プログラム呼び出しインスタンスです。特定プログラムのプログラム呼び出しインスタンスは、指定のジョブ内の複数のスレッドの中に存在します。

以下の図は、ジョブ、スレッド、実行単位、およびプログラム呼び出しインスタンスの間の関係を示しています。

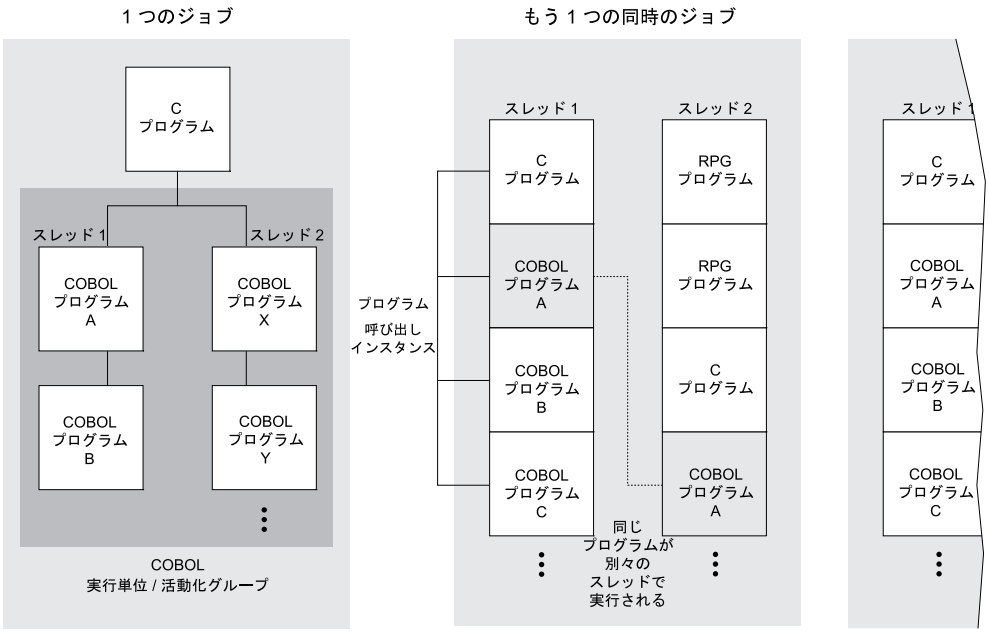


図 86. マルチスレッド化の概念の概略図

ILE COBOL は、プログラム・スレッドの開始または管理をサポートする COBOL ステートメントを持っていませんが、COBOL プログラムは、これを実行する API を使用することができます。 ILE COBOL プログラムは、マルチスレッド化環境のスレッドの中で実行することができます。つまり、ILE COBOL プログラムは、1 つのジョブの複数のスレッド内で実行している他のアプリケーションによって呼び出すか、またはスレッド内の複数のプログラム呼び出しインスタンスとして呼び出すことができます。

本章の後の部分には、マルチスレッド化環境用の ILE COBOL プログラムを作成するのに役立つ情報が記載されています。

この章では次のことについて説明します。
- マルチスレッド化環境でどのように言語エレメントが解釈されるか
- マルチスレッド化サポートの THREAD をいつ選択するか
- マルチスレッド化環境での制御転送
- マルチスレッド化環境での ILE COBOL の使用例

マルチスレッド化環境でどのように言語エレメントが解釈されるか

ILE COBOL プログラムはジョブ内の別々のスレッドとして実行することができるため、言語エレメントは、以下の 2 つの方法で解釈されることに注意してください。

実行単位有効範囲
言語エレメントは ILE COBOL 実行単位の実行の期間の間は存続し、スレッド内の他のプログラムで使用することができます。

プログラム呼び出しインスタンス有効範囲
言語エレメントは、特定のプログラム呼び出しインスタンス内でだけ存続します。

これらの有効範囲の 2 つのタイプは、以下の 2 つの文脈において重要です。

参照 項目がそこから参照されることができる場所を記述します。たとえば、データ項目が「実行単位」の参照有効範囲を持っている場合、実行単位内のどのプログラム呼び出しインスタンスも、データ項目を参照することができます。

状態 項目がストレージ内にどれだけ長く存続するかを記述します。たとえば、データ項目が「プログラム呼び出しインスタンス」の状態有効範囲を持っている場合、データ項目は、プログラム呼び出しインスタンスが実行している間だけストレージに残ります。

以下の表に、さまざまな ILE COBOL 言語エレメントの参照および状態の有効範囲を要約します。

言語エレメント	参照の有効範囲	状態の有効範囲
ADDRESS-OF 特殊レジスター	関連レコードと同じ	プログラム呼び出しインスタンス
DB-FORMAT-NAME 特殊レジスター	実行単位	プログラム呼び出しインスタンス
DEBUG-ITEM 特殊レジスター	構文チェックのみ	
ファイル	実行単位	実行単位
FORMAT OF 特殊レジスター	関連 ID と同じ	関連 ID と同じ
指標データ	プログラム	プログラム呼び出しインスタンス
LENGTH OF 特殊レジスター	関連 ID と同じ	関連 ID と同じ
LINAGE-COUNTER 特殊レジスター	関連ファイルと同じ	関連ファイルと同じ
LINKAGE-SECTION データ	実行単位	基礎になるデータの有効範囲に基づく
LOCAL-STORAGE データ	プログラム	プログラム呼び出しインスタンス
LOCALE OF 特殊レジスター	関連 ID と同じ	関連 ID と同じ
RETURN-CODE	実行単位	プログラム呼び出しインスタンス
WHEN-COMPILED 特殊レジスター	実行単位	実行単位
WORKING-STORAGE データ	実行単位	実行単位
SORT-RETURN 特殊レジスター	実行単位	プログラム呼び出しインスタンス

実行単位の範囲を指定したエレメントの処理

実行単位の範囲を指定したエレメントは、複数のモジュールで共用できるストレージを持っています。共用ストレージの例は、以下のとおりです。

- 外部ファイルおよび共用ファイル
- 外部データ項目
- モジュール間の CALL BY REFERENCE

実行単位内で、各モジュールはロックを持ち、ILE COBOL は、一度にモジュールの 1 つのコピーだけが実行単位内で実行するようにします。実行単位の有効範囲を持つリソースがある場合、アプリケーションの中で、ロジックを使用して複数のスレッドからそのデータへのアクセスを同期化するのは、プログラマーの責任です。以下の 1 つまたは両方を実行することができます。

- 実行単位の範囲を指定したリソースが、複数のスレッドから同時にアクセスされないように、アプリケーションを構成します。

- 別々のスレッドから同時にリソースにアクセスする場合は、C が提供する機能を使用するか、または mutex を作成したり処理したりするための Pthread mutex サポートや MI 組み込み関数などのプラットフォーム機能によって、アクセスを同期化します。詳しくは、以下の URL で、**プログラミング (Programming)** のトピックにリストされている、マルチスレッド・アプリケーション (Multithreaded Applications) のドキュメントを参照してください。

  `http://www.as400.ibm.com/infocenter/`

プログラム呼び出しインスタンスの範囲を指定した要素の処理

これらの言語エレメントで、ストレージは、個別プログラム呼び出しインスタンスごとに割り振られます。したがって、プログラムが複数のスレッド間で複数回呼び出されたとしても、それが呼び出されるごとに別々のストレージを割り振ります。たとえば、プログラム X が複数のスレッドで呼び出されると、X のそれぞれのプログラム呼び出しインスタンスは、それが所有する一連のリソース (ストレージなど) を取得します。

これらの言語エレメントに関連するストレージは、プログラム呼び出しインスタンスの範囲を指定されているので、データが複数のスレッドにわたってアクセスされるのが保護されており、アクセス同期を心配する必要はありません。ただし、このデータは、それが明示的に渡されない限り、プログラム呼び出し間で共用することはできません。

マルチスレッド化サポートのための THREAD の選択

THREAD(SERIALIZE) PROCESS オプションを、マルチスレッド化 Java アプリケ
ーションと相互作用するすべてのモジュールでコード記述する必要があります。
COBOL は、明らかに自動ストレージを使用するだけのプログラムおよびプロシー
ジャーにおいてさえも、静的ストレージに強く依存しています。
THREAD(SERIALIZE) は、この静的ストレージを正しく処理しているかを確かめる
ために必要です。対象はJava メソッドへの呼び出しを含むモジュールだけではあり
ません。アプリケーションの Java 部分がマルチスレッドで実行されている可能性が
ある場合、、Java との対話中に呼び出される可能性のある、すべてのモジュールに
対しても適用されます。

マルチスレッド化サポートのための PROCESS ステートメントの THREAD オプションで、SERIALIZE を選択します。 SERIALIZE を指定したコンパイルは、スレッド化サポートのための ILE COBOL 実行時環境を準備します。ただし、SERIALIZE を指定してコンパイルすると、プログラムのパフォーマンスが低下することがあります。実行単位内のすべてのプログラムを SERIALIZE を指定してコンパイルする必要があります。すなわち、SERIALIZE を指定してコンパイルしたプログラムと NOTHREAD を指定してコンパイルしたプログラムを 1 つの実行単位内で混合することができません。

デフォルト・オプションは THREAD(NOTHREAD) です。 THREAD PROCESS ステートメント・オプションについての詳細は 69 ページの『THREAD オプション』を参照してください。

THREAD についての言語に関する制約事項

THREAD(SERIALIZE) が有効である場合、以下の言語エレメントはサポートされておらず、コンパイラーによって重大エラー・メッセージ (重大度 30) のフラグが付けられます。

- ALTER ステートメント
- プロシージャー名なしの GO TO ステートメント
- PROGRAM-ID 段落での INITIAL 句
- STOP リテラル・ステートメント
- STOP RUN
- WITH DEBUGGING MODE 文節

DDM データ域の使用は、マルチスレッド化環境では許されません。

マルチスレッド化環境では、UPSI スイッチを使用しないようにすることをお勧めします。これは、あるスレッドがスイッチを設定して、それを検査する前に、別のスレッドがそのスイッチを再び設定する可能性があるからです。

マルチスレッド化環境での制御転送

マルチスレッド化環境の ILE COBOL プログラムを作成する場合、以下の制御転送における問題に注意してください。

CALL および CANCEL

単一スレッド化環境においては、呼び出されたプログラムは、それが実行単位内で最初に呼び出され、しかも、呼び出されたプログラムの CANCEL の後に最初に呼び出される、初期状態にあります。

EXIT PROGRAM

EXIT PROGRAM は、すべての場合にスレッドを終了することなしに、プログラムの呼び出し側に戻ります。メイン・プログラムからの EXIT PROGRAM は、コメントとして処理されます。

GOBACK

メイン・プログラムからの GOBACK が呼び出し側に戻ることを除いては、EXIT PROGRAM と同じです。実行単位内で呼び出された ILE COBOL プログラムすべてが GOBACK または EXIT PROGRAM 経由でそれぞれの呼び出し側に戻った場合、この判別をすることができます。

マルチスレッド化環境での ILE COBOL の制限

いくつかの ILE COBOL アプリケーションは、サブシステムまたは他のアプリケーションに依存します。マルチスレッド化環境では、これらの依存関係によって、以下に示すような ILE COBOL プログラム上の制限がいくつか出てきます。

SORT/MERGE

SORT および MERGE は、一度に 1 つのスレッド内でのみ活動状態にする必要があります。ただし、これを COBOL ランタイム環境が実施することはないので、アプリケーションで制御する必要があります。

外部ファイルおよび共用ファイル
外部ファイルおよび共用ファイルは、複数のスレッドから同時にアクセスまたは更新してはなりません。ただし、これを COBOL ランタイム環境が実施することはないので、アプリケーションで制御する必要があります。

一般的に、実行単位内のアプリケーションが認識できるリソースへの同期アクセスは、アプリケーションの責任で行う必要があります。

マルチスレッド化環境での ILE COBOL の使用例

この例は、2 つの ILE COBOL スレッドを作成し、それらの ILE COBOL スレッドが終わるのを待ってから、終了する ILE COBOL メイン・プロシージャーで構成されています。

マルチスレッド化の例のサンプル・コード

この例には、以下の 3 つのサンプル・コードがあります。

THRCBL QCBLLESRC
ILE COBOL スレッドを作成し、それらのスレッドが終わるのを待ってから、終了する ILE COBOL メイン・プロシージャー。

SUBA QCBLLESRC
THRCBL が作成したスレッドによって呼び出される ILE COBOL プロシージャー。

SUBB QCBLLESRC
THRCBL が作成したスレッドによって呼び出される 2 番目の ILE COBOL プロシージャー。

THRCBL QCBLLESRC のサンプル・コードは 411 ページの図 87 に示されています。

```
PROCESS NOMONOPRC OPTIONS THREAD(SERIALIZE).
IDENTIFICATION DIVISION.
PROGRAM-ID. THRCBL.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
  special-names.  system-console is oper1.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 pthread_attr_t typedef.
   05 FILLER PIC 9(8) usage binary occurs 4 times.
   05 FILLER USAGE POINTER.
01 pthread_t typedef.
   05 FILLER USAGE POINTER.
   05 FILLER PIC 9(8) usage binary.
   05 FILLER PIC 9(8) usage binary.
   05 FILLER PIC 9(8) usage binary.
   05 FILLER PIC 9(8) usage binary.
   05 FILLER USAGE POINTER.
01 PROC-SUBA-PTR USAGE PROCEDURE-POINTER.
01 PROC-SUBB-PTR USAGE PROCEDURE-POINTER.
01 attr type pthread_attr_t.
01 rc PIC 9(8) usage binary value 0.
01 group1.
   05 thread type pthread_t occurs 10 times.
01 joinStatus0 USAGE POINTER.
01 joinStatus1 USAGE POINTER.
PROCEDURE DIVISION.
TEST1-INIT.
    SET PROC-SUBA-PTR TO ENTRY PROCEDURE "SUBA".
    SET PROC-SUBB-PTR TO ENTRY PROCEDURE "SUBB".

* スレッド属性オブジェクトを作成
    call procedure "pthread_attr_init" using attr
                   returning rc.

* スレッドを結合可能として定義
    call procedure "pthread_attr_setdetachstate" using attr
                   by value 0 size 4
                   returning rc.
```

図 87. THRCBL QCBLLESRC のソース・コード (1/2)

```
      * スレッドの作成開始
           call procedure "pthread_create" using thread(1) attr
                         by value PROC-SUBA-PTR omitted
                         returning rc.
           call procedure "pthread_create" using thread(2) attr
                         by value PROC-SUBB-PTR omitted
                         returning rc.
      * スレッドの結合開始
           call procedure "pthread_join" using by value thread(1)
                          by reference joinStatus0
                          returning rc.
           call procedure "pthread_join" using by value thread(2)
                          by reference joinStatus1
                          returning rc.

      * スレッド属性オブジェクトの破棄
           call procedure "pthread_attr_destroy" using attr
                          returning rc.
```

図 87. THRCBL QCBLLESRC のソース・コード (2/2)

SUBA QCBLLESRC のサンプル・コードは、図 88 に示されています。

```
       PROCESS NOMONOPRC OPTIONS THREAD(SERIALIZE).
       IDENTIFICATION DIVISION.
       PROGRAM-ID. SUBA.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
         special-names.  system-console is oper1.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 one-line pic x(11).
       PROCEDURE DIVISION.
       TEST1-INIT.
           move "IN SUBA" TO ONE-LINE.
           DISPLAY one-line UPON oper1.
```

図 88. SUBA QCBLLESRC のソース・コード

SUBB QCBLLESRC のサンプル・コードは、413 ページの図 89 に示されています。

```
PROCESS NOMONOPRC OPTIONS THREAD(SERIALIZE).
IDENTIFICATION DIVISION.
PROGRAM-ID. SUBB.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
  special-names.  system-console is oper1.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 one-line pic x(11).
PROCEDURE DIVISION.
TEST1-INIT.
    move "IN SUBB" TO ONE-LINE.
    DISPLAY one-line UPON oper1.
```

図 89. SUBB QCBLLESRC のソース・コード

マルチスレッド化の例の作成および実行

マルチスレッド化の例を作成し、実行するには、以下のステップに従ってください。

1. 3 つの ILE COBOL モジュールを作成します。
   - ILE COBOL モジュール THRCBL を作成するには、以下のように入力します。

     ```
     CRTCBLMOD MODULE(THRCBL) SRCFILE(*CURLIB/QCBLLESRC) DBGVIEW(*ALL)
     ```

   - ILE COBOL モジュール SUBA を作成するには、以下のように入力します。

     ```
     CRTCBLMOD MODULE(SUBA) SRCFILE(*CURLIB/QCBLLESRC) DBGVIEW(*ALL)
     ```

   - ILE COBOL モジュール SUBB を作成するには、以下のように入力します。

     ```
     CRTCBLMOD MODULE(SUBB) SRCFILE(*CURLIB/QCBLLESRC) DBGVIEW(*ALL)
     ```

2. 3 つのモジュールを使用して、プログラム THREAD を作成します。
   - THREAD プログラムを作成するには、以下のように入力します。

     ```
     CRTPGM PGM(THREAD) MODULE(*CURLIB/THRCBL *CURLIB/SUBA *CURLIB/SUBB)
     ```

3. SPAWN コマンドを作成し、マルチスレッド化プログラム THREAD を呼び出します。

   ```
   SPAWN MYLIB/THREAD DEBUG(2)
   ```

   SPAWN コマンドの作成方法については、URL http://www.ibm.com/systems/i/infocenter/ で、『**プログラミング**』トピックにリストされている、『マルチスレッド・アプリケーション (Multithreaded Applications)』文書を参照してください
4. プログラムの出力を表示するには、DSPMSG QSYSOPR を入力してください。出力は、どのスレッドが最初に実行するかによりますが、以下のようなスレッドの順序を表示します。

   ```
   IN SUBB
   IN SUBA
   ```

   または次のようになります。

   ```
   IN SUBA
   IN SUBB
   ```

第 16 章 ILE COBOL のエラーおよび例外の処理

ILE COBOL には、プログラムの実行中に起こり得るエラー状態を予期したり訂正したりするために役立つ特殊なエレメントが含まれています。コードが完全な場合でも、プログラムで使用するシステム機能でエラーが起こることがあります。

それらを処理するためのコードをプログラムに挿入することによって、起こり得るエラー条件を予期できます。プログラムにエラー処理コードがない場合、予期しなかった方法でそのプログラムが実行されてしまい、データ・ファイルが破壊されたり、誤った出力が作成されたりすることがあります。エラー処理コードがないと、問題があることに気付かない場合さえあります。

エラー処理コードによって取られるアクションには、状態の処理と継続、メッセージの発行、またはプログラムの停止の試行などが含まれます。最低でも、エラー状態を識別するためのエラー・メッセージをコーディングすることをお勧めします。

ILE COBOL プログラムを実行する際に、重大エラーが起こることがあります。エラー発生時に活動状態だった ILE COBOL 文節または句によって、ILE COBOL の特定の文節または句が実行されます。

この章では、次の処理を行う方法について説明します。
- エラー処理バインド可能 API の使用
- 意図的なダンプの開始
- ストリング操作中のエラー処理
- 算術演算のエラー処理
- 入出力操作中のエラー処理
- ソート・マージ操作中のエラー処理
- CALL ステートメントの例外処理
- ユーザー作成のエラー処理ルーチンの作成

ILE 条件処理

AS/400 システムでは、プログラムが互いに状況を通信する方法がいくつかあります。主な方法の 1 つは、IBM i メッセージを送信することです。

IBM i メッセージには複数のタイプがあります。それには、照会、情報、完了、エスケープ、および通知メッセージがあります。たとえば、コンパイルが成功した場合に ILE COBOL コンパイラーから送られる最後のメッセージは、次のような LNC0901 です。

プログラム *program-name* がライブラリー *library-name* に *date* の *time* に作成されました。

メッセージ LNC0901 は完了メッセージです。 コンパイルが失敗すると、次のようなメッセージ LNC9001 を受け取ります。

```
Compile failed.  Program-name not created.
```

メッセージ LNC9001 はエスケープ・メッセージです。

ILE 条件と IBM i メッセージはよく似ています。エスケープ、状況、通知、および機能チェック・メッセージは条件であり、各 ILE 条件には関連する IBM i メッセージがあります。

IBM i メッセージは、宣言してメッセージ・モニターを使用可能にすると処理できますが、これと同様に ILE 条件も **ILE 異常事態処理ルーチン**を登録することによって処理できます。 ILE 異常事態処理ルーチンを使用すると、実行時に例外処理プロシージャーを登録でき、例外が起こるとこの手順に制御が与えられます。異常事態処理ルーチンを登録するには、ユーザー作成異常事態処理ルーチン登録 (CEEHDLR) バインド可能 API を使用してください。

プログラム・オブジェクトまたは ILE プロシージャーが呼び出されると、新しい呼び出しスタック項目が作成されます。各呼び出しスタック項目には、それぞれ呼び出しメッセージ待ち行列が対応しています。この呼び出しメッセージ待ち行列は、プログラム・オブジェクトが呼び出される場合はプログラム・メッセージ待ち行列であり、ILE プロシージャーが呼び出される場合はプロシージャー・メッセージ待ち行列です。 ILE では、呼び出しスタック項目にメッセージを送ると、関連するプログラム・オブジェクトまたは ILE プロシージャーにメッセージを送ることができます。

同様に、呼び出しスタック項目に条件を通知すると、関連するプログラム・オブジェクトまたは ILE プロシージャーに条件を通知できます。プログラム・オブジェクトに条件を通知する際には、ILE バインド可能 API を使用します。条件管理バインド可能 API のリストについては、「*ILE 概念*」の ILE バインド可能 API についての部分を参照してください。

各呼び出しスタック項目ごとに、複数の ILE 異常事態処理ルーチンを登録できます。 1 つの呼び出しスタック項目に対して複数の ILE 条件ハンドラーが登録されている場合、システムは後入れ先出し (LIFO) の順にこれらのハンドラーを呼び出します。 ILE 異常事態処理ルーチンをさまざまな優先順位で登録することもできます。 ILE COBOL が使用するのは、これらの優先順位のうちのわずかなものだけです。 85〜225 の範囲で約 10 個の異なる優先順位があります。 ILE 異常事態処理ルーチンは、優先順位の数値が小さいものから順に呼び出されます。

ILE で、特定の呼び出しスタック項目で例外条件が処理されないなら、未処理の例外メッセージは、直前の呼び出しスタック項目メッセージ待ち行列にパーコレートされます。この状態が起きると、例外処理は直前の呼び出しスタック項目から続行します。管理境界に達するか、または例外メッセージが処理されるまで、未処理の例外条件のパーコレーションは続行します。未処理の例外メッセージが管理境界にパーコレートされると、それは機能チェックに変換されます。

次に機能チェックが、元の例外条件を発行した呼び出しスタック項目に処理されるか、または管理境界にパーコレートされます。機能チェックが処理されると、通常の処理が継続して例外処理は終了します。機能チェックが管理境界にパーコレートされると、ILE はアプリケーションが予期しないエラーで終了したと見なします。総称障害例外メッセージ CEE9901 が、ILE から管理境界の呼び出し側に送られます。

プログラム・オブジェクトまたは ILE プロシージャーで例外条件が起きると、それは、まずそのプログラム・オブジェクトまたは ILE プロシージャーの呼び出しスタ

ック項目用に登録されている ILE 異常事態処理ルーチンによって処理されます。呼び出しスタック項目用に ILE 異常事態処理ルーチンが登録されていない場合、例外条件は HLL 固有のエラー・ハンドラーによって処理されます。 HLL 固有のエラー・ハンドラーは、エラーを処理するために定義される言語機能です。 ILE COBOL での HLL 固有エラー処理には、入出力エラー処理用の USE 宣言や、ON SIZE ERROR および INVALID KEY などの、有効範囲がステートメントである条件句の命令が含まれます。

例外条件が HHL 固有のエラー処理で処理されないなら、前述のように未処理の例外条件が直前の呼び出しスタック項目メッセージ待ち行列にパーコレートされます。

ILE 条件処理についての詳細は、「*ILE 概念*」の、エラー処理についての部分、および例外と条件の管理についての部分を参照してください。

ILE COBOL プログラムの終了

ILE COBOL プログラムは、以下のものによって終了することがあります。

- ILE COBOL ステートメント (EXIT PROGRAM、STOP RUN、または GOBACK)
- 照会メッセージに対する応答
- 暗黙の STOP RUN ステートメントまたは EXIT PROGRAM ステートメント
- 別の ILE 言語で、ILE COBOL STOP RUN ステートメントに相当するもの。たとえば、ILE C の `exit()` 関数。
- 別の ILE 言語で、ILE COBOL 異常 STOP RUN ステートメントに相当するもの。たとえば、ILE C の `abort()` 関数。
- 呼び出し先の ILE プロシージャーまたはプログラム・オブジェクトが、呼び出し側 ILE COBOL プログラムの範囲を超えて送るエスケープ・メッセージ。
- 呼び出し側 ILE COBOL プログラムが実行している活動化グループが、呼び出し先 ILE プロシージャーまたはプログラム・オブジェクトによって終了すること。

メイン ILE COBOL プログラムにその次の実行可能ステートメントがない場合 (ILE COBOL サブプログラムの場合は暗黙の EXIT PROGRAM)、すなわちプログラムの最後のステートメントまで処理が実行された場合は、暗黙のうちに STOP RUN ステートメントが実行されます。

照会メッセージは、ILE COBOL ステートメント (すなわち STOP リテラル) に応じて出されることもありますが、通常はプログラムで重大エラーが起きた場合や、ILE COBOL 操作が正常に完了しない場合に出されます。 (例: LNR7205、LNR7207、および LNR7208。) 照会メッセージを調べることによって、例外エラーが起きた後に取るアクションを判別できます。

COBOL 照会メッセージに対する一般的な応答として、C、D、F、および G (取り消し、取り消しおよびダンプ、取り消しおよびすべてのダンプ、続行) の 4 つがあります。最初の 3 つの応答を (最終ステップとして) 行うと、異常な STOP RUN が暗黙のうちに実行されます。

メイン ILE COBOL プログラム中の暗黙または明示の STOP RUN ステートメント、または GOBACK ステートメントによって、最も近い管理境界に緊急終了条件が通知されます。次の 2 つの方法でその緊急終了条件を処理できます。

- 管理境界に達する前に、登録されているエラー・ハンドラーを使用するか、または

  注: 異常事態処理ルーチンを登録するには、ユーザー作成異常事態処理ルーチン登録 (CEEHDLR) バインド可能 API を使用してください。例外ハンドラーの詳細については、「*ILE 概念*」を参照してください。
- 管理境界に達した場合は、管理境界より後のプログラムをすべて終了し、管理境界の前のプログラムに制御を戻す。

この管理境界がハード管理境界である場合は、活動化グループ (実行単位) が終了します。

STOP RUN が正常に実行されず、ハード管理境界に達した場合は、管理境界の前のプログラムに CEE9901 エスケープ・メッセージが出されます。

エラー処理バインド可能アプリケーション・プログラム・インターフェース (API) の使用

ILE COBOL でのエラー処理のレベルには 2 つあります。まず、各優先順位で登録されている異常事態処理ルーチンに、条件を処理する機会が与えられます。管理境界に達した時点で条件が未処理のままであれば、機能チェック条件が送られます。各 ILE COBOL ILE プロシージャーには、ILE 異常事態処理ルーチンが機能チェック処理用に優先順位 205 で登録されています。この機能チェック異常事態処理ルーチンは、次のバインド可能 API による処理が行われないと、COBOL 照会メッセージを出します。

- COBOL エラー・ハンドラー検索 (QlnRtvCobolErrorHandler)

  COBOL エラー・ハンドラー検索 (QlnRtvCobolErrorHandler) API は、この API の呼び出し側である活動化グループの現行の ILE COBOL エラー処理プロシージャーの名前を検索するためのものです。
- COBOL エラー・ハンドラー設定 (QlnSetCobolErrorHandler)

  COBOL エラー・ハンドラー設定 (QlnSetCobolErrorHandler) API は、この API の呼び出し側である活動化グループの ILE COBOL エラー処理プロシージャーの ID を指定するためのものです。

上記の API は、ILE COBOL プログラム内の例外処理にのみ影響を与えます。これらのすべての API の詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「プログラミング」カテゴリーの中の『*CL および API*』セクションの COBOL API に関するセクションを参照してください。

注: これらの API を使用する際は、CRTCBLMOD コマンドまたは CRTBNDCBL コマンドの OPTION パラメーターに *NOMONOPRC 値を指定しなければなりません。

意図的なダンプの開始

COBOL ダンプ (QlnDumpCobol) バインド可能 API を使用して、ILE COBOL プログラムの定様式ダンプを意図的に開始できます。 QlnDumpCobol API は、次のものを定義する 6 つのパラメーターを受け入れます。

- プログラム・オブジェクト名
- ライブラリー名
- モジュール・オブジェクト名
- プログラム・オブジェクトのタイプ
- ダンプのタイプ
- エラー・コード

次の例で、QlnDumpCobol API を呼び出す方法と、その結果の操作を示します。

```
WORKING-STORAGE SECTION.
01  ERROR-CODE.
    05  BYTES-PROVIDED     PIC S9(6)  BINARY VALUE ZERO.
    05  BYTES-AVAILABLE    PIC S9(6)  BINARY VALUE ZERO.
    05  EXCEPTION-ID       PIC X(7).
    05  RESERVED-X         PIC X.
    05  EXCEPTION-DATA     PIC X(64).
01  PROGRAM-NAME           PIC X(10).
01  LIBRARY-NAME           PIC X(10).
01  MODULE-NAME            PIC X(10).
01  PROGRAM-TYPE           PIC X(10).
01  DUMP-TYPE              PIC X.
PROCEDURE DIVISION.
    MOVE LENGTH OF ERROR-CODE  TO BYTES-PROVIDED.
    MOVE "MYPROGRAM"           TO PROGRAM-NAME.
    MOVE "TESTLIB"             TO LIBRARY-NAME.
    MOVE "MYMOD1"              TO MODULE-NAME.
    MOVE "*PGM"                TO PROGRAM-TYPE.
    MOVE "D"                   TO DUMP-TYPE.
    CALL PROCEDURE "QlnDumpCobol"  USING PROGRAM-NAME,
                                         LIBRARY-NAME, MODULE-NAME,
                                         PROGRAM-TYPE, DUMP-TYPE,
                                         ERROR-CODE.
```

これにより、ライブラリー TESTLIB 中のプログラム・オブジェクト MYPROGRAM 中のモジュール・オブジェクト MYMOD1 の COBOL ID の定様式ダンプを得ることができます (オプション D)。

```
WORKING-STORAGE SECTION.
01  ERROR-CODE.
    05  BYTES-PROVIDED     PIC S9(6)  BINARY VALUE ZERO.
    05  BYTES-AVAILABLE    PIC S9(6)  BINARY VALUE ZERO.
    05  EXCEPTION-ID       PIC X(7).
    05  RESERVED-X         PIC X.
    05  EXCEPTION-DATA     PIC X(64).
01  PROGRAM-NAME           PIC X(10).
01  LIBRARY-NAME           PIC X(10).
01  MODULE-NAME            PIC X(10).
01  PROGRAM-TYPE           PIC X(10).
01  DUMP-TYPE              PIC X.
PROCEDURE DIVISION.
    MOVE LENGTH OF ERROR-CODE  TO BYTES-PROVIDED.
    MOVE "*SRVPGM"             TO PROGRAM-TYPE.
    MOVE "F"                   TO DUMP-TYPE.
    CALL PROCEDURE "QlnDumpCobol"  USING OMITTED, OMITTED,
                                         OMITTED, PROGRAM-TYPE,
                                         DUMP-TYPE, ERROR-CODE.
```

これにより、QlnDumpCobol API を呼び出したサービス・プログラムの COBOL ID とファイル関連情報の定様式ダンプを得ることができます (オプション F)。

QlnDumpCobol API に対する入力パラメーターのどれかに無効なデータがあるとダンプは実行されず、エラー・メッセージが生成されるか、または、例外データが戻されます。 BYTES-PROVIDED フィールドがゼロの場合はエラー・メッセージが生成されます。 BYTES-PROVIDED フィールドの値がゼロでない場合は、ERROR-CODE パラメーターに例外データが戻され、エラー・メッセージは生成されません。

ユーザーが定様式ダンプ内のプログラムの変数の値を見ることができないようにするには、以下のいずれか 1 つを実行します。

- プログラム識別情報を除去して、プログラム内にデバッグ・データが存在しないようにする。
- プログラムを実行するのに十分な権限をユーザーに付与するが、定様式ダンプの実行権限は付与しない。それには、*OBJOPR および *EXECUTE 権限を与えます。
- このプログラムでは QlnDumpCobol を呼び出さないでください。

QlnDumpCobol API の詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「*プログラミング*」カテゴリーの中の『*CL および API*』セクションの COBOL API に関するセクションを参照してください。

プログラム状況構造体

プログラム状況構造体は、プログラム中でエラーが発生したときにエラー情報を提供するサブフィールドを含む、事前定義の構造体です。 PROGRAM STATUS 文節を使用して、エラー情報を受け取るデータ項目を指定することで、これらのサブフィールドにアクセスします。プログラム状況構造体および PROGRAM STATUS 文節の詳細については、「*WebSphere Development Studio: ILE COBOL 解説書*」を参照してください。

ストリング操作中のエラー処理

データに対して、STRING または UNSTRING の操作を実行する際に、エラーが発生する場合があります。 STRING ステートメントと UNSTRING ステートメントには ON OVERFLOW 句があり、この句で通常のストリング・オーバーフロー・エラー状態を処理することができます。 STRING ステートメントの場合、暗黙または明示のポインターが次の値のときに、ON OVERFLOW 句が実行されます。

- 1 より小さい。
- 受け取りフィールドの長さより大きい。

UNSTRING ステートメントの場合は、次のときに ON OVERFLOW 句が実行されます。

- 暗黙または明示のポインターの値が 1 より小さい。
- 暗黙または明示のポインターの値が送り側フィールドの長さより大きい。

- すべての受け取りフィールドが処理されたが、送り側フィールドに未検査の文字がある。

ON OVERFLOW 句で処理されないその他のエラー条件があると、一般的には MCH メッセージが出されます。多くの場合、このメッセージは、機能チェック異常事態処理ルーチンによって処理されます。機能チェック異常事態処理ルーチンが呼び出されないようにするには、CEEHDLR API を使用して独自の異常事態処理ルーチンを登録し、MCH メッセージをキャッチすることができます。

オーバーフロー条件が発生した場合、STRING ステートメントまたは UNSTRING ステートメントの ON OVERFLOW 句を使用することによって、実行したいエラー処理ステップを識別します。 STRING ステートメントや UNSTRING ステートメントに ON OVERFLOW 文節がない場合は、次の順次ステートメントに制御が渡され、完了していない操作は通知されません。

ON OVERFLOW 句の詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の STRING および UNSTRING ステートメントの部分を参照してください。

算術演算のエラー処理

算術演算においては、特定の典型的なエラーが発生することがあります。そのような典型的なエラーが発生すると MCH メッセージが出されます。

ON SIZE ERROR 句

ADD、SUBTRACT、MULTIPLY、DIVIDE、および COMPUTE ステートメントの ON SIZE ERROR 句は次の処理を行います。

- 2 進数と 10 進数のオーバーフロー・メッセージを出すことを可能にする。この 2 進数と 10 進数のオーバーフロー・メッセージは MCH1210 です。 10 進数のゼロ除算メッセージは MCH1211 です。
- 2 進数、10 進数、および浮動小数点数のオーバーフロー・メッセージ、およびその他の算術演算 MCH メッセージをキャッチする異常事態処理ルーチンを登録する。浮動小数点オーバーフロー・メッセージには、MCH1206 (オーバーフロー) および MCH1207 (アンダーフロー) が含まれています。

  2 進数および 10 進数オーバーフロー・メッセージとは違い、浮動小数点数オーバーフローは、ON SIZE ERROR 句があっても使用可能にはなりません。浮動小数点数オーバーフローは、ジョブ・レベルでオン / オフにします。デフォルトでは、浮動小数点数オーバーフロー・メッセージは必ず出されます。 このため ILE COBOL は、ON SIZE ERROR 句がコーディングされている場合は別として、そのようなメッセージを無視します。浮動小数点数オーバーフローをオン / オフにすることについては 422 ページの『浮動小数点計算のエラー処理』を参照してください。

ILE COBOL は、前述の異常事態処理ルーチンを優先順位 85 で登録します。ユーザー異常事態処理ルーチンは、優先順位 165 で登録され、前述の異常事態処理ルーチンが例外を処理しない場合に限り制御を受け取ります。

ON SIZE ERROR 句がコーディングされていない場合、2 進数および 10 進数オーバーフロー・メッセージは出されず、浮動小数点数メッセージは無視されます。CEEHDLR API を使用してユーザー異常事態処理ルーチンを登録していない場合、その他のすべての算術 MCH メッセージは、多くの場合、機能チェック異常事態処理ルーチンで処理されます。

サイズ・エラー条件は、次の状況で発生します。

- 算術演算の結果が、その結果を入れる固定小数点フィールドより大きい。
- ゼロ除算
- ゼロのゼロ乗
- ゼロの負数乗
- 負数の小数乗
- 浮動小数点数オーバーフローまたはアンダーフロー

算術演算における一般的なエラーとして、サイズ・エラー (MCH1210) と 10 進数データ・エラー (MCH1202) があります。ほとんどの MCH エラーは、ILE COBOL で直接には検出されず、オペレーティング・システムで検出され、システム・メッセージが表示されます。 ILE COBOL はそのようなメッセージをモニターして、SIZE ERROR 命令ステートメントを実行するか、それとも実行時メッセージ (LNR7200) を出してプログラムを終了するかを決定する内部ビットを設定します。

LNR7200 メッセージを送らないようにするためには、CEEHDLR API を使用して MCH メッセージを処理するユーザー異常事態処理ルーチンを登録するか、または COBOL バインド可能 API を使用して、LNR72xx 照会メッセージを処理する ILE COBOL エラー・ハンドラーをコーディングすることができます。

ILE COBOL は、算術演算中にゼロ除算の結果として起こったエラーは検出します。 ILE COBOL がこの種のエラーを検出すると、SIZE ERROR 命令ステートメントが実行されます。

一般に、システム・メッセージ MCH1210 が出されるのは、1 つの 2 進数または 10 進数の数字フィールドを別のフィールドに移動したときに、その受け取り側が小さすぎる場合です。このエラーは ILE COBOL でモニターされ、やはり SIZE ERROR 命令ステートメントが出されます。

LNR7200 は、通常は ILE COBOL プログラム中でモニターされていない重大エラーが起こると出される実行時メッセージです。

モニターされていない重大エラーが起きた時に出されるメッセージの典型的なものとして、システム・メッセージ MCH1202 があります。この種のエラーが発生すると、ILE COBOL 実行時メッセージ LNR7200 (または、ILE COBOL プログラムによって呼び出されたプログラム中でエラーが発生した場合は LNR7204) が出されます。モニターされていない重大エラーの例としては、そのほかにシステム・メッセージ MCH3601 および MCH0601 もあります。

浮動小数点計算のエラー処理

IBM i に用意されている一群の計算属性 (CA) MI 命令は、浮動小数点演算に関する情報を検索したり、浮動小数点演算の方法を変更したりするためのものです。 た

とえば、SETCA (計算属性の設定) MI 命令は、特定の浮動小数点例外が起きないようにし、さらに丸めを行うかどうかについても指定します。デフォルトでは、浮動小数点演算結果は必ず 丸められ、無効オペランド以外のすべての例外を通知します。

浮動小数点数に関して、発生しないようになっている例外は次のとおりです。

1. オーバーフロー
2. アンダーフロー
3. ゼロ除算
4. 不正確な結果
5. 無効オペランド

ON SIZE ERROR 句の処理については、ILE COBOL では最初の 3 つが通知される必要があります。

ILE COBOL でも、小数点以下のうち最も近い桁への丸めが必要であるため、CA MI 命令を使用して丸めを行わないようにした場合、残りの数字が落とされて、不正確な結果になってしまいます。

入出力操作中のエラー処理

エラー処理は、入出力ステートメントの処理中に、他の方法では通知されない可能性のある重大エラーを捕らえるのに役立ちます。入出力操作においては、重要なエラー処理のための句と文節がいくつかあります。これらは以下のとおりです。

- AT END 句
- INVALID KEY 句
- NO DATA 句
- USE AFTER EXCEPTION/ERROR 宣言型プロシージャー
- FILE STATUS 文節

入出力操作中に、システムはエラーを検出してメッセージを送ります。続いて ILE COBOL がこのメッセージをモニターします。さらに、ILE COBOL はシステムがサポートしない入出力操作においても、いくつかのエラーを検出します。入出力操作中にエラーを検出する方法には関係なく、結果は常にゼロ以外の内部ファイル状況、実行時メッセージ、またはその両方になります。

エラー処理の重要な特徴は、入出力ステートメントの AT END/INVALID KEY 句、USE AFTER EXCEPTION/ERROR プロシージャー、または SELECT ステートメントの FILE STATUS 文節がファイルにない場合に、入出力ステートメントの処理中にエラーが発生すると、実行時メッセージを出されることです。

入出力エラーについては、重大でない入出力エラーが起こった後に、プログラムの実行を続行するかしないかを選択できることに注意してください。 ILE COBOL は訂正アクションを実行しません。エラー処理コードをプログラム設計に挿入することによりプログラムの続行を選択する場合は、適切なエラー・リカバリー手順もコーディングしなければなりません。

入出力ステートメントと特に関係のあるエラー処理の句や文節に加えて、ユーザー定義の ILE 異常事態処理ルーチンおよび ILE COBOL エラー処理 API を使用して入出力エラーを処理することもできます。

ILE COBOL は、入出力ステートメントごとに異常事態処理ルーチンを登録して、さまざまな入出力関連条件をキャッチします。これらの異常事態処理ルーチンは優先順位 185 で登録されているので、ユーザー定義の異常事態処理ルーチンが最初に制御を受け取ります。

前述のように、ファイルに対して該当する AT END、INVALID KEY、USE プロシージャー、または FILE STATUS 文節がない場合にエラー発生すると、ILE COBOL の実行時メッセージが出されます。メッセージ LNR7057 はエスケープ・メッセージです。このメッセージは、ユーザー定義の異常事態処理ルーチンで処理できます。このメッセージを処理できる異常事態処理ルーチンがない場合は、メッセージ LNR7057 が機能チェックとして再び送られます。

ILE COBOL には機能チェック異常事態処理ルーチンがあり、ILE COBOL エラー処理 API が定義されていない場合には、それによって最終的に照会メッセージ LNR7207 が出されます。

入出力 verb の処理

次の図は、USE プロシージャーと (NOT) AT END、(NOT) INVALID KEY、および NO DATA 命令ステートメントがいつ実行されるのかを示すものです。

この図でファイル状況とは、内部ファイル状況のことです。

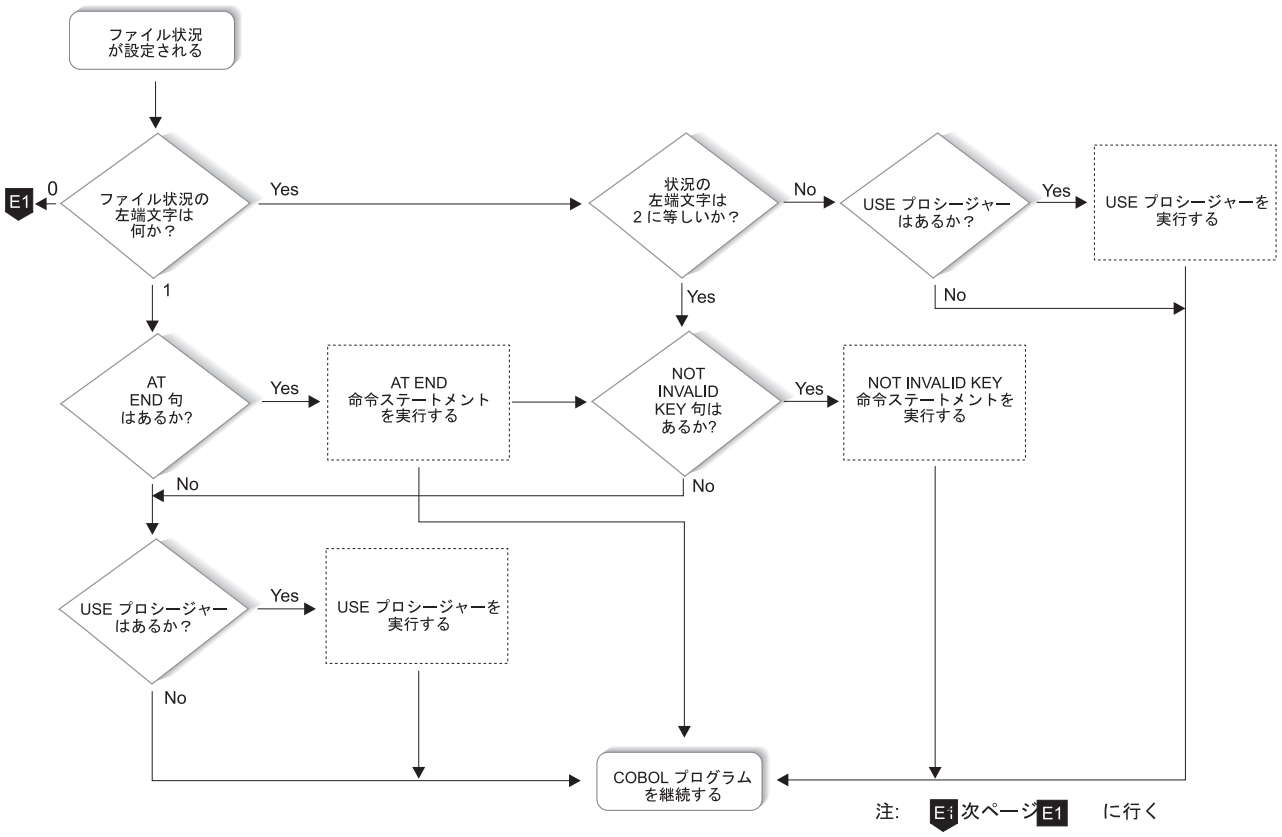


図 90. 入出力 verb の処理 (1/2)

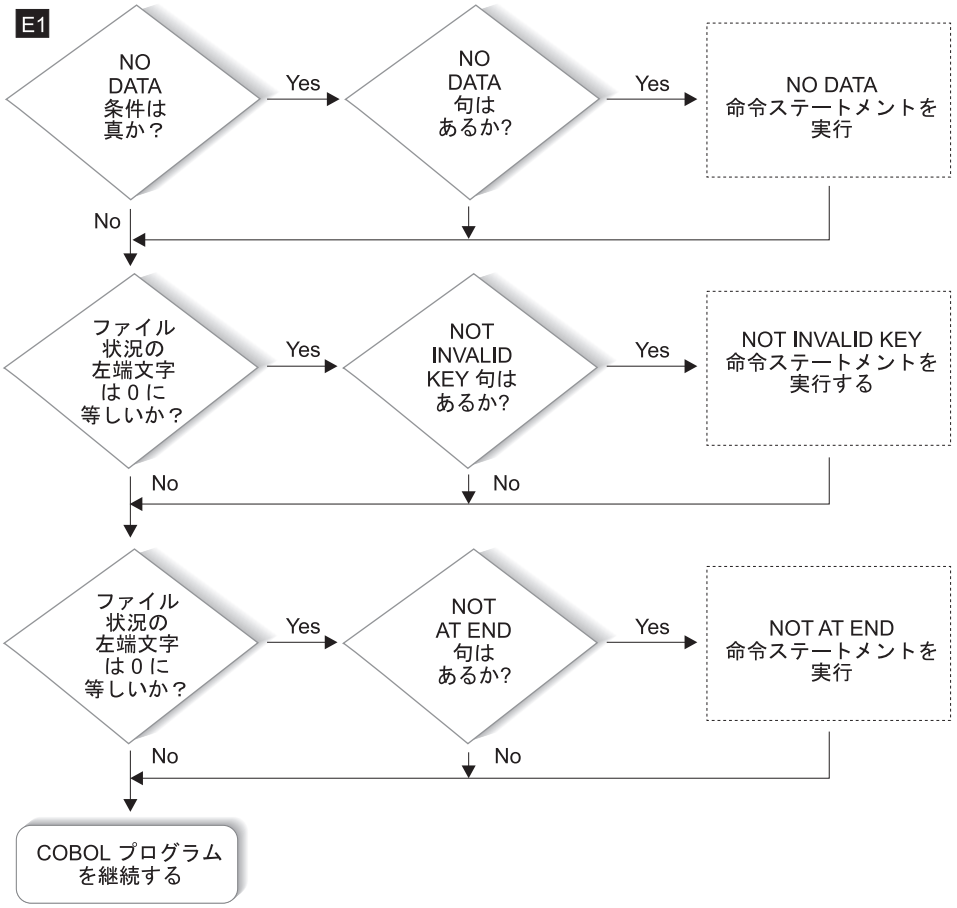


図 90. 入出力 verb の処理 (2/2)

注: 図の中で、実際のステートメントに適用できる部分に従ってください。

ファイル終了条件の検出 (AT END 句)

ファイル終了条件は、エラーである場合とそうでない場合があります。多くの設計では、ファイル終了は意図的に順次読み取られ、AT END 条件が予期されています。

しかし、ファイル終了条件がエラーを反映する場合もあります。 READ ステートメントの AT END 句をコーディングすることにより、プログラム設計に基づいて、それぞれの場合の処理を行います。

AT END 句をコーディングすると、ファイル終了条件が生じた時点で、この句で識別される命令ステートメントが実行されます。 AT END 句をコーディングしない場合には、関連する USE AFTER EXCEPTION/ERROR 宣言が実行されます。

コーディングした NOT AT END 句が実行されるのは、READ ステートメントが正常に完了した場合だけです。ファイル終了以外の条件で READ 操作が失敗した場合、AT END 句と NOT AT END 句は両方とも実行されません。その代わり、関連する USE AFTER EXCEPTION/ERROR 宣言型プロシージャーが実行された後で READ ステートメントの最後に制御が渡されます。

AT END 句も USE AFTER EXCEPTION/ERROR 宣言型プロシージャーもコーディングしておらず、かつファイルに STATUS KEY 文節をコーディングしている場合は、ファイル終了条件を検出した入出力ステートメントの後にある次の順次命令に制御が渡されます。この時点で状況キーを参照し、該当するアクションを行ってエラーを処理するようにコーディングする必要があります。

無効キー条件の検出 (INVALID KEY 句)

索引キーや相対キーの障害のために入出力エラーが起こると、INVALID KEY 句で識別される命令ステートメントに制御が与えられます。 INVALID KEY 句は、索引ファイルや相対ファイルの READ、START、WRITE、REWRITE、および DELETE ステートメントに組み込めます。

INVALID KEY 句は、次の点で USE AFTER EXCEPTION/ERROR 宣言と異なります。

- INVALID KEY 句は限られたタイプのエラーについて操作を行うが、USE AFTER EXCEPTION/ERROR 宣言はほとんどの形式のエラーで有効。
- INVALID KEY 句は入出力 verb に直接コーディングされるが、USE AFTER EXCEPTION/ERROR 宣言部分は個別にコーディングされる。
- INVALID KEY 句は 1 つの入出力操作に特有だが、USE AFTER EXCEPTION/ERROR 宣言部分の方が一般的。

無効キー条件になる入出力ステートメントに INVALID KEY 句を指定すると、INVALID KEY 句で識別される命令ステートメントに制御が転送されます。この場合、コーディングした USE AFTER EXCEPTION/ERROR 宣言部分は実行されません。

指定した NOT INVALID KEY 句は、ステートメントが正常に完了した場合に限り実行されます。無効キー以外の条件で操作が失敗した場合、INVALID KEY 句と NOT INVALID KEY 句は両方とも実行されません。その代わり、関連する USE AFTER EXCEPTION/ERROR 宣言部分が実行された後で、入出力ステートメントの最後に制御が渡されます。

状況キーを評価し、特定の無効キー条件を調べるには、INVALID KEY 句と一緒に FILE STATUS 文節を使用してください。

たとえば、得意先マスター・レコードを含むファイルがあり、トランザクション更新ファイル中の情報を使用してこのレコードの一部を更新する必要がある場合を考えてみます。各トランザクション・レコードを読み取り、マスター・ファイル中で対応するレコードを検出し、必要な更新を行います。 2 つのファイルにはレコードごとに得意先番号のフィールドがあり、マスター・ファイルにはレコードごとに固有の得意先番号があります。

通勤者 (COMMUTER) レコードのマスター・ファイルの FILE-CONTROL 項目には、索引付き編成およびランダム・アクセスを定義し、基本レコード・キーとして MASTER-COMMUTER-NUMBER、またファイル状況キーとして COMMUTER-FILE-STATUS を定義するステートメントが含まれています。次の例では、INVALID KEY 句と FILE STATUS 文節とを使用して、入出力ステートメントのエラーの原因をより明確に判別する方法を示します。

```
.
. (read the update transaction record)
.
MOVE "TRUE" TO TRANSACTION-MATCH
MOVE UPDATE-COMMUTER-NUMBER TO MASTER-COMMUTER-NUMBER
READ MASTER-COMMUTER-FILE INTO WS-CUSTOMER-RECORD
   INVALID KEY
      DISPLAY "MASTER CUSTOMER RECORD NOT FOUND"
      DISPLAY "FILE STATUS CODE IS: " COMMUTER-FILE-STATUS
      MOVE "FALSE" TO TRANSACTION-MATCH
END-READ
```

EXCEPTION/ERROR 宣言型プロシージャーの使用 (USE ステートメント)

ILE COBOL プログラムには、1 つまたは複数の USE AFTER EXCEPTION/ERROR 宣言型プロシージャーをコーディングできます。入出力エラーが発生すると、このプログラムに制御が与えられます。次のものを使用できます。

- ファイル・オープン・モード (INPUT、OUTPUT、I-O、または EXTEND) ごとに個別のプロシージャー
- 特定のファイルごとに個別のプロシージャー
- ファイルのグループに個別のプロシージャー

プログラムの PROCEDURE DIVISION の宣言セクションに、上記の各プロシージャーを入れます。宣言作成の詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

プロシージャーでは、訂正アクションの試行、操作の再試行、続行、またはプログラムの終了を選択できます。エラーをさらに分析したい場合は、USE AFTER EXCEPTION/ERROR 宣言型プロシージャーと状況キーを組み合わせて使用できます。

GLOBAL ファイルの場合は、ILE COBOL プログラムごとに独自の USE AFTER EXCEPTION/ERROR 宣言型プロシージャーを使用できます。

USE AFTER EXCEPTION/ERROR 宣言自体を、GLOBAL として宣言できます。入出力エラー発生時に複数の宣言部分が実行される場合は、特別な優先順位の規則に従います。この規則が適用されると、条件にかなう最初の宣言だけが実行用に選択されます。選択される宣言は、宣言の実行に関する規則に従っていなければなりません。宣言選択においての優先順位は次のとおりです。

1. プログラム中のファイル特有の宣言 (形式は USE AFTER ERROR ON *ファイル名-1*) のうち、修飾条件となるステートメントを含むもの
2. プログラム中のモード特有の宣言 (形式 USE AFTER ERROR ON INPUT のもの) のうち、修飾条件となるステートメントを含むもの
3. ファイル特有の宣言のうち、GLOBAL 句を指定していて、かつ修飾条件を検査されたプログラムを直接含むプログラム中のもの
4. モード特有の宣言のうち、GLOBAL 句が指定されていて、かつ修飾条件を検査されたプログラムを直接含むプログラム中のもの
5. プログラムがネストしている場合、親に対して、規則 3 と 4 が再帰的に適用されます。

エラーが発生した後に制御をプログラムに戻したい場合は、USE AFTER EXCEPTION/ERROR 宣言型プロシージャーを作成してください。このプロシージャーを作成しないなら、エラーが発生した後にジョブが取り消されたり異常終了したりすることがあります。

各 USE AFTER EXCEPTION/ERROR 宣言型プロシージャーは、同一の ILE COBOL プログラムの他の宣言型プロシージャーや宣言以外の部分の呼び出しとは別個の呼び出しとして実行されます。したがって、CEEHDLR API を呼び出して宣言型プロシージャーから ILE 異常事態処理ルーチンを登録すると、その ILE 異常事態処理ルーチンは USE AFTER EXCEPTION/ERROR 宣言型プロシージャーで例外が発生した場合に限り呼び出され、ILE COBOL プログラムの他の部分で例外が発生しても呼び出されません。

ファイル状況キーによるエラーのタイプの判別

ファイル状況キーの更新は、ファイルに対する個々の入出力操作の後で、2 桁のファイル状況キーに値を入れることによって行われます。通常は、先頭桁がゼロの場合は操作が成功したことを示し、2 桁ともゼロの場合は「異常が報告されなかった」ことを意味します。

FILE-CONTROL 項目を使用することにより、ILE COBOL プログラムで使用する各ファイルの編成とアクセス方式を指定しなければなりません。また、この項目には FILE STATUS 文節もコーディングできます。

FILE STATUS 文節は、入出力操作の結果のコピーを入れるための 1 つまたは 2 つのデータ項目 (WORKING-STORAGE セクションにコーディングされているもの) を指定します。 1 つ目の項目のコピーは外部ファイル状況と呼ばれます。TRANSACTION ファイルを使用する場合は、もう 1 つの結果レコードがあります。これは外部戻りコードと呼ばれ、外部メジャー戻りコードと外部マイナー戻りコードとで構成されます。

ILE COBOL は、これら 2 つのデータ項目に対応する情報を ILE COBOL ファイル・フィールド記述子 (FFD) に保持します。 ILE COBOL のこれら 2 つのデータ項目のコピーは、それぞれ内部ファイル状況と内部戻りコードと呼ばれます。この章では、*ファイル状況*、および *(メジャー / マイナー) 戻りコード* は、特に指定がない限り ILE COBOL のコピーのことです。

入出力ステートメントの処理中に、次の 3 つの方法のどれかでファイル状況を更新できます。ファイル状況の内容によって、エラー処理プロシージャーを実行するかどうかが判別されます。

エラー処理プロシージャーは、入出力操作が失敗した後に制御を握ります。これはゼロ以外のファイル状況で示されます。 これらのプロシージャーが実行される前に、ファイル状況は外部ファイル状況にコピーされます。

ファイル状況は次の 3 つの方法のいずれかで設定されます。

- A 方式 (すべてのファイル)

  ILE COBOL は、ファイル制御ブロック中の変数の内容を調べます。内容が予期したものと異なる場合は、ゼロ以外のファイル状況が設定されます。このように

して設定されるほとんどのファイル状況は、ILE COBOL ファイル・フィールド記述子 (FFD) およびシステム・ユーザー・ファイル制御ブロック (UFCB) の検査結果です。

- B 方式 (トランザクション・ファイル)

  ILE COBOL は、システムからのメジャーおよびマイナー戻りコードを調べます。メジャー戻りコードがゼロでない場合、メジャー戻りコードとマイナー戻りコードで構成される戻りコードがファイル状況に変換されます。メジャー戻りコードがゼロの場合、ファイル状況は A 方式または C 方式で設定されている可能性があります。

  サブファイルの READ、WRITE、および REWRITE 操作の場合は、A 方式と C 方式だけが適用されます。

  戻りコードのリストと関連するファイル状況については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の『ファイル構造サポートの要約および状況キー値』を参照してください。

- C 方式 (すべてのファイル)

  ILE COBOL がデータ管理機能を呼び出して入出力操作を実行すると、システムからメッセージが送られます。続いて、ILE COBOL がそのメッセージをモニターし、それに応じてファイル状況を設定します。各 ILE COBOL 入出力操作は、サービス・プログラムの中の、ILE COBOL コンパイラーによって提供されるルーチンで処理されます。次にこのルーチンは、データ管理機能を呼び出して、入出力操作を実行します。ほとんどの場合、サービス・プログラム中のルーチンに対する呼び出しの前後で単一のメッセージ・モニターが有効にされます。

  各入出力操作のメッセージ・モニターは、典型的な入出力例外を処理し、「メッセージ・モニターは、受け取った CPF メッセージに基づいてファイル状況を設定します」 という意味の CPF メッセージを出します。メッセージ・モニターが処理するメッセージのリストについては、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の『ファイル構造サポートの要約および状況キー値』を参照してください。

  この方法でメッセージ・モニターを使用すると、プログラム中のその他の入出力操作タイプには関係なく、入出力操作の種類ごとに一貫したファイル状況が設定されます。メッセージ・モニターの詳細については 432 ページの『異常事態処理ルーチンによるメッセージ処理』を参照してください。

MAP 0010: ファイル状況の設定方法

001

– 入出力操作開始。
– 内部ファイル状況をリセット。
– A 方式: ファイル制御ブロック中の変数の内容を検査。 (たとえばファイルが適切にオープンされているかどうかを検査。)

ファイル制御ブロック中の変数は予期されているとおりに設定されているか?
Yes No

002

– エラー発生を示す内部ファイル状況を設定する。
– ステップ 006 に進む。

003

– データ管理機能を呼び出して入出力操作を実行する。

データ管理機能は例外を戻すか?
Yes No

004

– A 方式: ファイル制御ブロック中の変数の内容を検査。

ファイル制御ブロック中の変数は予期されているとおりに設定されているか?
Yes No

005

– エラー発生を示す内部ファイル状況を設定する。
– ステップ 006 に進む。

006

– 内部ファイル状況を、ファイル状況文節に指定されている外部ファイル状況に移動。
– 内部ファイル状況に基づいてエラー処理コードを実行する。

007

ファイルはトランザクション・ファイルか?
Yes No

008

– C 方式: データ管理機能から送られる CPF メッセージに従って内部ファイル状況を設定する。

– 431 ページのステップ 004 に進む。

009

システムからのメジャー戻りコードとマイナー戻りコードは使用可能か?
Yes No

010

– C 方式: データ管理機能から送られる CPF メッセージに従って内部ファイル状況を設定する。
– 431 ページのステップ 004 に進む。

011

– B 方式: システムから使用可能なメジャー戻りコードとマイナー戻りコードに基づいて内部ファイル状況を設定する。
– 431 ページのステップ 004 に進む。

メジャー戻りコードとマイナー戻りコードの解釈

プログラム中に TRANSACTION ファイルを指定すると、SELECT ステートメントの FILE STATUS 文節に、外部ファイル状況と、外部 (メジャーおよびマイナー) 戻りコードの 2 つのデータ名を含めることができます。 429 ページの『ファイル状況キーによるエラーのタイプの判別』に示されているように、ファイル状況は、3 つの方法のいずれかで設定できます。しかし、トランザクション入出力によりデータ管理が呼び出された後は、システムによって戻りコードが設定されます。その結果、システム・メッセージが出されるほとんどのエラー状態にも、戻りコードが対応しています。

戻りコードはファイル状況値と似ています。すなわち、システムから送られる CPF メッセージは、ILE COBOL 実行時例外ハンドラーによってグループにまとめられ、CPF メッセージの各グループを使用して 1 つまたは複数のファイル状況が設定されます。同様に、各メジャー戻りコードも、CPF メッセージのグループによって生成されます。 (マイナー戻りコードは必ずしも同じではありません。) ファイル状況と戻りコードの主な違いは、CPF メッセージのグループ化に関する違いです。

ILE COBOL は TRANSACTION ファイルの戻りコードだけを設定しますが、他のタイプのファイル (印刷出力ファイルなど) も戻りコードは設定されます。それらのファイルの戻りコードには、I-O-FEEDBACK 操作から ACCEPT を使用してアクセスします。

異常事態処理ルーチンによるメッセージ処理

異常事態処理ルーチンは、システムか他の ILE プロシージャーまたはプログラム・オブジェクトから送られるメッセージを、ILE プロシージャーまたはプログラム・オブジェクトで処理するための手段を提供します。 1 つの異常事態処理ルーチンで 1 つまたは複数のメッセージを処理できます。

異常事態処理ルーチンはある点で USE プロシージャーに似ています。 USE プロシージャーが入出力エラーに応じて行うアクションを指定する方法と同様の方法で、

異常事態処理ルーチンはマシン・インターフェース (MI) 命令の処理中にエラーが発生した際に行うアクションを指定します。 MI 命令エラーはシステム・メッセージによって通知され、各 ILE COBOL ステートメントは 1 つまたは複数の MI 命令で構成されます。

異常事態処理ルーチンには次の 2 つのタイプがあります。

- 異常事態処理ルーチンの 1 つのタイプは、プログラム全体について活動状態になっているものです。この種の異常事態処理ルーチンは、総称エラー状態を処理するように設計されています。
- もう 1 つのタイプの異常事態処理ルーチンは、ステートメントごとに活動状態になります。この種の異常事態処理ルーチンの典型的な使用方法は、入出力操作のモニターです。この種の異常事態処理ルーチンは、ファイル状況を設定し、SIZE ERROR、END-OF-PAGE、および OVERFLOW 条件を示します。

ソート・マージ操作のエラー処理

SORT または MERGE 操作中のエラーを検出するには、SORT-RETURN 特殊レジスターを使用します。 SORT-RETURN 特殊レジスターには、SORT 操作や MERGE 操作が成功したか失敗したかを示す戻りコードが入れられます。SORT-RETURN 特殊レジスターの戻りコードは、操作が成功した場合は 0、操作が失敗した場合は 16 です。

エラー宣言または入出力プロシージャー中で SORT-RETURN 特殊レジスターを 16 に設定することによって、すべてのレコードの処理が終わる前に SORT/MERGE 操作を終了することができます。レコードが戻されるか解放される前に、操作が終了します。

SORT-RETURN 特殊レジスターには、次のような暗黙の定義があります。

```
01     SORT-RETURN  GLOBAL   PIC S9(4) USAGE BINARY VALUE ZERO.
```

ネストされたプログラムで使用する場合、SORT-RETURN 特殊レジスターは、最外部の ILE COBOL プログラムで暗黙のうちに GLOBAL として定義されます。

SORT-RETURN 特殊レジスターの詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の SORT および MERGE ステートメントの部分を参照してください。

CALL ステートメントの例外処理

CALL 操作自体が失敗した場合には、CALL ステートメントに関する例外条件が生じます。たとえば、システムがストレージ不足か、または呼び出し先プログラムを見つけられない場合があります。この場合、CALL ステートメントに ON EXCEPTION 文節または ON OVERFLOW 文節がないと、アプリケーションが異常終了することがあります。 ON EXCEPTION 文節か ON OVERFLOW 文節を使用することによって、例外条件を検出して、異常終了しないようにし、独自のエラー処理ルーチンを実行することができます。たとえば、

```
CALL "REPORTA"
   IN LIBRARY "MYLIB"
      ON EXCEPTION
          DISPLAY "Program REPORTA not available."
END-CALL
```

プログラム REPORTA が使用できない場合、またはライブラリー MYLIB にない場合は、ON EXCEPTION 文節で制御が続行します。

ON EXCEPTION 句および ON OVERFLOW 句は、CALL 操作自体のエラーによる例外しか処理しません。

CALL 操作によって通知される ON EXCEPTION 条件は、優先順位 130 で登録されている異常事態処理ルーチンによって処理されます。この優先順位では、CALL ステートメントの存在する特定の呼び出しスタック項目に通知される条件だけが処理されます。この優先順位では、ユーザー作成の異常事態処理ルーチンが、通知された条件を参照する機会がなくなることがあります。

アプリケーションの中の CALL ステートメントに ON EXCEPTION 句および ON OVERFLOW 句がない場合に、CALL ステートメントが失敗すると、例外は ILE 条件処理で処理されます。 ILE 条件処理の概要については 415 ページの『ILE 条件処理』を参照してください。

ユーザー作成のエラー処理ルーチン

ON EXCEPTION 句、ON SIZE ERROR 句、および他の ILE COBOL 言語のセマンティクスを使用すると、プログラム実行時に発生する可能性のあるほとんどのエラー状態を処理できます。しかし、マシン・チェックなどの異常なエラー状態が発生すると、ILE COBOL は照会メッセージを出し、それにより重大エラーが発生した後に行う必要のあるアクションを判別できます。

しかし、ILE および ILE COBOL には、ユーザー作成の ILE 異常事態処理ルーチンを使用するメカニズムが用意されています。このメカニズムにより、照会メッセージを出す前に異常なエラー状態を処理できます。 ILE 異常事態処理を使用すると、独自のエラー処理ルーチンを作成してエラー状態を処理することにより、プログラムの実行を続行することができます。

ユーザー作成の異常事態処理ルーチンの優先順位は 165 です。この優先順位を設定されたユーザー作成異常事態処理ルーチンは、通知された条件を、入出力異常事態処理ルーチンまたは ILE デバッガー異常事態処理ルーチンより前に参照できます。

ILE がユーザー作成エラー処理ルーチンに制御を渡すようにするには、まず入り口点を識別して ILE に登録しなければなりません。例外ハンドラーを登録するには、プロシージャー・ポインターをユーザー作成異常事態処理ルーチン登録 (CEEHDLR) のバインド可能 API に渡します。 ILE COBOL プログラムを例外ハンドラーとして使用する場合、登録できるのは最外部の ILE COBOL プログラムだけです。 ILE COBOL では、非再帰的プログラムの場合、再帰が不可能なので、ILE COBOL プログラムを例外ハンドラーとして登録する場合は、このプログラムを、必ず活動化グループ中で 1 回だけしか呼び出せないようにするか、再帰的プログラムにする必要があります。

例外ハンドラーの詳細については、「*ILE 概念*」を参照してください。 プロシージャー・ポインター・データ項目を使用すると、プロシージャー入り口点の入り口アドレスを ILE サービスに渡せます。プロシージャー・ポインター・データ項目の詳細については 402 ページの『プロシージャー・ポインターによる入り口点アドレスの受け渡し』を参照してください。ユーザー作成異常事態処理ルーチンはいくつでも登録できます。複数のユーザー作成異常事態処理ルーチンを登録すると、後入れ先出し法 (LIFO) の順序で制御がハンドラーに渡されます。

ユーザー作成異常事態処理ルーチン登録抹消 (CEEHDLU) API を使用して、ユーザー作成異常事態処理ルーチンを登録取り消しすることもできます。

一般的な例外とそのいくつかの原因

MCH1202 10 進数データ・エラー

- それ以前に有効なデータが数値基本項目に保管されていないのに、その項目がソースとして使用された。 この項目に VALUE 文節を加えるか、または MOVE ステートメントを使用して値を初期設定する必要があります。
- 非数値データを数字項目に入れようとした。
- それ以前に不良データがプログラム中のサブファイルに書き込まれた。 サブファイルのデータは表示画面に書き込まれるまで妥当性検査されません。したがって、サブファイル制御レコードの WRITE の時点で 1202 エラーが発生することがありますが、実際は不良データはサブファイルにそれ以前に挿入されていました。

MCH0601 ポインター例外

- LINKAGE SECTION (リンケージ・セクション) の一部が、割り振られているスペースを超えて拡張された。

  たとえば、LINKAGE SECTION の項目のアドレスを設定し、その基本データ項目に MOVE を使用したために 1 つまたは複数の項目が上記のスペースを超えて拡張されると、MCH0601 が出されます。

  ポインターの使用については 377 ページの『第 14 章 ILE COBOL プログラムでのポインターの使用』を参照してください。

MCH0602 ポインター位置合わせ

- 呼び出し側プログラムの WORKING-STORAGE SECTION でのポインター位置合わせが、呼び出し先プログラムの LINKAGE SECTION での位置合わせと一致していない。 16 バイトの境界に位置合わせしなければなりません。

  ポインターの使用については 377 ページの『第 14 章 ILE COBOL プログラムでのポインターの使用』を参照してください。

MCH0603 範囲検査エラー

- 添え字の値が配列の下限より小さいか、配列の上限より大きいか、または文字ストリングを定義した複合オペランドによって基本文字ストリングの範囲外に文字ストリングが定義された。

MCH3601 ポインター・エラー

- レコードまたはレコード中のフィールドに対する参照が行われたが、関連ファイルがクローズされているか、または一度もオープンされていない。

  たとえば、ファイルの OPEN が失敗したが、そのファイルに対してその他の入出力ステートメント処理を実行しようとした場合があります。ファイル状況を調べてから他の入出力を試行する必要があります。

CPF2415 要求の終わり

- ジョブ入力ストリームからの入力を受け入れようとしたが、システムはバッチ・モードで実行しているので入力を使用できない。

障害の後でのリカバリー

障害の後に何らかのリカバリーを行うことができます。次の 2 つの分野でこのリカバリーを行うことができます。

- コミットメント制御によるファイルのリカバリー
- TRANSACTION ファイルのリカバリー

コミットメント制御によるファイルのリカバリー

障害の後にシステムを再始動すると、コミットメント制御下のファイルは、最後のコミットメント境界での状況に自動的に復元されます。コミットメント制御の詳細については 469 ページの『コミットメント制御の使用』を参照してください。

コミットメント制御は、活動化グループ・レベルおよびジョブ・レベルの 2 つのレベルに範囲を限定できます。詳細については、「*ILE 概念*」の『コミットメント制御範囲指定』を参照してください。

ジョブまたは活動化グループが (ユーザー・エラーまたはシステム・エラーが原因で) 異常終了すると、コミットメント制御下のファイルは、ジョブの終了や活動化グループの終了の処理の一部として、最後のコミットメント境界でのそのファイルの状況に復元されます。コミットメント制御境界は、プログラムで選択したコミットメント制御の有効範囲に基づいて決まります。

コミットメント制御下のファイルはシステムや処理の失敗後にロールバックされるので、この機能を使用して再始動に役立てることができます。ジョブを再始動する必要が生じた場合に役立つ可能性のあるデータを保管するためのレコードを別個に作成することができます。この再始動データには、合計、カウンター、レコード・キー値、相対キー値、またアプリケーションからのその他の関連処理情報などの項目を入れることができます。

コミットメント制御下のファイルに上記の再始動データを保持しておくなら、COMMIT ステートメントを出した時点で、この再始動データもデータベースに永続的に保管されます。ジョブや処理の失敗後に ROLLBACK が行われたなら、エラーより前に正常に行われた処理の範囲でレコードを検索できます。上記の方式はプログラミング技法の一提案に過ぎず、常に適切というものではなく、アプリケーションに応じて異なることに注意してください。

TRANSACTION ファイルのリカバリー

オペレーターが介入したり、ワークステーションや通信装置をオン / オフに構成変更したりすることなく、TRANSACTION ファイルの入出力エラーを回復できる場合もあります。

TRANSACTION ファイルに対して潜在的にリカバリー可能な入出力エラーが発生した場合、アプリケーション・プログラム中で取る必要のあるリカバリー・ステップのほかにも、システムはエラー・リカバリーを試行するためのアクションを開始します。システムが行うアクションの詳細については、「*Communications Management*」を参照してください。

入出力操作後にファイル状況を調べることによって、アプリケーション・プログラムは TRANSACTION ファイルの入出力エラーからリカバリーできるかどうかを判別できます。ファイル状況キーの値が 9N の場合、アプリケーション・プログラムは入出力エラーから回復できます。リカバリー手順は、アプリケーション・プログラムの一部としてコーディングしなければならず、TRANSACTION ファイルが装置を 1 つ獲得したのか、それとも複数の装置を接続したのかによって異なります。

装置を 1 つ獲得したファイルでは、以下のとおりです。

1. 入出力エラーが発生した TRANSACTION ファイルをクローズする。
2. ファイルを再オープンする。
3. 失敗した操作を再試行するのに必要なステップを処理する。使用されているプログラム装置のタイプによっては、これに複数のステップが関係してくる場合があります。 (たとえば、最後の入出力操作が READ だった場合は、READ ステートメントを処理する前に処理された 1 つまたは複数の WRITE ステートメントを繰り返す必要が生じることがあります。) リカバリー手順の詳細については、「*ICF Programming*」を参照してください。

複数の装置を獲得したディスプレイ・ファイルでは、以下のとおりです。

1. TRANSACTION ファイルに関する入出力エラーを発生させたプログラム装置を DROP する。
2. その同じプログラム装置を ACQUIRE する。
3. 上記のステップ 3 を参照。

複数の装置を獲得した ICF ファイルでは、以下のとおりです。

1. その同じプログラム装置を ACQUIRE する。
2. 上記のステップ 3 を参照。

複数の装置を獲得したディスプレイ・ファイルでは、以下のとおりです。

アプリケーション・プログラムのリカバリーの試行は、通常は 1 回だけでなければなりません。

リカバリーの試行が失敗した場合は、以下のとおりです。

- ファイルに接続されているプログラム装置が 1 つだけの場合、STOP RUN、EXIT PROGRAM、または GOBACK ステートメントを処理してプログラムを終了し、エラーの原因を突きとめる。

- ファイルが獲得したプログラム装置が複数である場合は、次のどちらかを行うことができます。
  - TRANSACTION ファイルに関する入出力エラーを発生させたプログラム装置がない状態で処理を継続し、後でその装置を再び獲得する。
  - プログラムを終了する。

TRANSACTION ファイルの入出力エラーを診断する際に役立つメジャー戻りコードとマイナー戻りコードについては、「*ICF Programming*」を参照してください。

439 ページの図 92 に、エラー・リカバリー手順の例を示します。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A*  DISPLAY FILE FOR ERROR RECOVERY EXAMPLE
     A*
     A                                      INDARA
     A          R FORMAT1                   CF01(01 'END OF PROGRAM')
     A*
     A                                  12 28'ENTER INPUT '
     A            INPUTFLD       5   I 12 42
     A                                  20 26'F1 - TERMINATE'
```

図 91. エラー・リカバリー手順の例 -- DDS

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/RECOVERY       ISERIES1   06/02/15 13:48:21        ページ    2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.  RECOVERY.
    3     000300 ENVIRONMENT DIVISION.
    4     000400 CONFIGURATION SECTION.
    5     000500 SOURCE-COMPUTER. IBM-ISERIES.
    6     000600 OBJECT-COMPUTER. IBM-ISERIES.
    7     000700 INPUT-OUTPUT SECTION.
    8     000800 FILE-CONTROL.
    9     000900     SELECT RECOVFILE
   10     001000         ASSIGN TO WORKSTATION-RECVFILE-SI
   11     001100         ORGANIZATION IS TRANSACTION
   12     001200         ACCESS MODE IS SEQUENTIAL
   13     001300         FILE STATUS IS STATUS-FLD, STATUS-FLD-2
   14     001400         CONTROL-AREA IS CONTROL-FLD.
   15     001500     SELECT PRINTER-FILE
   16     001600         ASSIGN TO PRINTER-QPRINT.
          001700
   17     001800 DATA DIVISION.
   18     001900 FILE SECTION.
   19     002000 FD  RECOVFILE.
   20     002100 01  RECOV-REC.
          002200     COPY DDS-ALL-FORMATS OF RECVFILE.
   21    +000001       05  RECVFILE-RECORD PIC X(5).                                    <-ALL-FMTS
         +000002*  INPUT FORMAT:FORMAT1    FROM FILE RECVFILE   OF LIBRARY CBLGUIDE     <-ALL-FMTS
         +000003*                                                                      <-ALL-FMTS
   22    +000004       05  FORMAT1-I     REDEFINES RECVFILE-RECORD.                   <-ALL-FMTS
   23    +000005           06 INPUTFLD             PIC X(5).                          <-ALL-FMTS
         +000006* OUTPUT FORMAT:FORMAT1    FROM FILE RECVFILE   OF LIBRARY CBLGUIDE     <-ALL-FMTS
         +000007*                                                                      <-ALL-FMTS
         +000008*      05  FORMAT1-O     REDEFINES RECVFILE-RECORD.                   <-ALL-FMTS
         +000008*      05  FORMAT1-O     REDEFINES RECVFILE-RECORD.                   <-ALL-FMTS
          002300
   24     002400 FD  PRINTER-FILE.
   25     002500 01  PRINTER-REC.
   26     002600    05  PRINTER-RECORD            PIC X(132).
          002700
   27     002800 WORKING-STORAGE SECTION.
          002900
   28     003000 01  I-O-VERB                    PIC X(10).
   29     003100 01  STATUS-FLD                  PIC X(2).
   30     003200    88  NO-ERROR                   VALUE "00".
   31     003300    88  ACQUIRE-FAILED             VALUE "9H".
   32     003400    88  TEMPORARY-ERROR            VALUE "9N".
   33     003500 01  STATUS-FLD-2                PIC X(4).
   34     003600 01  CONTROL-FLD.
   35     003700    05  FUNCTION-KEY              PIC X(2).
   36     003800    05  PGM-DEVICE-NAME           PIC X(10).
   37     003900    05  RECORD-FORMAT             PIC X(10).
   38     004000 01  END-INDICATOR               PIC 1   INDICATOR 1
          004100                                   VALUE B"0".
   39     004200    88  END-NOT-REQUESTED          VALUE B"0".
   40     004300    88  END-REQUESTED              VALUE B"1".
   41     004400 01  USE-PROC-FLAG               PIC 1
          004500                                   VALUE B"1".
```

図 92. エラー・リカバリー手順の例 (1/3)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/RECOVERY        ISERIES1   06/02/15 13:48:21        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   42     004600    88  USE-PROC-NOT-EXECUTED        VALUE B"0".
   43     004700    88  USE-PROC-EXECUTED            VALUE B"1".
   44     004800 01  RECOVERY-FLAG                PIC 1
          004900                                     VALUE B"0".
   45     005000    88  NO-RECOVERY-DONE            VALUE B"0".
   46     005100    88  RECOVERY-DONE               VALUE B"1".
   47     005200 01  HEADER-LINE.
   48     005300    05  FILLER                     PIC X(60)
          005400                                     VALUE SPACES.
   49     005500    05  FILLER                     PIC X(72)
          005600                                     VALUE "ERROR REPORT".
   50     005700 01  DETAIL-LINE.
   51     005800    05  FILLER                     PIC X(15)
          005900                                     VALUE SPACES.
   52     006000    05  DESCRIPTION                PIC X(25)
          006100                                     VALUE SPACES.
   53     006200    05  DETAIL-VALUE               PIC X(92)
          006300                                     VALUE SPACES.
   54     006400 01  MESSAGE-LINE.
   55     006500    05  FILLER                     PIC X(15)
          006600                                     VALUE SPACES.
   56     006700    05  DESCRIPTION                PIC X(117)
          006800                                     VALUE SPACES.
   57     006900 PROCEDURE DIVISION.
   58     007000 DECLARATIVES.
          007100 HANDLE-ERRORS SECTION.
          007200     USE AFTER STANDARD ERROR PROCEDURE ON RECOVFILE.  1
          007300 DISPLAY-ERROR.
   59     007400     SET USE-PROC-EXECUTED TO TRUE.
   60     007500     WRITE PRINTER-REC FROM HEADER-LINE
          007600          AFTER ADVANCING PAGE
          007700     END-WRITE
   61     007800     MOVE "ERROR OCCURED IN" TO DESCRIPTION OF DETAIL-LINE.
   62     007900     MOVE I-O-VERB TO DETAIL-VALUE OF DETAIL-LINE.
   63     008000     WRITE PRINTER-REC FROM DETAIL-LINE
          008100          AFTER ADVANCING 5 LINES
          008200     END-WRITE
   64     008300     MOVE "FILE STATUS =" TO DESCRIPTION OF DETAIL-LINE.
   65     008400     MOVE STATUS-FLD TO DETAIL-VALUE OF DETAIL-LINE.  2
   66     008500     WRITE PRINTER-REC FROM DETAIL-LINE
          008600          AFTER ADVANCING 2 LINES
          008700     END-WRITE
   67     008800     MOVE "EXTENDED FILE STATUS =" TO DESCRIPTION OF DETAIL-LINE.
   68     008900     MOVE STATUS-FLD-2 TO DETAIL-VALUE OF DETAIL-LINE.
   69     009000     WRITE PRINTER-REC FROM DETAIL-LINE
          009100          AFTER ADVANCING 2 LINES
          009200     END-WRITE
   70     009300     MOVE "CONTROL-AREA =" TO DESCRIPTION OF DETAIL-LINE.
   71     009400     MOVE CONTROL-FLD TO DETAIL-VALUE OF DETAIL-LINE.
   72     009500     WRITE PRINTER-REC FROM DETAIL-LINE
          009600          AFTER ADVANCING 2 LINES
          009700     END-WRITE.
          009800 CHECK-ERROR.
   73     009900     IF TEMPORARY-ERROR AND NO-RECOVERY-DONE THEN
   74     010000         MOVE "***ERROR RECOVERY BEING ATTEMPTED***"  3
```

図 92. エラー・リカバリー手順の例 (2/3)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/RECOVERY       ISERIES1   06/02/15 13:48:21        ページ    4
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
        010100             TO  DESCRIPTION OF MESSAGE-LINE
   75   010200         WRITE PRINTER-REC FROM MESSAGE-LINE
        010300             AFTER ADVANCING 3 LINES
        010400         END-WRITE
   76   010500         SET RECOVERY-DONE TO TRUE
   77   010600         DROP PGM-DEVICE-NAME FROM RECOVFILE
   78   010700         ACQUIRE PGM-DEVICE-NAME FOR RECOVFILE  4
        010800     ELSE
   79   010900         IF RECOVERY-DONE THEN  5
   80   011000             MOVE "***ERROR AROSE FROM RETRY AFTER RECOVERY***"
        011100                 TO  DESCRIPTION OF MESSAGE-LINE
   81   011200             WRITE PRINTER-REC FROM MESSAGE-LINE
        011300                 AFTER ADVANCING 3 LINES
        011400             END-WRITE
   82   011500             MOVE "***PROGRAM ENDED***"
        011600                 TO  DESCRIPTION OF MESSAGE-LINE
   83   011700             WRITE PRINTER-REC FROM MESSAGE-LINE
        011800                 AFTER ADVANCING 2 LINES
        011900             END-WRITE
   84   012000                 CLOSE RECOVFILE
        012100                       PRINTER-FILE
   85   012200                 STOP RUN
        012300         ELSE
   86   012400             SET NO-RECOVERY-DONE TO TRUE
        012500         END-IF
        012600     END-IF
   87   012700     MOVE "***EXECUTION CONTINUES***"
        012800         TO DESCRIPTION OF MESSAGE-LINE.
   88   012900     WRITE PRINTER-REC FROM MESSAGE-LINE
        013000         AFTER ADVANCING 2 LINES
        013100     END-WRITE.
        013200 END DECLARATIVES.
        013300
        013400 MAIN-PROGRAM SECTION.
        013500 MAINLINE.
   89   013600     MOVE "OPEN" TO I-O-VERB.
   90   013700     OPEN I-O     RECOVFILE
        013800          OUTPUT  PRINTER-FILE.
   91   013900     PERFORM I-O-PARAGRAPH UNTIL END-REQUESTED.  6
   92   014000     CLOSE RECOVFILE
        014100           PRINTER-FILE.
   93   014200     STOP RUN.
        014300
        014400 I-O-PARAGRAPH.
   94   014500     PERFORM UNTIL USE-PROC-NOT-EXECUTED OR NO-RECOVERY-DONE  7
   95   014600          MOVE "WRITE" TO I-O-VERB
   96   014700          SET USE-PROC-NOT-EXECUTED TO TRUE
   97   014800          WRITE RECOV-REC FORMAT IS "FORMAT1"
        014900               INDICATOR IS END-INDICATOR
        015000          END-WRITE
        015100     END-PERFORM
   98   015200     MOVE "READ" TO I-O-VERB.
   99   015300     SET USE-PROC-NOT-EXECUTED TO TRUE.
  100   015400     SET NO-RECOVERY-DONE TO TRUE.
  101   015500     READ RECOVFILE FORMAT IS "FORMAT1"
        015600         INDICATOR IS END-INDICATOR  8
        015700     END-READ
  102   015800     IF NO-ERROR THEN
  103   015900         PERFORM SOME-PROCESSING
        016000     END-IF.
        016100
        016200 SOME-PROCESSING.
        016300*   (たとえば、データベース処理などを挿入します。)
        016400
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 92. エラー・リカバリー手順の例 (3/3)

1 RECOVFILE に関する入出力エラーの発生時に行う処理を定義します。

2 問題を診断する際に役立つ情報を出力します。

3 ファイル状況が 9N (一時エラー) で、この入出力操作に関するエラー・リカバリーがこれ以前に試行されていない場合、この時点でエラー・リカバリーが試行されます。

4 リカバリー手順として、入出力エラーの発生したプログラム装置をドロップしてから再獲得します。

5 プログラムのループを避けるため、リカバリーがそれ以前に試行されていた場合、ここでは試行されません。

6 プログラムの主部は、ユーザーが F1 キーを押してプログラムに終了を通知するまで、装置の読み書きを実行します。

7 WRITE 操作が失敗してリカバリーした場合、WRITE が再試行されます。

8 READ 操作が失敗した場合、装置に再書き込みしてから READ を再試行することにより処理が続行されます。

ヌル可能フィールドを使用した操作中のエラーの処理

プログラムでヌル可能フィールドを参照する際、ILE COBOL コンパイラーは、フィールドが実際にヌルかどうかの検査は行いません。ヌル可能として参照されるフィールドに、そのフィールドのヌル・マップおよびヌル・キー・マップのヌル値が実際に入っているかどうか (言い換えれば、0 または 1) の確認は、プログラマーが行わなければなりません。 プログラムでフィールドがヌル可能と定義されても、データベースでヌル可能と定義されていない場合には、ILE COBOL による検査は行われず、フィールド内にあるものは実行時にすべて使用されます。プログラムの初期設定時、外部記述ファイル用のフィールドはゼロに設定されます。プログラム記述ファイルについて、プログラムの初期設定時にヌル可能フィールドがゼロに設定されるかどうかの確認は、プログラマーが行わなければなりません。

ファイルが*ヌル可能*であり、ALWNULL 属性が指定されていない 場合は、ヌル値を持つレコードを読み取ろうとしても、ファイル状況 90 で失敗します。

ファイルが*ヌル可能*ではなく、ASSIGN 文節の ALWNULL 属性が指定されている場合は、データベースからヌル・マップおよびヌル・キー・マップがゼロとして戻されます。また、ヌル・マップおよびヌル・キー・マップは、データベースに渡される際に無視されます。

ロケール操作中のエラーの処理

ILE COBOL のロケールには、以下の 3 つのタイプがあります。

- DEFAULT ロケール
- CURRENT ロケール
- 特定ロケール

特定ロケールは、SPECIAL-NAMES 段落および SET LOCALE ステートメントで参照されます。 SPECIAL-NAMES 段落の特定ロケールの例を、以下に示します。

```
 SPECIAL-NAMES.  LOCALE "MYLOCALE" IN LIBRARY "MYLIB" IS newlocale.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 group-item.
   05  num-edit PIC $99.99 SIZE 8 LOCALE newlocale.
PROCEDURE DIVISION.
    MOVE 40 to num-edit.
```

上記の例では、特定ロケールの簡略名 newlocale が定義されています。この簡略名は、変数 num-edit の定義で使用されます。 簡略名はプログラムで参照されるので、上記のプログラムが初めて呼び出される際に、実行時の ILE COBOL は、ライブラリー MYLIB のロケール MYLOCALE を検出して、それを記憶域にロードしようとします。

i5/OS のロケールは、タイプが *LOCALE のオブジェクトであり、他の i5/OS オブジェクト同様、ライブラリー内に存在し、それに特定の権限を割り当てられています。 COBOL プログラムで定義され、かつ参照されるロケール簡略名は、プログラムが最初に呼び出される際に解決されます。考えられる障害のタイプを、以下に示します。

- ロケールが、指定したライブラリーに存在しない。
- ロケールのライブラリーが存在しない。
- ロケールまたはロケール・ライブラリーに対する権限が十分でない。

これらのタイプの障害は、ほとんどの他の i5/OS オブジェクトで一般的なものです。 上記シナリオはどの場合も、エスケープ・メッセージ (通常は LNR7096) が出されます。 ロケール・オブジェクトは、検出された後、実行時の ILE COBOL によってロードしなければなりません。ロケール・オブジェクトをロードするには、さまざまなスペースの割り振りが必要です。使用できるスペースがない場合は、エスケープ・メッセージ (通常は、LNR7070) が出されます。

SET LOCALE には考えられる形式がいくつかあります。特定のロケールを参照できる 2 つの基本形式を次に示します。

```
 SPECIAL-NAMES.  LOCALE "ALOCALE" IS alocale.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 group-item.
    05 num-edit PIC +$9(4).99 SIZE 10 LOCALE alocale.
* num-edit2 is based on the current locale
    05 num-edit2 PIC +$9(4).99 SIZE 10 LOCALE.
    05 locale-name PIC X(10) VALUE "FRANCE".
    05 locale-lib  PIC X(10) VALUE "MYLIB".
    MOVE 345.67 TO num-edit.
* set the current locale to "ALOCALE" in library "*LIBL".
    SET LOCALE LC_ALL FROM alocale.
    MOVE 678.02 TO num-edit2.

* set the current locale to "FRANCE" in library "MYLIB".
    SET LOCALE LC_ALL FROM locale-name
        IN LIBRARY locale-lib.
    MOVE 678.02 TO num-edit2.
```

最初の形式は SPECIAL-NAMES 段落のロケール簡略名を参照し、前の例の場合同様、プログラムが最初に呼び出される時に解決され、ロードされます。 2 番目の SET ステートメントでは、ロケール名は ID locale-name の内容から選ばれ、ロケールが存在するライブラリーは ID locale-lib の内容から選ばれます。この場合、ロケール・オブジェクトの解決とロードは、SET ステートメントの実行時に行われます。この形式の SET ステートメントで、ロケールが解決できない場合は、エスケープ・メッセージ (通常は LNR7098) が出されます。メッセージが出される理由のタイプは、前述の LNR7096 の場合と同じです。

第 3 部 ILE COBOL 入出力考慮事項

第 17 章 ファイルの定義

この章では次のことについて説明します。
- プログラム記述ファイルの定義
- 外部記述ファイルの定義
- データ記述仕様 (DDS) を使用したファイルの記述
- ILE COBOL プログラムでの外部記述ファイルの使用

ファイル記述のタイプ

i5/OS でのすべての入出力操作にとって重要な要素は、ファイルです。 プログラムが使用する各ファイルの記述は、オペレーティング・システムによって保持されます。オペレーティング・システムに対するファイルの記述には、ファイルのタイプ (データベースやデバイスなど) に関する情報、ファイル内のレコードの長さ、および各フィールドとその属性の記述が含まれています。ファイルは、IDDU、SQL/400® コマンド、または DDS によって、オペレーティング・システムに対してフィールド・レベルで記述されます。 (例えば、CRTPF コマンドを使用して) DDS を指定せずにファイルを作成した場合でも、そのファイルにはまだ 1 つのフィールド記述が含まれています。その単一フィールドの名前は、そのファイルと同じで、そのレコード長は、作成コマンドで指定したレコード長です。

ファイルは、以下の 2 つの方法で定義できます。
- **プログラム記述ファイル**は、プログラマーによって、 ILE COBOL プログラム内のデータ部 (Data Division) にフィールド・レベルで記述されます。
- **外部記述ファイル**の場合、ILE COBOL コンパイラーは、システム上のファイルの記述を使用して、ILE COBOL プログラム内のフィールド・レベルにあるファイルを記述しているデータ部 (Data Division) に ILE COBOL ソースを生成します。ファイルは、プログラムをコンパイルする前に作成する必要があります。

外部記述ファイルおよびプログラム記述ファイルはどちらも、ILE COBOL プログラムの INPUT-OUTPUT SECTION および FILE SECTION 内で定義する必要があります。外部記述ファイルの FILE SECTION 内のレコード記述は、形式 2 の COPY ステートメントで定義されています。抽出されるのは、フィールド・レベルの記述だけです。EXTERNALLY-DESCRIBED-KEY が RECORD KEY として指定されている場合、RECORD KEY を構成しているフィールドも DDS から抽出されます。形式 2 の COPY ステートメントの詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

ファイルが外部記述またはプログラム記述されている場合は、プロシージャー部 (Procedure Division) 内の実際のファイル処理は同じになります。

プログラム記述ファイルの定義

プログラム記述ファイルのレコードおよびフィールドを記述するには、形式 2 の COPY ステートメントを使用せずに、直接 ILE COBOL プログラムの FILE SECTION にレコード記述をコーディングします。

このプログラムを実行するには、あらかじめこのファイルがシステムに存在している必要があります。唯一の例外は、CRTCBLMOD/CRTBNDCBL コマンドで OPTION(*CRTF) を指定して動的ファイル作成を使用する場合です。詳細については、33 ページの『CRTCBLMOD コマンドのパラメーター』の OPTION パラメーターの説明を参照してください。

ファイルを作成するには、ファイル作成コマンドの 1 つを使用します。 DDS は、ファイル作成コマンドで使用することができます。ILE COBOL 索引付きファイルの場合は、キー順アクセス・パスを作成する必要があります。ファイルの作成時に、DDS でキーを指定します。ILE COBOL プログラム内のレコード・キーは、ファイルが作成されたときに定義されたキーに一致する必要があります。これらのキーの値が一致しない場合でもファイル操作を続けることはできますが、間違ったレコード・キーがシステムに渡される可能性があります。間違ったレコード・キーに明らかに正しいキー値が偶然含まれていた場合は、入出力操作は正常に実行されますが、その操作は間違ったデータに対して行われることになります。したがって、データの整合性が損なわれる可能性があります。この問題が発生しないようにするには、可能な場合は外部記述ファイルを使用する必要があります。

外部記述ファイルの定義

ファイルの外部記述には、以下の項目が含まれます。

- レコード・フォーマット仕様。これには、レコード内のフィールドの記述が含まれています。
- アクセス・パス仕様。これは、レコードの検索方法を記述します。

これらの仕様は、外部ファイル記述およびファイルの作成に使用する IBM i コマンドから得られたものです。

外部記述ファイルは、プログラム記述ファイルよりも以下の点で優れています。

- ILE COBOL プログラムのコーディング量が少ない。多くのプログラムが同じファイルを使用する場合は、オペレーティング・システムにフィールドを一度だけ定義して、すべてのプログラムでそれを使用することができます。これによって、そのファイルを使用するプログラムごとに別個にレコード記述をコーディングする必要がなくなります。
- プログラミング・エラーが発生する可能性が低くなる。多くの場合、プログラムは、ファイルのレコード・フォーマットを変更し、次にプログラム内のコーディングをまったく変更せずに、そのファイルを使用するプログラムを再コンパイルすることにより更新できます。
- ファイル記述のレベル検査。ILE COBOL プログラム内のファイル、およびシステム上の実際のファイルの記述のレベル検査は、(LVLCHK(*NO) がファイル作成コマンドまたは指定変更コマンドで指定されていない限り) そのファイルが開かれたときに実行されます。プログラム内のファイルの記述が実際のファイルと一致しない場合は、オープン操作が失敗し、ファイル状況が 39 になります。
- 索引付きファイルの場合、EXTERNALLY-DESCRIBED-KEY が RECORD KEY 文節で指定されている場合は、実際のファイルでもレコード・キーがそのファイルの ILE COBOL プログラムの記述内と同じ位置を占めていることを保証することができます。また、プログラム記述ファイルでは使用できない不連続キーを使用することもできます。

- 改良された文書。同一ファイルを使用するプログラムでは、整合性のあるレコード・フォーマットおよびフィールド名が使用されます。
- 外部記述出力ファイルに対して処理する編集を DDS で指定することができる。

プログラムで外部記述ファイルを使用するには、まずそのファイルを記述するための DDS を作成し、実際のファイルそれ自体を作成します。

データ記述仕様 (DDS) を使用したファイルの記述

データ記述仕様 (DDS) を使用すると、オペレーティング・システムに対してフィールド・レベルでファイルを記述することができます。 DDS では、外部記述ファイル内の各レコード・フォーマットは、固有のレコード・フォーマット名で識別されます。

レコード・フォーマット仕様は、レコード内のフィールド、およびレコード内のフィールドの位置を記述します。フィールドは、レコード内に DDS で指定された順に配置されています。一般にフィールド記述に含まれるのは、フィールド名、フィールド・タイプ (文字、バイナリー、外部 10 進、内部 10 進、内部浮動小数点)、およびフィールド長 (数値フィールド内の小数点以下の桁数を含む) です。フィールド属性は、物理ファイルまたは論理ファイルのレコード・フォーマットで指定するのではなく、フィールド参照ファイルに定義することができます。(450 ページの図 93 を参照。)

レコード・フォーマットのキーは、DDS で指定されます。形式 2 の COPY ステートメントを使用した場合は、フォーマットのキーを DDS に定義する方法を示したソース・プログラム・リストに、コメントのテーブルが生成されます。

さらに、DDS キーワードを使用して以下を行うことができます。

- フィールドに編集コードを指定する (EDTCDE)。
- ファイルに重複キー値を使用できないように指定する (UNIQUE)。
- レコード・フォーマットまたはフィールドにテキスト記述を指定する (TEXT)。

データベース・ファイルに対して有効な DDS キーワードの詳細リストについては、 Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「*プログラミング*」カテゴリーを参照してください。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
      A**FLDREF    DSTREF     DISTRIBUTION APPLICATION FIELDS REFERENCE
      A          R DSTREF                     TEXT('DISTRIBUTION FIELD REF')
      A* COMMON FIELDS USED AS REFERENCE
    1 A            BASDAT         6  0       EDTCDE(Y)
      A                                       TEXT('BASE DATE FIELD')
      A* FIELDS USED BY CUSTOMER MASTER FILE
    2 A            CUST           5          CHECK(MF)
      A                                       COLHDG('CUSTOMER' 'NUMBER')
      A            NAME          20          COLHDG('CUSTOMER NAME')
    3 A            ADDR      R               REFFLD(NAME)
      A                                       COLHDG('CUSTOMER ADDRESS')
      A            CITY      R               REFFLD(NAME)
      A                                       COLHDG('CUSTOMER CITY')
    2 A            STATE          2          CHECK(MF)
      A                                       COLHDG('STATE')
      A            SRHCOD         6          CHECK(MF)
      A                                       COLHDG('SEARCH' 'CODE')
      A                                       TEXT('CUSTOMER NUMBER SEARCH CODE')
    2 A            ZIP            5  0       CHECK(MF)
      A                                       COLHDG('ZIP' 'CODE')
    4 A            CUSTYP         1  0       RANGE(1 5)
      A                                       COLHDG('CUST' 'TYPE')
      A                                       TEXT('CUSTOMER TYPE 1=GOV 2=SCH 3=B+
      A                                       US 4=PT 5=OTH')
    5 A            ARBAL          8  2       COLHDG('ACCTS REC' 'BALANCE')
      A                                       EDTCDE(J)
    6 A            ORDBAL    R               REFFLD(ARBAL)
      A                                       COLHDG('A/R AMT IN' 'ORDER FILE')
      A            LSTAMT    R               REFFLD(ARBAL)
      A                                       COLHDG('LAST' 'AMOUNT' 'PAID')
    7 A                                       TEXT('LAST AMOUNT PAID IN A/R')
      A            LSTDAT    R               REFFLD(ARBAL)
      A                                       COLHDG('LAST' 'DATE' 'PAID  ')
      A                                       TEXT('LAST DATE PAID IN A/R')
      A            CRDLMT         8  2       COLHDG('CUSTOMER' 'CREDIT' 'LIMIT')
      A                                       EDTCDE(J)
      A            SLSYR         10  2       COLHDG('CUSTOMER' 'SALES' 'THIS YEAR')
      A                                       EDTCDE(J)
      A            SLSLYR        10  2       COLHDG('CUSTOMER' 'SALES' 'LAST YEAR')
      A                                       EDTCDE(J)
```

図 93. フィールド参照ファイルの例

このフィールド参照ファイルの例 (図 93) は、CUSMSTL (カスタマー・マスター論理) ファイル (452 ページの図 94 に示されています) によって使用されるフィールドの定義を示しています。フィールド参照ファイルには通常、他のファイルが使用するフィールドの定義も含まれています。以下で、このフィールド参照ファイルのエントリーの一部について説明します。

1 BASDAT フィールドは、キーワード EDTCDE (Y) で示されているように、Y 編集コードで編集されます。このフィールドが ILE COBOL プログラムの外部記述出力ファイルで使用されている場合、COBOL によって生成されたフィールドは、DDS で指定されているデータ型と互換性があります。このフィールドは、レコードが書き込まれると編集されます。このフィールドがプログラム記述出力ファイルで使用されている場合、このファイル内の DDS と互換性を維持するのは、ユーザーの責任です。このファイルの作成時に DDS を使用しない場合は、ILE COBOL プログラム内のフィールドを適切に編集するのもユーザーの責任になります。

2 CHECK(MF) 項目は、ディスプレイ・ワークステーションからこの項目を入

力する場合は、全桁入力必須フィールドであることを指定します。全桁入力必須とは、フィールドの文字はすべて、ディスプレイ・ワークステーションから入力する必要がある、ということです。

3 ADDR および CITY フィールドは、REFFLD キーワードで示されているように、NAME フィールドに指定されているのと同じ属性を共用します。

4 RANGE キーワードは CUSTYP フィールドに対して指定されており、ディスプレイ・ワークステーションからこのフィールドに入力できるのは、1 から 5 の範囲の数字だけであることを保証します。

5 COLHDG キーワードは、適用業務開発ツール・セットのツールで使用されている場合は、フィールドの列ヘッドを提供します。

6 ARBAL フィールドは、キーワード EDTCDE(J) で示されているように、J 編集コードで編集されます。

7 テキスト記述 (TEXT キーワード) は、いくつかのフィールドに提供されます。TEXT キーワードは文書化の目的に使用され、さまざまなリストで使用されます。

ILE COBOL プログラムでの外部記述ファイルの使用

形式 2 の COPY ステートメントをコーディングすると、ファイル記述をプログラムに組み込むことができます。外部記述ファイルからの情報は ILE COBOL コンパイラーによって取り出され、ILE COBOL データ構造が生成されます。

以下のページでは、DDS の使用例と、形式 2 の COPY ステートメントを使用した結果生成される ILE COBOL コードを示します。(形式 2 の COPY ステートメントの詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。)

- 452 ページの図 94 は、論理ファイルの DDS を、 453 ページの図 95 は、生成された ILE COBOL コードを示しています。
- 453 ページの図 96 は同じファイルについて説明していますが、ALIAS キーワードを含んでおり、 454 ページの図 97 は、生成された ILE COBOL コードを示しています。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
    1 A** LOGICAL  CUSMSTL   CUSTOMER MASTER FILE
      A                                    2 UNIQUE
      A        3 R CUSREC                    PFILE(CUSMSTP)
      A                                      TEXT('CUSTOMER MASTER RECORD')
      A            CUST
      A            NAME
      A            ADDR
      A            CITY
      A            STATE
      A            ZIP
      A            SRHCOD
      A            CUSTYP
      A            ARBAL
      A            ORDBAL
      A            LSTAMT
      A            LSTDAT
      A            CRDLMT
      A            SLSYR       5
      A            SLSLYR
      A        4 K CUST
```

図 94. 論理ファイルのデータ記述仕様の例

1 カスタマー・マスター物理ファイル (CUSMSTP) を処理するための論理ファイルが定義され、CUSMSTL という名前が付けられています。

2 UNIQUE キーワードは、このファイルには重複キー値を使用できないことを示しています。

3 CUSMSTL ファイルに 1 つのレコード・フォーマット (CUSREC) が定義されています。このファイルは、物理ファイル CUSMSTP を基にしています。

4 CUST ファイルがこのファイルのキー・フィールドとして識別されています。

5 論理ファイルに対してフィールド属性 (長さ、データ・タイプ、小数点以下の桁数など) が DDS で指定されていない場合、それらの属性は、物理ファイル内の対応するフィールドから取得されます。論理ファイルの DDS で指定されているフィールド属性はすべて、物理ファイル内のそれに対応するフィールドの属性を指定変更します。物理ファイル内のフィールドの定義は、フィールド参照ファイルを参照することもできます。フィールド参照ファイルとは、フィールド名とその定義 (サイズやタイプなど) で構成されるデータ記述ファイルを指します。フィールド参照ファイルを使用する場合は、複数のレコード・フォーマットで使用されている同じフィールドは、フィールド参照ファイルに一度だけ定義する必要があります。フィールド参照ファイルの詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベース」カテゴリーの中の『*DB2 Universal Database for AS/400*』セクションを参照してください。

450 ページの図 93 は、データベース・ファイルで使用されるフィールドの属性を定義するフィールド参照ファイルの例を示しています。

```
                          ソ ー ス

STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S コピー名   変更日付


  18     000200 01  CUSTOMER-INVOICE-RECORD.
         000210     COPY DDS-CUSREC OF CUSMSTL.
         +000001*    I-O FORMAT:CUSREC     FROM FILE CUSMSTL    OF LIBRARY TESTLIB           CUSREC
         +000002*                          CUSTOMER MASTER RECORD                           CUSREC
         +000003*           USER SUPPLIED KEY BY RECORD KEY CLAUSE                          CUSREC
  19     +000004       05  CUSREC.                                                          CUSREC
  20     +000005           06 CUST                  PIC X(5).                              CUSREC
         +000006*                 CUSTOMER NUMBER                                          CUSREC
  21     +000007           06 NAME                  PIC X(25).                             CUSREC
         +000008*                 CUSTOMER NAME                                            CUSREC
  22     +000009           06 ADDR                  PIC X(20).                             CUSREC
         +000010*                 CUSTOMER ADDRESS                                         CUSREC
  23     +000011           06 CITY                  PIC X(20).                             CUSREC
         +000012*                 CUSTOMER CITY                                            CUSREC
  24     +000013           06 STATE                 PIC X(2).                              CUSREC
         +000014*                 STATE                                                    CUSREC
  25     +000015           06 ZIP                   PIC S9(5)       COMP-3.                CUSREC
         +000016*                 ZIP CODE                                                 CUSREC
  26     +000017           06 SRHCOD                PIC X(6).                              CUSREC
         +000018*                 CUSTOMER NUMBER SEARCH CODE                              CUSREC
  27     +000019           06 CUSTYP                PIC S9(1)       COMP-3.                CUSREC
         +000020*                 CUSTOMER TYPE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT               CUSREC
  28     +000021           06 ARBAL                 PIC S9(6)V9(2)  COMP-3.                CUSREC
         +000022*                 ACCOUNTS REC. BALANCE                                    CUSREC
  29     +000023           06 ORDBAL                PIC S9(6)V9(2)  COMP-3.                CUSREC
         +000024*                 A/R AMT. IN ORDER FILE                                   CUSREC
  30     +000025           06 LSTAMT                PIC S9(6)V9(2)  COMP-3.                CUSREC
         +000026*                 LAST AMT. PAID IN A/R                                    CUSREC
  31     +000027           06 LSTDAT                PIC S9(6)       COMP-3.                CUSREC
         +000028*                 LAST DATE PAID IN A/R                                    CUSREC
  32     +000029           06 CRDLMT                PIC S9(6)V9(2)  COMP-3.                CUSREC
         +000030*                 CUSTOMER CREDIT LIMIT                                    CUSREC
  33     +000031           06 SLSYR                 PIC S9(8)V9(2)  COMP-3.                CUSREC
         +000032*                 CUSTOMER SALES THIS YEAR                                 CUSREC
  34     +000033           06 SLSLYR                PIC S9(8)V9(2)  COMP-3.                CUSREC
         +000034*                 CUSTOMER SALES LAST YEAR                                 CUSREC
```

図 95. 形式 2 の COPY ステートメント (DDS) の結果の例

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A** LOGICAL  CUSMSTL   CUSTOMER MASTER FILE
     A                                      UNIQUE
     A          R CUSREC                    PFILE(CUSMSTP)
     A                                      TEXT('CUSTOMER MASTER RECORD')
     A            CUST                      ALIAS(CUSTOMER_NUMBER)
     A            NAME                    1 ALIAS(CUSTOMER_NAME)
     A            ADDR                      ALIAS(ADDRESS)
     A            CITY
     A            STATE
     A            ZIP
     A            SRHCOD                    ALIAS(SEARCH_CODE)
     A            CUSTYP                    ALIAS(CUSTOMER_TYPE)
     A            ARBAL                     ALIAS(ACCT_REC_BALANCE)
     A            ORDBAL
     A            LSTAMT
     A            LSTDAT
     A            CRDLMT
     A            SLSYR
     A            SLSLYR
     A          K CUST
```

図 96. ALIAS を指定したデータ記述仕様の例

1 これは、ALIAS キーワードに関連付けられている名前です。この名前は、プログラム内に組み込まれます。別名は DDS ALIAS オプションによって

使用可能になる代替名で、この代替名によって最大 30 文字のデータ名を ILE COBOL プログラム内に組み込むことができます。

```
                              ソ ー ス

STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S コピー名    変更日付


  18     002000 01  CUSTOMER-INVOICE-RECORD.
         002100     COPY DDS-CUSREC OF CUSMSTL ALIAS.
        +000001*    I-O FORMAT:CUSREC     FROM FILE CUSMSTL    OF LIBRARY TESTLIB           CUSREC
        +000002*                          CUSTOMER MASTER RECORD                            CUSREC
        +000003*          USER SUPPLIED KEY BY RECORD KEY CLAUSE                            CUSREC
  19    +000004        05  CUSREC.                                                          CUSREC
  20    +000005            06 CUSTOMER-NUMBER       PIC X(5).                             CUSREC
        +000006*                 CUSTOMER NUMBER                                           CUSREC
  21    +000007            06 CUSTOMER-NAME         PIC X(25).                            CUSREC
        +000008*                 CUSTOMER NAME                                             CUSREC
  22    +000009            06 ADDRESS-DDS           PIC X(20).                            CUSREC
        +000010*                 CUSTOMER ADDRESS                                          CUSREC
  23    +000011            06 CITY                  PIC X(20).                            CUSREC
        +000012*                 CUSTOMER CITY                                             CUSREC
  24    +000013            06 STATE                 PIC X(2).                             CUSREC
        +000014*                 STATE                                                     CUSREC
  25    +000015            06 ZIP                   PIC S9(5)       COMP-3.               CUSREC
        +000016*                 ZIP CODE                                                  CUSREC
  26    +000017            06 SEARCH-CODE           PIC X(6).                             CUSREC
        +000018*                 CUSTOMER NUMBER SEARCH CODE                               CUSREC
  27    +000019            06 CUSTOMER-TYPE         PIC S9(1)       COMP-3.               CUSREC
        +000020*                 CUSTOMER TYPE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT                CUSREC
  28    +000021            06 ACCT-REC-BALANCE      PIC S9(6)V9(2)  COMP-3.               CUSREC
        +000022*                 ACCOUNTS REC. BALANCE                                     CUSREC
  29    +000023            06 ORDBAL                PIC S9(6)V9(2)  COMP-3.               CUSREC
        +000024*                 A/R AMT. IN ORDER FILE                                    CUSREC
  30    +000025            06 LSTAMT                PIC S9(6)V9(2)  COMP-3.               CUSREC
        +000026*                 LAST AMT. PAID IN A/R                                     CUSREC
  31    +000027            06 LSTDAT                PIC S9(6)       COMP-3.               CUSREC
        +000028*                 LAST DATE PAID IN A/R                                     CUSREC
  32    +000029            06 CRDLMT                PIC S9(6)V9(2)  COMP-3.               CUSREC
        +000030*                 CUSTOMER CREDIT LIMIT                                     CUSREC
  33    +000031            06 SLSYR                 PIC S9(8)V9(2)  COMP-3.               CUSREC
        +000032*                 CUSTOMER SALES THIS YEAR                                  CUSREC
  34    +000033            06 SLSLYR                PIC S9(8)V9(2)  COMP-3.               CUSREC
        +000034*                 CUSTOMER SALES LAST YEAR                                  CUSREC
```

図 97. ALIAS キーワードを指定した形式 2 の COPY ステートメント (DD) の結果の例

形式 2 の COPY ステートメントを使用してプログラム内にファイルの外部記述を配置する以外にも、標準のレコード定義および再定義を使用して、外部ファイルを記述したり、一連のフィールドのグループ定義を提供したりすることもできます。プログラム記述定義がファイルの外部定義と互換性があることを保証するのは、プログラマーの責任です。

455 ページの図 98 は、ILE COBOL プログラムを i5/OS 上のファイルに関連付け
て、DDS から外部ファイル記述を利用する方法を示しています。

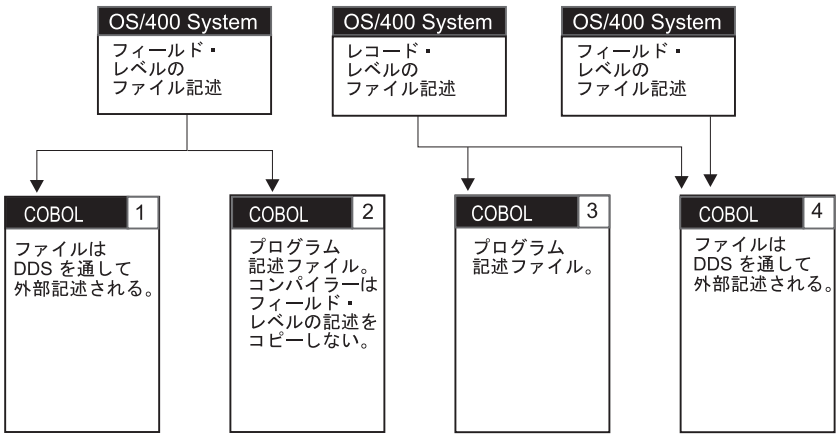


図 98. ILE COBOL と i5/OS ファイルの関連付け方法を示した例

1 ILE COBOL プログラムは、オペレーティング・システムに対して定義されているファイルのフィールド・レベル記述を使用します。レコード記述を得るために形式 2 の COPY ステートメントをコーディングします。コンパイラーは、コンパイル時に、外部フィールド・レベルの記述をコピーし、それを構文的に正しい ILE COBOL レコード記述に変換します。このファイルは、コンパイル時に存在している必要があります。

2 外部記述ファイルは ILE COBOL プログラムでプログラム記述ファイルとして使用されます。外部記述ファイルのレコード記述全体が ILE COBOL プログラムにコーディングされます。このファイルは、コンパイル時に存在している必要はありません。

3 ファイルは、レコード・レベルまでに限り、オペレーティング・システムに対して記述されます。レコード記述全体を ILE COBOL プログラムにコーディングする必要があります。このファイルは、コンパイル時に存在している必要はありません。

4 コンパイル時にファイル名を指定できますが、実行時には別のファイル名を指定することができます。ILE COBOL 形式 2 の COPY ステートメントは、コンパイル時にファイルのレコード記述を生成します。実行時は、プログラムが別のファイルにアクセスするよう、別のライブラリー・リストまたはファイル指定変更コマンドを使用することができます。コンパイル時にコピーされたファイル記述は、実行時に使用される入力レコードを記述するために使用されます。

注: 外部記述ファイルの場合、2 つのファイル・フォーマットは同一である必要があります。同一でない場合は、レベル検査エラーが発生します。

非キー順レコード検索とキー順レコード検索の指定

外部記述ファイルの記述には、ファイルからレコードを取り出す方法を記述するアクセス・パスが含まれます。レコードは、到着順 (非キー順) アクセス・パス、またはキー順アクセス・パスに基づいて取り出されます。外部記述データベース・ファイルのアクセス・パスの詳細については、 Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベース」カテゴリーの中の『*DB2 Universal Database for AS/400*』セクションを参照してください。

到着順アクセス・パスは、レコードがファイルに保管された順序に基づいています。レコードは、ファイルの末尾にのみ追加されます。

キー順アクセス・パスの場合、ファイルからレコードを取り出す順序は、DDS でファイルに対して定義されているキー・フィールドの内容に基づいています。例えば、 452 ページの図 94 に示されている DDS では、CUST がキー・フィールドとして定義されています。キー順アクセス・パスは、レコードが追加または削除されたとき、あるいはキー・フィールドの内容が変化したときに更新されます。キー順アクセス・パスの場合、1 つ以上のフィールドを DDS に定義して、レコード・フォーマットのキー・フィールドとして使用することができます。ファイル内のすべてのレコード・フォーマットが同じキー・フィールドを持つ必要はありません。例えば、注文ヘッダー・レコードの ORDER フィールドはキー・フィールドとして定義し、オーダー明細レコードの ORDER および LINE フィールドはキー・フィールドとして定義することができます。

入出力操作に対してフォーマットを指定しなかった場合、ファイルのキーは、そのファイル内のレコード・フォーマットの有効なキーによって決定されます。ファイルのキーは、以下のようにして決定されます。

- ファイル内のすべてのレコード・フォーマットが、同じ数のキー・フィールドを属性が同一の DDS で定義している場合、*ファイルのキー* は、それらレコード・フォーマットのキー内のすべてのフィールドで構成されます。(対応するフィールドの名前が同じである必要はありません。) 例えば、ファイルに 3 つのレコード・フォーマットが含まれており、各レコード・フォーマットのキーが、フィールド A、B、および C で構成されている場合、このファイルのキーは、フィールド A、B、および C で構成されます。すなわち、ファイルのキーは、レコードのキーと同一です。
- ファイル内のすべてのフォーマットが同じキー・フィールドを持っていない場合、ファイルのキーは、すべてのレコード・フォーマットに*共通する* キー・フィールドで構成されます。例えば、ファイルに 3 つのレコード・フォーマットがあり、キー・フィールドが以下のように定義されているとします。
  - REC1 にはキー・フィールドが含まれている。
  - REC2 にはキー・フィールド A および B が含まれている。
  - REC3 にはキー・フィールド A、B、および C が含まれている。

  この場合、ファイルのキーは、すべてのレコード・フォーマットに共通のキー・フィールドであるフィールド A です。
- すべてのレコード・フォーマットに共通するキー・フィールドがない場合、そのファイルへのキーによる参照は常にそのファイル内の最初のレコードを戻します。

ILE COBOL では、処理するレコードを識別するのに、索引付きファイルに RECORD KEY を指定する必要があります。 ILE COBOL は、キー値をファイルまたはレコードのキーと比較して、キーが RECORD KEY 値に一致するレコードに対して指定された操作を処理します。

RECORD KEY IS EXTERNALLY-DESCRIBED-KEY が指定されている場合は、以下のようになります。

- FORMAT 句が指定されている場合、コンパイラーは、レコード域内の指定されたフォーマットのキー・フィールドから検索引数を作成します。

- FORMAT 句が指定されていない場合、コンパイラーは、プログラムでそのファイルに対して定義されている最初のレコード・フォーマットの、レコード域内のキー・フィールドから検索引数を作成します。

注: ILE COBOL で処理される複数のキー・フィールドを含むファイルの場合、RECORD KEY IS EXTERNALLY-DESCRIBED-KEY が指定されている場合を除き、ILE COBOL プログラムが使用するレコード・フォーマットでは、キー・フィールドは隣接している必要があります。

外部記述ファイルのレベル検査

ILE COBOL プログラムが外部記述ファイルを使用している場合、オペレーティング・システムは、レベル検査関数 (LVLCHK) を提供します。この関数は、ファイルのフォーマットがコンパイル以降変更されていないことを保証します。

外部記述ファイルが使用されている場合 (すなわち、レコード記述が形式 2 の COPY ステートメントを使用してファイルに定義されていた場合)、コンパイラーは常にレベル検査に必要な情報を提供します。レベル検査されるのは、形式 2 の COPY ステートメントによってファイルの FD の項目にコピーされたフォーマットだけです。レベル検査関数は、実行時に、ファイルの作成、変更、または指定変更コマンドに対して行われた選択に基づいて開始されます。ファイル作成コマンドのデフォルトでは、レベル検査が要求されます。レベル検査が要求されていた場合は、ファイルが開かれたときにレベル検査がレコード・フォーマット・ベースで検査が行われます。レベル検査エラーが発生した場合、ILE COBOL はファイル状況を 39 に設定します。

CRTxxxF、CHGxxxF、または OVRxxxF CL コマンドに対して LVLCHK(*NO) が指定されており、既存のフォーマットを使用してファイルが再作成された場合、そのフォーマットを使用している既存の ILE COBOL プログラムは、フォーマットの変更内容によっては再コンパイルしないと動作しない可能性があります。

レベル検査を行わないで ILE COBOL プログラムでファイルを使用する場合には、十分に注意する必要があります。レベル検査または再コンパイルを行わないで ILE COBOL プログラムを使用すると、プログラムが失敗し、データが破壊される危険性があります。

注: ILE COBOL コンパイラーは、プログラム記述ファイルのレベル検査を提供していません。

レベル検査の詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベース」カテゴリーの中の『*DB2 Universal Database for AS/400*』セクションを参照してください。

第 18 章 ファイルの処理

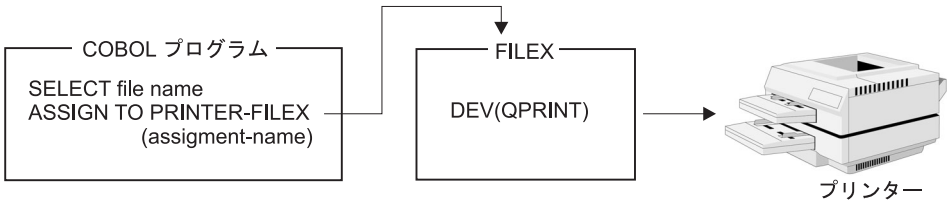
i5/OS で COBOL がファイルを使用して処理を行う方法は数多くあります。 この章では次のことについて説明します。

- ファイルと入出力装置との関連付け
- ファイル属性の変更
- ファイル入出力の宛先変更
- ファイルのロックおよび解除
- 入力レコードの非ブロック化および出力レコードのブロック化
- ファイルにアクセスするためのオープン・データ・パスの共用
- ファイル状況とフィードバック域の使用
- コミットメント制御の使用
- ファイルのソートおよびマージ
- CVTOPT データ・タイプによるデータ項目の宣言

ファイルと入出力装置との関連付け

ファイルは、入出力として使用される装置とプログラムとの接続リンクとして使用されます。実際に装置と関連付けることは、ファイルをオープンする時点で行われます。このタイプの入出力制御を使用すると、プログラムを変更することなく、そのプログラムで使用されているファイル属性 (および、場合によっては装置) を変更できます。

ILE COBOL 言語では、ファイル制御記入項目の ASSIGN 文節の ASSIGNMENT-NAME 項目に指定されているファイル名を使用してファイルを指します。このファイル名は、次のようにシステム・ファイル記述を指します。

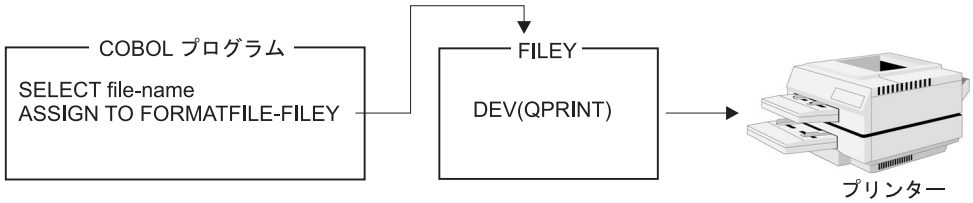


ASSIGN 文節の ILE COBOL 装置名は、選択したファイルで処理できる ILE COBOL 機能を定義します。コンパイル時には、特定の ILE COBOL 機能は特定の ILE COBOL 装置タイプに限り有効です。この点で、ILE COBOL は装置依存です。装置依存の例を次に示します。

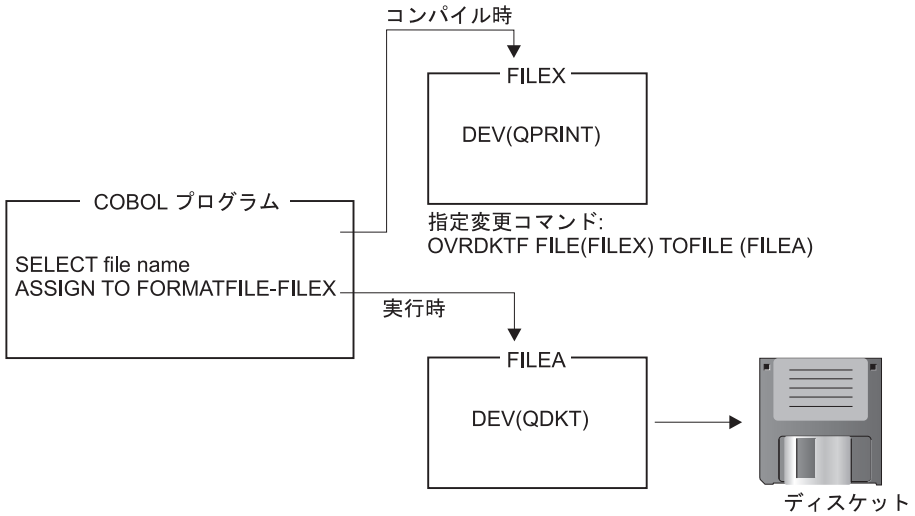
- SUBFILE 操作は WORKSTATION 装置に限り有効です。
- 標識は WORKSTATION 装置か FORMATFILE 装置に限り有効です。
- LINAGE は PRINTER 装置に限り有効です。
- OPEN INPUT WITH NO REWIND は TAPEFILE 装置に限り有効です。

たとえば、ILE COBOL プログラムの中でファイル名 FILEY が FORMATFILE 装置と関連付けられる場合を考えてみます。装置 FORMATFILE は独立装置タイプで

す。したがって、有効な行制御やページ制御の仕様がないので、FORMATFILE ファイルの WRITE ステートメントに ADVANCING 句を指定することはできません。プログラムが実行されると、実際の入出力装置が FILEY の記述中に指定されます。たとえばプリンターの場合は、次のようにデフォルトかまたは DDS で定義されている行制御およびページ制御しか使用できません。



CL コマンドを使用することによって、指定されているファイル記述のパラメーターを指定変更するか、あるいはコンパイル時または実行時にファイルを宛先変更できます。ファイルを宛先変更すると、次のように、コンパイル時にあるファイルを指定し、実行時には別のファイルを指定することができます。



上記の例では、ディスケット・ファイル指定変更 (OVRDKTF) コマンドを使用することにより、コンパイル時に指定されたのとまったく異なる装置ファイルを使用してプログラムを実行できます。

注: FORMATFILE 装置を入力に使用することはできません。入力できる装置 (DISKETTE 装置など) から FORMATFILE 装置に入出力を指定変更して入力操作をしようとすると、予期しない結果になることがあります。

ファイル指定変更がすべて有効なわけではありません。実行時に、処理されるファイルに対して ILE COBOL プログラムの中の仕様が有効かどうかを確認する検査が行われます。 ILE COBOL プログラムがファイル制御ブロックおよび I/O 要求で渡す仕様が誤っていると、その I/O 操作は失敗します。プログラムに装置指定が含まれていても、オペレーティング・システムでファイルの宛先変更がなされる可能性があります。たとえば、ILE COBOL 装置名が PRINTER で、プログラムが実際に使用するファイルがプリンターでない場合、オペレーティング・システムは ILE COBOL の印刷行送りとスキップの仕様を無視します。

オペレーティング・システムで許されない、ファイルが使用できなくなることもある、別のファイルの宛先変更もあります。たとえば、ILE COBOL 装置名が DATABASE または DISK で、プログラムにキー順 READ 操作が指定されていて、プログラムが実際に使用するファイルがディスク・ファイルまたはデータベース・ファイルでない場合、そのファイルは使用できなくなります。

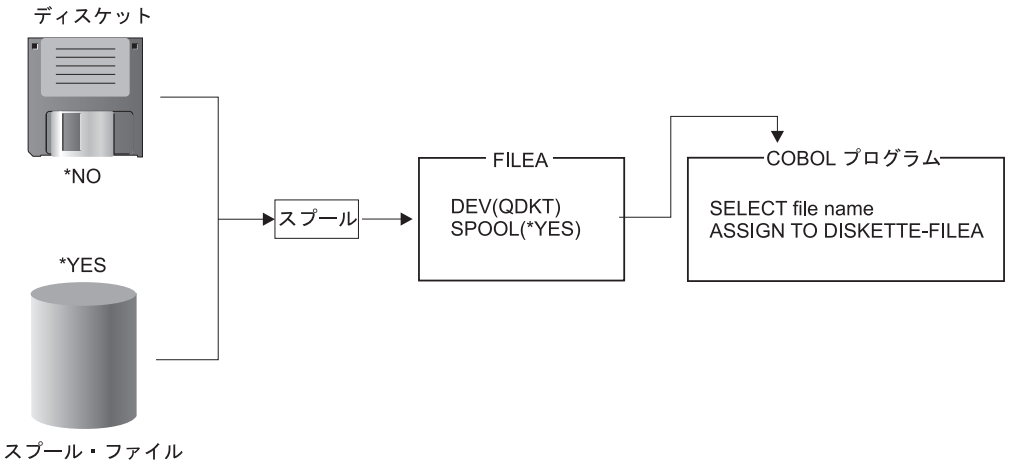
i5/OS システムには入出力スプール機能があります。個々の i5/OS システム・ファイル記述には、実行時にそのファイルにスプールが使用されたかどうかを判別するためのスプール属性があります。 ILE COBOL プログラムは、スプールが使用されたことを認識しません。ファイルを読み書きする相手となる実際の物理装置は、スプール読み取りプログラムまたはスプール書き込みプログラムによって判別されます。スプーリングの詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベース」カテゴリーの中の『*DB2 Universal Database for AS/400*』セクションを参照してください。

入出力スプールの指定

入力スプール

入力スプールは、バッチ・ジョブのインライン・データ・ファイルに限り有効です。 ILE COBOL がスプール・ファイルからの入力データを読み取る場合、ILE COBOL はデータのスプール元の装置を認識しません。

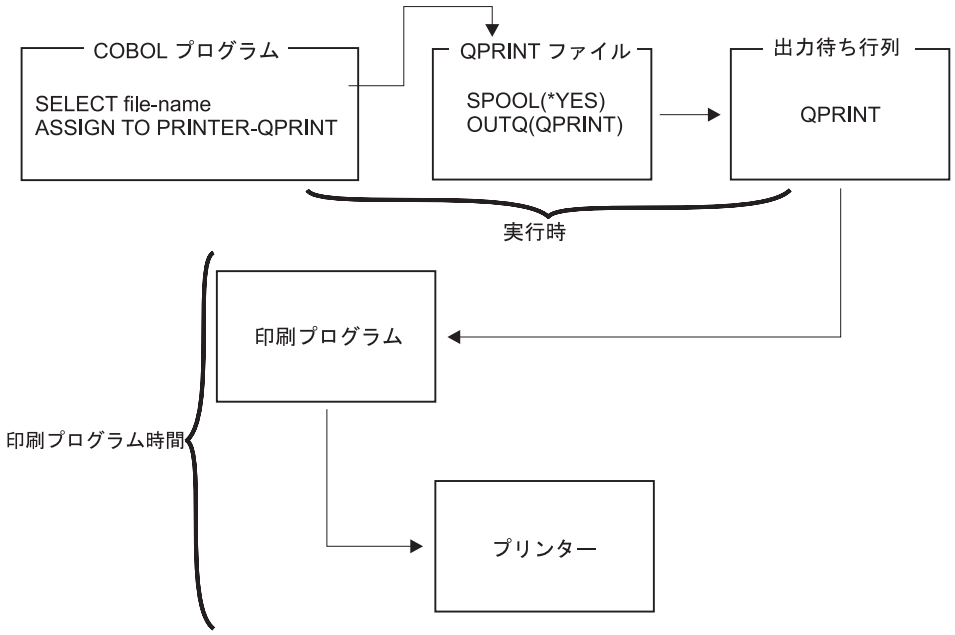
データは、次のようにしてスプール・インライン・ファイルから読み取られます。



インライン・データ・ファイルの詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベースおよびファイル・システム」カテゴリーを参照してください。

出力スプール

出力スプールは、バッチ・ジョブと対話式ジョブに限り有効です。 ILE COBOL にシステム名で指定されているファイル記述には、次の例のようにスプール仕様が記述されています。



実行時にファイル指定変更コマンドを使用することによって、ファイル記述中で指定されているスプール・オプション (印刷するコピー数など) を指定変更できます。また、i5/OS スプール・サポートを使用することによって、プログラム実行後にファイルを宛先変更できます。 たとえば、プリンターの出力先を別の装置 (ディスケットなど) に指定することができます。

ファイル属性の指定変更

指定変更は、ILE COBOL プログラムがファイルをオープンする前に指定しなければなりません。システムはファイル指定変更コマンドを使用して、オープンするファイルとそのファイルの属性を判別します。ファイルの指定変更の有効範囲は、呼び出しレベル、活動化グループ・レベル、またはジョブ・レベルになります。

呼び出しレベルの有効範囲の場合、特定の呼び出しレベルで出される指定変更は、呼び出しがどの活動化グループに含まれているかに関係なく、呼び出しレベルの後のすべての呼び出しについて有効です。また、指定変更を出した呼び出しレベルに制御が戻ると無効になります。

活動化グループの有効範囲の場合、その活動化グループの中で実行されているすべてのプログラム・オブジェクトに指定変更が適用され、活動化グループが終了するか、指定変更が明示的に削除される時点まで有効です。

注: デフォルト活動化グループ (*DFTACTGRP) では、活動化グループ・レベルの有効範囲が指定されると、実際の指定変更の有効範囲は呼び出しレベルになります。

ジョブ・レベルの有効範囲の場合、ジョブの中のすべてのプログラム・オブジェクトに指定変更が適用され、そのジョブが終了するか、指定変更が明示的に削除される時点まで活動状態になります。

指定変更の有効範囲を指定するには、指定変更 CL コマンドの OVRSCOPE パラメーターを使用します。有効範囲を明示指定しないなら、指定変更のデフォルトの有効範囲は、指定変更をどこで発行したかによって異なります。指定変更がデフォルト活動化グループから出されると、有効範囲は呼び出しレベルになります。指定変更がデフォルト以外の活動化グループから出されると、有効範囲は活動化グループ・レベルになります。

ファイルを指定変更する最も単純な形式は、ファイルの属性のいくつかを指定変更することです。たとえば、プリンター・ファイルを作成する際に、FILE(OUTPUT) と COPIES(2) を指定します。続いて、ILE COBOL プログラムを実行する前に、印刷される出力コピーの数を 3 に変更できます。指定変更コマンドは次のようになります。

```
OVRPRTF FILE(OUTPUT) COPIES(3)
```

ファイル入出力の宛先変更

ファイルを指定変更するもう 1 つの形式は、ILE COBOL プログラムを宛先変更して別のファイルにアクセスすることです。指定変更する際に*同じ* タイプのファイルに (たとえば、あるプリンター・ファイルを別のプリンター・ファイルに) 宛先変更すると、そのファイルは元のファイルと同じ方法で処理されます。

指定変更する際に*別の* タイプのファイルに宛先変更すると、その指定変更ファイルは元のファイルが処理されるはずだった方法と同じ方法で処理されます。 ILE COBOL プログラム中の装置依存の仕様のうち、指定変更装置に適用されないものは、システムによって無視されます。

ファイル宛先変更がすべて有効なわけではありません。たとえば、ILE COBOL プログラムの索引ファイルは、キー順アクセス・パスを使用した他の索引ファイルにのみ指定変更できます。

データベース・ファイルを指定変更してすべてのメンバーを処理することによって、複数のメンバーを処理することができます。次の例外に注意してください。

- ILE COBOL プログラムのコンパイル時に使用されるデータベース・ソース・ファイルを指定変更して、すべてのメンバーを処理することはできません。 OVRDBF MBR(*ALL) を指定すると、コンパイルが終了します。
- COPY ステートメント用に使用されるデータベース・ファイルを指定変更して、すべてのメンバーを処理することはできません。 OVRDBF MBR(*ALL) を指定すると、COPY ステートメントは無視されます。

ファイル指定変更は必ず適切に適用されていなければなりません。有効なファイル
宛先変更、無視される装置依存の特性、および想定されるデフォルトの詳細につい
ては、 http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の
「プログラミング」カテゴリーを参照してください。

ファイルのロックおよび解放

オペレーティング・システムでは、ジョブ・ステップで使用されるファイルをロック状態 (排他、読み取り可排他、更新共用、更新不可共用、または読み取り共用) にすることができます。ファイルをロック状態にする場合には、オブジェクトの割り振り (ALCOBJ) コマンドを使用します。

デフォルトでは、ILE COBOL プログラムがファイルをオープンする際、オペレーティング・システムはデータベース・ファイルを次のロック状態にします。

OPEN タイプ	ロック状態
INPUT	読み取り共用
I-O	更新共用
EXTEND	更新共用
OUTPUT	更新共用

読み取り共用ロック状態の場合、他のユーザーは、読み取り共用、更新共用、更新不可共用、または読み取り可排他ロック状態でファイルをオープンできますが、ファイルの排他使用は指定できません。更新共用ロック状態の場合、他のユーザーは、読み取り共用または更新共用ロック状態でファイルをオープンできます。

オペレーティング・システムは、装置ファイルを読み取り共用ロック状態にし、装置を読み取り可排他ロック状態にします。他のユーザーはファイルをオープンできますが、同じ装置を使用することはできません。

注: ILE COBOL プログラムが OUTPUT 用に物理ファイルをオープンすると、そのファイルは、メンバーをクリアするのに必要な時間だけ排他ロックの対象になります。

リソースの割り振りおよびロック状態の詳細については、Web サイト
http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベースおよびファイル・システム」カテゴリーを参照してください。

レコードのロックおよび解放

ILE COBOL プログラムがデータベース・ファイルに READ を実行する場合、そのファイルが入出力用にオープンされると、そのレコードがロック状態になって他のプログラムはそのレコードを更新できなくなります。すなわち、ファイルを入力用にオープンする場合は他のプログラムからそのレコードを読み取れますが、入出力用にオープンする場合はそれができません。同様に、入出力モードでオープンされたファイルの START 操作が成功すると、そのファイルのそのレコードの位置はロック状態になります。

コミットメント制御を行っている場合と行っていない場合の、レコードをロックする期間については 471 ページの図 99 を参照してください。

READ ステートメントまたは START ステートメントを実行する際に、入出力 (更新) モードでオープンされているファイルのレコードをロックしないようにするためには、NO LOCK 句を使用できます。 READ WITH NO LOCK ステートメントを使用すると、以前に READ ステートメントまたは START ステートメントによ

ってロックされたレコードがアンロックされます。また、READ WITH NO LOCK ステートメントで読み取られるレコードもロックされません。 START WITH NO LOCK ステートメントを使用すると、以前に START ステートメントまたは READ ステートメントによってロックされたレコードがアンロックされます。この句の詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の READ ステートメントおよび START ステートメントに関する部分を参照してください。

1 個の物理ファイルに基づく論理ファイルの場合、その物理ファイル中のレコードがロック状態になります。論理ファイルが複数の物理ファイルに基づいている場合は、各物理ファイルの中の 1 つのレコードがロック状態になります。

2 つめのファイルで同一の基本物理レコードを更新しようとすると、他のプログラムだけでなく元のプログラムにもこのロックが適用されます。

注: ランダム・アクセスか動的アクセスを使用して、索引付き編成または相対編成のファイルを入出力用にオープンすると、どの入出力 verb に関する入出力操作 (WRITE を除く) が失敗した場合でも、レコードはアンロックされます。WRITE 操作は更新操作とは見なされません。したがって、レコードのロックは解放されません。

更新用に読み取ったデータベース・レコードの解放については、Web サイト
http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベースおよびファイル・システム」カテゴリーを参照してください。

ファイルにアクセスするためのオープン・データ・パスの共用

経路指定ステップの中の別のプログラムを使用してすでにファイルをオープンしてある場合は、ILE COBOL プログラムで、同一のオープン・データ・パス (ODP) を使用してファイルにアクセスすることができます。

注: 通常、1 つのジョブには経路指定ステップが 1 つだけ含まれています。経路指
定のステップについては、Web サイト http://www.ibm.com/systems/i/infocenter/
にある **i5/OS Information Center** の「データベース」カテゴリーの中の『*DB2 Universal Database for AS/400*』セクションを参照してください。

共用 ODP には次の規則が適用されます。

1. ファイルを作成するコマンド (CRTxxxF)、変更コマンド (CHGxxxF)、またはファイル指定変更コマンド (OVRxxxF) 中には、SHARE(*YES) を指定しなければなりません。
2. 共用 ODP のあるファイルをプログラムが初めてオープンし、かつ現在もオープンされている場合、同一の経路指定ステップ中の共用 OPEN 操作は、標準 OPEN 操作より実行が速くなります。他の入出力操作の速度に影響はありません。
3. 別のプログラムの中でファイルを使用する際には、一貫性を保つ必要があります。同一の共用ファイルを使用する他のプログラムが、ファイルに対する入出力操作を実行すると、現在のファイル位置に影響があります。

入力レコードの非ブロック化および出力レコードのブロック化

1 つの**ブロック**に、複数のレコードを入れることができます。入力レコードの非ブロック化と出力レコードのブロック化は、次の条件のもとで起こります。

1. CRTCBLMOD または CRTBNDCBL コマンドの OPTION パラメーターで *NOBLK が設定されており (BLOCK CONTAINS 文節はある場合もない場合もある)、また以下の条件の**すべて**が満たされている。
   a. ファイルに ACCESS IS SEQUENTIAL が指定されている。
   b. プログラムの中でファイルが INPUT または OUTPUT 用にオープンされている。
   c. ファイルが、DISK、DATABASE、DISKETTE、または TAPEFILE に割り当てられている。
   d. そのファイルに対して、START ステートメントが指定されていない。

   RELATIVE 相対編成の場合、OPEN OUTPUT でブロック化は実行されません。

   BLOCK CONTAINS を指定してもそれは無視されます。ブロック化するレコード数は、システムが決めます。
2. CRTCBLMOD コマンドまたは CRTBNDCBL コマンドの OPTION パラメーターに *BLK が指定されており (BLOCK CONTAINS 文節はある場合もない場合もある)、かつ次の条件を**すべて**満たしている。
   a. ファイルに ACCESS IS SEQUENTIAL または ACCESS IS DYNAMIC が指定されている。
   b. プログラムの中でファイルが INPUT または OUTPUT 用にオープンされている。
   c. ファイルが、DISK、DATABASE、DISKETTE、または TAPEFILE に割り当てられている。

   RELATIVE 相対編成の場合、OPEN OUTPUT でブロック化は実行されません。

   BLOCK CONTAINS 文節は、ブロック化されるレコード数を制御します。DISKETTE ファイルの場合は、ブロック化するレコード数は常にシステムが決めます。

上記のすべての条件を満たしている場合でも、特定のオペレーティング・システムの制約事項のために、ブロック化や非ブロック化が行われないことがあります。この場合、際立ったパフォーマンスの向上はありません。

直接アクセス索引ファイルを使用している場合は、READ PRIOR および READ NEXT を使用してブロック化を実行できます。 READ PRIOR および READ NEXT を使用してブロック化を実行する際には、ブロックにレコードが残っている間に宛先を変更することはできません。レコードをブロックからクリアするには、ランダム READ またはランダム START などのランダム操作を指定するか、あるいは順次 READ FIRST または READ LAST を使用してください。

正しくない宛先変更が行われると、ファイル状況 9U になります。ファイルをクローズして再オープンしないと、それ以上の入出力は行えません。

OVRDBF コマンドに SEQONLY(*NO) を指定すると、実行時にブロック化を指定変更できます。

ディスクおよびデータベース・ファイルの場合は、BLOCK CONTAINS が使用され、ブロック化因数としてゼロが指定されるかまたは計算される場合、システムがブロック化因数を決めます。

指定したブロック化因数が変更される場合があります。

レコードのブロックの読み書きが行われる場合、入出力フィードバック域にブロックの中のレコード数が記述されます。 ILE COBOL によって複数のレコードのブロック化または非ブロック化が行われるファイルの場合、I-O-FEEDBACK 域は、読み取りまたは書き込みごとには更新されません。その次のブロックが読み書きされる時点で更新されます。

ブロック化が有効なデータベース・ファイルの場合、さまざまなプログラムで変更が行われると、変更された時点ですべての変更内容を参照できないことがあります。

ファイルに代替レコード・キーがある場合は、ブロック化は暗黙的に使用不可となります。

データベース・ファイルに変更を加える際のブロック化の影響、およびブロック化因数の変更については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベース」カテゴリーの中の『*DB2 Universal Database for AS/400*』セクションを参照してください。

ファイル状況とフィードバック域の使用

オープン・ファイルと関連のある OPEN-FEEDBACK 域と I-O-FEEDBACK 域のデータを ID に転送する際には、形式 3 の ACCEPT ステートメントを使用してください。このステートメントを指定することの詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の『ACCEPT ステートメント』を参照してください。

FILE STATUS

FILE STATUS 文節を指定すると、当該ファイルを明示的または暗黙のうちに参照する入出力要求が出されるたびに、システムはある値を状況キー・データ項目の中に入れます。この 2 文字の値は、ステートメントの実行状況を示します。入力レコードの非ブロック化と出力レコードのブロック化が行われる場合、IBM i 例外を生じさせるファイル状況の値は、ブロックが処理される場合に限り設定されます。レコードのブロック化の詳細については 466 ページの『入力レコードの非ブロック化および出力レコードのブロック化』を参照してください。

OPEN-FEEDBACK 域

OPEN-FEEDBACK 域は、OPEN 操作に関する情報が入っているオープン・データ・パス (ODP) の一部です。この情報は OPEN 処理中に設定され、そのファイルがオープンされている間、使用することができます。

この区域は、プログラムが使用しているファイルに関する情報を提供します。この区域に含まれる情報は、次のとおりです。

- 現在オープンしているファイルに関する情報。ファイル名やファイルのタイプなど。
- オペランドであるファイルのタイプに基づく情報。プリンターのサイズ、画面サイズ、ディスケット・ラベル、またはテープ・ラベルなど。

注: 正常にオープンされた OPTIONAL INPUT ファイルには、OPEN-FEEDBACK 域情報がありません。

I-O-FEEDBACK 域

システムは、ブロックがオペレーティング・システムとプログラムの間で転送されるたびに、I-O-FEEDBACK 域を更新します。 1 つのブロックには、1 つまたは複数のレコードを入れることができます。

COBOL によって複数のレコードのブロック化または非ブロック化が行われるファイルの場合、I-O-FEEDBACK 域は、読み取りまたは書き込みごとには更新されません。プログラム中のそれぞれの読み取りまたは書き込み操作の後に、I-O-FEEDBACK 情報が必要な場合には、次のどちらかの処置をとることができます。

- 466 ページの『入力レコードの非ブロック化および出力レコードのブロック化』に示されている条件のいずれも満たさないようにして、コンパイラーがブロック化コードおよび非ブロック化コードを生成しないようにする。
- データベース・ファイルによる指定変更 (OVRDBF) CL コマンドに SEQONLY(*NO) を指定する。

コンパイラーがブロック化および非ブロック化コードを生成しないようにする方が、SEQONLY(*NO) を指定するよりも効果的です。

コンパイラーがブロック化および非ブロック化コードを生成する場合でも、特定の IBM i の制限によって、ブロック化および非ブロック化が処理されないようにすることができます。この場合、際立ったパフォーマンスの向上はありません。しかし、各読み取りまたは書き込み操作の後で、I-O-FEEDBACK 域は更新されます。

I-O-FEEDBACK 域には、最後に正常に実行された入出力操作に関する情報が入れられます。その情報には、装置名、装置タイプ、AID 文字、および一部の装置に関するエラー情報が含まれます。この区域は、共通域と装置依存域の 2 つの部分から構成されています。装置依存域の長さと内容は、ファイルに関連する装置タイプによって異なります。この区域は I-O-FEEDBACK 共通域の後にあり、共通域および該当する装置依存域を入れるために十分な大きさの受け取り側 ID を指定することによって、入手することができます。

OPEN-FEEDBACK 域と I-O-FEEDBACK 域にあるデータ域のレイアウトと説明については、Webサイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベース」カテゴリーの中の『*DB2 Universal Database for AS/400*』セクションを参照してください。

コミットメント制御の使用

コミットメント制御は、次の処理を可能にする機能です。

- 同一ジョブ中のデータベース・ファイルに対する変更の同期化
- データベースに永続的に入れるべきでない変更内容の取り消し
- 変更が完了するまで、変更されるレコードをロックする
- ジョブまたはシステムの障害から回復するためのテクニック

データベース・レコードを同期変更する方がよいアプリケーションもあります。変更が有効であるとプログラムが判別すると、データベースに永続的な変更が加えられます (COMMIT ステートメントが処理されます)。変更が無効か、または処理中に問題が発生すると、変更は取り消されます (ROLLBACK ステートメントが処理されます)。 (OUTPUT 用のオープンの後にファイルの消去が行われた場合には、ROLLBACK を処理しても、消去されたレコードがファイルに復元されることはありません。) コミットメント制御のもとにない ファイルのレコードに対する変更は、必ず永続的になります。この変更は、以降の COMMIT ステートメントまたは ROLLBACK ステートメントの影響を受けることはありません。

COMMIT または ROLLBACK が正常に処理される地点は、それぞれコミットメント境界になります。 (プログラムの中で COMMIT または ROLLBACK がまだ出されていない場合、コミットメント制御のもとにあるいずれかのファイルを初めてオープンするときにコミットメント境界が作成されます。) 変更のコミットやロールバックを行った場合、影響を受けるのは、直前のコミットメント境界以降に行われた変更だけです。

コミットメント境界で変更を同期化すると、障害が発生した後の再始動またはリカバリーの手順が簡単になります。詳細については 436 ページの『障害の後でのリカバリー』を参照してください。

データベース・ファイルでコミットメント制御を使用すると、そのファイルのレコードは以下のいずれかのロック・レベルの対象になります。

- **高ロック・レベル**

  高ロック・レベルは、コミットメント制御開始 (STRCMTCTL) CL コマンドの LCKLVL(*ALL) パラメーターで指定します。高ロック・レベル (*ALL) を使用すると、COMMIT または ROLLBACK が正常に処理されるまで、コミットメント制御下のファイルにアクセスしているすべての レコード (入力用または出力用) がロックされます。
- **カーソル固定ロック・レベル**

  カーソル固定ロック・レベルは、コミットメント制御開始 (STRCMTCTL) CL コマンドの LCKLVL(*CS) パラメーターで指定します。カーソル固定ロック・レベル (*CS) を使用すると、コミットメント制御のもとでオープンしたファイルにアクセスしているすべてのレコードがロックされます。読み取られても変更や削除が行われないレコードは、別のレコードが読み取られる時点でアンロックされます。変更、追加、または削除が行われるレコードは、COMMIT ステートメントまたは ROLLBACK ステートメントが正常に処理されるまでロックされています。
- **低ロック・レベル**

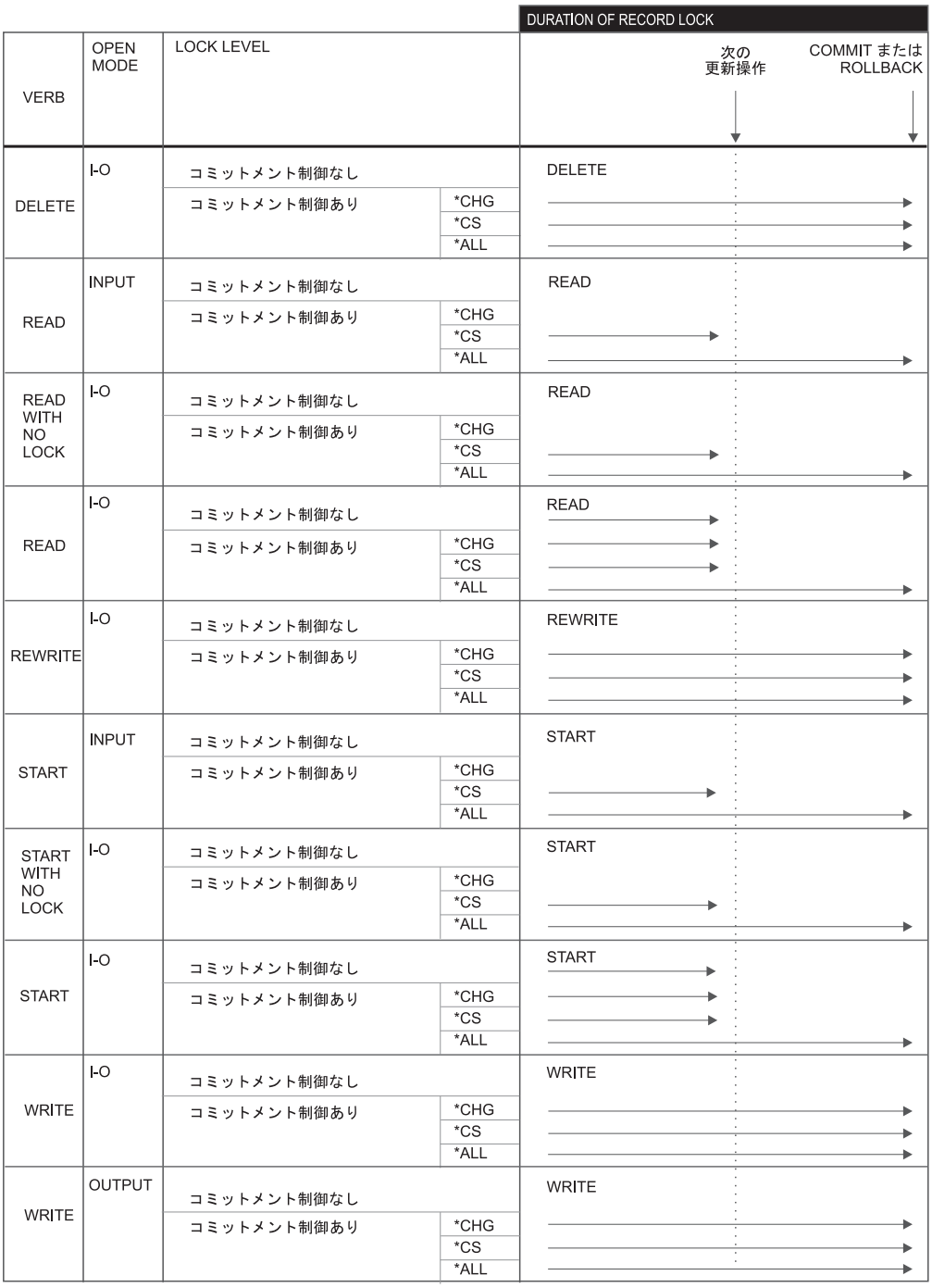
低ロック・レベルは、コミットメント制御開始 (STRCMTCTL) CL コマンドの LCKLVL(*CHG) パラメーターで指定します。低ロック・レベル (*CHG) を使用すると、コミットメント更新用に読み取られた (コミットメント制御のもとでオープンしたファイルの) すべてのレコードがロックされます。レコードが変更、追加、または削除されると、COMMIT ステートメントまたは ROLLBACK ステートメントが正常に処理されるまで、そのレコードはロックされています。更新操作用にアクセスしたが、変更されることなく解除されるレコードは、アンロックされます。

ロックされたレコードは、同一ジョブの中で同一の物理ファイルまたは論理ファイルを使用した場合に限り変更できます。

ロック・レベルは、ロックされたレコードを読み取れるかどうかも管理します。高ロック・レベル (*ALL) を使用すると、データベース・ファイル中のロックされたレコードは読み取れません。低ロック・レベル (*CHG) を使用した場合、データベース・ファイルをジョブの中で INPUT としてオープンするか、または I-O として READ WITH NO LOCK を使用してオープンすれば、そのファイルの中のロックされたレコードを読み取ることができます。

ファイルがコミットメント制御下にない 他のジョブは、ファイルが INPUT としてオープンされている場合、使用されるロック・レベルには関係なく、ロックされたレコードを常に読み取れます。ロックされたレコードを他のジョブが読み取る可能性があるため、データにアクセスできるのは、*そのデータがデータベースに永続的にコミットされる前* です。ロックされたレコードを別のジョブが読み取った*後に* ROLLBACK ステートメントを処理した場合、アクセスされるデータにはデータベースの内容が反映されていません。

471 ページの図 99 に、コミットメント制御が行われている場合と行われていない場合に、ファイルのレコードをロックする際の考慮事項を示します。



注: 更新操作には、同一のファイルについての START、READ、REWRITE、または DELETE 操作 (成功するか失敗するかに関係なく)、およびそのファイルのクローズが含まれます。WRITE 操作は更新操作とは見なされません。したがって、WRITE 操作の結果としてロックは設定または解放されません。

図 99. コミットメント制御が行われている場合と行われていない場合でのレコードのロックについての考慮事項

コミットメント制御下のファイルは、最後のコミットメント境界以降に行われた変更の状況に影響を与えずに、クローズまたはオープンすることができます。この場合にも、変更を永続的なものにするために COMMIT を出すか、変更を取り消すに

は ROLLBACK を出さなければなりません。 COMMIT ステートメントが処理されると、ファイルは、処理前と同じオープンまたはクローズの状態に置かれます。

i5/OS バージョン 2 リリース 3 モディフィケーション 0 以前のライセンス・プログラムをもっている場合は、同一コミットメント定義のもとで、ジョブ中でオープンしたすべてのファイルは、同一ジャーナルにジャーナル処理する必要があります。 バージョン 3 リリース 1 以降では、ほとんどの場合、この制約事項は関係なくなりました。 ジャーナル管理とその関連機能の詳細について、およびコミットメント制御の詳細については、「*Recovering your system*」を参照してください。

さらに、コミットメント制御は、i5/OS 制御言語 (CL) を使用して ILE COBOL の外部で指定しなければなりません。コミットメント制御開始 (STRCMTCTL) コマンドは、コミットメント制御の機能を確立し、レコードをロックするレベルを高レベル (*ALL)、カーソル固定レベル (*CS)、または低レベル (*CHG) に設定します。

STRCMTCTL コマンドを使用してコミットメント制御を開始すると、システムによって**コミットメント定義**が作成されます。各コミットメント定義は、コミットメント制御の有効範囲に基づいて、STRCMTCTL コマンドを出したジョブ、またはジョブ中の活動化グループだけに認識されます。コミットメント定義には、ジョブまたはジョブの中の活動化グループの中の、コミットメント制御のもとで変更されるリソースに関する情報が入れられます。コミットメント定義中のコミットメント制御情報は、システムによりコミットメント・リソース変更として維持管理されます。

STRCMTCTL コマンドは、ファイルのコミットメント制御を自動的に開始するわけではありません。このファイルも、ILE COBOL プログラム内の I-O-CONTROL 段落の COMMITMENT CONTROL 文節に指定しなければなりません。コミットメント制御環境を正常終了する際には、コミットメント制御の終了 (ENDCMTCTL) コマンドを使用してください。このコマンドを使用すると、コミットメント制御下のデータベース・ファイルに関する未コミット変更がすべて取り消されます。 (暗黙の ROLLBACK が処理されます。) STRCMTCTL コマンドおよび ENDCMTCTL コマンドの詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「プログラミング」カテゴリーの中の『*CL および API*』セクションを参照してください。

コミットメント制御の詳細については、「*Recovering your system*」を参照してください。

注: 変更済みの未コミット・データを読み取れないようにする機能はコミットメント制御の機能なので、コミットメント制御のもとで実行している場合にしか使用できません。コミットメント制御を拡張しても通常の (未コミットの) データベース・サポートを変更することはできず、入力専用にオープンしたファイルを読み取る場合にロックされたレコードも読み取ることが可能です。ファイルを使用する際に一貫性を保つようにしてください。ファイルは常に、コミットメント制御のもとで実行するか、またはコミットメント制御のもとでは実行しないかを明確にしておいてください。

注: コミットメント制御は、ブロッキングがファイルのレコードに対して実行されていない場合に、有効なだけです。 OVRDBF コマンドに SEQONLY(*NO) を

指定すると、実行時にブロック化を防ぐことできます。ブロック化について
は、 466 ページの『入力レコードの非ブロック化および出力レコードのブロック化』を参照してください。
#

コミットメント制御の有効範囲の指定

ジョブ中で実行されているプログラム・オブジェクトによって、複数のコミットメント定義が開始され使用される場合があります。ジョブの個々のコミットメント定義により、関連リソースのある個々のトランザクションが識別されます。それらのリソースは、ジョブで開始された他のすべてのコミットメント定義には関係なくコミットしたりロールバックしたりすることができます。

コミットメント定義の**有効範囲**は、ジョブ内で実行されているプログラムのうち、どのプログラムがそのコミットメント定義を使用するかを指定するものです。コミットメント定義の有効範囲は次の 2 つです。

- 活動化グループ・レベル
- ジョブ・レベル

コミットメント定義の有効範囲は、STRCMTCTL コマンドの CMTSCOPE パラメーターで指定します。

コミットメント定義のデフォルトの有効範囲は、STRCMTCTL コマンドを出すプログラムの活動化グループです。 その活動化グループ中で実行するプログラム・オブジェクトだけがコミットメント定義を使用します。 OPM デフォルト活動化グループに対して活動化グループ・レベルで開始されたコミットメント定義は、デフォルト活動化グループ (*DFTACTGRP) コミットメント定義として認識されます。個々の活動化グループには、それぞれ独自のコミットメント定義があります。

コミットメント定義をジョブに限定することもできます。活動化グループのコミットメント定義が活動化グループ・レベルで開始されていない場合、その活動化グループ中で実行しているプログラム・オブジェクトは、ジョブ・レベルのコミットメント定義を使用します。このことは、ジョブに関する他のプログラム・オブジェクトがすでにジョブ・レベルのコミットメント定義を開始している場合に当てはまります。ジョブごとに開始できるジョブ・レベル・コミットメント定義は 1 つだけです。

特定の活動化グループについて、その活動化グループ中で実行するプログラム・オブジェクトが使用できるコミットメント定義は 1 つだけです。活動化グループ中で実行されるプログラム・オブジェクトは、コミットメント定義をジョブ・レベルでも活動化グループ・レベルでも使用できます。しかし、同時に両方のレベルのコミットメント定義を使用することはできません。

ILE COBOL プログラムがコミットメント制御の操作を実行する場合、要求用に使用するコミットメント定義が直接指定されるわけではありません。その代わりに、要求を行うプログラム・オブジェクトが実行している活動化グループに基づいて、使用されるコミットメント定義をシステムが判別します。

コミットメント定義が ILE 活動化グループに限定されている場合、活動化グループが正常終了すると、その定義に関連するファイルはクローズされ、暗黙のうちにコミットされます。活動化グループが異常終了すると、活動化グループに限定されたコミットメント定義の関連ファイルは、ロールバックされてクローズされます。

コミットメント制御の有効範囲の詳細については、「*ILE 概念*」を参照してください。

コミットメント制御を使用する例

475 ページの図 102 に、銀行の環境でコミットメント制御を使用する例を示します。プログラムは、トランザクションを処理して、ある口座から別の口座に送金します。トランザクション中に問題が起きなければ、変更内容がデータベース・ファイルにコミットされます。口座番号が不適切だったり金額が足りなかったりして送金が行われないと、ROLLBACK が出されて変更内容が取り消されます。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ACCOUNT MASTER PHYSICAL FILE -- ACCTMST
     A
     A                                      UNIQUE
     A          R ACCNTREC
     A            ACCNTKEY       5S
     A            NAME          20
     A            ADDR          20
     A            CITY          20
     A            STATE          2
     A            ZIP            5S
     A            BALANCE       10S 2
     A          K ACCNTKEY
```

図 100. コミットメント制御の使用例 - アカウント・マスター・ファイル DDS

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* PROMPT SCREEN FILE NAME 'ACCTFMTS'
     A*
     A                                   1 INDARA
     A          R ACCTPMT
     A                                     TEXT('CUSTOMER ACCOUNT PROMPT')
     A
     A                                     CA01(15 'END OF PROGRAM')
     A                                     PUTRETAIN OVERLAY
     A                                   1  3'ACCOUNT MASTER UPDATE'
     A                                   3  3'FROM ACCOUTN NUMBER'
     A            ACCTFROM       5Y 0I  3 23CHECK(ME)
     A  99                                 ERRMSG('INVALID FROM ACCOUNT +
     A                                     NUMBER' 99)
     A  98                                 ERRMSG('INSUFFICIENT FUNDS IN FROM +
     A                                     ACCOUNT' 98)
     A                                   4  3'TO ACCOUNT NUMBER'
     A            ACCTTO         5Y 0I  4 23CHECK(ME)
     A  97                                 ERRMSG('INVALID TO ACCOUNT +
     A                                     NUMBER' 97)
     A                                   5  3'AMOUNT TRANSFERRED'
     A            TRANSAMT      10Y02I  5 23
     A          R ERRFMT
     A  96                               6  5'INVALID FILE STATUS'
     A  95                               7  5'INVALID KEY IN REWRITE'
     A  94                               8  5'EOF CONDITION IN READ'
```

図 101. コミットメント制御の使用例 - プロンプト画面 DDS

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/ACCOUNT         ISERIES1   06/02/15 13:53:23        ページ    2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     ACCOUNT.
    3     000300 ENVIRONMENT DIVISION.
    4     000400 CONFIGURATION SECTION.
    5     000500 SOURCE-COMPUTER. IBM-ISERIES.
    6     000600 OBJECT-COMPUTER. IBM-ISERIES.
    7     000700 INPUT-OUTPUT SECTION.
    8     000800 FILE-CONTROL.
    9     000900     SELECT ACCOUNT-FILE  ASSIGN TO DATABASE-ACCTMST
   11     001000         ORGANIZATION IS INDEXED
   12     001100         ACCESS IS DYNAMIC
   13     001200         RECORD IS EXTERNALLY-DESCRIBED-KEY
   14     001300         FILE STATUS IS ACCOUNT-FILE-STATUS.
   15     001400     SELECT DISPLAY-FILE ASSIGN TO WORKSTATION-ACCTFMTS-SI  1
   17     001500         ORGANIZATION IS TRANSACTION.
          001600*********************************************************************
   18     001700 I-O-CONTROL.
   19     001800     COMMITMENT CONTROL FOR ACCOUNT-FILE.  2
          001900*********************************************************************
   20     002000 DATA DIVISION.
   21     002100 FILE SECTION.
   22     002200 FD  ACCOUNT-FILE.
   23     002300 01  ACCOUNT-RECORD.
          002400     COPY DDS-ALL-FORMATS OF ACCTMST.
   24    +000001       05  ACCTMST-RECORD PIC X(82).                                <-ALL-FMTS
         +000002*    I-O FORMAT:ACCNTREC   FROM FILE ACCTMST    OF LIBRARY CBLGUIDE  <-ALL-FMTS
         +000003*                                                                   <-ALL-FMTS
         +000004*THE KEY DEFINITIONS FOR RECORD FORMAT  ACCNTREC                    <-ALL-FMTS
         +000005*  NUMBER              NAME               RETRIEVAL       ALTSEQ   <-ALL-FMTS
         +000006*   0001   ACCNTKEY                      ASCENDING         NO      <-ALL-FMTS
   25    +000007       05  ACCNTREC      REDEFINES ACCTMST-RECORD.                 <-ALL-FMTS
   26    +000008           06 ACCNTKEY              PIC S9(5).                     <-ALL-FMTS
   27    +000009           06 NAME                  PIC X(20).                     <-ALL-FMTS
   28    +000010           06 ADDR                  PIC X(20).                     <-ALL-FMTS
   29    +000011           06 CITY                  PIC X(20).                     <-ALL-FMTS
   30    +000012           06 STATE                 PIC X(2).                      <-ALL-FMTS
   31    +000013           06 ZIP                   PIC S9(5).                     <-ALL-FMTS
   32    +000014           06 BALANCE               PIC S9(8)V9(2).                <-ALL-FMTS
          002500
   33     002600 FD  DISPLAY-FILE.
   34     002700 01  DISPLAY-REC.
          002800     COPY DDS-ALL-FORMATS OF ACCTFMTS.
   35    +000001       05  ACCTFMTS-RECORD PIC X(20).                               <-ALL-FMTS
         +000002*  INPUT FORMAT:ACCTPMT    FROM FILE ACCTFMTS   OF LIBRARY CBLGUIDE  <-ALL-FMTS
         +000003*                          CUSTOMER ACCOUNT PROMPT                  <-ALL-FMTS
   36    +000004       05  ACCTPMT-I     REDEFINES ACCTFMTS-RECORD.                <-ALL-FMTS
   37    +000005           06 ACCTFROM              PIC S9(5).                     <-ALL-FMTS
   38    +000006           06 ACCTTO                PIC S9(5).                     <-ALL-FMTS
   39    +000007           06 TRANSAMT              PIC S9(8)V9(2).                <-ALL-FMTS
         +000008* OUTPUT FORMAT:ACCTPMT    FROM FILE ACCTFMTS   OF LIBRARY CBLGUIDE  <-ALL-FMTS
         +000009*                          CUSTOMER ACCOUNT PROMPT                  <-ALL-FMTS
         +000010*      05  ACCTPMT-O     REDEFINES ACCTFMTS-RECORD.                <-ALL-FMTS
         +000011*  INPUT FORMAT:ERRFMT     FROM FILE ACCTFMTS   OF LIBRARY CBLGUIDE  <-ALL-FMTS
```

図 102. コミットメント制御の使用例 (1/4)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/ACCOUNT         ISERIES1   06/02/15 13:53:23        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
        +000012*                                                                            <-ALL-FMTS
        +000013*      05  ERRFMT-I      REDEFINES ACCTFMTS-RECORD.                         <-ALL-FMTS
        +000014* OUTPUT FORMAT:ERRFMT     FROM FILE ACCTFMTS   OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000015*                                                                            <-ALL-FMTS
        +000016*      05  ERRFMT-O      REDEFINES ACCTFMTS-RECORD.                         <-ALL-FMTS
         002900
   40    003000 WORKING-STORAGE SECTION.
   41    003100 77  ACCOUNT-FILE-STATUS           PIC X(2).
   42    003200 77  IND-ON                       PIC 1    VALUE B"1".
   43    003300 77  IND-OFF                      PIC 1    VALUE B"0".
   44    003400 01  DISPFILE-INDICS.
         003500     COPY DDS-ALL-FORMATS-INDIC OF ACCTFMTS.  3
   45   +000001       05  ACCTFMTS-RECORD.                                                 <-ALL-FMTS
        +000002*  INPUT FORMAT:ACCTPMT    FROM FILE ACCTFMTS   OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000003*                          CUSTOMER ACCOUNT PROMPT                           <-ALL-FMTS
   46   +000004           06 ACCTPMT-I-INDIC.                                              <-ALL-FMTS
   47   +000005                07 IN15             PIC 1  INDIC 15.                      <-ALL-FMTS
        +000006*                          END OF PROGRAM                                    <-ALL-FMTS
   48   +000007                07 IN97             PIC 1  INDIC 97.                      <-ALL-FMTS
        +000008*                          INVALID TO ACCOUNT NUMBER                         <-ALL-FMTS
   49   +000009                07 IN98             PIC 1  INDIC 98.                      <-ALL-FMTS
        +000010*                          INSUFFICIENT FUNDS IN FROM ACCOUNT                <-ALL-FMTS
   50   +000011                07 IN99             PIC 1  INDIC 99.                      <-ALL-FMTS
        +000012*                          INVALID FROM ACCOUNT NUMBER                       <-ALL-FMTS
        +000013* OUTPUT FORMAT:ACCTPMT    FROM FILE ACCTFMTS   OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000014*                          CUSTOMER ACCOUNT PROMPT                           <-ALL-FMTS
   51   +000015           06 ACCTPMT-O-INDIC.                                              <-ALL-FMTS
   52   +000016                07 IN97             PIC 1  INDIC 97.                      <-ALL-FMTS
        +000017*                          INVALID TO ACCOUNT NUMBER                         <-ALL-FMTS
   53   +000018                07 IN98             PIC 1  INDIC 98.                      <-ALL-FMTS
        +000019*                          INSUFFICIENT FUNDS IN FROM ACCOUNT                <-ALL-FMTS
   54   +000020                07 IN99             PIC 1  INDIC 99.                      <-ALL-FMTS
        +000021*                          INVALID FROM ACCOUNT NUMBER                       <-ALL-FMTS
        +000022*  INPUT FORMAT:ERRFMT     FROM FILE ACCTFMTS   OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000023*                                                                            <-ALL-FMTS
        +000024*          06 ERRFMT-I-INDIC.                                                <-ALL-FMTS
        +000025* OUTPUT FORMAT:ERRFMT     FROM FILE ACCTFMTS   OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000026*                                                                            <-ALL-FMTS
   55   +000027           06 ERRFMT-O-INDIC.                                               <-ALL-FMTS
   56   +000028                07 IN94             PIC 1  INDIC 94.                      <-ALL-FMTS
   57   +000029                07 IN95             PIC 1  INDIC 95.                      <-ALL-FMTS
   58   +000030                07 IN96             PIC 1  INDIC 96.                      <-ALL-FMTS
         003600
   59    003700 PROCEDURE DIVISION.
   60    003800 DECLARATIVES.
         003900 ACCOUNT-ERR-SECTION SECTION.
         004000     USE AFTER STANDARD EXCEPTION PROCEDURE ON ACCOUNT-FILE.
         004100 ACCOUNT-ERR-PARAGRAPH.
   61    004200     IF ACCOUNT-FILE-STATUS IS NOT EQUAL "23" THEN
   62    004300         MOVE IND-ON TO IN96 OF ERRFMT-O-INDIC  4
         004400     ELSE
   63    004500         MOVE IND-ON TO IN95 OF ERRFMT-O-INDIC  5
         004600     END-IF
   64    004700     WRITE DISPLAY-REC FORMAT IS "ERRFMT"
         004800           INDICATORS ARE ERRFMT-O-INDIC
```

図 102. コミットメント制御の使用例 (2/4)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/ACCOUNT         ISERIES1   06/02/15 13:53:23        ページ    4
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
        004900     END-WRITE
   65   005000     CLOSE DISPLAY-FILE
        005100           ACCOUNT-FILE.
   66   005200     STOP RUN.
        005300
        005400 DISPLAY-ERR-SECTION SECTION.
        005500     USE AFTER STANDARD EXCEPTION PROCEDURE ON DISPLAY-FILE.
        005600 DISPLAY-ERR-PARAGRAPH.
   67   005700     MOVE IND-ON TO IN94 OF ERRFMT-O-INDIC
   68   005800     WRITE DISPLAY-REC FORMAT IS "ERRFMT"
        005900          INDICATORS ARE ERRFMT-O-INDIC
        006000     END-WRITE
   69   006100     CLOSE DISPLAY-FILE
        006200           ACCOUNT-FILE.
   70   006300     STOP RUN.
        006400 END DECLARATIVES.
        006500
        006600 MAIN-PROGRAM SECTION.
        006700 MAINLINE.
   71   006800     OPEN I-O DISPLAY-FILE
        006900          I-O ACCOUNT-FILE.
   72   007000     MOVE ZEROS TO ACCTPMT-I-INDIC
        007100                   ACCTPMT-O-INDIC.
   73   007200     PERFORM WRITE-READ-DISPLAY.
   74   007300     PERFORM VERIFY-ACCOUNT-NO UNTIL IN15 EQUAL IND-ON.
   75   007400     CLOSE DISPLAY-FILE
        007500           ACCOUNT-FILE.
   76   007600     STOP RUN.
        007700
        007800 VERIFY-ACCOUNT-NO.
   77   007900     PERFORM VERIFY-TO-ACCOUNT.
   78   008000     IF IN97 OF ACCTPMT-O-INDIC EQUAL IND-OFF THEN
   79   008100         PERFORM VERIFY-FROM-ACCOUNT.
   80   008200     PERFORM WRITE-READ-DISPLAY.
        008300
        008400 VERIFY-FROM-ACCOUNT.
   81   008500     MOVE ACCTFROM TO ACCNTKEY.
   82   008600     READ ACCOUNT-FILE
   83   008700         INVALID KEY MOVE IND-ON TO IN99 OF ACCTPMT-O-INDIC
        008800     END-READ
   84   008900     IF IN99 OF ACCTPMT-O-INDIC EQUAL IND-ON THEN  6
        009000*
   85   009100         ROLLBACK
        009200*
        009300     ELSE
   86   009400         PERFORM UPDATE-FROM-ACCOUNT
        009500     END-IF.
        009600
        009700 VERIFY-TO-ACCOUNT.
   87   009800     MOVE ACCTTO TO ACCNTKEY.
   88   009900     READ ACCOUNT-FILE
   89   010000         INVALID KEY MOVE IND-ON TO IN97 OF ACCTPMT-O-INDIC  7
        010100     END-READ
   90   010200     IF IN97 OF ACCTPMT-O-INDIC EQUAL IND-ON THEN
        010300*
```

図 102. コミットメント制御の使用例 (3/4)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/ACCOUNT        ISERIES1   06/02/15 13:53:23        ページ     5
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   91     010400         ROLLBACK  8
          010500*
          010600     ELSE
   92     010700         PERFORM UPDATE-TO-ACCOUNT
          010800     END-IF.
          010900
          011000 UPDATE-TO-ACCOUNT.
   93     011100     ADD TRANSAMT TO BALANCE.
   94     011200     REWRITE ACCOUNT-RECORD.
          011300
          011400 UPDATE-FROM-ACCOUNT.
   95     011500     SUBTRACT TRANSAMT FROM BALANCE.
   96     011600     REWRITE ACCOUNT-RECORD.
   97     011700     IF BALANCE IS LESS THAN 0 THEN
   98     011800         MOVE IND-ON TO IN98 OF ACCTPMT-O-INDIC
          011900*
   99     012000         ROLLBACK  9
          012100*
          012200     ELSE
          012300*
  100     012400         COMMIT  10
          012500*
          012600     END-IF.
          012700
          012800 WRITE-READ-DISPLAY.
  101     012900     WRITE DISPLAY-REC FORMAT IS "ACCTPMT"
          013000         INDICATORS ARE ACCTPMT-O-INDIC  11
          013100     END-WRITE
  102     013200     MOVE ZEROS TO ACCTPMT-I-INDIC
          013300                   ACCTPMT-O-INDIC.
  103     013400     READ DISPLAY-FILE RECORD
          013500         INDICATORS ARE ACCTPMT-I-INDIC
          013600     END-READ.
          013700
          013800
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 102. コミットメント制御の使用例 (4/4)

1 プログラム用に別個の標識域を用意します。

2 COMMITMENT CONTROL 文節で、コミットメント制御のもとに置かれるファイルを指定します。この文節で指定されているファイルは、COMMIT verb と ROLLBACK verb の影響を受けます。

3 形式 2 の COPY ステートメント中に標識属性 INDIC を指定することにより、プログラムの中で標識を使用するためのデータ記述項目を WORKING-STORAGE の中に定義します。

4 無効なファイル状況の場合に IN96 が設定されます。

5 REWRITE 操作で INVALID KEY 条件の場合に IN95 が設定されます。

6 入力された口座番号が送金元の口座として無効な場合には、IN99 が設定されます。

7 入力された口座番号が送金先の口座として無効な場合には、IN97 が設定されます。

8 READ において INVALID KEY 条件が発生すると、ROLLBACK が使用され、最初の READ の後のレコードに対するレコード・ロックが解除されます。

9 送金できない (標識が設定されている) 場合には、ROLLBACK ステートメントが処理されます。コミットメント制御下のデータベース・ファイルに加えられた変更は、すべて取り消されます。

10 送金が有効だった (標識が設定されていない) 場合には、COMMIT ステー

トメントが処理され、コミットメント制御下のデータベース・ファイルに加えられた変更は、すべて永続的な変更になります。

11 標識で制御されるワークステーションのディスプレイ装置に関するオプションに、INDICATORS 句が必要です。

ファイルのソートおよびマージ

レコードを特定の順序に並べ替えることは、データ処理における一般的な要件です。このようなレコードの順序付けは、ソート操作やマージ操作を使用して行うことができます。

- **ソート**操作は、順序が正しくない入力を受け入れて、指定された順序で出力します。
- **マージ**操作は、2 つ以上のソート済みのファイルを比較して、それらのファイルを順序を保って結合します。

ファイルのソートやマージを実行する際には、次の処理を行う必要があります。

1. 必要に応じて、ソートまたはマージのための入出力ファイルを記述する。
   - それには、INPUT-OUTPUT SECTION の FILE-CONTROL 段落でファイルを選択し、DATA DIVISION の FILE SECTION 中で FD (ファイル記述) 項目を使用してファイルを記述します。
2. ソート・ファイルおよびマージ・ファイルを記述する。
   - それには、INPUT-OUTPUT SECTION の FILE-CONTROL 段落でソート・ファイルまたはマージ・ファイルを選択し、DATA DIVISION の FILE SECTION 中で SD (ソート記述) 項目を使用してファイルを記述します。
3. ソート操作またはマージ操作を指定する。
   - それには、PROCEDURE DIVISION の SORT ステートメントまたは MERGE ステートメントを実行します。

ファイルの記述

ソート・ファイルおよびマージ・ファイルは、環境部の SELECT ステートメントとデータ部の SD (ソート記述) 項目を使用して記述する必要があります。例として 480 ページの図 103 を参照してください。 SD 項目に記述されるソート・ファイルまたはマージ・ファイルは、ソートまたはマージの操作中に使用される作業用ファイルです。このファイルについて入出力ステートメントを実行することはできません。

ソートまたはマージの操作の入出力用に使用されるファイルを記述するには、データ部で FD (ファイル記述) 項目を指定してください。 WORKING-STORAGE SECTION か LINKAGE SECTION だけに定義されているレコードのソートまたはマージを行うこともできます。 WORKING-STORAGE SECTION または LINKAGE SECTION のデータ項目のソートまたはマージだけを行い、ソートまたはマージ操作の入出力としてファイルを使用しない場合でも、ソート・ファイルまたはマージ・ファイルに SD 項目と FILE-CONTROL 項目が必要です。

どの SD 項目にもレコード記述が含まれていなければなりません。たとえば、次のようにします。

```
SD  SORT-WORK-1.
01  SORT-WORK-1-AREA.
    05  SORT-KEY-1      PIC X(10).
    05  SORT-KEY-2      PIC X(10).
    05  FILLER          PIC X(80).
```

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/SMPLSORT       ISERIES1   06/02/15 13:54:42       ページ     2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.  SMPLSORT.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600 SOURCE-COMPUTER. IBM-ISERIES.
    6     000700 OBJECT-COMPUTER. IBM-ISERIES.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
          001000*
          001100*  ソート・ファイルが文書として扱われるように名前を割り当てる
          001200*
    9     001300     SELECT SORT-WORK-1
   10     001400            ASSIGN TO DISK-SORTFILE1.
   11     001500     SELECT SORT-WORK-2
   12     001600            ASSIGN TO DISK-SORTFILE1.
   13     001700     SELECT INPUT-FILE
   14     001800            ASSIGN TO DISK-INFILE.
          001900
   15     002000 DATA DIVISION.
   16     002100 FILE SECTION.
   17     002200 SD  SORT-WORK-1.
   18     002300 01  SORT-WORK-1-AREA.
   19     002400     05  SORT-KEY-1             PIC X(10).
   20     002500     05  SORT-KEY-2             PIC X(10).
   21     002600     05  FILLER                 PIC X(80).
          002700
   22     002800 SD  SORT-WORK-2.
   23     002900 01  SORT-WORK-2-AREA.
   24     003000     05  SORT-KEY               PIC X(5).
   25     003100     05  FILLER                 PIC X(25).
          003200
   26     003300 FD INPUT-FILE.
   27     003400 01  INPUT-RECORD                PIC X(100).
          003500
          003600*    .
          003700*    .
          003800*    .
          003900
   28     004000 WORKING-STORAGE SECTION.
   29     004100 01  EOS-SW                      PIC X.
   30     004200 01  FILLER.
   31     004300     05  TABLE-ENTRY  OCCURS 100 TIMES
          004400                      INDEXED BY X1       PIC X(30).
          004500*    .
          004600*    .
          004700*    .
          004800
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 103. ソート・プログラムの ENVIRONMENT DIVISION と DATA DIVISION の項目

ソート・ファイルおよびマージ・ファイルは、手続き部 (PROCEDURE DIVISION) の SORT ステートメントまたは MERGE ステートメントを使用して処理します。このステートメントでは、レコードのフィールドのうち、ソートまたはマージの順序を指定するキー・フィールドを指定します。キーは昇順か降順に指定することができ、また複数キーを指定した場合にはその 2 つを組み合わせて使用するよう指定することができます。

同一の ILE COBOL プログラムで SORT ステートメントと MERGE ステートメントを混用できます。 1 つの ILE COBOL プログラムにいくつでもソート操作またはマージ操作を指定でき、そのおのおのに独自の入力プロシージャーまたは出力プロシージャーを指定できます。

ILE COBOL プログラム中では、次の操作を含む複数のソート操作またはマージ操作を実行できます。
- 同一のソート操作またはマージ操作を複数回呼び出す。
- 複数の異なるソート操作またはマージ操作を行う。

ただし、1 つの操作を完了してから別の操作を開始しなければなりません。

ファイルのソート

ソート操作は、順序が正しくない入力を受け入れて、指定された順序で出力します。

SORT...INPUT PROCEDURE ステートメントを使用してソートする**前に**、ソート・レコードに対して実行する**入力**プロシージャーを指定することができます。

SORT...OUTPUT PROCEDURE ステートメントを使用してソートした**後に**、ソート・レコードに対して実行する**出力**プロシージャーを指定することができます。

入力または出力プロシージャーは、レコードの追加、削除、変更、編集、またはその他の修正を行うのに使用します。

SORT ステートメントを使用して、次の処理を行うことができます。
- WORKING-STORAGE SECTION または LINKAGE SECTION 中のデータ項目 (テーブルを含む) をソートする。
- SORT...USING ステートメントを使用して、事前処理なしで、レコードを読み込んで新しいファイルの中に直接入れる。
- SORT...GIVING ステートメントを使用して、追加処理なしで、ソートされたレコードをファイルに直接転送する。

多くの場合、ソート操作を含む ILE COBOL プログラムは、1 つの入力プロシージャーによって 1 つまたは複数の入力ファイルが読み取られて操作されるように編成されています。入力プロシージャーの中では、RELEASE ステートメントによってレコードをソート・ファイル中に入れます。ソート操作を開始する前にレコードの変更や処理を行わない場合は、SORT ステートメントの USING 句を使用して、指定された入力ファイルの未変更のレコードを新しいファイルに入れます。

ソート操作の完了後に、RETURN ステートメントを使用して、ソートされたレコードを出力プロシージャーの中で一度に 1 つずつ変更することができます。レコードの変更や処理を行わない場合は、SORT ステートメントの GIVING 句を使用して、出力ファイルの名前を指定し、ソートされたレコードを出力ファイルに書き出します。

SORT、RELEASE、および RETURN ステートメントについては、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

ファイルのマージ

マージ操作は、2 つ以上のソート済みのファイルを比較して、それらのファイルを順序を保って結合します。

マージ操作の後に使用される出力プロシージャーにアクセスすることにより、MERGE...OUTPUT PROCEDURE ステートメントを使用して出力レコードを変更できます。

SORT ステートメントの場合と異なり、MERGE ステートメントでは入力プロシージャーは指定できません。 MERGE...USING ステートメントを使用しなければなりません。

マージ操作の前に入力ファイルを順番に並べる必要はありません。マージ操作が、それらのファイルを順番に並べて結合し、順序よく並んだ 1 つのファイルにします。

手続き部 (PROCEDURE DIVISION) に MERGE ステートメントがあると、マージ処理が開始されます。このマージ操作は入力ファイルのレコードのキーを比較し、出力プロシージャーの RETURN ステートメントまたは GIVING 句に名前を指定されているファイルに、順序よくレコードを一度に 1 つずつ渡します。

マージ済みレコードを処理したい場合には、出力プロシージャーの中で RETURN ステートメントを使用することによって、ILE COBOL プログラムで一度に 1 つずつ処理できます。マージ済みレコードの変更や処理を行わない場合には、MERGE ステートメントの GIVING 句を使用することにより、マージ済みレコードの書き込み先としてマージ出力ファイルの名前を指定できます。

ソート基準の指定

SORT ステートメントでは、レコードをソートする基準となるキーを指定します。そのキーは、ソートされるレコードのレコード記述の中に定義されていなければなりません。次の例では、SORT-GRID-LOCATION と SORT-SHIFT を SORT ステートメント中で使用する前に、それらを DATA DIVISION で定義していることに注目してください。

```
 DATA DIVISION.

⋮
 SD  SORT-FILE.
 01  SORT-RECORD.
     05  SORT-KEY.
         10  SORT-SHIFT               PIC X(1).
         10  SORT-GRID-LOCATION       PIC X(2).
         10  SORT-REPORT              PIC X(3).
     05  SORT-EXT-RECORD.
         10  SORT-EXT-EMPLOYEE-NUM    PIC X(6).
         10  SORT-EXT-NAME            PIC X(30).
         10  FILLER                   PIC X(73).
 PROCEDURE DIVISION.

⋮
       SORT SORT-FILE
            ON ASCENDING KEY SORT-GRID-LOCATION SORT-SHIFT
            INPUT PROCEDURE 600-SORT3-INPUT
            OUTPUT PROCEDURE 700-SORT3-OUTPUT.

⋮
```

上記の例のように、複数のキーをソートする場合には、重要性の順で降順にキーを指定してください。上記の例には、入力プロシージャーと出力プロシージャーの使

用方法も示されています。レコードを処理してからソートしたい場合は入力プロシージャーを使用し、レコードをソートしてから追加の処理を行いたい場合は出力プロシージャーを使用してください。

ソート・キーの長さに関する制約事項

キーの最大数は制限されていませんが、キーの全長が 2000 バイトを超えてはなりません。

浮動小数点数に関する考慮事項

SORT (および MERGE 操作) のキー・データ項目が浮動小数点数である場合があります。キーが外部浮動小数点項目の場合は、文字データとして扱われます。したがって、レコードのソート順序は、使用する照合順序によって決まります。キーが内部浮動小数点項目の場合は、順序は数値順になります。

日時データ・タイプに関する考慮事項

キー・データ項目は、SORT および MERGE の両操作の場合、日時クラスのデータ
項目であることがあります。一般に、日時クラスの項目は、項目が i5/OS DDS の
日付、時刻、およびタイム・スタンプの形式と一致している、日付、時刻、および
タイム・スタンプ項目としてのみソートされます。その他の場合はすべて、文字デ
ータとして処理されます。 文字データとして扱われる日時クラスの項目は、SORT
または MERGE の操作中には有効な照合順序を無視します。 ILE COBOL プログ
ラムで日時データ・タイプを使用する際の詳細については 212 ページの『日時データ・タイプを処理する』を参照してください。

ソートの場合に日時項目として扱われる DDS データ・タイプを、以下に示します。

- DATE 形式 *MDY
- DATE 形式 *DMY
- DATE 形式 *EUR
- DATE 形式 *USA
- TIME 形式 *USA

ヌル値に関する考慮事項

キー・データ項目は、SORT および MERGE の両操作の場合、ヌル値を持つことがあります。データベース・ファイルでは、照合順序の最高値をヌル値が占めます。ただし、ヌル値を含むヌル可能ファイルの SORT および MERGE を行えるようにするには、まず、ASSIGN 文節に ALWNULL キーワードを指定して、ファイルをヌル可能と定義する必要があります。

代替照合順序

EBCDIC、ASCII、または他の照合順序に基づいてレコードをソートできます。デフォルトの照合順序は EBCDIC か、CONFIGURATION SECTION に指定した PROGRAM COLLATING SEQUENCE です。 SORT ステートメントの COLLATING SEQUENCE 句を使用すると、PROGRAM COLLATING SEQUENCE で名前を指定した照合順序を指定変更できます。その結果、さまざまな照合順序を使用して、プログラム中で複数のソートを行えます。

プログラムが実行時に使用する照合順序を、ILE COBOL ソース・プログラムのコンパイル時に指定することもできます。使用される照合順序を指定するには、

CRTCBLMOD コマンドおよび CRTBNDCBL コマンドの SRTSEQ パラメーターと LANGID パラメーターを使用します。コンパイル時に照合順序を指定する方法の説明については 57 ページの『CRTCBLMOD の国別言語ソート・シーケンスの指定』を参照してください。コンパイル時に指定された照合順序を指定変更するには、OBJECT-COMPUTER 段落に PROGRAM COLLATING SEQUENCE 文節を指定するか、または SORT ステートメントの COLLATING SEQUENCE 句を使用します。

ASCII ファイルをソートする際には、ASCII 照合順序を要求する必要があります。この要求を行うには、SORT ステートメントの COLLATING SEQUENCE *英字名* 句を使用してください (*英字名* は SPECIAL-NAMES 段落で STANDARD-1 として定義されているものです)。 SORT ステートメントまたは SORT ステートメントを指定変更する MERGE ステートメントに COLLATING SEQUENCE 句が指定されていない場合は、OBJECT-COMPUTER 段落の PROGRAM COLLATING SEQUENCE 文節に名前を指定することもできます。

入力プロシージャーの作成

入力ファイルのレコードを、ソート・プログラムに渡す前に処理する必要がない場合は、SORT...USING を使用します。 SORT...USING *ファイル名* を使用した場合、そのファイルをオープンし、レコードを読み取り、そのレコードをソート・プログラムに渡し、そのファイルをクローズするための入力プロシージャーが ILE COBOL コンパイラーによって生成されます。

SORT ステートメントが実行されている時は入力ファイルをオープンしないでください。入力ファイルのレコードを、ソート・プログラムに渡す前に処理したい場合は、SORT ステートメントの INPUT PROCEDURE オプションを使用してください。

各入力プロシージャーは、段落またはセクションのどちらかでなければなりません。たとえば、WORKING-STORAGE セクションのテーブルのレコードをソート・ファイルに渡すには、以下のようにします。

```
 PROCEDURE DIVISION.

⋮
       SORT SORT-FILE
            ON ASCENDING KEY SORT-KEY
            INPUT PROCEDURE 600-SORT3-INPUT-PROC

⋮
 600-SORT3-INPUT-PROC SECTION.
       PERFORM WITH TEST AFTER
               VARYING X1 FROM 1 BY 1 UNTIL X1 = 100
            RELEASE SORT-RECORD  FROM  TABLE-ENTRY(X1)
       END-PERFORM.
```

入力プロシージャーには、レコードを処理してソート操作に渡すためのコードが含まれます。入力プロシージャーを使用して、次の処理を行えます。

- データ項目を WORKING-STORAGE セクションからソート・ファイルに渡す (RELEASE)。
- プログラム中のどこかですでに読み取られているレコードを渡す。

- レコードを入力ファイルから読み取り、選択か処理を行い、ソート・ファイルに渡す。

レコードをソート・ファイルに転送する際には、すべての入力プロシージャーに少なくとも 1 つの RELEASE または RELEASE FROM ステートメントが含まれていなければなりません。

出力プロシージャーの作成

ソートされたレコードをソート・ファイルから追加の処理を行わずに別のファイルに転送する場合は、SORT...GIVING を使用します。 SORT...GIVING *ファイル名* を使用した場合、そのファイルをオープンし、レコードを戻し、そのレコードを書き込み、そのファイルをクローズするための出力プロシージャーが ILE COBOL コンパイラーによって生成されます。 SORT ステートメントの実行時には、GIVING 句で指定したファイルがオープンされていてはなりません。

ソートされたレコードの選択、編集、または他の変更を行ってから、そのレコードをソート作業ファイルから別のファイル中に書き出したい場合は、SORT ステートメントの OUTPUT PROCEDURE 句を使用してください。

出力プロシージャーでは、RETURN ステートメントを使用して、ソート済みの各レコードをその出力プロシージャーで使用できるようにしなければなりません。出力プロシージャーには、RETURN ステートメントによって、使用可能なレコードを一度に 1 つずつ処理するのに必要なステートメントを入れることができます。

RETURN の代わりに RETURN INTO を使用すれば、WORKING-STORAGE セクションまたは出力域にレコードを戻して処理することができます。 AT END 句を RETURN ステートメントとともに使うこともできます。 AT END 句の中の命令ステートメントは、すべてのレコードがソート・ファイルから戻された後で実行されます。

各出力プロシージャーには少なくとも 1 つの RETURN ステートメントまたは RETURN INTO ステートメントが入っていなければなりません。また、各出力プロシージャーは、セクションまたは段落のどちらかでなければなりません。

入力プロシージャーと出力プロシージャーに関する制約事項

入力プロシージャーおよび出力プロシージャー内のステートメントには、次の制約事項が適用されます。

- 入力プロシージャーと出力プロシージャーには、SORT ステートメントや MERGE ステートメントを入れることはできません。
- 入力プロシージャーと出力プロシージャーには、STOP RUN、EXIT PROGRAM、または GOBACK ステートメントを入れることはできません。
- 別のプログラムに対する CALL ステートメントは含めることができます。ただし、呼び出し先プログラムで、SORT ステートメントまたは MERGE ステートメントを実行することはできません。
- 入力プロシージャーおよび出力プロシージャーでは、ALTER、GO TO、および PERFORM ステートメントを使用して、その入力プロシージャーまたは出力プロシージャーの外のプロシージャー名を参照できます。しかし、GO TO または PERFORM ステートメントの後で入力プロシージャーまたは出力プロシージャー

に戻らなければなりません。 GO TO ステートメントまたは PERFORM ステートメントの結果として実行される COBOL プロシージャーに、SORT ステートメントまたは MERGE ステートメントを入れることはできません。

- PROCEDURE DIVISION の残りの部分で、入力プロシージャーまたは出力プロシージャーの内側への制御の移動を行うことはできません (宣言セクションから制御が戻る場合は例外)。
- ソートまたはマージの操作中には、SD データ項目が使用されます。 RETURN ステートメントが実行される前に、この項目を出力プロシージャーで使用しないでください。

ソートまたはマージが正常に実行されたかどうかの判別

ソート操作またはマージ操作が完了すると、戻りコードまたは完了コードが SORT-RETURN 特殊レジスターに保管されます。 SORT-RETURN 特殊レジスターには、ソート操作またはマージ操作が正常に実行された場合には戻りコード 0 が入り、ソート操作またはマージ操作が失敗した場合には 16 が入ります。

個々の SORT ステートメントまたは MERGE ステートメントが実行されるごとに、SORT-RETURN 特殊レジスターの内容が変更されます。個々の SORT ステートメントまたは MERGE ステートメントごとに、正常終了に関するテストを行う必要があります。たとえば、

```
 PROCEDURE DIVISION.

⋮
       SORT SORT-WORK-2
            ON ASCENDING KEY SORT-KEY
            INPUT PROCEDURE 600-SORT3-INPUT-PROC
            OUTPUT PROCEDURE 700-SORT3-OUTPUT-PROC.
       IF SORT-RETURN NOT EQUAL TO 0
            DISPLAY "SORT ENDED ABNORMALLY. SORT-RETURN = " SORT-RETURN

⋮
 600-SORT3-INPUT-PROC SECTION.

⋮
 700-SORT3-OUTPUT-PROC SECTION.

⋮
```

ソート操作またはマージ操作の未完終了

SORT-RETURN 特殊レジスターを使用するなら、SORT 操作や MERGE 操作を完了前に終了させることができます。エラー宣言または入出力プロシージャー中で SORT-RETURN 特殊レジスターを 16 に設定することによって、すべてのレコードの処理が終わる前にソート操作またはマージ操作を終了することができます。ソート操作またはマージ操作は、レコードが戻されたり渡されたりする前に終了します。その後で制御は次の SORT ステートメントまたは MERGE ステートメントに戻ります。

可変長レコードのソート

可変長レコードが含まれるファイルでは、単一のレコード長ではなく、最小レコード長と最大レコード長があります。

可変長レコードのソートまたはマージを行う場合は、キー・データ名で参照されるすべてのデータ項目が、レコードの最初の *n* 個の文字位置の中に入っていなければなりません (*n* はファイルに指定されたレコードの最小サイズ)。

SORT ステートメントの処理時に、SORT ステートメントに指定されている KEY のレコード長がレコードの最小サイズを超えると、ILE COBOL コンパイラーはエラー・メッセージを出します。

次の場合にはソート・レコードが切り捨てられます。

- 入力ファイル・レコードの最大レコード長が、ソート・ファイル・レコードの最大レコード長より大きい場合
- ソート・ファイル・レコードの最大レコード長が、出力ファイル・レコードの最大レコード長より大きい場合

切り捨てが行われるとコンパイル時エラー・メッセージが出されます。実行時には診断メッセージが出されます。

次の場合には、ソート・レコードにブランクが埋め込まれます。

- 入力ファイル・レコードの最小レコード長が、ソート・ファイル・レコードの最小レコード長より小さい場合
- ソート・ファイル・レコードの最小レコード長が、出力ファイル・レコードの最小レコード長より小さい場合

レコードにブランクが埋め込まれるとコンパイル時通知メッセージが出されます。実行時にはメッセージは出されません。

ファイルのソートおよびマージの例

488 ページの図 104 は、現在の売上と年間の売上のソート・ファイルを作成する方法を示すものです。

まず、現在の売上に関する SORT ステートメントが実行されます。このソート操作に関する入力プロシージャーは SCREEN-DEPT です。レコードは、部門については昇順でソートされ、各部門の純売上 (SALES) については降順でソートされます。次に、このソートに関する出力が印刷されます。

ソート操作が完了すると、現在の売上 (CURRENT-SALES) のレコードと年間の売上 (YTD-SALES) のレコードがマージされます。このファイルのレコードは、部門番号については昇順、各部門の従業員番号については昇順、月 (MONTH) については昇順にマージされ、更新された年間基本マスター・ファイルを作成します。

マージ処理が終了すると、更新された年間マスター・ファイルを印刷します。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/SORTMERG        ISERIES1   06/02/15 13:56:03        ページ     2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.  SORTMERGE.
          000300************************************************************
          000400* THIS IS A SORT/MERGE EXAMPLE USING AN INPUT PROCEDURE *
          000500************************************************************
    3     000600 ENVIRONMENT DIVISION.
    4     000700 CONFIGURATION SECTION.
    5     000800 SOURCE-COMPUTER. IBM-ISERIES.
    6     000900 OBJECT-COMPUTER. IBM-ISERIES.
    7     001000 INPUT-OUTPUT SECTION.
    8     001100 FILE-CONTROL.
    9     001200     SELECT WORK-FILE
   10     001300            ASSIGN TO DISK-WRK.
   11     001400     SELECT CURRENT-SALES-FILE-IN
   12     001500            ASSIGN TO DISK-CURRIN.
   13     001600     SELECT CURRENT-SALES-FILE-OUT
   14     001700            ASSIGN TO DISK-CURROUT.
   15     001800     SELECT YTD-SALES-FILE-IN
   16     001900            ASSIGN TO DISK-YTDIN.
   17     002000     SELECT YTD-SALES-FILE-OUT
   18     002100            ASSIGN TO DISK-YTDOUT.
   19     002200     SELECT PRINTER-OUT
   20     002300            ASSIGN TO PRINTER-PRTSUMM.
          002400
   21     002500 DATA DIVISION.
   22     002600 FILE SECTION.
   23     002700 SD  WORK-FILE.
   24     002800 01  SALES-RECORD.
   25     002900   05 EMPL-NO                 PIC 9(6).
   26     003000   05 DEPT                    PIC 9(2).
   27     003100   05 SALES                   PIC 9(7)V99.
   28     003200   05 NAME-ADDR               PIC X(61).
   29     003300   05 MONTH                   PIC X(2).
   30     003400 FD CURRENT-SALES-FILE-IN.
   31     003500 01 CURRENT-SALES-IN.
   32     003600   05 EMPL-NO                 PIC 9(6).
   33     003700   05 DEPT                    PIC 9(2).
   34     003800      88 ON-SITE-EMPLOYEE     VALUES 0 THRU 6, 8.
   35     003900   05 SALES                   PIC 9(7)V99.
   36     004000   05 NAME-ADDR               PIC X(61).
   37     004100   05 MONTH                   PIC X(2).
   38     004200 FD CURRENT-SALES-FILE-OUT.
   39     004300 01 CURRENT-SALES-OUT.
   40     004400   05 EMPL-NO                 PIC 9(6).
   41     004500   05 DEPT                    PIC 9(2).
   42     004600   05 SALES                   PIC 9(7)V99.
   43     004700   05 NAME-ADDR               PIC X(61).
   44     004800   05 MONTH                   PIC X(2).
   45     004900 FD YTD-SALES-FILE-IN.
   46     005000 01 YTD-SALES-IN.
   47     005100   05 EMPL-NO                 PIC 9(6).
   48     005200   05 DEPT                    PIC 9(2).
   49     005300   05 SALES                   PIC 9(7)V99.
```

図 104. SORT / MERGE の使用例 (1/3)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/SORTMERG        ISERIES1   06/02/15 13:56:03        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   50     005400   05 NAME-ADDR               PIC X(61).
   51     005500   05 MONTH                   PIC X(2).
   52     005600 FD YTD-SALES-FILE-OUT.
   53     005700 01 YTD-SALES-OUT.
   54     005800   05 EMPL-NO                  PIC 9(6).
   55     005900   05 DEPT                     PIC 9(2).
   56     006000   05 SALES                    PIC 9(7)V99.
   57     006100   05 NAME-ADDR               PIC X(61).
   58     006200   05 MONTH                   PIC X(2).
   59     006300 FD PRINTER-OUT.
   60     006400 01 PRINT-LINE.
   61     006500   05 RECORD-LABEL            PIC X(25).
   62     006600   05 DISK-RECORD-DISPLAY     PIC X(80).
          006700
   63     006800 WORKING-STORAGE SECTION.
   64     006900 01 SALES-FILE-IN-EOF-STATUS  PIC X        VALUE "F".
   65     007000    88 SALES-FILE-IN-END-OF-FILE           VALUE "T".
   66     007100 01 SALES-FILE-OUT-EOF-STATUS PIC X        VALUE "F".
   67     007200    88 SALES-FILE-OUT-END-OF-FILE          VALUE "T".
   68     007300 01 YTD-SALES-OUT-EOF-STATUS  PIC X        VALUE "F".
   69     007400    88 YTD-SALES-OUT-END-OF-FILE           VALUE "T".
          007500
   70     007600 PROCEDURE DIVISION.
          007700 MAIN-PROGRAM SECTION.
          007800 MAINLINE.
          007900
   71     008000     OPEN INPUT CURRENT-SALES-FILE-IN
          008100                CURRENT-SALES-FILE-OUT
          008200                YTD-SALES-FILE-OUT
          008300          OUTPUT PRINTER-OUT.
          008400*
          008500*    現在の売上をソート
          008600*
   72     008700     SORT WORK-FILE
          008800          ON ASCENDING KEY DEPT OF SALES-RECORD
          008900          ON DESCENDING KEY SALES OF SALES-RECORD
          009000          INPUT PROCEDURE SCREEN-DEPT
          009100          GIVING CURRENT-SALES-FILE-OUT.
   73     009200     READ CURRENT-SALES-FILE-OUT
   74     009300         AT END SET SALES-FILE-OUT-END-OF-FILE TO TRUE
          009400     END-READ.
   75     009500     PERFORM UNTIL SALES-FILE-OUT-END-OF-FILE
   76     009600         MOVE "SORTED CURRENT SALES "
          009700             TO RECORD-LABEL OF PRINT-LINE
   77     009800         MOVE CURRENT-SALES-OUT TO DISK-RECORD-DISPLAY
   78     009900         WRITE PRINT-LINE
   79     010000         READ CURRENT-SALES-FILE-OUT
   80     010100             AT END SET SALES-FILE-OUT-END-OF-FILE TO TRUE
          010200         END-READ
          010300     END-PERFORM.
          010400*
          010500*    年次報告書を更新
          010600*
   81     010700     MERGE WORK-FILE
          010800          ON ASCENDING KEY DEPT OF SALES-RECORD
```

図 104. SORT / MERGE の使用例 (2/3)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/SORTMERG       ISERIES1   06/02/15 13:56:03        ページ     4
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
          010900          ON ASCENDING KEY EMPL-NO OF SALES-RECORD
          011000          ON ASCENDING KEY MONTH OF SALES-RECORD
          011100          USING YTD-SALES-FILE-IN
          011200                CURRENT-SALES-FILE-IN
          011300          GIVING YTD-SALES-FILE-OUT.
          011400*
          011500*    年次報告書を印刷
          011600*
    82    011700     READ YTD-SALES-FILE-OUT
    83    011800         AT END SET YTD-SALES-OUT-END-OF-FILE TO TRUE
          011900     END-READ.
    84    012000     PERFORM UNTIL YTD-SALES-OUT-END-OF-FILE
    85    012100         MOVE "MERGED YTD SALES ",
          012200             TO RECORD-LABEL OF PRINT-LINE
    86    012300         MOVE YTD-SALES-OUT TO DISK-RECORD-DISPLAY
    87    012400         WRITE PRINT-LINE
    88    012500         READ YTD-SALES-FILE-OUT
    89    012600             AT END SET YTD-SALES-OUT-END-OF-FILE TO TRUE
          012700         END-READ
          012800     END-PERFORM.
          012900
    90    013000     CLOSE CURRENT-SALES-FILE-IN
          013100           CURRENT-SALES-FILE-OUT
          013200           YTD-SALES-FILE-OUT
          013300           PRINTER-OUT.
    91    013400     STOP RUN.
          013500
          013600 SCREEN-DEPT SECTION.
          013700 SCREEN-DEPT-PROCEDURE.
          013800
    92    013900     READ CURRENT-SALES-FILE-IN
    93    014000         AT END SET SALES-FILE-IN-END-OF-FILE TO TRUE
          014100     END-READ.
    94    014200     PERFORM UNTIL SALES-FILE-IN-END-OF-FILE
    95    014300         MOVE "UNSORTED CURRENT SALES ",
          014400             TO RECORD-LABEL OF PRINT-LINE
    96    014500         MOVE CURRENT-SALES-IN TO DISK-RECORD-DISPLAY
    97    014600         WRITE PRINT-LINE
    98    014700         IF ON-SITE-EMPLOYEE
    99    014800             MOVE CURRENT-SALES-IN TO SALES-RECORD
   100    014900             RELEASE SALES-RECORD
          015000         END-IF
   101    015100         READ CURRENT-SALES-FILE-IN
   102    015200             AT END SET SALES-FILE-IN-END-OF-FILE TO TRUE
          015300         END-READ
          015400     END-PERFORM.
          015500
                         * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 104. SORT / MERGE の使用例 (3/3)

SAA データ・タイプを使用したデータ項目の宣言

ILE COBOL コンパイラーでは、外部記述ファイルおよび SAA データベース・データ・タイプの可変長フィールドを、標準 COBOL データ項目に変換することができます。変換できる SAA データ・タイプは、可変長フィールド、日付、時刻、タイム・スタンプ、DBCS グラフィック、および浮動小数点フィールドです。 ILE COBOL は、これらの可変長フィールドを制限付きでサポートしています。

可変長フィールド

CRTCBLMOD コマンドまたは CRTBNDCBL コマンドの CVTOPT パラメーターに *VARCHAR、または PROCESS ステートメントに VARCHAR オプションを指定すると、可変長フィールドをプログラムで使用することができます。 *VARCHAR を指定すると、ILE COBOL プログラムは外部記述ファイルの可変長フィールドを ILE COBOL グループ項目に変換します。

そのようなグループ項目の例を次に示します。

```
06  ITEM1.
    49  ITEM1-LENGTH     PIC S9(4) COMP-4.
    49  ITEM1-DATA       PIC X(n).
```

n は可変長フィールドの最大長です。プログラムの中で、PIC S9(4) COMP-4 はこのタイプの他の宣言と同様に扱われ、PIC X(n) は標準的な英数字として扱われます。

*VARCHAR を指定しないと、可変長フィールドは無視され、ILE COBOL プログラム中の FILLER フィールドとして宣言されます。 *NOVARCHAR を指定しない場合、項目は次のように宣言されます。

```
06  FILLER     PIC x(n+2).
```

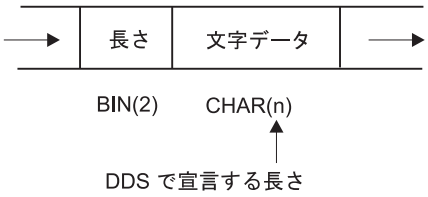
構文については 41 ページの『CVTOPT パラメーター』にある CVTOPT パラメーターを参照してください。

プログラムでは、生成されたデータ部分に対して有効な、あらゆる文字操作を実行できます。ただし、フィールドの構造上、長さ部分は有効な 2 進データでなければなりません。それが負の場合または最大フィールド長より大きい場合、このデータは無効です。

フィールドの最初の 2 バイトに有効な 2 進数が入っていない場合に、そのフィールドを含むレコードに WRITE や REWRITE を実行しようとするとエラーになり、ファイル状況 90 が戻されます。

可変長フィールドを指定する場合、次の条件が適用されます。

- データ部に形式 2 の COPY ステートメントが使用されている場合に可変長フィールドが検出されると、それは ILE COBOL プログラム中で固定長文字フィールドとして宣言されます。
- 1 バイト文字フィールドの場合、宣言された ILE COBOL フィールドの長さは、DDS フィールドの 1 バイト文字数に 2 バイトを加えた数値です。
- DBCS グラフィック・データ・フィールドの場合、宣言された ILE COBOL フィールドの長さは、DDS フィールドの中の DBCS グラフィック文字の数を 2 倍した数値に 2 バイトを加えた数値です。グラフィック・データ・タイプの詳細については 504 ページの『DBCS グラフィック・フィールド』を参照してください。 ILE COBOL フィールドの追加の 2 バイトには、可変長フィールドの現在の長さを表す 2 進数が入れられます。 492 ページの図 105 に、可変長フィールドの ILE COBOL フィールド長を示します。



1 バイト文字フィールドの場合: 2 + n = ILE COBOL フィールド長

DBCS グラフィック・データ・タイプの場合: 2 + 2(n) = ILE COBOL フィールド長

図 105. 可変長フィールドの ILE COBOL フィールド長

- ILE COBOL プログラムは、宣言された固定長フィールドに対して有効な、あらゆる文字操作を実行できます。ただし、フィールドの構造上、このフィールドの最初の 2 バイトには有効な 2 進データが入っていなければなりません (現行フィールド長データは、0 より小さいか、または DDS フィールド長より大きい値である場合は無効です)。フィールドの最初の 2 バイトに無効なフィールド長データが入っている場合は、入出力操作エラーになります。ファイル状況 90 が戻されます。
- *VARCHAR を指定しないと、FILLER に値を代入することはできないため、可変長フィールドで WRITE 操作を実行する際に問題が発生する可能性があります。 2 バイト・フィールドに、フィールドで許可されている範囲より長い値 (`X'4040'` など) が入れられることがあります。この場合、入出力エラーになります。
- SORT/MERGE キーに可変長フィールドを使用することはできません。SORT/MERGE キーに可変長フィールドを使用すると、構造全体が英数字データ項目として比較されます。

可変長フィールドを使用するプログラムの例については 506 ページの『可変長 DBCS グラフィック・フィールドの使用例』を参照してください。

日付、時刻、およびタイム・スタンプのフィールド

ILE COBOL プログラムでは、次の 2 つの方法で、DDS の日付、時刻、およびタイム・スタンプのフィールドを使用することができます。
- 日時クラスの日付、時刻、またはタイム・スタンプのデータ項目として
- 英数字フィールドとして

日時クラス

DDS の日付、時刻、およびタイム・スタンプのフィールドは、CRTCBLMOD または CRTBNDCBL の CVTOPT パラメーターの *DATETIME オプションによって、ILE COBOL の FILLER 項目として、あるいはその DDS 名によって宣言することができます。 *NODATETIME が指定された場合、DDS の日付、時刻、およびタイム・スタンプのフィールドは、ILE COBOL の FILLER 項目として宣言されます。*DATETIME が指定された場合、DDS の日付、時刻、およびタイム・スタンプの項目は、ILE COBOL の DDS 名で宣言されます。

デフォルトでは、DDS の日付、時刻、およびタイム・スタンプのフィールドは、COBOL 英数字データ項目を作成します。すなわち、COPY DDS は、DDS の日付、時刻、またはタイム・スタンプのフィールドごとに、PIC X(n) を生成します。

FORMAT 文節を生成し、したがって COBOL 日時クラス項目を作成するには、以下のように CVTOPT 値を指定しなければなりません。
- DDS 日付フィールドには *DATE
- DDS 時刻フィールドには *TIME
- DDS タイム・スタンプ・フィールドには *TIMESTAMP

上記 CVTOPT パラメーター値に相当する PROCESS ステートメント・オプションは、それぞれ DATE、TIME、および TIMESTAMP です。

日時クラスの項目の処理に関する詳細については 212 ページの『日時データ・タイプを処理する』を参照してください。

DDS のゾーン、パック、および文字のフィールドは、DATFMT キーワードを持つことができます。一般に、この種のフィールドは、COPY DDS の発生時に PICTURE 文節を生成します。その結果の COBOL 項目は、数値ゾーン、数値パック、または英数字データ・タイプです。ただし、これらの項目には、COPY DDS を用いて FORMAT 文節を生成することができます (この場合は、日時クラスの COBOL 日付データ項目が作成されます)。 CVTOPT パラメーターの *CVTTODATE 値を指定すると、DATFMT キーワードを指定した DDS のゾーン、パック、および文字のフィールドで日付データ項目が生成されます。 CVTOPT パラメーターの *NOCVTTODATE 値は、それぞれ数値ゾーン、数値パック、または英数字フィールドを生成します。これらの 2 つの値は、CVTTODATE および NOCVTTODATE オプションとして、PROCESS ステートメントにも存在します。

表 25 および 494 ページの表 26 には、ゾーン、パック、および文字の DDS フィールドに許可された DATFMT パラメーター、および CVTOPT(*CVTTODATE) 変換パラメーターが指定されている場合に COPY DDS から生成される DATFMT パラメーターと同等の ILE COBOL の形式がリストされています。

表 25 は、文字およびゾーンのフィールドです。 USAGE DISPLAY を想定しています。

表 25. 文字およびゾーンのフィールドに使用できる DATFMT パラメーター

IBM i形式	COBOL 生成の形式	説明	形式	長さ
*MDY	%m%d%y	月日年	mmddyy	6
*DMY	%d%m%y	日月年	ddmmyy	6
*YMD	%y%m%d	年月日	yymmdd	6
*JUL	%y%j	年間通算日	yyddd	5
*ISO	@Y%m%d	ISO (国際標準化機構)	yyyymmdd	8
*USA	%m%d@Y	IBM USA 標準規格	mmddyyyy	8
*EUR	%d%m@Y	IBM 欧州標準規格	ddmmyyyy	8
*JIS	@Y%m%d	JIS (日本工業規格) 西暦	yyyymmdd	8
*CMDY	@C%m%d%y	世紀月日年	cmmddyy	7
*CDMY	@C%d%m%y	世紀日月年	cddmmyy	7
*CYMD	@C%y%m%d	世紀年月日	cyymmdd	7
*MDYY	%m%d@Y	月日年	mmddyyyy	8
*DMYY	%d%m@Y	日月年	ddmmyyyy	8

表 25. 文字およびゾーンのフィールドに使用できる DATFMT パラメーター (続き)

IBM i形式	COBOL 生成の形式	説明	形式	長さ
*YYMD	@Y%m%d	年月日	yyyymmdd	8
*YM	%y%m	年月	yymm	4
*MY	%m%y	月年	mmyy	4
*YYM	@Y%m	年月	yyyymm	6
*MYY	%m@Y	月年	mmyyyy	6
*LONGJUL	@Y%j	年間通算日	yyyyddd	7

表 26 は、パック・フィールド用です。 USAGE PACKED-DECIMAL が生成されます。

表 26. パック・フィールドに使用できる DATFMT パラメーター

IBM i形式	COBOL 生成の形式	説明	形式	長さ
*MDY	%m%d%y	月日年	mmddyy	4
*DMY	%d%m%y	日月年	ddmmyy	4
*YMD	%y%m%d	年月日	yymmdd	4
*JUL	%y%j	年間通算日	yyddd	3
*ISO	@Y%m%d	ISO (国際標準化機構)	yyyymmdd	5
*USA	%m%d@Y	IBM USA 標準規格	mmddyyyy	5
*EUR	%d%m@Y	IBM 欧州標準規格	ddmmyyyy	5
*JIS	@Y%m%d	JIS (日本工業規格) 西暦	yyyymmdd	5
*CMDY	@C%m%d%y	世紀月日年	cmmddyy	4
*CDMY	@C%d%m%y	世紀日月年	cddmmyy	4
*CYMD	@C%y%m%d	世紀年月日	cyymmdd	4
*MDYY	%m%d@Y	月日年	mmddyyyy	5
*DMYY	%d%m@Y	日月年	ddmmyyyy	5
*YYMD	@Y%m%d	年月日	yyyymmdd	5
*YM	%y%m	年月	yymm	3
*MY	%m%y	月年	mmyy	3
*YYM	@Y%m	年月	yyyymm	4
*MYY	%m@Y	月年	mmyyyy	4
*LONGJUL	@Y%j	年間通算日	yyyyddd	4

英数字クラス

このセクションでは、日付、時刻、およびタイム・スタンプのデータ項目を、ILE COBOL プログラムの英数字フィールドとして使用する方法を説明します。これを、 492 ページの『日時クラス』で説明した日時クラスの日付、時刻、およびタイム・スタンプのデータ項目を使用した場合と比較してください。

デフォルトでは、DDS の日付、時刻、またはタイム・スタンプのフィールドは、固定長文字フィールドとして ILE COBOL プログラムに入れられます。 ILE COBOL プログラムは、有効な、あらゆる文字操作を、固定長フィールドで実行することができます。これらの操作は、英数字データ項目に関する標準の COBOL 規則に従っ

ています。 CRTCBLMOD および CRTBNDCBL コマンドの *NODATE、*NOTIME、および *NOTIMESTAMP CVTOPT パラメーター値を使用すると、COPY DDS は英数字 COBOL データ項目を生成します。これらの CVTOPT パラメーター値も、それぞれ NODATE、NOTIME、および NOTIMESTAMP の PROCESS ステートメント上にあります。

日付、時刻、およびタイム・スタンプのフィールドがプログラムに入るのは、CRTCBLMOD または CRTBNDCBL コマンドの CVTOPT パラメーターの *DATETIME オプションか、PROCESS ステートメントの DATETIME オプションを指定した場合に限られます。 CVTOPT パラメーターの説明とその構文については 41 ページの『CVTOPT パラメーター』を参照してください。 *DATETIME が指定されていない場合は、日付、時刻、およびタイム・スタンプ・フィールドは無視され、ユーザーの ILE COBOL プログラムで FILLER フィールドとして宣言されます。

日付、時刻、およびタイム・スタンプのデータ・タイプには、それぞれ独自の形式があります。

日付、時刻、またはタイム・スタンプ情報を含んでいるフィールドがプログラムによって更新され、その更新された情報がデータベースに戻される場合、フィールドの形式は、そのフィールドがデータベースから取り出されたときの形式と全く同じでなければなりません。同じ形式を使用しなければ、エラーが発生します。各データ・タイプの有効な形式については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベースおよびファイル・システム」カテゴリーを参照してください。

CL コマンドを使用してソース・ステートメントを入力する方法に関する情報を入手するには、 Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** のカテゴリー「プログラミング」の中の『*CL および API*』セクションを参照してください。

日付、時刻、またはタイム・スタンプ・フィールドに適正な値を転送する前にレコードを WRITE しようとすると、WRITE 操作は失敗し、ファイル状況 90 が戻されます。 日付、時刻、またはタイム・スタンプのフィールドでキー・フィールドを使用しようとすると、READ または START 操作でもエラーが発生し、そのフィールドには適切な値が入りません。

プログラムの中の日付、時刻、タイム・スタンプの項目を FILLER として宣言した場合、このフィールドをシステムで受け入れられる値には設定できないので、このフィールドを含むレコードには WRITE を実行しないようにしてください。

ILE COBOL の英数字データ・タイプとして生成される DDS の日付、時刻、およびタイム・スタンプのフィールドは、SORT/MERGE キーとして指定することができますが、日付、時刻、およびタイム・スタンプのデータ項目としてではなく、英数字データ項目として比較されます。

*DATETIME コンパイラー・オプションが *DATE を処理する例

496 ページの図 106 では、DDS 日付項目 DATEITEM を定義しています。このセクションでは、DDS 日付項目がどのように影響するかだけを説明します。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
      A          R DATETIME
      A*
      A            VARITEM      100          VARLEN
      A*
      A            TIMEITEM       T          TIMFMT(*HMS)
      A            DATEITEM       L          DATFMT(*YMD)
      A            TIMESTAMP      Z
```

図 106. 日時フィールドを定義する DDS ファイル

以下の例では、CVTOPT パラメーターの *DATE オプションと一緒に CVTOPT パラメーターの *DATETIME オプションを指定する組み合わせと、その組み合わせが、 DATEITEM をプログラムに入れる方法にどのように影響するかを示します。

例 1: *NODATETIME を *NODATE と一緒に指定すると、DATEITEM は、以下のようにプログラムに入れられます。

```
05 FILLER    PIC X(8).
```

例 2: *DATETIME を *NODATE と一緒に指定すると、DATEITEM は、以下のようにプログラムに入れられます。

```
05 DATEITEM    PIC X(8).
```

例 3: *DATETIME を *DATE と一緒に指定すると、DATEITEM は、以下のようにプログラムに入れられます。

```
05 DATEITEM    FORMAT DATE '%y/%m/%d'.
```

例 4: *NODATETIME を *DATE と一緒に指定すると、DATEITEM は、以下のようにプログラムに入れられます。

```
05 FILLER    FORMAT DATE '%y/%m/%d'.
```

ヌル可能フィールド

ヌル可能フィールドとは、ヌル値を保持できるフィールドです。**ヌル値**とは、すべての非ヌル値と区別される特殊値のことで、情報がないことを意味します。たとえば、ヌル値とは、ゼロの値、すべてのブランク、または 16 進数のゼロと同じではありません。また、どの値とも、他のヌル値とさえも等しくはなりません。

データベース・レコードのフィールドごとに、フィールドがヌル値かどうかを示す 1 バイトの値があります。フィールドが、*ヌル値ならば* 中に値 1 が含まれ、*ヌル値でなければ* 値 0 が含まれます。この値のストリングを**ヌル・マップ**と呼び、ヌル可能データベース・ファイルのレコードごとに 1 つのヌル・マップがあります。ヌル可能データベース・ファイルのレコード様式ごとに、独自のヌル・マップを持ちます。

ファイルにはキーも付けられ、キーには、**ヌル・キー・マップ**が入っています。ヌル・キー・マップとは、同じように定義した値の個別のストリングで、キーの各フィールドに 1 つあります。キー付きヌル可能データベース・ファイルのレコードごとに、1 つのヌル・キー・マップがあります。キー付きヌル可能データベース・ファイルのレコード様式ごとに、独自のヌル・キー・マップを持ちます。

ヌル・マップの中の値は、WORKING-STORAGE SECTION でのヌル・マップの定義方法によって、ブールの場合もあれば、英数字の場合もあります。外部記述ファイルを使用していて、COPY-DDS ステートメント WITH NULL-MAP を指定すると、ブール値を指定した 1 つまたは複数のヌル・マップがセットアップされます。
COPY-DDS ステートメント WITH NULL-MAP-ALPHANUM を指定すると、英数字値を指定した 1 つまたは複数のヌル・マップがセットアップされます。
COPY-DDS ステートメント WITH NULL-KEY-MAP では、ブール値を指定した 1 つまたは複数のヌル・キー・マップが生成されます。プログラム記述ファイルを使用すると、WORKING-STORAGE SECTION で、ヌル・マップをブールまたは英数字のいずれかに定義することができます。

NULL-MAP-ALPHANUM は、ヌル文字マップへ受け入れられる、あるいはヌル文字マップから送ることができる値の範囲を拡大して、0 または 1 以外の値を組み入れます。ヌル文字マップ・フィールド内で、このフィールドがヌルであることを示す値は 1 だけです。ヌル・マップで送ることができる、または受け入れることができる 0 または 1 以外の値についての詳細は、 Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベース」カテゴリーの中の『*DB2 Universal Database for AS/400*』セクションを参照してください。

入出力ステートメントに NULL-MAP (NULL-KEY-MAP) 句を指定すると、ヌル可能フィールドを含むデータベース・レコードが ILE COBOL プログラムによってアクセスされる際、そのレコードのヌル・キー・マップ (存在する場合) とヌル・マップが、プログラムのヌル・マップ (ヌル・キー・マップ) のコピーとの間でコピーされます。入出力ステートメントでの NULL-MAP および NULL-KEY-MAP 句の使用に関する詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

ヌル可能ファイルの入出力、レコードの位置指定、およびヌル可能キー付きファイルでのレコードの削除については、以下の部分で説明します。

- 『入出力操作でのヌル・マップおよびヌル・キー・マップの使用』
- 498 ページの『データベース・ファイルのヌル可能レコードの位置指定』
- 498 ページの『データベース・ファイルのヌル可能レコードの削除』.

ヌル可能フィールドのエラー状態の処理に関する詳細については 442 ページの『ヌル可能フィールドを使用した操作中のエラーの処理』を参照してください。ヌル可能フィールドの定義、および COPY DDS ステートメントでのヌル可能フィールドの使用に関する詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

入出力操作でのヌル・マップおよびヌル・キー・マップの使用

ヌル可能フィールドでの入出力操作は、以下の入出力ステートメントで NULL-MAP IS または NULL-KEY MAP IS 句を使用して行うことができます。

- READ (形式 1、2 および 3)
- WRITE (形式 1 および 2)
- REWRITE (形式 1)

これらの句は、レコードとそのキーを定義するヌル・マップとヌル・キー・マップに関する、システムのデータ管理の設定値を処理します。これらの句で指定された設定値は、添え字を付けるか、参照変更することができます。

ASSIGN 文節に ALWNULL 属性が指定されていて、WRITE または REWRITE ステートメントで NULL-MAP IS 句を指定しないと、B'0' のストリングが渡されます。このレコード内のフィールドはすべて、ヌル値ではない と見なされます。ファイルが索引ファイルで、かつ NULL-MAP IS 句を指定している場合は、NULL-KEY-MAP IS 句も指定しなければなりません。キー・フィールドの場合は、ヌル・キー・マップの値がヌル・マップの対応する値と必ず同じになるようにしなければなりません。

ASSIGN 文節に ALWNULL 属性が指定されていて、READ ステートメントで NULL-MAP IS 句を指定しないと、ヌル・マップには READ の前に入っていた同じ値が入ります。ヌル可能キーの場合も、NULL-KEY-MAP IS 句を指定しなかった場合は、同じことが起こります。ファイルが索引ファイルで、かつ NULL-MAP IS 句を指定している場合は、NULL-KEY-MAP IS 句も指定しなければなりません。

ヌル可能フィールドを処理できる入出力ステートメントの詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

データベース・ファイルのヌル可能レコードの位置指定

データベース・ファイルのヌル可能レコードの位置を決めるには、START ステートメントで NULL-KEY-MAP IS 句を使用します。この句のオブジェクトは、添え字を付けるか、参照変更することができます。 START ステートメントで参照されるキー・フィールドのいずれかがヌル可能で、かつ NULL-KEY-MAP IS 句が使用されない場合は、代わりにすべてゼロのヌル・マップが使用されます。

NULL-KEY-MAP IS 句を用いてデータベースのヌル可能レコードの位置を決める際の詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

データベース・ファイルのヌル可能レコードの削除

データベース・ファイルのヌル可能レコードを削除するには、DELETE ステートメントで NULL-KEY-MAP IS 句を使用します。この句のオブジェクトは、添え字を付けるか、参照変更することができます。 DELETE ステートメントで参照されるキー・フィールドのいずれかがヌル可能で、かつ NULL-KEY-MAP IS 句が使用されない場合は、代わりにすべてゼロのヌル・マップが使用されます。

NULL-KEY-MAP IS 句を用いてデータベースのヌル可能レコードを削除する際の詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

ヌル・マップおよびヌル・キー・マップの使用例

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* THIS IS THE STUDENT INFORMATION FILE  - NULLSTDT
     A
     A          R PERSON
     A            FNAME         20
     A            LNAME         30
     A            MARK           3P         ALWNULL
```

図 107. ヌル・マップおよびヌル・キー・マップの使用例 - STUDENT INFORMATION FILE (学生情報ファイル) DDS

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* THIS IS THE CAR INFORMATION FILE  - NULLCAR
     A
     A                                     UNIQUE
     A          R CARS
     A            CARMODEL      25A         ALWNULL
     A            YEAR           4P
     A            OPTIONS        2P
     A            PRICE          7P 2
     A          K CARMODEL
```

図 108. ヌル・マップおよびヌル・キー・マップの使用例 - CAR INFORMATION FILE (カー情報ファイル) DDS

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/NULLMAP         ISERIES1   06/02/15 14:20:55        ページ     2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID. NULLMAP.
    3     000300 ENVIRONMENT DIVISION.
    4     000400 CONFIGURATION SECTION.
    5     000500  SOURCE-COMPUTER.   IBM-ISERIES.
    6     000600  OBJECT-COMPUTER.   IBM-ISERIES.
    7     000700 INPUT-OUTPUT SECTION.
    8     000800  FILE-CONTROL.
    9     000900      SELECT NULLSTDT
   10     001000          ASSIGN TO DATABASE-NULLSTDT-ALWNULL  1
   11     001100          ORGANIZATION IS SEQUENTIAL
   12     001200          ACCESS IS SEQUENTIAL
   13     001300          FILE STATUS IS NULLSTDT-STATUS.
   14     001400      SELECT NULLCAR
   15     001500          ASSIGN TO DATABASE-NULLCAR-ALWNULL
   16     001600          ORGANIZATION IS INDEXED
   17     001700          ACCESS IS DYNAMIC
   18     001800          RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
   19     001900          FILE STATUS IS NULLCAR-STATUS.
   20     002000 DATA DIVISION.
   21     002100 FILE SECTION.
   22     002200   FD NULLSTDT.
   23     002300   01 NULLSTDT-REC.
          002400      COPY DDS-ALL-FORMATS OF NULLSTDT.
   24    +000001       05  NULLSTDT-RECORD PIC X(52).                                  <-ALL-FMTS
         +000002*    I-O FORMAT:PERSON     FROM FILE NULLSTDT   OF LIBRARY CBLGUIDE   <-ALL-FMTS
         +000003*                                                                      <-ALL-FMTS
   25    +000004       05  PERSON        REDEFINES NULLSTDT-RECORD.                   <-ALL-FMTS
   26    +000005           06 FNAME                PIC X(20).                         <-ALL-FMTS
   27    +000006           06 LNAME                PIC X(30).                         <-ALL-FMTS
   28    +000007           06 MARK                 PIC S9(3)         COMP-3.  2       <-ALL-FMTS
         +000008*                  (ヌル可能フィールド)                                <-ALL-FMTS
   29     002500   FD NULLCAR.
   30     002600   01 NULLCAR-REC.
          002700      COPY DDS-ALL-FORMATS OF NULLCAR.
   31    +000001       05  NULLCAR-RECORD PIC X(34).                                  <-ALL-FMTS
         +000002*    I-O FORMAT:CARS       FROM FILE NULLCAR    OF LIBRARY CBLGUIDE   <-ALL-FMTS
         +000003*                                                                      <-ALL-FMTS
         +000004*THE KEY DEFINITIONS FOR RECORD FORMAT  CARS                          <-ALL-FMTS
         +000005*  NUMBER              NAME               RETRIEVAL       ALTSEQ     <-ALL-FMTS
         +000006*   0001   CARMODEL                      ASCENDING         NO        <-ALL-FMTS
   32    +000007       05  CARS          REDEFINES NULLCAR-RECORD.                    <-ALL-FMTS
   33    +000008           06 CARMODEL             PIC X(25).                         <-ALL-FMTS
         +000009*                  (ヌル可能フィールド)                                <-ALL-FMTS
   34    +000010           06 YEAR                 PIC S9(4)         COMP-3.          <-ALL-FMTS
   35    +000011           06 OPTIONS              PIC S9(2)         COMP-3.          <-ALL-FMTS
   36    +000012           06 PRICE                PIC S9(5)V9(2)    COMP-3.          <-ALL-FMTS
          002800
   37     002900 WORKING-STORAGE SECTION.
   38     003000 01 NULLSTDT-STATUS      PIC XX VALUE " ".
   39     003100 01 NULLCAR-STATUS      PIC XX VALUE " ".
   40     003200 01 NULLSTDT-NM.
          003300    COPY DDS-ALL-FORMATS OF NULLSTDT
```

図 109. ヌル・マップおよびヌル・キー・マップの使用例 (1/4)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/NULLMAP         ISERIES1   06/02/15 14:20:55        ページ    3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
         003400           WITH NULL-MAP.  3
        +000001*    NULL MAP:  PERSON     FROM FILE NULLSTDT   OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000002*                                                                            <-ALL-FMTS
   41   +000003       05  PERSON-NM.  4                                                     <-ALL-FMTS
   42   +000004           06 FILLER               PIC X(2)  VALUE ZEROS.                   <-ALL-FMTS
   43   +000005           06 MARK-NF              PIC 1  VALUE B"0".  5                    <-ALL-FMTS
   44    003500 01 NULLCAR-NKM.
         003600    COPY DDS-ALL-FORMATS OF NULLCAR
         003700           WITH NULL-KEY-MAP
         003800           WITH NULL-MAP.
        +000001*    NULL MAP:  CARS       FROM FILE NULLCAR    OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000002*                                                                            <-ALL-FMTS
        +000003*   NULL KEY MAP:  6                                                         <-ALL-FMTS
   45   +000004       05  CARS-NKM.                                                        <-ALL-FMTS
   46   +000005           06 CARMODEL-NF          PIC 1  VALUE B"0".                       <-ALL-FMTS
   47   +000006       05  CARS-NM.                                                         <-ALL-FMTS
   48   +000007           06 CARMODEL-NF          PIC 1  VALUE B"0".                       <-ALL-FMTS
   49   +000008           06 FILLER               PIC X(3)  VALUE ZEROS.                   <-ALL-FMTS
         003900
   50    004000 PROCEDURE DIVISION.
         004100 MAINLINE.
   51    004200     OPEN OUTPUT NULLSTDT.
   52    004300     MOVE "JOHN"    TO FNAME OF PERSON.
   53    004400     MOVE "SMITH"   TO LNAME OF PERSON.
   54    004500     MOVE B"1"      TO MARK-NF OF PERSON-NM.  7
   55    004600     WRITE NULLSTDT-REC
         004700           NULL-MAP IS PERSON-NM.
   56    004800     CLOSE NULLSTDT.
         004900
   57    005000     OPEN INPUT  NULLSTDT.
   58    005100     MOVE " "       TO FNAME OF PERSON.
   59    005200     MOVE " "       TO LNAME OF PERSON.
   60    005300     MOVE B"0"      TO MARK-NF OF PERSON-NM.
   61    005400     READ NULLSTDT NULL-MAP IS PERSON-NM.
   62    005500     IF FNAME   OF PERSON    = "JOHN"  AND
         005600        LNAME   OF PERSON    = "SMITH" AND
         005700        MARK-NF OF PERSON-NM = B"1"    AND  8
         005800        NULLSTDT-STATUS     = "00"
   63    005900        DISPLAY "NAME IS CORRECT"
         006000     ELSE
   64    006100        DISPLAY "NAME IS NOT CORRECT"
         006200     END-IF.
   65    006300     CLOSE NULLSTDT.
         006400
   66    006500     OPEN EXTEND NULLSTDT.
   67    006600     MOVE "TOM"     TO FNAME OF PERSON.
   68    006700     MOVE "JONES"   TO LNAME OF PERSON.
   69    006800     MOVE B"1"      TO MARK-NF OF PERSON-NM.
   70    006900     WRITE NULLSTDT-REC NULL-MAP IS PERSON-NM.
   71    007000     CLOSE NULLSTDT.
         007100
   72    007200     OPEN INPUT  NULLSTDT.
   73    007300     MOVE " "       TO FNAME OF PERSON.
   74    007400     MOVE " "       TO LNAME OF PERSON.
   75    007500     MOVE B"0"      TO MARK-NF OF PERSON-NM.
```

図 109. ヌル・マップおよびヌル・キー・マップの使用例 (2/4)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/NULLMAP         ISERIES1   06/02/15 14:20:55        ページ     4
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
         007600
   76    007700     READ NULLSTDT
         007800          NULL-MAP IS PERSON-NM.
   77    007900     READ NULLSTDT
         008000          NULL-MAP IS PERSON-NM.
   78    008100     IF FNAME   OF PERSON    = "TOM"   AND
         008200        LNAME   OF PERSON    = "JONES" AND
         008300        MARK-NF OF PERSON-NM = B"1"    AND
         008400        NULLSTDT-STATUS     = "00"
   79    008500        DISPLAY "NAME IS CORRECT"
         008600     ELSE
   80    008700        DISPLAY "NAME IS NOT CORRECT"
   81    008800        DISPLAY "NAME IS: " FNAME " " LNAME
         008900     END-IF.
   82    009000     CLOSE NULLSTDT.
         009100
   83    009200     OPEN EXTEND NULLSTDT.
   84    009300     MOVE "PETER"   TO FNAME OF PERSON.
   85    009400     MOVE "STONE"   TO LNAME OF PERSON.
   86    009500     MOVE B"1"      TO MARK-NF OF PERSON-NM.
   87    009600     WRITE NULLSTDT-REC
         009700           NULL-MAP IS PERSON-NM.
   88    009800     CLOSE NULLSTDT.
         009900
   89    010000     OPEN I-O NULLSTDT.
   90    010100     MOVE " "       TO FNAME OF PERSON.
   91    010200     MOVE " "       TO LNAME OF PERSON.
   92    010300     MOVE B"1"      TO MARK-NF OF PERSON-NM.
   93    010400     READ NULLSTDT
         010500          NULL-MAP IS PERSON-NM.
   94    010600     READ NULLSTDT
         010700          NULL-MAP IS PERSON-NM.
   95    010800     READ NULLSTDT
         010900          NULL-MAP IS PERSON-NM.
   96    011000     MOVE "BRICK" TO LNAME   OF PERSON.
   97    011100     MOVE B"0"    TO MARK-NF OF PERSON-NM.
   98    011200     REWRITE NULLSTDT-REC NULL-MAP IS PERSON-NM.
   99    011300     CLOSE NULLSTDT.
         011400
  100    011500     OPEN I-O   NULLSTDT.
  101    011600     MOVE " "       TO FNAME OF PERSON.
  102    011700     MOVE " "       TO LNAME OF PERSON.
  103    011800     MOVE B"1"      TO MARK-NF OF PERSON-NM.
  104    011900     READ NULLSTDT
         012000          NULL-MAP IS PERSON-NM.
  105    012100     READ NULLSTDT
         012200          NULL-MAP IS PERSON-NM.
  106    012300     READ NULLSTDT
         012400          NULL-MAP IS PERSON-NM.
  107    012500     IF FNAME   OF PERSON    = "PETER" AND
         012600        LNAME   OF PERSON    = "BRICK" AND
         012700        MARK-NF OF PERSON-NM = B"0"    AND
         012800        NULLSTDT-STATUS     ="00"
  108    012900        DISPLAY "NAME IS CORRECT"
         013000     ELSE
```

図 109. ヌル・マップおよびヌル・キー・マップの使用例 (3/4)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/NULLMAP         ISERIES1   06/02/15 14:20:55        ページ    5
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
  109     013100        DISPLAY "NAME IS NOT CORRECT"
  110     013200        DISPLAY "NAME IS: " FNAME " " LNAME
          013300     END-IF.
  111     013400     CLOSE NULLSTDT.
          013500*-------------------------------------------------------------*
          013600* WRITE records to indexed NULLCAR.                           *
          013700*-------------------------------------------------------------*
  112     013800     OPEN OUTPUT NULLCAR.
  113     013900     MOVE B"0"                TO CARMODEL-NF OF CARS-NKM.
  114     014000     MOVE B"0"                TO CARMODEL-NF OF CARS-NM.
  115     014100     MOVE "SUPERCAR"          TO CARMODEL    OF CARS.
  116     014200     MOVE 1995                TO YEAR        OF CARS.
  117     014300     MOVE 2                   TO OPTIONS     OF CARS.
  118     014400     MOVE 14799               TO PRICE       OF CARS.
  119     014500     WRITE NULLCAR-REC NULL-KEY-MAP IS CARS-NKM
          014600                       NULL-MAP IS CARS-NM.
  120     014700     MOVE "FASTCAR"           TO CARMODEL    OF CARS.
  121     014800     MOVE 1997                TO YEAR        OF CARS.
  122     014900     MOVE 5                   TO OPTIONS     OF CARS.
  123     015000     MOVE 18799               TO PRICE       OF CARS.
  124     015100     WRITE NULLCAR-REC NULL-KEY-MAP IS CARS-NKM
          015200                       NULL-MAP IS CARS-NM.  9
  125     015300     MOVE B"1"                TO CARMODEL-NF OF CARS-NKM.
  126     015400     MOVE B"1"                TO CARMODEL-NF OF CARS-NM.
  127     015500     MOVE 1996                TO YEAR        OF CARS.
  128     015600     MOVE 5                   TO OPTIONS     OF CARS.
  129     015700     MOVE 16199               TO PRICE       OF CARS.
  130     015800     WRITE NULLCAR-REC NULL-KEY-MAP IS CARS-NKM
          015900                       NULL-MAP IS CARS-NM.
  131     016000     CLOSE NULLCAR.
          016100
  132     016200     OPEN I-O NULLCAR.
  133     016300     MOVE B"0"                TO CARMODEL-NF OF CARS-NKM.
  134     016400     MOVE B"0"                TO CARMODEL-NF OF CARS-NM.
  135     016500     MOVE "SUPERCAR"          TO CARMODEL    OF CARS.
  136     016600     MOVE 0                   TO YEAR        OF CARS.
  137     016700     MOVE 0                   TO OPTIONS     OF CARS.
  138     016800     MOVE 0                   TO PRICE       OF CARS.
  139     016900     READ NULLCAR
          017000          NULL-KEY-MAP IS CARS-NKM
          017100          NULL-MAP     IS CARS-NM.
  140     017200     READ NULLCAR NEXT
          017300          NULL-KEY-MAP IS CARS-NKM
          017400          NULL-MAP     IS CARS-NM.
  141     017500     IF CARMODEL-NF OF CARS-NKM      = B"1"         AND
          017600        YEAR        OF CARS          = 1996         AND
          017700        OPTIONS     OF CARS          = 5            AND
          017800        PRICE       OF CARS          = 16199        AND
          017900        NULLCAR-STATUS             = "00"
  142     018000        DISPLAY "CAR IS CORRECT"
          018100     ELSE
  143     018200        DISPLAY "CAR IS NOT CORRECT"
  144     018300        DISPLAY "CAR IS: " CARMODEL " " YEAR " " OPTIONS " " PRICE
  145     018400        DISPLAY "NULLCAR-STATUS " NULLCAR-STATUS
          018500     END-IF.
  146     018600     CLOSE NULLCAR.
          018700
  147     018800     STOP RUN.
                         * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 109. ヌル・マップおよびヌル・キー・マップの使用例 (4/4)

500 ページの図 109 のサンプル・プログラムは、データベース・ファイルのヌル・キー・マップとヌル・マップを使用して、有効な学生とカー・モデルを追跡する方法の例です。

1 データベース・ファイル NULLSTDT をヌル可能と定義します。

2 データ項目 MARK を定義します。フィールドが DDS の ALWNULL キーワードによりヌル可能と定義されたため、メッセージ (ヌル可能フィールド) が下に表示されます。

3 COPY DDS ステートメントおよび WITH NULL-MAP 句を用いて、ヌル可能 DDS ファイル NULLSTDT がプログラムに入れられます。

4 ヌル・マップ PERSON-NM が定義されます。

5 B″0″ の値によって、データ項目 MARK-NF は*非ヌル* に初期設定されます。ヌル可能フィールドに B″1″ の値を指定すると、ヌルになります。

6 ヌル・キー・マップ CARS-NKM が定義されます。

7 NULL-MAP IS PERSON-NM 句によってレコード NULLSTDT-REC が書き込まれ、書き込み操作中のヌル・マップの使い方を示します。 NULL MAP IS 句は、他のあらゆる入出力操作においても使用されます。

8 MARK-NF OF PERSON-NM データ項目が、ヌル値 (B″1″) であるかを検査されます。

9 NULLCAR-REC レコードが書き込まれ、ヌル・キー・マップおよびヌル・マップはともに、NULL-KEY-MAP IS および NULL-MAP IS 句を用いて参照されなければなりません。

DBCS グラフィック・フィールド

DBCS グラフィック・データ・タイプは、各文字が 2 バイトで表される文字ストリングです。 DBCS グラフィック・データ・タイプには、シフトアウト (SO) 文字やシフトイン (SI) 文字は含まれません。次の図に、1 バイト文字データと DBCS グラフィック・データの違いを示します。

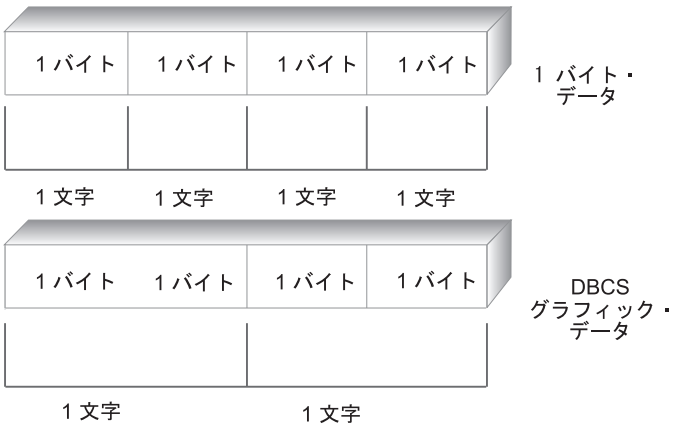


図 110. 1 バイト文字データとグラフィック・データとの比較

DBCS グラフィック・データを ILE COBOL プログラムに含めることができるのは、CRTCBLMOD コマンドまたは CRTBNDCBL コマンドの CVTOPT パラメーターに *PICXGRAPHIC 値か *PICGGRAPHIC 値を指定した場合か、PROCESS ステートメントに CVTPICXGRAPHIC オプションか CVTPICGGRAPHIC オプションを指定した場合だけです。この指定を行わないと、グラフィック・データは無視され、ILE COBOL プログラム中で FILLER フィールドとして宣言されます。CVTOPT パラメーターの説明とその構文については 33 ページの『CRTCBLMOD コマンドのパラメーター』を参照してください。

DBCS グラフィック・データを指定した場合、次の条件が適用されます。

- DBCS グラフィック・データは、固定長英数字フィールドまたは DBCS フィールドとして ILE COBOL プログラムの中にコピーされます。
- DBCS グラフィック・データの 1 文字 の長さは 2 バイトです。

- *PICXGRAPHIC を指定した場合、各固定長 DBCS グラフィック・データ・フィールド (たとえば 504 ページの図 110 に示されているように PIC G(2) DISPLAY-1 で定義されたフィールド) の長さは、フィールドのバイト数の長さ (示された例では、4 バイトの長さ) になります。 *PICGGRAPHIC を指定した場合は、各固定長 DBCS グラフィック・データ・フィールドの長さは、2 バイト文字の数 (例では 2 文字の長さ) になります。

可変長 DBCS グラフィック・フィールド

DBCS グラフィック・データ・タイプと組み合わせて可変長フィールドを使用して、可変長 DBCS グラフィック・データを指定できます。可変長 DBCS グラフィック・データを指定するには、CRTCBLMOD コマンドまたは CRTBNDCBL コマンドの CVTOPT パラメーターに *VARCHAR および *PICXGRAPHIC を指定するか、または PROCESS ステートメントに VARCHAR オプションおよび CVTPICXGRAPHIC オプションを指定します。

CVTOPT(*NOVARCHAR *NOPICGGRAPHIC)、CVTOPT(*NOVARCHAR *PICGGRAPHIC)、CVTOPT(*NOVARCHAR *NOPICXGRAPHIC) または CVTOPT(*NOVARCHAR *PICXGRAPHIC) のどれかを指定した場合、ILE COBOL コンパイラーが可変長 DBCS グラフィック・データ項目を検出すると、プログラムの内容は次のようになります。

```
            06 FILLER           PIC X(2n+2).
*             (Variable-length field)
```

n は DDS フィールドの中の文字数です。

CVTOPT(*VARCHAR *NOPICGGRAPHIC) または CVTOPT(*VARCHAR *NOPICXGRAPHIC) を指定した場合、ILE COBOL コンパイラーが可変長 DBCS グラフィック・データ項目を検出すると、プログラムの内容は次のようになります。

```
            06 NAME
*             (Variable-length field)
              49 NAME-LENGTH        PIC S9(4) COMP-4.
*                         (Number of 2-byte characters)
              49 FILLER             PIC X(2n).
```

n は DDS フィールドの中の DBCS 文字の数です。

CVTOPT(*VARCHAR *PICXGRAPHIC) を指定した場合、ILE COBOL コンパイラーが可変長 DBCS グラフィック・データ項目を検出すると、プログラムの内容は次のようになります。

```
            06 NAME
*             (Variable-length field)
              49 NAME-LENGTH        PIC S9(4) COMP-4.
*                         (Number of 2-byte characters)
              49 NAME-DATA          PIC X(2n).
```

n は DDS フィールドの中の DBCS 文字の数です。

CVTOPT(*VARCHAR *PICGGRAPHIC) を指定した場合、ILE COBOL コンパイラーが可変長 DBCS グラフィック・データ項目を検出すると、プログラムの内容は次のようになります。

```
          06 NAME
      *             (Variable-length field)
                 49 NAME-LENGTH        PIC S9(4) COMP-4.
      *                         (Number of 2-byte characters)
                 49 NAME-DATA          PIC G(n) DISPLAY-1.
```

n は DDS フィールドの中の DBCS 文字の数です。

可変長 DBCS グラフィック・フィールドの使用例

図 111 は、DDS ファイルで可変長 DBCS グラフィック・データ項目を定義する例です。 507 ページの図 112 は、*PICXGRAPHIC を指定した形式 2 の COPY ステートメントを使用した ILE COBOL プログラムと、そのプログラムがコンパイルされた結果のリストです。 508 ページの図 113 は、*PICGGRAPHIC を指定した可変長 DBCS グラフィック・データ項目を使用した ILE COBOL プログラムです。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R SAMPLEFILE
     A*
     A            VARITEM      100          VARLEN
     A*
     A            TIMEITEM       T          TIMFMT(*HMS)
     A            DATEITEM       L          DATFMT(*YMD)
     A            TIMESTAMP      Z
     A*
     A            GRAPHITEM    100G
     A            VGRAPHITEM   100G          VARLEN
```

図 111. 可変長グラフィック・データ・フィールドを定義する DDS ファイル

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/PGM1           ISERIES1   06/02/15 14:31:24        ページ    2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
         000100 process varchar datetime cvtpicxgraphic
    1    000200 Identification division.
    2    000300   Program-id.  pgm1.
         000400
    3    000500 Environment division.
    4    000600 Configuration section.
    5    000700   Source-computer.  ibm-iSeries.
    6    000800   Object-computer.  ibm-iSeries.
    7    000900 Input-output section.
    8    001000   File-control.
    9    001100     Select file1
   10    001200       assign to database-samplefi                                        00/08/15
   11    001300       organization is sequential
   12    001400       access is sequential
   13    001500       file status is fs1.
         001600
   14    001700 Data division.
   15    001800 File section.
   16    001900 fd  file1.
   17    002000 01  record1.
         002100 copy dds-all-formats of samplefi.                                       00/08/15
   18   +000001       05  SAMPLEFI-RECORD PIC X(546).                                <-ALL-FMTS
        +000002*    I-O FORMAT:SAMPLEFILE FROM FILE SAMPLEFI   OF LIBRARY CBLGUIDE     <-ALL-FMTS
        +000003*                                                                       <-ALL-FMTS
   19   +000004       05  SAMPLEFILE    REDEFINES SAMPLEFI-RECORD.                   <-ALL-FMTS
   20   +000005           06 VARITEM.                                                 <-ALL-FMTS
        +000006*                (可変長フィールド)                                     <-ALL-FMTS
   21   +000007                49 VARITEM-LENGTH   PIC S9(4) COMP-4.                  <-ALL-FMTS
   22   +000008                49 VARITEM-DATA     PIC X(100).                        <-ALL-FMTS
   23   +000009           06 TIMEITEM             PIC X(8).                          <-ALL-FMTS
        +000010*                (時刻フィールド)                                       <-ALL-FMTS
   24   +000011           06 DATEITEM             PIC X(8).                          <-ALL-FMTS
        +000012*                (日付フィールド)                                       <-ALL-FMTS
   25   +000013           06 TIMESTAMP            PIC X(26).                         <-ALL-FMTS
        +000014*                (タイム・スタンプ・フィールド)                         <-ALL-FMTS
   26   +000015           06 GRAPHITEM            PIC X(200).                        <-ALL-FMTS
        +000016*                (グラフィック・フィールド)                             <-ALL-FMTS
   27   +000017           06 VGRAPHITEM.                                              <-ALL-FMTS
        +000018*                (可変長フィールド)                                     <-ALL-FMTS
   28   +000019                49 VGRAPHITEM-LENGTH                                   <-ALL-FMTS
        +000020                                   PIC S9(4) COMP-4.                   <-ALL-FMTS
        +000021*                (2 バイト文字の数)                                     <-ALL-FMTS
   29   +000022                49 VGRAPHITEM-DATA  PIC X(200).                        <-ALL-FMTS
        +000023*                (グラフィック・フィールド)                             <-ALL-FMTS
   30    002200 Working-Storage section.
   31    002300 77  fs1      pic x(2).
         002400
   32    002500 Procedure division.
         002600 Mainline.
   33    002700     stop run.
                    * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

*図 112. 可変長 DBCS グラフィック・データ項目と *PICXGRAPHIC を使用する ILE COBOL プログラム*

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/DBCSPICG       ISERIES1   06/02/15 14:48:02       ページ     2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
         000100 process varchar datetime cvtpicggraphic                                          00/08/21
    1    000200 Identification division.
    2    000300   Program-id.  dbcspicg.                                                        00/08/21
         000400
    3    000500 Environment division.
    4    000600 Configuration section.
    5    000700   Source-computer.  ibm-iSeries.
    6    000800   Object-computer.  ibm-iSeries.
    7    000900 Input-output section.
    8    001000   File-control.
    9    001100     Select file1
   10    001200       assign to database-samplefi                                               00/08/15
   11    001300       organization is sequential
   12    001400       access is sequential
   13    001500       file status is fs1.
         001600
   14    001700 Data division.
   15    001800 File section.
   16    001900 fd  file1.
   17    002000 01  record1.
         002100 copy dds-all-formats of samplefi.                                                00/08/15
   18   +000001       05  SAMPLEFI-RECORD PIC X(546).                                  <-ALL-FMTS
        +000002*    I-O FORMAT:SAMPLEFILE FROM FILE SAMPLEFI   OF LIBRARY CBLGUIDE     <-ALL-FMTS
        +000003*                                                                       <-ALL-FMTS
   19   +000004       05  SAMPLEFILE    REDEFINES SAMPLEFI-RECORD.                    <-ALL-FMTS
   20   +000005           06 VARITEM.                                                 <-ALL-FMTS
        +000006*                (可変長フィールド)                                     <-ALL-FMTS
   21   +000007               49 VARITEM-LENGTH   PIC S9(4) COMP-4.                   <-ALL-FMTS
   22   +000008               49 VARITEM-DATA     PIC X(100).                         <-ALL-FMTS
   23   +000009           06 TIMEITEM             PIC X(8).                           <-ALL-FMTS
        +000010*                (時刻フィールド)                                       <-ALL-FMTS
   24   +000011           06 DATEITEM             PIC X(8).                           <-ALL-FMTS
        +000012*                (日付フィールド)                                       <-ALL-FMTS
   25   +000013           06 TIMESTAMP            PIC X(26).                          <-ALL-FMTS
        +000014*                (タイム・スタンプ・フィールド)                         <-ALL-FMTS
   26   +000015           06 GRAPHITEM            PIC G(100) DISPLAY-1.               <-ALL-FMTS
        +000016*                (グラフィック・フィールド)                             <-ALL-FMTS
   27   +000017           06 VGRAPHITEM.                                              <-ALL-FMTS
        +000018*                (可変長フィールド)                                     <-ALL-FMTS
   28   +000019               49 VGRAPHITEM-LENGTH                                    <-ALL-FMTS
        +000020                                  PIC S9(4) COMP-4.                    <-ALL-FMTS
        +000021*                (2 バイト文字の数)                                     <-ALL-FMTS
   29   +000022               49 VGRAPHITEM-DATA  PIC G(100) DISPLAY-1.               <-ALL-FMTS
        +000023*                (グラフィック・フィールド)                             <-ALL-FMTS
   30    002200 Working-Storage section.
   31    002300 77  fs1      pic x(2).
         002400
   32    002500 Procedure division.
         002600 Mainline.
   33    002700     stop run.
                    * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

*図 113. 可変長 DBCS グラフィック・データ項目と *PICGGRAPHIC を使用する ILE COBOL プログラム*

浮動小数点フィールド

CRTCBLMOD または CRTBNDCBL コマンドの CVTOPT パラメーターに *FLOAT を指定するか、または PROCESS ステートメントの FLOAT オプションを指定することによって、内部浮動小数点数をプログラムに含めることができます。

*FLOAT を指定した場合、浮動小数点データ・タイプは、その DDS 名および USAGE として COMP-1 (単精度) または COMP-2 (倍精度) と共にプログラム中に含められます。 *FLOAT を指定しない場合、浮動小数点データ・タイプは、USAGE が 2 進数の FILLER フィールドとして宣言されます。

たとえば、以下の DDS によって単精度浮動小数点フィールドに *FLOAT を指定した場合、

```
  COMP1          9F         FLTPCN(*SINGLE)
```

プログラムに含められるデータは、以下のようになります。

```
06  COMP1               COMP-1.
```

上記の DDS で *FLOAT を指定しない場合 (または *NOFLOAT を指定した場合)、生成される DDS フィールドは以下のようになります。

```
06  FILLER      PIC 9(5)        COMP-4.
```

一般に、浮動小数点データ項目は、10 進数データ項目が使用されるところであればどこにでも使用できます

第 19 章 外部接続装置へのアクセス

この章では、ILE COBOL が外部接続装置と対話する方法について説明します。外部接続装置とは、プリンター、テープ装置、ディスケット装置、ディスプレイ装置、および他システムなどの外部接続のハードウェアです。

ILE COBOL から外部接続装置にアクセスするには、装置ファイルを使用します。**装置ファイル**は、ディスプレイ装置、プリンター、テープ装置、ディスケット装置、および通信回線で接続されている他のシステムなどの外部接続装置に対するアクセスを提供するファイルです。

装置ファイルの種類

ILE COBOL プログラムでシステムの装置の読み書きを行うには、その前に装置が構成される時点で、オペレーティング・システムが装置のハードウェア機能を識別するための装置記述を作成しなければなりません。装置ファイルは、装置を使用する方法を指定します。 ILE COBOL プログラムでは、特定の装置ファイルを参照することにより、システムに対して記述されている方法で装置を使用します。装置ファイルは、ILE COBOL プログラムからの出力データを装置への出力用にフォーマット設定したり、ILE COBOL プログラムのために装置からの入力データをフォーマット設定したりします。

外部接続装置と、それにアクセスするために使用する装置ファイルのリストを、表 27 に示します。

表 27. 装置ファイルとそれに関連する外部接続装置

装置ファイル	関連する外部接続装置	CL コマンド	ILE COBOL 装置名	ILE COBOL のデフォルト・ファイル名
プリンター・ファイル	プリンターへのアクセスを提供し、印刷出力の形式を記述する。	CRTPRTF CHGPRTF OVRPRTF	PRINTER FORMATFILE	QPRINT
テープ・ファイル	テープ装置に保管されているデータ・ファイルへのアクセスを提供する。	CRTTAPF CHGTAPF OVRTAPF	TAPEFILE	QTAPE
ディスケット・ファイル	ディスケット装置に保管されているデータ・ファイルへのアクセスを提供する	CRTDKTF CHGDKTF OVRDKTF	DISKETTE	QDKT
ディスプレイ・ファイル	ディスプレイ装置へのアクセスを提供する。	CRTDSPF CHGDSPF OVRDSPF	WORKSTATION	
ICF ファイル	システム上のプログラムが、別のシステムのプログラムと通信できるようにする	CRTICFF CHGICFF OVRICFF	WORKSTATION	

装置ファイルには使用される装置を識別するファイル記述が含まれていますが、データは含まれていません。

プリンターへのアクセス

ILE COBOL プログラムから、プリンターへの印刷出力を作成するには、1 つまたは複数のプリンター・ファイルに対して WRITE ステートメントを出します。 IBM 提供のプリンター・ファイル (QPRINT など) を使用することもできますし、印刷ファイル作成 (CRTPRTF) コマンドを使用して、独自のファイルを作成することもできます。 CRTPRTF コマンドの詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「*プログラミング*」カテゴリーの中の『*CL および API*』セクションを参照してください。

ILE COBOL プログラムの中でプリンター・ファイルを使用する場合には、次の処理を行わなければなりません。

- 環境部 (ENVIRONMENT DIVISION) の FILE-CONTROL 段落のファイル制御記入項目を使用してプリンター・ファイルの名前を指定する。
- データ部 (DATA DIVISION) のファイル記述項目を使用してプリンター・ファイルを記述する。

プリンター・ファイルで有効なファイル操作は WRITE、OPEN、および CLOSE です。

プリンター・ファイルの名前の指定

ILE COBOL のプログラムの中でプリンター・ファイルを使用するには、環境部 (ENVIRONMENT DIVISION) の FILE-CONTROL 段落内のファイル制御記入項目を使用してプリンター・ファイルの名前を指定しなければなりません。
FILE-CONTROL 段落の説明については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。 ILE COBOL プログラム中では複数のプリンター・ファイルを使用できますが、各プリンター・ファイルの名前は固有でなければなりません。

プリンター・ファイルは、プログラム記述ファイルまたは外部記述ファイルのどちらでもかまいません。

プログラムに記述するプリンター・ファイルの名前は、FILE-CONTROL 段落で次のようにして指定します。

```
FILE-CONTROL.
    SELECT printer-file-name
        ASSIGN TO PRINTER-printer_device_name
        ORGANIZATION IS SEQUENTIAL.
```

外部記述のプリンター・ファイルは、FILE-CONTROL 段落で次のようにして指定します。

```
FILE-CONTROL.
    SELECT printer-file-name
        ASSIGN TO FORMATFILE-printer_device_name
        ORGANIZATION IS SEQUENTIAL.
```

ファイルを選択するには SELECT 文節を使用します。そのファイルは、DATA DIVISION の FD 項目で指定されているものでなければなりません。

プリンター・ファイルをプリンターと関連付けるには、ASSIGN 文節を使用します。プログラムに記述するプリンター・ファイルを使用する場合は、ASSIGN 文節

に装置タイプ PRINTER を指定してください。外部記述プリンター・ファイルを使用する場合は、ASSIGN 文節に装置タイプ FORMATFILE を指定してください。

プリンター・ファイルの名前を指定する場合は、ファイル制御記入項目に ORGANIZATION IS SEQUENTIAL を使用してください。

プリンター・ファイルの記述

環境部 (ENVIRONMENT DIVISION) でプリンター・ファイルの名前を指定した後で、データ部 (DATA DIVISION) のファイル記述項目でそのプリンター・ファイルを記述しなければなりません。ファイル記述項目の詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。プリンター・ファイルを記述するには、形式 4 のファイル記述項目を使用します。

プリンター・ファイルは、プログラム記述にしたり外部記述にしたりすることができます。プログラム記述のプリンター・ファイルは、装置 PRINTER に割り当てられます。外部記述プリンター・ファイルは、装置 FORMATFILE に割り当てられます。 FORMATFILE を使用した場合は AS/400 の機能を最大限活用でき、PRINTER を使用した場合はプログラムの移植性が大きくなります。

外部記述プリンター・ファイルを使用することは、プログラム記述のプリンター・ファイルを使用する場合に比べて次の利点があります。

- 1 つの WRITE ステートメントで複数行印刷できる。 1 つの WRITE ステートメントで複数の行を書き込んでいて END-OF-PAGE 条件に達した場合、すべての行を印刷した後で END-OF-PAGE 命令ステートメントが処理されます。オーバーフロー域に行を印刷したり、END-OF-PAGE 命令ステートメントが処理される前に改ページすることもできます。

  FORMATFILE 使用時に END-OF-PAGE 条件になる場合については 518 ページの図 116 に示されています。
- 標識値に基づいて、オプションでフィールドを印刷することができる。
- フィールド値の編集を簡単に定義できる。
- 印刷形式の維持管理が簡単である (特に複数のプログラムが使用する場合)。

FORMATFILE ファイルに ADVANCING 句を使用すると、コンパイル・エラーが出されます。 FORMATFILE ファイルにおいて行の ADVANCING は、SKIPA や SKIPB などの DDS キーワードによって、また行番号の使用によって管理されます。

プログラム記述プリンター・ファイルの記述

プログラム記述のプリンター・ファイルは、装置 PRINTER に割り当てなければなりません。 DATA DIVISION の中にプログラム記述のプリンター・ファイルを記述した簡単なファイル記述項目は、次のようになります。

```
FD  print-file.
01  print-record              PIC X(132).
```

LINAGE 文節を使用して行送りと改ページの制御を処理する: すべての行送りと改ページの制御は、プログラム記述プリンター・ファイルのファイル記述項目の中に LINAGE 文節を指定することによって、コンパイラーの生成するコードにより内部的に要求することができます。

```
FD  print-file
    LINAGE IS integer-1 LINES
           WITH FOOTING AT integer-2
           LINES AT TOP integer-3
           LINES AT BOTTOM integer-4.
01  print-record              PIC X(132).
```

用紙の位置指定が行われるのは、最初の WRITE ステートメントの実行時だけです。プリンターの用紙は新しい物理ページになり、LINAGE-COUNTER は 1 に設定されます。プリンター・ファイルが共用されている場合に他のプログラムがレコードをこのファイルに書き込んだ場合も、依然として ILE COBOL の WRITE ステートメントが最初の WRITE ステートメントであると見なされます。そのファイルの最初の WRITE ステートメントでない場合でも、用紙の位置指定は ILE COBOL コンパイラーによって処理されます。

WRITE ステートメントの行送りや改ページは、すべて内部で制御されます。用紙の位置指定が ILE COBOL コンパイラーに適切に定義されていないなら、ページの物理サイズは無視されます。 LINAGE 文節があり PRINTER に割り当てられているファイルの場合、改ページ操作は、論理ページ (ページ本体) の終わりにまで進んでから、下部マージンと上部マージンの分だけ行送りするということです。

LINAGE 文節を使用するとパフォーマンスが低下します。 LINAGE 文節は必要な場合にだけ使用するようにしてください。物理ページ送りが受け入れられる場合、LINAGE 文節は必要ありません。

外部記述プリンター・ファイル (FORMATFILE) の記述

外部記述プリンター・ファイルは、装置 FORMATFILE に割り当てなければなりません。 FORMATFILE という語が使用されているのは、そのようなファイルの WRITE ステートメントでは FORMAT 句が有効であり、そのファイルの DDS においてデータのフォーマット設定が指定されることによります。 DATA DIVISION の中に外部記述プリンター・ファイルを記述した簡単なファイル記述項目は、次のようになります。

```
FD  print-file.
01  print-record.
    COPY DDS-ALL-FORMATS-O OF print-file-dds.
```

使用する FORMATFILE ファイルの DDS を作成してください。 DDS の作成方法
については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS**
Information Center の「*プログラミング*」カテゴリーを参照してください。

FORMATFILE ファイルの DDS を作成したなら、形式 2 の COPY ステートメントを使用して、プリンター・ファイルのデータ・レコードのレイアウトを記述してください。 ILE COBOL プログラムをコンパイルすると、形式 2 の COPY により、プリンター・ファイルを記述するためのデータ部 (DATA DIVISION) ステートメントが作成されます。すべての様式に対して 1 つのストレージ域を生成するためには、形式 2 の COPY ステートメントの DDS-ALL-FORMATS-O オプションを使用してください。

FORMATFILE の装置を指定すれば、次の 2 つの方法で印刷出力のフォーマット設定を行えます。

1. WRITE ステートメントに指定されている FORMAT 句に適切な値を使用することにより、印刷の形式と順序を選択する。たとえば、ページごとに 1 つの形式を 1 回ずつ使用してヘッディングを作成し、別の形式を使用してそのページの明細行を作成する、というようにします。
2. 標識の値を設定し、WRITE ステートメントの INDICATOR 句によってその標識を渡すことにより、各様式の印刷出力のための適切なオプションを選択する。たとえば、フィールドに下線を付けたり、形式印刷前後にブランク行を作成したり、特定のフィールドの印刷をスキップしたりすることができます。

FORMATFILE に割り当てられたファイルには LINAGE 文節を使用しないでください。もし使用すると、LINAGE 文節を無視することを示すコンパイル時エラー・メッセージが出されます。

プリンター・ファイルへの書き出し

プリンター・ファイルに書き出すには、その前にそのファイルをオープンしておかなければなりません。プリンター・ファイルをオープンするには、形式 1 の OPEN ステートメントを使用します。プリンター・ファイルは OUTPUT 用にオープンしてください。

```
OPEN OUTPUT printer-file-name.
```

プリンター・ファイルに出力を送信するには、WRITE ステートメントを使用します。プログラム記述プリンター・ファイルに出すには、形式 1 の WRITE ステートメントを使用してください。外部記述プリンター・ファイルに書き出すには、形式 3 の WRITE ステートメントを使用してください。

関数名 CSP に関連した簡略名をプリンター・ファイルの WRITE ステートメントの ADVANCING 句に指定すると、ADVANCING 0 LINES を指定した場合と同じ効果が得られます。関数名 C01 に関連した簡略名をプリンター・ファイルの WRITE ステートメントの ADVANCING 句に指定すると、ADVANCING PAGE を指定した場合と同じ効果が得られます。

装置タイプ FORMATFILE に ASSIGN したファイルの WRITE ステートメントに ADVANCING 句を指定することはできません。

プリンター・ファイルを使用し終えた時点で、それをクローズしてください。プリンター・ファイルをクローズするには、形式 1 の CLOSE ステートメントを使用します。 ファイルをクローズすると、そのファイルは、それを再びオープンするまで処理できなくなります。

```
CLOSE printer-file-name.
```

ILE COBOL プログラムでの FORMATFILE ファイルの使用例

このプログラムは、社員データ・ファイルに基づいて、すべての男性従業員に関する明細従業員レコードを印刷します。入力レコードは、従業員番号の昇順に並べます。入力ファイルと出力ファイルは両方とも外部記述です。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* PHYSICAL FILE DDS FOR PERSONNEL FILE IN FORMATFILE EXAMPLE
     A
     A                                      UNIQUE
     A          R PERSREC
     A            EMPLNO         6S
     A            NAME          30
     A            ADDRESS1      35
     A            ADDRESS2      20
     A            BIRTHDATE      6
     A            MARSTAT        1
     A            SPOUSENAME    30
     A            NUMCHILD       2S
     A          K EMPLNO
```

図 114. ILE COBOL プログラムでの FORMATFILE ファイルの使用例 -- 物理ファイル DDS

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* PRINTER FILE DDS FOR FORMATFILE EXAMPLE
     A*
     A                                     1 INDARA REF(PERSFILE)
     A          R HEADING 2                  SKIPB(1) SPACE(3)  3
     A                                      15'PERSONNEL LISTING'
     A                                        UNDERLINE
     A                                      33'- ORDERED BY'
     A            ORDERTYPE     15          46
     A                                      80DATE EDTCDE(Y)
     A                                      93TIME   4
     A                                     115'PAGE:'
     A                                      +1PAGNBR EDTCDE(3)
     A*
     A          R DETAIL 5                   SPACEA(3) 6
     A*    LINE 1
     A                                       1'NAME:'
     A            NAME      R              11UNDERLINE
     A                                      55'EMPLOYEE NUMBER:'
     A            EMPLNO    R              73
     A                                      87'DATE OF BIRTH:'
     A            BIRTHDATE R             103SPACEA(1) 7
     A*    LINE 2
     A                                       1'ADDRESS:'
     A            ADDRESS1  R              11
     A                                      55'MARITAL STATUS:'
     A            MARSTAT   R              73
     A  01                                  87'SPOUSE''S NAME:'
     A  01  8     SPOUSENAMER             103
     A*    LINE 3
     A            ADDRESS2  R              11SPACEB(1)
     A                                      55'CHILDREN:'
     A            NUMCHILD  R              73EDTCDE(3) 9
```

図 115. ILE COBOL プログラムでの FORMATFILE ファイルの使用例 -- プリンター・ファイル DDS

1 INDARA は、そのファイルのために別個の標識域を使用することを指定します。

2 HEADING は、各ページのヘッディングを提供する様式の名前です。

3 SKIPB(1) と SPACEA(3) は、以下の処理を行うためのものです。
1. 様式 HEADING の印刷前に、次ページの行 1 にスキップする。

2. 様式 HEADING の印刷後に、ブランクを 3 行入れる。

4 DATE、TIME、および PAGNBR を使用して、様式 HEADING の印刷時に現在日付、時刻、およびページ番号が自動的に印刷されるようにします。

5 DETAIL は、社員データ・ファイル中の従業員ごとに明細行を印刷するために使用される様式名です。

6 SPACEA(3) により、個々の従業員明細行の後にブランクを 3 行入れます。

7 SPACEA(1) により、BIRTHDATE フィールドの印刷の後に 1 行のブランク行が印刷されます。その結果、それ以降の同じ形式のフィールドが、新しい行に印刷されます。

8 01 は、様式 DETAIL の印刷時に、ILE COBOL プログラムが標識 01 をオンにして渡す場合に限り、このフィールドが印刷されることを意味します。

9 EDTCDE(3) は、この数値フィールドの印刷時に先行ゼロを除去するために使用されています。

```
5722WDS V5R4M0  060210 LN  IBM                          CBLGUIDE/FRMTFILE       ISERIES1   06/02/15 14:35:57        ページ     2
                                   ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     FRMTFILE.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600 SOURCE-COMPUTER. IBM-ISERIES.
    6     000700 OBJECT-COMPUTER. IBM-ISERIES.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
    9     001000     SELECT PERSREPT ASSIGN TO FORMATFILE-PERSREPT-SI  1
   11     001100         ORGANIZATION IS SEQUENTIAL.
   12     001200     SELECT PERSFILE ASSIGN TO DATABASE-PERSFILE
   14     001300         ORGANIZATION IS INDEXED
   15     001400         ACCESS MODE IS SEQUENTIAL
   16     001500         RECORD IS EXTERNALLY-DESCRIBED-KEY.
          001600
   17     001700 DATA DIVISION.
   18     001800 FILE SECTION.
   19     001900 FD  PERSREPT.
   20     002000 01  PERSREPT-REC.
          002100     COPY DDS-ALL-FORMATS-O OF PERSREPT.  2
   21    +000001       05  PERSREPT-RECORD PIC X(130).                                    <-ALL-FMTS
         +000002* OUTPUT FORMAT:HEADING    FROM FILE PERSREPT   OF LIBRARY CBLGUIDE      <-ALL-FMTS
         +000003*                                                                        <-ALL-FMTS
   22    +000004       05  HEADING-O     REDEFINES PERSREPT-RECORD.                     <-ALL-FMTS
   23    +000005           06 ORDERTYPE             PIC X(15).                          <-ALL-FMTS
         +000006* OUTPUT FORMAT:DETAIL     FROM FILE PERSREPT   OF LIBRARY CBLGUIDE      <-ALL-FMTS
         +000007*                                                                        <-ALL-FMTS
   24    +000008       05  DETAIL-O      REDEFINES PERSREPT-RECORD.  3                   <-ALL-FMTS
   25    +000009           06 NAME                 PIC X(30).                          <-ALL-FMTS
   26    +000010           06 EMPLNO                PIC S9(6).                          <-ALL-FMTS
   27    +000011           06 BIRTHDATE             PIC X(6).                           <-ALL-FMTS
   28    +000012           06 ADDRESS1              PIC X(35).                          <-ALL-FMTS
   29    +000013           06 MARSTAT               PIC X(1).                           <-ALL-FMTS
   30    +000014           06 SPOUSENAME            PIC X(30).                          <-ALL-FMTS
   31    +000015           06 ADDRESS2              PIC X(20).                          <-ALL-FMTS
   32    +000016           06 NUMCHILD              PIC S9(2).                          <-ALL-FMTS
   33     002200 FD  PERSFILE.
   34     002300 01  PERSFILE-REC.
          002400     COPY DDS-ALL-FORMATS-O OF PERSFILE.
   35    +000001       05  PERSFILE-RECORD PIC X(130).                                    <-ALL-FMTS
         +000002*    I-O FORMAT:PERSREC    FROM FILE PERSFILE   OF LIBRARY CBLGUIDE      <-ALL-FMTS
         +000003*                                                                        <-ALL-FMTS
         +000004*THE KEY DEFINITIONS FOR RECORD FORMAT  PERSREC                         <-ALL-FMTS
         +000005*  NUMBER              NAME               RETRIEVAL       ALTSEQ       <-ALL-FMTS
         +000006*   0001   EMPLNO                        ASCENDING         NO          <-ALL-FMTS
   36    +000007       05  PERSREC       REDEFINES PERSFILE-RECORD.                     <-ALL-FMTS
   37    +000008           06 EMPLNO                PIC S9(6).                          <-ALL-FMTS
   38    +000009           06 NAME                  PIC X(30).                          <-ALL-FMTS
   39    +000010           06 ADDRESS1              PIC X(35).                          <-ALL-FMTS
   40    +000011           06 ADDRESS2              PIC X(20).                          <-ALL-FMTS
   41    +000012           06 BIRTHDATE             PIC X(6).                           <-ALL-FMTS
   42    +000013           06 MARSTAT               PIC X(1).                           <-ALL-FMTS
```

図 116. ILE COBOL プログラムでの FORMATFILE ファイルの使用例 (1/2)

```
5722WDS V5R4M0  060210 LN  IBM                          CBLGUIDE/FRMTFILE       ISERIES1   06/02/15 14:35:57        ページ    3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
   43    +000014           06 SPOUSENAME           PIC X(30).                                <-ALL-FMTS
   44    +000015           06 NUMCHILD             PIC S9(2).                                <-ALL-FMTS
          002500
   45     002600 WORKING-STORAGE SECTION.
   46     002700 77  HEAD-ORDER                  PIC X(15)
          002800                                   VALUE "EMPLOYEE NUMBER".
   47     002900 01  PERSREPT-INDICS.
          003000     COPY DDS-ALL-FORMATS-O-INDIC OF PERSREPT.  4
   48    +000001       05  PERSREPT-RECORD.                                                   <-ALL-FMTS
         +000002* OUTPUT FORMAT:HEADING    FROM FILE PERSREPT   OF LIBRARY CBLGUIDE          <-ALL-FMTS
         +000003*                                                                           <-ALL-FMTS
         +000004*           06 HEADING-O-INDIC.                                             <-ALL-FMTS
         +000005* OUTPUT FORMAT:DETAIL     FROM FILE PERSREPT   OF LIBRARY CBLGUIDE          <-ALL-FMTS
         +000006*                                                                           <-ALL-FMTS
   49    +000007           06 DETAIL-O-INDIC.                                              <-ALL-FMTS
   50    +000008                07 IN01            PIC 1  INDIC 01.                        <-ALL-FMTS
          003100
   51     003200 77  EOF-FLAG                     PIC X(1)
          003300                                   VALUE "0".
   52     003400    88  NOT-END-OF-FILE           VALUE "0".
   53     003500    88  END-OF-FILE               VALUE "1".
   54     003600 77  MARRIED                      PIC X(1)
          003700                                   VALUE "M".
          003800
   55     003900 PROCEDURE DIVISION.
          004000 MAIN-PROGRAM SECTION.
          004100 MAINLINE.
   56     004200     OPEN INPUT PERSFILE
          004300          OUTPUT PERSREPT.
   57     004400     PERFORM HEADING-LINE.
   58     004500     PERFORM UNTIL END-OF-FILE
   59     004600         READ PERSFILE
   60     004700             AT END SET END-OF-FILE TO TRUE
   61     004800             NOT AT END PERFORM PRINT-RECORD  5
          004900         END-READ
          005000     END-PERFORM
   62     005100     CLOSE PERSFILE
          005200           PERSREPT.
   63     005300     STOP RUN.
          005400
          005500 PRINT-RECORD.
   64     005600     MOVE CORR PERSREC TO DETAIL-O.  6
                *** CORRESPONDING items for statement 64:
                ***     EMPLNO
                ***     NAME
                ***     ADDRESS1
                ***     ADDRESS2
                ***     BIRTHDATE
                ***     MARSTAT
                ***     SPOUSENAME
                ***     NUMCHILD
                *** End of CORRESPONDING items for statement 64
   65     005700     IF MARSTAT IN PERSFILE-REC IS EQUAL MARRIED THEN  7
   66     005800         MOVE B"1" TO IN01 IN DETAIL-O-INDIC
          005900     ELSE
   67     006000         MOVE B"0" TO IN01 IN DETAIL-O-INDIC
          006100     END-IF
   68     006200     WRITE PERSREPT-REC FORMAT IS "DETAIL"  8
          006300         INDICATORS ARE DETAIL-O-INDIC
   69     006400         AT EOP PERFORM HEADING-LINE  9
          006500     END-WRITE.
          006600
          006700 HEADING-LINE.
   70     006800     MOVE HEAD-ORDER TO ORDERTYPE
   71     006900     WRITE PERSREPT-REC FORMAT IS "HEADING"
          007000     END-WRITE.
          007100
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 116. ILE COBOL プログラムでの FORMATFILE ファイルの使用例 (2/2)

1 外部記述プリンター・ファイルは、装置 FORMATFILE に割り当てられます。 SI は、DDS の中で別個の標識域が指定されているということを示します。

2 形式 2 の COPY ステートメントを使用して、プリンター・ファイルのためのフィールドをプログラムの中にコピーします。

3 様式 DETAIL のフィールドは、別個の 3 行に印刷されますが、1 つのレコードで定義されていることに注意してください。

4 形式 2 の COPY ステートメントを使用して、プリンター・ファイル中で使用される標識をプログラムの中にコピーします。

5 段落 PROCESS-RECORD においては、従業員レコードごとに PRINT-RECORD を処理します。

6 従業員レコード中のすべてのフィールドを様式 DETAIL のレコードに移動します。

7 従業員が既婚の場合は標識 01 がオンになります。独身の場合はこの標識はオフになり、DETAIL の配偶者名フィールドが印刷されないようにします。

8 印刷制御のための標識 01 を渡して、様式 DETAIL を印刷します。

9 ページごとの行数が超過した場合に、END-OF-PAGE となります。様式 HEADING が新しいページに印刷されます。

テープ装置に保管されているファイルへのアクセス

テープ装置との間でレコードを読み書きするには、**テープ・ファイル**を使用します。テープ装置に保管されているファイルは、次の 2 つのカテゴリーに分けられます。

- **順次単一ボリューム:** 1 つのボリュームに全体が入っている 1 つの順次ファイル。このボリュームには複数のファイルが入れられることがあります。
- **順次マルチボリューム:** 複数のボリュームに入っている 1 つの順次ファイル。

テープ・ファイルの作成 (CRTTAPF) コマンドを使用することにより、独自のテープ・ファイルを作成できます。 CRTTAPF コマンドの詳細については、Web サイト http://www.ibm.com/eserver/iseries/infocenter にある **i5/OS Information Center** の「プログラミング」カテゴリーの中の『*CL および API*』セクションを参照してください。 もう 1 つの方法として、IBM 提供のデフォルトのテープ・ファイル QTAPE を使用することもできます。テープ・ファイルは、使用するテープ装置を識別します。

テープ装置に保管されているファイルを ILE COBOL プログラムで使用するには、次のことを行わなければなりません。

- 環境部 (ENVIRONMENT DIVISION) の FILE-CONTROL 段落のファイル制御記入項目を使用してファイルの名前を指定する。
- データ部 (DATA DIVISION) のファイル記述項目を使用してファイルを記述する。

テープ装置は順次アクセスしかできないので、テープ装置には順次ファイルしか保管できません。 テープ装置に保管されているファイルのレコードには、固定長レコードまたは可変長レコードが可能です。

テープ装置で有効なファイル操作は OPEN、CLOSE、READ、および WRITE です。

テープ装置に保管されているファイルの名前の指定

テープ装置に保管されている順次ファイルを ILE COBOL プログラムで使用するには、ENVIRONMENT DIVISION の FILE-CONTROL 段落でファイル制御記入項目を使用してファイルの名前を指定しなければなりません。 FILE-CONTROL 段落の説明については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

ファイルの名前は、FILE-CONTROL 段落の中で以下のようにして指定します。

```
FILE-CONTROL.
    SELECT sequential-file-name
        ASSIGN TO TAPEFILE-tape_device_name
        ORGANIZATION IS SEQUENTIAL.
```

ファイルを選択するには SELECT 文節を使用します。そのファイルは、DATA DIVISION の FD 項目で指定されているものでなければなりません。

ファイルをテープ装置と関連付けるには、ASSIGN 文節を使用します。テープ・ファイルを使用するには、ASSIGN 文節に装置タイプ TAPEFILE を指定しなければなりません。

テープ・ファイルを用いてアクセスするファイルの名前を指定する場合は、ファイル制御記入項目に ORGANIZATION IS SEQUENTIAL を使用してください。

テープ装置に格納されているファイルの記述

環境部 (ENVIRONMENT DIVISION) で順次ファイルの名前を指定した後で、データ部 (DATA DIVISION) の中でファイル記述項目を使用してファイルを記述しなければなりません。ファイル記述項目の詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。テープ・ファイルを使用してアクセスする順次ファイルを記述するには、形式 3 のファイル記述項目を使用します。

テープ・ファイルにはデータ記述仕様 (DDS) がありません。テープ装置に保管されている順次ファイルは、プログラム記述ファイルでなければなりません。 ILE COBOL プログラムでは、テープ装置との間で受け渡しされるデータをテープ・ファイル記述に指定された方法で配置できるようなレコード様式で、フィールドを記述する必要があります。

DATA DIVISION の中で、テープ・ファイルによってアクセスする順次ファイルを記述する簡単なファイル記述項目は、以下のようなものになります。

```
FD  sequential-file-name.
01  sequential-file-record.
    05  record-element-1   PIC ... .
    05  record-element-2   PIC ... .
    05  record-element-3   PIC ... .
.
.
.
```

可変長レコードのテープ・ファイルの記述

可変長レコードのファイルをテープ装置に保管することができます。ファイルの最大レコード長と最小レコード長を定義するには、ファイルの FD 項目に形式 3 の RECORD 文節を指定します。

DATA DIVISION の中で可変長レコードの順次ファイルを記述する簡単なファイル記述項目は、以下のようなものになります。

```
FILE SECTION.
FD  sequential-file-name
    RECORD IS VARYING IN SIZE
              FROM integer-6 TO integer-7
              DEPENDING ON data-name-1.
01  minimum-sized-record.
    05  minimum-sized-element     PIC X(integer-6).
01  maximum-sized-record.
    05  maximum-sized-element     PIC X(integer-7).
⋮
WORKING-STORAGE SECTION.
77  data-name-1                   PIC 9(5).
⋮
```

ファイルの中のレコードの最小レコード・サイズは、*integer-6* によって定義されます。ファイルの中のレコードの最大レコード・サイズは、*integer-7* によって定義されます。 *integer-6* で指定されるものよりレコード長が短いファイルや *integer-7* で指定されるものよりレコード長が長いファイルのレコード記述は作成しないようにしてください。この規則を破るレコード記述があると、ILE COBOL コンパイラーからコンパイル時エラー・メッセージが出されます。 ILE COBOL コンパイラーは、レコード記述によって導き出される限界値を使用することになります。また、ILE COBOL コンパイラーは、レコード記述の中にレコード長が *integer-7* になるようなレコード記述がない場合にも、コンパイル時エラー・メッセージを出します。

可変長レコードに対して READ ステートメントまたは WRITE ステートメントが実行される場合、そのレコードのサイズは *data-name-1* の内容で定義されます。

可変長レコードの処理方法については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の形式 3 の RECORD 文節の部分を参照してください。

テープ装置に保管されているファイルの読み書き

テープ装置に保管されているファイルの読み書きを行うには、その前にまずファイルをオープンしなければなりません。ファイルをオープンするには、形式 1 の OPEN ステートメントを使用します。テープ装置に保管されているファイルから読み取る場合は、INPUT モードでオープンする必要があります。テープ装置に保管されているファイルに書き込む場合は、OUTPUT モードまたは EXTEND モードでオープンする必要があります。テープ装置に保管されているファイルを I-O モードでオープンすることは**できません**。 OPEN ステートメントのいくつかの例を次に示します。

```
OPEN INPUT sequential-file-name.
OPEN OUTPUT sequential-file-name.
OPEN EXTEND sequential-file-name.
```

テープ装置に保管されている順次ファイルから読み取るには、形式 1 の READ ステートメントを使用します。 READ ステートメントは、ILE COBOL プログラムがファイル中の次の論理レコードを使用できるようにします。順次マルチボリューム・ファイルで、READ ステートメントの処理中にボリュームの終わりが検出されたものの、論理的なファイルの終わりに達していない場合には、以下のアクションが示されている順番で行われます。

1. 標準的な終了ボリューム・ラベル・プロシージャーが処理される。

2. ボリュームの切り替えが行われる。
3. 標準的な開始ボリューム・ラベル・プロシージャーが実行される。
4. 次のボリュームの最初のデータ・レコードを使用可能にする。

ILE COBOL プログラムは、読み取り操作中に上記のアクションが取られたという標識を受け取りません。

テープ装置に保管されているレコードを順次ファイルに書き込むには、形式 1 の WRITE ステートメントを使用します。順次マルチボリューム・ファイルで、WRITE ステートメントの処理中にボリュームの終わりが検出された場合、以下に示すアクションがこの順に行われます。
1. 標準的な終了ボリューム・ラベル・プロシージャーが実行される。
2. ボリュームの切り替えが行われる。
3. 標準的な開始ボリューム・ラベル・プロシージャーが実行される。
4. 次のボリュームにデータ・レコードが書き込まれる。

ボリュームの終わり状態になっても、そのことを示す標識は ILE COBOL プログラムには返されません。

テープ装置に保管されているファイルの使用が終ったなら、それをクローズしなければなりません。ファイルをクローズするには、形式 1 の CLOSE ステートメントを使用します。 ファイルをクローズすると、そのファイルは、それを再びオープンするまで処理できなくなります。

```
CLOSE sequential-file-name.
```

CLOSE ステートメントには、ボリュームを巻き戻したりアンロードしたりするオプションもあります。

通常は、テープ・ファイルに対して CLOSE ステートメントを実行する時点で、ボリュームの巻き戻しを実行します。しかし、ファイルをクローズした後でも現行ボリュームをその時点の位置のままにしておきたい場合は、CLOSE ステートメントに NO REWIND 句を指定してください。 NO REWIND を指定すると、リールは巻き戻されません。

順次マルチボリューム・テープ・ファイルの場合に REEL/UNIT FOR REMOVAL 句を使用すると、現行ボリュームの巻き戻しとアンロードが行われます。続いて、ボリュームが除去されたことがシステムに通知されます。

ボリュームの巻き戻しとアンロードについては、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の中の形式 1 の CLOSE ステートメントに関する説明を参照してください。

可変長レコードのテープ・ファイルの読み書き

可変長レコードをテープ・ファイルに読み書きする場合、その可変長レコードの最大長が、必ずテープの最大レコード長以下になるようにしてください。テープの最大レコード長は、それが OUTPUT 用にオープンされた時点で決められます。 テープの最大レコード長がテープに書き込まれるどの可変長レコードよりも短いなら、それらのレコードはテープの最大長で切り捨てられます。

テープ装置に保管されている順次ファイルから読み取るには、形式 1 の READ ステートメントを使用します。 READ ステートメントは、ILE COBOL プログラムがファイル中の次の論理レコードを使用できるようにします。

READ 操作が正常に実行された場合、*data-name-1* が指定されているなら、読み取ったレコードの文字位置数がそれに入れられます。 READ 操作が失敗した場合、*data-name-1* は READ 操作が試行される前の値のままです。

READ ステートメントに INTO 句を指定した場合、現行レコードの中で、暗黙の MOVE ステートメントの中の送り出し項目として加わっている文字位置数は、以下のようにして決められます。

- *data-name-1* が指定されている場合は *data-name-1* の内容。または、
- *data-name-1* が指定されていない場合は、読み取ったレコードの中の文字位置数。

READ ステートメント実行時に、読み込んだレコードの中の文字位置の数がそのファイルのレコード記述項目で指定されている最小レコード長より小さい場合は、レコード域のうち読み込まれた有効な最後の文字の右の部分にブランクが埋め込まれます。読み込んだレコードの中の文字位置の数がそのファイルのレコード記述項目で指定されている最大レコード長より大きい場合は、レコードは、レコード記述項目で指定されている最大レコード・サイズの右側で切り捨てられます。読み込んだレコードの長さが、ファイルのファイル記述項目で定義されている最小レコード長と最大レコード長の間にない場合には、ファイル状況 04 が戻されます。

テープ装置に保管されている可変長レコードを順次ファイルに書き込むには、形式 1 の WRITE ステートメントを使用します。書き込むレコードの長さは、*data-name-1* に指定します。 *data-name-1* を指定しない場合、書き込むレコードの長さは以下のようにして決められます。

- レコードに OCCURS...DEPENDING ON 項目が含まれている場合には、固定部分と、WRITE ステートメント実行時のオカレンスの数によって記述されるテーブル部分との合計。
- レコードに OCCURS...DEPENDING ON 項目が含まれていない場合には、レコード定義の中での文字位置数。

ディスケット装置に保管されているファイルへのアクセス

ディスケット装置の中にあって、基本交換形式、H 交換形式、または I 交換形式で初期設定されているディスケットのレコードを読み書きするには、**ディスケット・ファイル**を使用します。ディスケット装置に保管されているファイルは、次の 2 つのカテゴリーに分けられます。

- **順次単一ボリューム:** 1 つのディスケットに全体が入っている 1 つの順次ファイル。このディスケットには複数のファイルが入れられることがあります。
- **順次マルチボリューム:** 複数のディスケットに入っている 1 つの順次ファイル。

ディスケット・ファイルの作成 (CRTDKTF) コマンドを使用することにより、独自
のディスケット・ファイルを作成できます。 CRTDKTF コマンドの詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「プログラミング」カテゴリーの中の『*CL および API*』セクションを参照してください。 もう 1 つの方法として、IBM 提供のデフォルトのディスケッ

ト・ファイル QDKT を使用することもできます。このディスケット・ファイルは、
使用されるディスケット装置を識別します。

ディスケット装置に保管されているファイルを ILE COBOL プログラムで使用するには、次のことを行わなければなりません。

- 環境部 (ENVIRONMENT DIVISION) の FILE-CONTROL 段落のファイル制御記入項目を使用してファイルの名前を指定する。
- データ部 (DATA DIVISION) のファイル記述項目を使用してファイルを記述する。

ディスケット装置は順次アクセスしかできないので、1 つのディスケット装置には 1 つの順次ファイルしか保管できません。ディスケット装置で有効なファイル操作は OPEN、CLOSE、READ、および WRITE です。

ディスケット装置に格納されているファイルの名前の指定

ディスケット装置に保管されている順次ファイルを ILE COBOL プログラムで使用するには、ENVIRONMENT DIVISION の FILE-CONTROL 段落でファイル制御記入項目を使用してファイルの名前を指定しなければなりません。 FILE-CONTROL 段落の説明については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

ファイルの名前は、FILE-CONTROL 段落の中で以下のようにして指定します。

```
FILE-CONTROL.
    SELECT sequential-file-name
        ASSIGN TO DISKETTE-diskette_device_name
        ORGANIZATION IS SEQUENTIAL.
```

ファイルを選択するには SELECT 文節を使用します。そのファイルは、DATA DIVISION の FD 項目で指定されているものでなければなりません。

ファイルをディスケット装置と関連付けるには、ASSIGN 文節を使用します。ディスケット・ファイルを使用するには、ASSIGN 文節に装置タイプ DISKETTE を指定しなければなりません。

ディスケット・ファイルを用いてアクセスするファイルの名前を指定する場合は、ファイル制御記入項目に ORGANIZATION IS SEQUENTIAL を使用してください。

ディスケット装置に保管されているファイルの記述

環境部 (ENVIRONMENT DIVISION) で順次ファイルの名前を指定した後で、データ部 (DATA DIVISION) の中でファイル記述項目を使用してファイルを記述しなければなりません。ファイル記述項目の説明については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。ディスケット・ファイルを使用してアクセスする順次ファイルを記述するには、形式 2 のファイル記述項目を使用します。

ディスケット・ファイルにはデータ記述仕様 (DDS) がありません。 ディスケット装置に保管されている順次ファイルは、プログラム記述ファイルでなければなりません。 ILE COBOL プログラムでは、ディスケット装置との間で受け渡しされるデータをディスケット・ファイル記述に指定された方法で配置できるようなレコード様式で、フィールドを記述する必要があります。

DATA DIVISION の中で、ディスケット・ファイルによってアクセスする順次ファイルを記述する簡単なファイル記述項目は、以下のようなものになります。

```
FD  sequential-file-name.
01  sequential-file-record.
    05  record-element-1   PIC ... .
    05  record-element-2   PIC ... .
    05  record-element-3   PIC ... .
⋮
```

ディスケット装置に保管されているファイルの読み書き

ディスケット装置に保管されているファイルの読み書きを行うには、その前にまずファイルをオープンしなければなりません。ファイルをオープンするには、形式 1 の OPEN ステートメントを使用します。ディスケット装置に保管されているファイルから読み取る場合は、INPUT モードでオープンする必要があります。ディスケット装置に保管されているファイルに書き込む場合は、OUTPUT モードまたは EXTEND モードでオープンする必要があります。ディスケット装置に保管されているファイルを I-O モードでオープンすることは**できません**。 OPEN ステートメントのいくつかの例を次に示します。

```
OPEN INPUT sequential-file-name.
OPEN OUTPUT sequential-file-name.
OPEN EXTEND sequential-file-name.
```

ディスケット装置に保管されている順次ファイルから読み取るには、形式 1 の READ ステートメントを使用します。 READ ステートメントは、ILE COBOL プログラムがファイル中の次の論理レコードを使用できるようにします。

入力ファイルからレコードを読み込む場合、COBOL プログラムに指定されているレコード長は、ディスケットのデータ・ファイル・ラベルに記述されているレコード長と同じでなければなりません。 COBOL プログラムに指定するレコード長がデータ・ファイルのレコード長と同じでないなら、レコードがプログラムに指定される長さになるよう埋め込みまたは切り捨てが行われます。

順次マルチボリューム・ファイルで、READ ステートメントの処理中にボリュームの終わりが検出されたものの、論理的なファイルの終わりに達していない場合には、以下のアクションが示されている順番で行われます。

1. 標準的な終了ボリューム・ラベル・プロシージャーが処理される。
2. ボリュームの切り替えが行われる。
3. 標準的な開始ボリューム・ラベル・プロシージャーが実行される。
4. 次のボリュームの最初のデータ・レコードを使用可能にする。

ILE COBOL プログラムは、読み取り操作中に上記のアクションが取られたという標識を受け取りません。

ディスケット装置に保管されているレコードを順次ファイルに書き込むには、形式 1 の WRITE ステートメントを使用します。

出力ファイルにレコードを書き込む場合、レコード長を COBOL プログラム中に指定しなければなりません。プログラムに指定したレコード長がディスケットの様式のレコード長より長い場合、診断メッセージがプログラムに送られ、レコードは切り捨てられます。各交換タイプごとにディスケット装置でサポートされている最大レコード長は、次のとおりです。

交換タイプ	サポートされる最大レコード長
基本交換	128 バイト
H 交換	256 バイト
I 交換	4096 バイト

順次マルチボリューム・ファイルで、WRITE ステートメントの処理中にボリュームの終わりが検出された場合、以下に示すアクションがこの順に行われます。
1. 標準的な終了ボリューム・ラベル・プロシージャーが実行される。
2. ボリュームの切り替えが行われる。
3. 標準的な開始ボリューム・ラベル・プロシージャーが実行される。
4. 次のボリュームにデータ・レコードが書き込まれる。

ボリュームの終わり状態になっても、そのことを示す標識は COBOL プログラムには返されません。

ディスケット装置に保管されているファイルを使用し終えた時点で、それをクローズしなければなりません。ファイルをクローズするには、形式 1 の CLOSE ステートメントを使用します。 ファイルをクローズすると、そのファイルは、それを再びオープンするまで処理できなくなります。

```
CLOSE sequential-file-name.
```

ディスプレイ装置ファイルと ICF ファイルへのアクセス

ILE COBOL プログラムとワークステーションなどのディスプレイ装置との間で情報を交換するには、ディスプレイ・ファイルを使用します。ディスプレイ・ファイルは、ディスプレイ装置に表示する情報の形式、およびディスプレイ装置との間でシステムがその情報を処理する方法を定義するために使用されます。 ILE COBOL は、TRANSACTION ファイルを使用することによって、対話式にディスプレイ装置との通信を行います。

あるシステムのプログラムにおいて、それと同じシステムやリモート・システムのプログラムと通信するには、システム間通信機能 (ICF) ファイルを使用します。 ILE COBOL は、TRANSACTION ファイルを使用してシステム間通信を行います。

ディスプレイ装置の TRANSACTION ファイルを使用したり ICF ファイルを使用したりする方法については 579 ページの『第 21 章 トランザクション・ファイルの使用』を参照してください。

第 20 章 DISK ファイルおよび DATABASE ファイルの使用

DATABASE および DISK という ILE COBOL 装置と関連付けられるデータベース・ファイルには、次のものがあります。

- 外部記述ファイル。フィールドは、DDS によって IBM i に対して記述されます。
- プログラム記述ファイル。フィールドは、そのファイルを使用するプログラムの中で記述されます。

データベース・ファイルは、物理ファイル作成 (CRTPF) または論理ファイル作成 (CRTLF) CL コマンドを使用して作成します。これらのコマンドの説明については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「*プログラミング*」カテゴリーの中の『*CL および API*』セクションを参照してください。

この章では次のことについて説明します。

- DISK ファイルと DATABASE ファイルの違い
- DISK ファイルと DATABASE ファイルの編成方法
- DISK ファイルと DATABASE ファイルのさまざまな処理方法

DISK ファイルと DATABASE ファイルの違い

ILE COBOL プログラム中のファイルを物理データベース・ファイルや単一様式論理データベース・ファイルに関連付けるには、DISK という装置タイプを使用します。装置タイプとして DISK を選んだ場合、ILE COBOL データベース拡張機能は使えません。 DISK 装置タイプは、動的ファイル作成 (索引ファイルを除く) および可変長レコードをサポートします。

ILE COBOL プログラム中のファイルをデータベース・ファイルや DDM ファイルに関連付けるには、DATABASE という装置タイプを使用します。装置タイプとして DATABASE を選んだ場合、ILE COBOL データベース拡張機能を使うことが可能になります。それらのデータベース拡張機能には、次のものが含まれています。

- コミットメント制御
- 重複レコード・キー
- レコード様式
- 外部記述ファイル
- ヌル可能ファイル

しかし、DATABASE 装置タイプでは、動的ファイル作成や可変長レコードがサポートされていません。

ファイル編成と i5/OS ファイル・アクセス・パス

ファイルの中のレコードにアクセスするためのアクセス・パスには、次の 2 種類のものがあります。

- キー順アクセス・パス
- 到着順アクセス・パス

キー順アクセス・パスのファイルは、ILE COBOL では、SEQUENTIAL、RELATIVE、または INDEXED 編成のファイルとして処理できます。

ILE COBOL においてキー順ファイルを相対ファイルとして処理するためには、それが物理ファイルであるか、またはメンバーが 1 つの物理ファイル・メンバーに基づく論理ファイルでなければなりません。 ILE COBOL においてキー順ファイルを順次ファイルとして処理するためには、それが物理ファイルであるか、または 1 つの物理ファイル・メンバーに基づいている論理ファイルのうち、選択 / 除外論理を含まないものでなければなりません。

到着順アクセス・パスのファイルは、ILE COBOL において RELATIVE または SEQUENTIAL 編成のファイルとして処理できます。そのファイルは、物理ファイルであるか、またはファイルの各メンバーが 1 つの物理ファイル・メンバーのみに基づく論理ファイルでなければなりません。

論理ファイルに対して順次アクセスが指定される場合、そのファイルの中のレコードは、ファイルのデフォルト・アクセス・パスによってアクセスされます。

DISK ファイルおよび DATABASE ファイルのファイル処理方法

DISK ファイルおよび DATABASE ファイルでは、次のような編成が可能です。

- SEQUENTIAL
- RELATIVE
- INDEXED

ファイル編成の種類ごとに、それぞれ固有のファイル処理方法が使用されます。

順次ファイルの処理

ILE COBOL 順次ファイルは、レコードの処理が、ファイルに入れられた順序、すなわち到着順に行われるファイルのことです。たとえば、ファイルで 10 番目にあるレコードは、10 番目のレコード位置を占めており、10 番目に処理されます。ファイルを順次ファイルとして処理するには、SELECT 文節に ORGANIZATION IS SEQUENTIAL を指定するか、または ORGANIZATION 文節を省略する必要があります。順次ファイルに対して可能なのは順次アクセスだけです。

順次ファイルにアクセスする標準的な COBOL プログラムを作成するには、特定の特性のファイルを作成する必要があります。表 28 に、それらの特性とそれを制御する方法を示します。

表 28. 標準的な COBOL プログラムからアクセス可能な順次ファイルの特性

特性	制御
ファイルは物理ファイルでなければならない。	CRTPF コマンドを使用してファイルを作成する。

表 28. 標準的な COBOL プログラムからアクセス可能な順次ファイルの特性 (続き)

特性	制御
ファイルは共用ファイルにはできない。	CRTPF CL コマンドに SHARE(*NO) を指定する。
ファイルにキーを指定することはできない。	ファイルのデータ記述仕様 (DDS) の中に、位置 17 が K である行を含めない。
ファイルのファイル・タイプは DATA でなければならない。	CRTPF CL コマンドに FILETYPE(*DATA) を指定する。
フィールド編集は使えない。	ファイル DDS に EDTCDE キーワードおよび EDTWRD キーワードを指定しないようにする。
行および位置情報を指定できない。	ファイル DDS の中のすべてのフィールド記述の位置 39 ～ 44 をブランクのままにする。
行送りキーワードとスキップ・キーワードを指定できない。	ファイル DDS に SPACEA、SPACEB、SKIPA、または SKIPB のキーワードを指定しないようにする。
標識は使えない。	ファイル DDS の中のすべての行の位置 9 ～ 16 をブランクのままにする。
日付、時刻、およびページ番号などのシステム提供関数を使用できない。	ファイル DDS に DATE、TIME、または PAGNBR のキーワードを指定しないようにする。
ファイルに対して選択 / 除外レベル・キーワードを使用できない。	ファイル DDS の中に、位置 17 が S または O である行を含めないようにする。COMP、RANGE、VALUES、または ALL のキーワードは指定しないようにする。
ファイル中のレコードを再使用できない。	CRTPF CL コマンドに REUSEDLT(*NO) を指定する。
ファイル中のレコードに NULL フィールドを含めることができない。	ファイル DDS に ALWNULL キーワードを指定しないようにする。

順次ファイルに格納されているデータにアクセスするには、OPEN、READ、WRITE、REWRITE、および CLOSE ステートメントを使用します。それらのステートメントのそれぞれについては、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

SEQUENTIAL 編成の物理データベース・ファイルを OUTPUT 用にオープンすると、それはすべてクリアされます。

I-O (更新) モードでオープンするファイルの中のレコードの順序を保って、そのファイル中のレコードを再使用できるようにするには、ファイルを作成したり変更したりしないでください。すなわち、REUSEDLT オプションを指定した物理ファイルの変更 (CHGPF) CL コマンドを使用しないでください。

注: ILE COBOL コンパイラーは、外部ファイルと関連付けられた装置が、割り当て名の装置部分で指定されたタイプのものかどうかの検査は行いません。割り当て名で指定された装置は、そのファイルが割り当てられている実際の装置と一

致している必要があります。詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の『ASSIGN 文節』を参照してください。

相対ファイルの処理

ILE COBOL 相対ファイルは、相対レコード番号によって処理されるファイルです。相対レコード番号によってファイルを処理するには、そのファイルに対する SELECT ステートメントで ORGANIZATION IS RELATIVE を指定する必要があります。相対ファイルに対しては、順次アクセス、レコード番号によるランダム・アクセス、または動的アクセスが可能です。 ILE COBOL 相対ファイルでは、キー順アクセス・パスは不可能です。

相対ファイルにアクセスする標準的な COBOL プログラムを作成するには、特定の特性のファイルを作成する必要があります。 表 29 に、それらの特性とそれを制御する方法を示します。

表 29. 標準的な COBOL プログラムからアクセス可能な相対ファイルの特性

特性	**制御**
ファイルは物理ファイルでなければならない。[1]	CRTPF コマンドを使用してファイルを作成する。
ファイルは共用ファイルにはできない。	CRTPF CL コマンドに SHARE(*NO) を指定する。
ファイルにキーを指定することはできない。	ファイルのデータ記述仕様 (DDS) の中に、位置 17 が K である行を含めない。
レコード検索の開始位置を指定することはできない。	POSITION パラメーターを指定した OVRDBF CL コマンドを出さないようにする。
ファイルに対して選択 / 除外レベル・キーワードを使用できない。	ファイル DDS の中に、位置 17 が S または O である行を含めないようにする。 COMP、RANGE、VALUES、または ALL のキーワードは指定しないようにする。
ファイル中のレコードを再使用できない。	CRTPF CL コマンドに REUSEDLT(*NO) を指定する。
ファイル中のレコードに NULL フィールドを含めることができない。	ファイル DDS に ALWNULL キーワードを指定しないようにする。
注: [1] メンバーが 1 つの物理ファイルに基づいている論理ファイルは、ILE COBOL 相対ファイルとして使用することができます。	

相対ファイルに格納されているデータにアクセスするには、OPEN、READ、WRITE、START、REWRITE、DELETE、および CLOSE ステートメントを使用します。それらのステートメントのそれぞれについては、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。 START ステートメントは、INPUT または I-O 用にオープンされて、順次アクセスまたは動的アクセスされるファイルにのみ適用されます。

順次アクセスされる相対ファイルの場合、START ステートメント以外では SELECT 文節の KEY 句は無視されます。 START ステートメントに KEY 句が指定されていない場合、SELECT 文節の RELATIVE KEY 句が使用され、KEY IS EQUAL であると見なされます。

相対ファイルがランダム・アクセスまたは動的アクセスされる場合、SELECT 文節の RELATIVE KEY 句が使用されます。

NEXT 句を指定できるのは、SEQUENTIAL または DYNAMIC アクセス・モードが指定されているファイルに対する READ ステートメントだけです。 NEXT が指定されている場合、SELECT 文節の KEY 句は無視されます。 RELATIVE KEY データ項目は、READ 操作での順次アクセスが指定されているファイルの相対レコード番号によって更新されます。

OUTPUT 用にオープンされている物理データベース・ファイルは、すべてクリアされます。 RELATIVE 編成のデータベース・ファイルのうち、動的またはランダム・アクセス・モードを指定されているものは、削除されたレコードによっても初期設定されます。動的またはランダム・アクセス・モードでアクセスされる非常に大きい相対ファイル (1,000,000 レコード以上) では、削除されたレコードについてファイルが初期設定されるため、OPEN OUTPUT 処理に時間がかかるのが普通です。初期設定の必要なファイルをオープンするのに必要な時間は、そのファイルの中のレコード数によって異なります。

ファイルに対する最初の OPEN ステートメントが OPEN OUTPUT ではない場合、その相対ファイルを使う前に、ファイルをクリアし、削除されたレコードについて初期設定する必要があります。 INZPFM コマンドの RECORDS パラメーターは、*DLT に指定する必要があります。 ILE COBOL によってクリア操作と初期設定操作が処理される場合には指定変更が適用されますが、CL コマンドによって処理される場合は指定変更が適用されません。詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「*プログラミング*」カテゴリーの中の『*CL および API*』セクションの CLRPFM および INZPFM コマンドの説明を参照してください。

順次アクセス・モードで OUTPUT 用にオープンされる新しい相対ファイルの処理は、これとは異なります。表 30 に、関係する条件をまとめます。

表 30. 相対出力ファイルの初期設定

ファイル・アクセスと CL の指定	オープン時の条件	クローズ時の条件	ファイル境界
順次 *INZDLT		書き込まれないレコードが初期設定される。[1]	すべて増分。
順次 *INZDLT *NOMAX サイズ		CLOSE は成功。[1] ファイル状況は 0Q。[2]	書き込まれたレコードの境界まで。
順次 *NOINZDLT			書き込まれたレコードの境界まで。
ランダムまたは動的	レコードは初期設定される。 ファイルはオープン。		すべて増分。

表 30. 相対出力ファイルの初期設定 (続き)

ファイル・アクセスと CL の指定	オープン時の条件	クローズ時の条件	ファイル境界
ランダムまたは動的 *NOMAX サイズ	OPEN 失敗。 ファイル状況は 9Q。[3]		ファイルは空。

注:

1. CLOSE ステートメント実行時には、削除されたレコードについて初期設定すべきレコードが非常に多数 (1,000,000 以上) 残っている場合、時間がかかるのが普通です。
2. ファイル・サイズを超えないようにしつつ、現行のレコード数を超えてファイル境界を拡張するためには、ファイルの処理の前に、INZPFM コマンドを使用することによって、削除されたレコードを追加してください。 0Q というファイル状況を受け取ったときに、それでもファイルにレコードを追加したい場合には、このようにする必要があります。相対ファイルを現在のサイズ以上に拡張しようとするどちらの試みも、境界違反となります。
3. 9Q のファイル状況から回復するためには、関連する実行時メッセージ・テキストの説明に従って、CHGPF コマンドを使用してください。

RELATIVE 編成の ILE COBOL ファイルの場合、物理ファイル・メンバー再編成 (RGZPFM) CL コマンドによって、次のことができます。

- ファイルから、削除されたレコードをすべて除去する。 ILE COBOL は、すべての相対ファイル・レコードを削除レコードに関して初期設定するので、明示的に書き込まれていないレコードはファイルから除去されます。ファイルの最初の削除レコードより後にあるレコードの相対レコード番号は、すべて変わります。
- ファイルにキーがあり、キー・シーケンスに合わせて到着順が変更される場合、相対レコード番号を変える (KEYFILE パラメーター)。

さらに、REUSEDLT オプションを指定した物理ファイル変更 (CHGPF) CL コマンドでは、削除されたレコードの再使用が可能になるため、ファイルの順次操作においてレコードの検索や書き込みの順序を変えることができます。

索引付きファイルの処理

索引付きファイルは、デフォルト・アクセス・パスがキー値に基づいて構築されているファイルです。索引付きファイルのキー順アクセス・パスを作成する 1 つの方法は、DDS を使う方法です。

索引付きファイルは、SELECT ステートメントの ORGANIZATION IS INDEXED 文節によって識別されます。

索引付きファイル中のレコードは、キー・フィールドによって識別されます。ユーザーは、SELECT ステートメントの RECORD KEY 文節で、キー・フィールドを指定します。索引付きファイルのレコード記述内で、RECORD KEY データ項目が定義されている必要があります。ファイルに対して複数のレコード記述がある場合、その 1 つにだけ RECORD KEY データ名が含まれていれば十分です。しかし、そのファイルの他のレコード記述を参照した場合、RECORD KEY データ項目を含むレコード記述内の同じ位置が、他のレコード記述の中で KEY としてアクセスされます。

ALTERNATE RECORD KEY 文節で代替キーを指定することもできます。代替キーを使用すると、索引付きファイルにアクセスし、基本キーの順序とは違う順序でレコードを読み取ることができます。

索引付きファイルでは、順次アクセス、キーによるランダム・アクセス、または動的アクセスが可能です。

索引付きファイルにアクセスする標準的な COBOL プログラムを作成するには、特定の特性のファイルを作成する必要があります。表 31 に、それらの特性とそれを制御する方法を示します。

表 31. 標準的な COBOL プログラムからアクセス可能な索引付きファイルの特性

特性	**制御**
ファイルは物理ファイルでなければならない。	CRTPF コマンドを使用してファイルを作成する。
ファイルは共用ファイルにはできない。	CRTPF CL コマンドに SHARE(*NO) を指定する。
ファイルにはキーを定義しなければならない。	ファイルのデータ記述仕様 (DDS) に、位置 17 に K を使用して、少なくとも 1 つのキー・フィールドを定義する。
レコード内での複数のキーは連続していなければならない。	ファイル DDS に単一のキー・フィールドを指定するか、またはキー・フィールドをキーの重要度の降順で連続して指定する。
キー・フィールドは英数字でなければならない。 数値にすることはできない。	DDS キー・フィールドとして使用するフィールドを定義する際に、位置 35 に A または H を指定する。
順序付けに使うキーの値には、各バイトの 8 ビットすべてが含まれている必要がある。	英数字キー・フィールドを指定する。
ファイルにキー値の重複したレコードを含めることはできない。	ファイル DDS の中で UNIQUE キーワードを指定する。
キーは昇順でなければならない。	ファイル DDS に DESCEND キーワードを指定しないようにする。
レコード検索の開始位置を指定することはできない。	POSITION パラメーターを指定した OVRDBF CL コマンドを出さないようにする。
ファイルに対して選択 / 除外レベル・キーワードを使用できない。	ファイル DDS の中に、位置 17 が S または O である行を含めないようにする。COMP、RANGE、VALUES、または ALL のキーワードは指定しないようにする。
ファイル中のレコードに NULL フィールドを含めることができない。	ファイル DDS に ALWNULL キーワードを指定しないようにする。

索引付きファイルに保管されているデータにアクセスするには、OPEN、READ、WRITE、START、REWRITE、DELETE、および CLOSE ステートメントを使用します。それらのステートメントのそれぞれについては、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。索引付きファイルにアクセスする場合、DATABASE ファイルでは FORMAT 句の指定はオプションであり、DISK ファイルでは FORMAT 句を使用できません。 FORMAT 句が指定されてい

ない場合には、ファイルのデフォルトの様式名が使用されます。ファイルのデフォルトの様式名は、そのファイル中で定義されている最初の様式名です。特殊レジスター DB-FORMAT-NAME を使用することによって、成功した最後の入出力操作で使用された様式名を検索することができます。

索引付きファイルからレコードを順次に読み込む場合、ILE COBOL プログラムの中でそのファイルがどのように記述されているかに応じて、レコードは到着順またはキー順で戻されます。レコードを到着順に取り出すには、次の

```
ORGANIZATION IS SEQUENTIAL
ACCESS IS SEQUENTIAL
```

を、索引付きファイルに対する SELECT ステートメントで使用します。レコードをキー・シーケンス (通常は昇順) で取り出すには、次の

```
ORGANIZATION IS INDEXED
ACCESS IS SEQUENTIAL
```

を、索引付きファイルに対する SELECT ステートメントで指定してください。

順次アクセスされる索引付きファイルの場合、START ステートメント以外では SELECT 文節の KEY 句は無視されます。 START ステートメントに KEY 句が指定されていない場合、SELECT 文節の RECORD KEY 句が使用され、KEY IS EQUAL であると見なされます。

ランダム・アクセスまたは動的アクセスされる索引付きファイルの場合、START ステートメント以外では SELECT 文節の KEY 句が使用されます。 START ステートメントに KEY 句が指定されていない場合、SELECT 文節の RECORD KEY 句が使用され、KEY IS EQUAL であると見なされます。

DYNAMIC アクセスの DATABASE ファイルの READ ステートメントには、NEXT、PRIOR、FIRST、または LAST を指定することができます。 SEQUENTIAL アクセスの DATABASE ファイルの READ ステートメントには、NEXT も指定することができます。 NEXT、PRIOR、FIRST、または LAST が指定されている場合、SELECT 文節の KEY 句は無視されます。

INDEXED 編成の物理データベース・ファイルを OUTPUT 用にオープンすると、それはすべてクリアされます。

有効な RECORD KEY

ファイルの DDS は、キー・フィールドとして使われるフィールドを指定します。ファイルに複数のキー・フィールドがある場合、RECORD KEY IS EXTERNALLY-DESCRIBED-KEY が指定されているのでない限り、各レコードのキー・フィールドは連続している必要があります。

DDS の中でファイルに対して 1 つのキー・フィールドしか指定されていない場合、RECORD KEY は、DDS で定義されているキー・フィールドと同じ長さの単一フィールドでなければなりません。

形式 2 の COPY ステートメントがファイルに対して指定されている場合、RECORD KEY 文節は次のどれかを指定している必要があります。

- その名前が COBOL 予約語である場合、DDS で使用されている名前の末尾に -DDS を付加した名前。

- そのファイルのプログラム記述レコード記述で定義されているデータ名で、DDS のキー・フィールド定義で定義されているのと同じ長さと同じ位置のもの。
- EXTERNALLY-DESCRIBED-KEY。このキーワードは、DDS の中で各レコード様式ごとに定義されているキーをファイル・アクセスに使用することを指定します。これらのキーは不連続でもかまいません。 1 つのレコード様式内の別々の位置に定義してもかまいません。

DDS で複数の連続したキー・フィールドが指定されている場合、RECORD KEY データ名は、DDS 中の複数のキー・フィールドの長さの合計と等しい長さの単一のフィールドでなければなりません。ファイルに対して形式 2 の COPY ステートメントが指定されている場合、そのファイルについて、レコード中での適切な長さと位置によって RECORD KEY データ名を定義しているプログラム記述のレコード記述もなければなりません。

連続項目は、DATA DIVISION の中の連続した基本項目またはグループ項目で、単一のデータ階層に含まれているものです。

部分キーの参照

START ステートメントを使うと、部分キーを使用することができます。 KEY IS 句は必要です。

部分キーを参照する検索引き数の指定に関する規則については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の『START ステートメント』を参照してください。

538 ページの図 117 に、プログラム記述ファイルを使用する START ステートメントの例を示します。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL              CBLGUIDE/STRTPGMD       ISERIES1   06/02/15 14:41:49       ページ    2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID. STRTPGMD.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600   SOURCE-COMPUTER. IBM-ISERIES.
    6     000700 OBJECT-COMPUTER. IBM-ISERIES.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
    9     001000     SELECT FILE-1 ASSIGN TO DISK-NAMES                                          00/08/15
   11     001100     ACCESS IS DYNAMIC RECORD KEY IS FULL-NAME IN FILE-1
   13     001200     ORGANIZATION IS INDEXED.
          001300
   14     001400 DATA DIVISION.
   15     001500 FILE SECTION.
   16     001600 FD  FILE-1.
   17     001700 01  RECORD-DESCRIPTION.
   18     001800     03 FULL-NAME.
   19     001900        05 LAST-AND-FIRST-NAMES.
   20     002000           07 LAST-NAME             PIC X(20).
   21     002100           07 FIRST-NAME            PIC X(20).
   22     002200        05 MIDDLE-NAME              PIC X(20).
   23     002300     03 LAST-FIRST-MIDDLE-INITIAL-NAME REDEFINES FULL-NAME
          002400                                    PIC X(41).
   24     002500     03 REST-OF-RECORD             PIC X(50).
          002600
   25     002700 PROCEDURE DIVISION.
          002800 MAIN-PROGRAM SECTION.
          002900 MAINLINE.
   26     003000     OPEN INPUT FILE-1.
          003100*
          003200* POSITION THE FILE STARTING WITH RECORDS THAT HAVE A LAST NAME OF
          003300* "SMITH"
   27     003400     MOVE "SMITH" TO LAST-NAME.
   28     003500     START FILE-1 KEY IS EQUAL TO LAST-NAME
   29     003600           INVALID KEY DISPLAY "NO DATA IN SYSTEM FOR " LAST-NAME
   30     003700                       GO TO ERROR-ROUTINE
          003800     END-START.
          003900*          .
          004000*          .
          004100*          .
          004200*
          004300* POSITION THE FILE STARTING WITH RECORDS THAT HAVE A LAST NAME OF
          004400* "SMITH" AND A FIRST NAME OF "ROBERT"
   31     004500     MOVE "SMITH" TO LAST-NAME.
   32     004600     MOVE "ROBERT" TO FIRST-NAME.
   33     004700     START FILE-1 KEY IS EQUAL TO LAST-AND-FIRST-NAMES
   34     004800           INVALID KEY DISPLAY "NO DATA IN SYSTEM FOR "
          004900                               LAST-AND-FIRST-NAMES
   35     005000                       GO TO ERROR-ROUTINE
          005100     END-START.
          005200*          .
          005300*          .
```

図 117. プログラム記述ファイルを使用する START ステートメント (1/2)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/STRTPGMD       ISERIES1   06/02/15 14:41:49        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
         005400*          .
         005500*
         005600* POSITION THE FILE STARTING WITH RECORDS THAT HAVE A LAST NAME OF
         005700* "SMITH", A FIRST NAME OF "ROBERT", AND A MIDDLE INITIAL OF "M"
         005800
   36    005900     MOVE "SMITH" TO LAST-NAME.
   37    006000     MOVE "ROBERT" TO FIRST-NAME.
   38    006100     MOVE "M" TO MIDDLE-NAME.
   39    006200     START FILE-1 KEY IS EQUAL TO LAST-FIRST-MIDDLE-INITIAL-NAME
   40    006300           INVALID KEY DISPLAY "NO DATA IN SYSTEM FOR "
         006400                               LAST-FIRST-MIDDLE-INITIAL-NAME
   41    006500                       GO TO ERROR-ROUTINE
         006600     END-START.
         006700
         006800
         006900 ERROR-ROUTINE.
   42    007000     STOP RUN.
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 117. プログラム記述ファイルを使用する START ステートメント (2/2)

図 118 と 540 ページの図 119 に、外部記述ファイルを使用する START ステートメントの例を示します。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                      UNIQUE
     A          R RDE                       TEXT('RECORD DESCRIPTION')
     A            FNAME         20          TEXT('FIRST NAME')
     A            MINAME         1          TEXT('MIDDLE INITIAL NAME')
     A            MNAME         19          TEXT('REST OF MIDDLE NAME')
     A            LNAME         20          TEXT('LAST NAME')
     A            PHONE         10  0       TEXT('PHONE NUMBER')
     A            DATA          40          TEXT('REST OF DATA')
     A          K LNAME
     A          K FNAME
     A          K MINAME
     A          K MNAME
```

図 118. 外部記述ファイルを使用する START ステートメント -- DDS

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/STRTEXTD       ISERIES1   06/02/15 14:43:17        ページ     2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID. STRTEXTD.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600   SOURCE-COMPUTER. IBM-ISERIES
    6     000700   OBJECT-COMPUTER. IBM-ISERIES.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
    9     001000     SELECT FILE-1 ASSIGN TO DATABASE-NAMES
   11     001100     ACCESS IS DYNAMIC RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
   13     001200     ORGANIZATION IS INDEXED.
          001300
   14     001400 DATA DIVISION.
   15     001500 FILE SECTION.
   16     001600 FD  FILE-1.
   17     001700 01  RECORD-DESCRIPTION.
   17     001700 01  RECORD-DESCRIPTION.
          001800     COPY DDS-RDE OF NAMES.
         +000001*    I-O FORMAT:RDE        FROM FILE NAMES      OF LIBRARY CBLGUIDE           RDE
         +000003*THE KEY DEFINITIONS FOR RECORD FORMAT  RDE                                   RDE
         +000004*  NUMBER              NAME               RETRIEVAL       ALTSEQ            RDE
         +000005*   0001   LNAME                          ASCENDING         NO              RDE
         +000006*   0002   FNAME                          ASCENDING         NO              RDE
         +000007*   0003   MINAME                         ASCENDING         NO              RDE
         +000008*   0004   MNAME                          ASCENDING         NO              RDE
   18    +000009       05  RDE.                                                             RDE
   19    +000010           06 FNAME                 PIC X(20).                            RDE
         +000011*                FIRST NAME                                                RDE
   20    +000012           06 MINAME                PIC X(1).                             RDE
         +000013*                MIDDLE INITIAL NAME                                       RDE
   21    +000014           06 MNAME                 PIC X(19).                            RDE
         +000015*                REST OF MIDDLE NAME                                       RDE
   22    +000016           06 LNAME                 PIC X(20).                            RDE
         +000017*                LAST NAME                                                 RDE
   23    +000018           06 PHONE                 PIC S9(10)        COMP-3.             RDE
         +000019*                PHONE NUMBER                                              RDE
   24    +000020           06 DATA-DDS              PIC X(40).                            RDE
         +000021*                REST OF DATA                                              RDE
   25     001900 66  MIDDLE-NAME RENAMES MINAME THRU MNAME.
          002000
   26     002100 PROCEDURE DIVISION.
          002200 MAIN-PROGRAM SECTION.
          002300 MAINLINE.
   27     002400     OPEN INPUT FILE-1.
          002500*
          002600* POSITION THE FILE STARTING WITH RECORDS THAT HAVE A LAST NAME
          002700* OF "SMITH"
   28     002800     MOVE "SMITH" TO LNAME.
   29     002900     START FILE-1 KEY IS EQUAL TO LNAME
   30     003000           INVALID KEY DISPLAY "NO DATA IN SYSTEM FOR " LNAME
   31     003100                       GO TO ERROR-ROUTINE
          003200     END-START.
```

図 119. 外部記述ファイルを使用する START ステートメント (1/2)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/STRTEXTD       ISERIES1   06/02/15 14:43:17        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
         003300*          .
         003400*          .
         003500*          .
         003600*
         003700* POSITION THE FILE STARTING WITH RECORDS THAT HAVE A LAST NAME
         003800* OF "SMITH" AND A FIRST NAME OF "ROBERT"
   32    003900     MOVE "SMITH" TO LNAME.
   33    004000     MOVE "ROBERT" TO FNAME.
   34    004100     START FILE-1 KEY IS EQUAL TO LNAME, FNAME
   35    004200           INVALID KEY DISPLAY "NO DATA IN SYSTEM FOR "
         004300                               LNAME " " FNAME
   36    004400                       GO TO ERROR-ROUTINE
         004500     END-START.
         004600*    .
         004700*    .
         004800*          .
         004900*
         005000* POSITION THE FILE STARTING WITH RECORDS THAT HAVE A LAST NAME OF
         005100* "SMITH", A FIRST NAME OF "ROBERT", AND A MIDDLE INITIAL OF "M"
   37    005200     MOVE "SMITH" TO LNAME.
   38    005300     MOVE "ROBERT" TO FNAME.
   39    005400     MOVE "M" TO MINAME.
   40    005500     START FILE-1 KEY IS EQUAL TO LNAME, FNAME, MINAME
   41    005600           INVALID KEY DISPLAY "NO DATA IN SYSTEM FOR "
         005700                               LNAME SPACE FNAME SPACE MINAME
   42    005800                       GO TO ERROR-ROUTINE
         005900     END-START.
         006000
         006100
         006200 ERROR-ROUTINE.
   43    006300     STOP RUN.
                    * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 119. 外部記述ファイルを使用する START ステートメント (2/2)

代替レコード・キー

代替キーは、代替索引と関連しており、代替索引は一時的な場合と永続的な場合があります。

一時的な代替索引は、ファイルが開かれたときに ILE COBOL が作成するものです。ファイルが閉じられると、その一時的な索引は消滅します。デフォルトでは、ILE COBOL は一時的な索引を作成しません。一時的な代替索引を使用するには、CRTARKIDX オプションを指定する必要があります。

ただし、ILE COBOL は、永続的な索引を検出することができれば、一時的な索引を作成する代わりにその永続的な索引をまだ使用します。永続的代替索引とは、ILE COBOL プログラムが終了しても、消えずに残るもののことです。永続的な索引は論理ファイルと関連するので、 COBOL プログラムで永続的な索引を使用するには、その前にまず論理ファイルを作成する必要があります。

論理ファイルの DDS 指定は、キー・フィールド (複数可) を除いて、物理ファイルの指定と同じである必要があります。論理ファイルのキー・フィールドは、対応する、代替キーのデータ項目と一致するように定義する必要があります。 ILE COBOL プログラムは、このような論理ファイルは、一切参照しないことを注意してください。

永続的な索引を使用すると、一時的なものを使用した場合よりもパフォーマンスが向上します。レコード域内の代替キー・データ項目の長さと開始位置は、対応する DDS フィールドの長さと開始位置に一致しなければなりません。また DDS キー・フィールドは基本キーと関連しているため、この DDS フィールドは、キー参照さ

れるフィールドであってはなりません。代替キー・データ項目が複数の DDS フィールドにマップする場合、代替キー・データ項目の開始位置は最初の DDS フィールドと一致し、代替キー・データ項目の長さはこのキーを構成するすべての DDS フィールドの長さの合計と等しくなければなりません。

EXTERNALLY-DESCRIBED-KEY 文節は、代替キーも持つファイルに関しては指定できません。

ある特定の入出力要求に使用するキーは、参照キーと呼びます。参照キーは、READ または START ステートメントで変更することができます。

論理ファイルを索引付きファイルとして処理する

1 つの論理ファイルに対して複数のレコード様式があり、そのおのおのにキー・フィールドが関連付けられていて、それを ILE COBOL で索引付きファイルとして処理する場合には、次の制約事項および考慮事項が適用されます。

- レコード様式選択プログラムが存在していて、論理ファイル作成 (CRTLF) コマンド、論理ファイル変更 (CHGLF) コマンド、またはデータベース・ファイル指定変更 (OVRDBF) コマンドの FMTSLR パラメーターで指定されている場合以外は、そのファイルに対するすべての WRITE ステートメントで FORMAT 句を指定する必要があります。
- アクセス・モードが RANDOM または DYNAMIC であり、そのファイルに対して DUPLICATES 句が指定されていない場合、すべての DELETE および REWRITE ステートメントで FORMAT 句を指定する必要があります。
- FORMAT 句が指定されていないなら、RECORD KEY データ項目のうちそのファイルのすべてのレコード様式に共通している部分だけが、入出力ステートメントのキーとしてシステムによって使用されることになります。 FORMAT 句が指定されているなら、RECORD KEY データ項目のうち指定されたレコード様式に対して定義されている部分だけが、システムによってキーとして使用されることになります。
- ファイル中のいずれかの様式の最初のキー・フィールドに *NONE が指定されているなら、レコードは順次アクセスのみ可能です。ファイルがランダムに読み込まれる場合、次のようになります。
  - 様式名が指定されている場合、指定された様式の最初のレコードが戻されます。
  - 様式名が指定されていない場合、ファイル中の最初のレコードが戻されます。

  どちらの場合でも、RECORD KEY データ項目の値は無視されます。
- プログラム定義のキー・フィールドの場合、
  - 各レコード様式中のキー・フィールドは連続していなければなりません。
  - 各レコード様式の最初のキー・フィールドは、各レコード内の同じ相対位置から始まっていなければなりません。
  - RECORD KEY データ項目の長さは、そのファイル中のいずれかの様式の最長のキーの長さと同じでなければなりません。
- EXTERNALLY-DESCRIBED-KEY の場合、
  - 各レコード様式中のキー・フィールドは不連続でも可能です。

– 1 つのレコード様式内の別々の位置にキー・フィールドを定義することができます。

図 120 と図 121 に、索引付きファイルのアクセス・パスを DDS を使用して記述する方法の例を示します。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R FORMATA                   PFILE(ORDDTLP)
     A                                      TEXT('ACCESS PATH FOR INDEXED FILE')
     A            FLDA          14
     A            ORDERN         5S 0
     A            FLDB         101
     A          K ORDERN
```

図 120. データ記述仕様を用いた索引付きファイルのアクセス・パスの定義

データ記述仕様を使用することにより、プログラムによって記述された索引付きファイルのアクセス・パスを作成することができます。

図 120 に示されている DDS は、論理ファイル ORDDTLL の FORMATA レコード様式のためのものであり、ORDERN という 5 桁の長さのフィールドがキー・フィールドとして定義されています。 ORDERN をキー・フィールドとして定義したことによって、このファイルのキー順アクセス・パスが確立されます。その他の 2 つのフィールド FLDA と FLDB は、このレコード中の残りの位置が文字フィールドであることを記述しています。

プログラムによって記述された入力ファイル ORDDTLL は、SELECT 文節の FILE-CONTROL セクションで、索引付きファイルとして記述されています。

FD 項目の各フィールドの ILE COBOL 記述は、DDS ファイル中の対応する記述と合致していなければなりません。 RECORD KEY データ項目は、レコードの位置 15 から始まる 5 桁の整数として定義されている必要があります。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R FORMATA                   PFILE(ORDDTLP)
     A                                      TEXT('ACCESS PATH FOR INDEXED FILE')
     A            FLDA          14
     A            ORDERN         5S 0
     A            ITEM           5
     A            FLDB          96
     A          K ORDERN
     A          K ITEM
```

図 121. 索引付きファイルのアクセス・パス (複合キー) を定義するためのデータ記述仕様

この例では、図 121 に示されている DDS が、論理ファイル ORDDTLL のレコード様式 FORMAT に対して 2 つのキー・フィールドを定義しています。プログラムによって記述されている索引付きファイルにおいて 2 つのフィールドを 1 つの複合キーとして使うためには、キー・フィールドがレコード中で連続している必要があります。

各フィールドの ILE COBOL 記述は、DDS ファイル中の対応する記述と合致していなければなりません。ファイル制御項目の RECORD KEY 文節で、レコードの位置 15 から始まる 10 文字の項目が定義されている必要があります。 DDS フィー

ルド ORDERN および ITEM の ILE COBOL 記述は、RECORD KEY 文節で定義されている 10 文字の項目に従属するものとなります。

様式選択プログラムの使用法および論理ファイル処理の詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベース」カテゴリーの中の『*DB2 Universal Database for AS/400*』セクションを参照してください。

キーの降順でのファイル処理

DDS でキーの降順を指定して作成されているファイルでは、READ ステートメントの NEXT、PRIOR、FIRST、および LAST 句は、キーの昇順が指定されているファイルの場合とはちょうど逆に作用します。キーの降順は、DDS でキー・フィールドの横の位置 45〜80 に DESCEND キーワードを指定することによって指定することができます。**キーの降順**の場合、キー・フィールドの値が最も高いものから最も低いものへと、データが配列されます。

たとえば、READ FIRST ではキー値が最も高いレコードが検索され、READ LAST ではキー値が最も低いレコードが検索されます。 READ NEXT は、キー値がそれより低い次のレコードを検索します。キーの降順となっているファイルでは、START 修飾子も逆になります。たとえば、START GREATER THAN を実行すると、現行のレコード・ポインターは現行キーより低いキーのレコードに移されます。

可変長レコード・ファイルの処理

可変長レコードは、DISK 装置タイプに関連付けられているデータベース・ファイルに対してのみサポートされています。

可変長レコードの DISK ファイルを記述する

ファイルの最大レコード長と最小レコード長を定義するには、ファイルの FD 項目に形式 2 の RECORD 文節を指定します。

DATA DIVISION の中で可変長レコードの順次ファイルを記述する簡単なファイル記述項目は、以下のようなものになります。

```
FILE SECTION.
FD  sequential-file-name
    RECORD IS VARYING IN SIZE
              FROM integer-6 TO integer-7
              DEPENDING ON data-name-1.
01  minimum-sized-record.
    05  minimum-sized-element     PIC X(integer-6).
01  maximum-sized-record.
    05  maximum-sized-element     PIC X(integer-7).
⋮
WORKING-STORAGE SECTION.
77  data-name-1                   PIC 9(5).
⋮
```

ファイルの中のレコードの最小レコード・サイズは、*integer-6* によって定義されます。ファイルの中のレコードの最大レコード・サイズは、*integer-7* によって定義されます。 *integer-6* で指定されるものよりレコード長が短いファイルや *integer-7* で指定されるものよりレコード長が長いファイルのレコード記述は作成しないようにしてください。この規則を破るレコード記述があると、ILE COBOL コンパイラー

からコンパイル時エラー・メッセージが出されます。また、ILE COBOL コンパイラーは、レコード記述の中にレコード長が *integer-7* になるようなレコード記述がない場合にも、コンパイル時エラー・メッセージを出します。

可変長レコードを含む**索引付きファイル**の場合、基本レコード・キーはレコードの最初の 'n' 個の文字位置に含まれていなければなりません ('n' はそのファイルに指定されている最小レコード・サイズ)。 FD 項目の処理において ILE COBOL コンパイラーは、RECORD KEY がレコードの固定部分の中に入っているかどうかを検査します。この規則に違反するキーがあると、エラー・メッセージが出されます。

可変長レコードの DISK ファイルのオープン

可変長レコードのデータベース・ファイルの OPEN ステートメントが正常に実行されるためには、次の条件が満たされている必要があります。

- ファイルの中のオープンする様式には、1 個の可変長フィールドがその様式の最後に含まれていなければなりません。
- オープンされるすべての様式において、固定長フィールドの合計は同じでなければなりません。
- すべての様式について、最小レコード長は固定長フィールドの合計以上、ファイルの最大レコード長以下でなければなりません。
- ファイルをキー順処理用にオープンする場合、キーには可変長フィールドが含まれていてはなりません。

上記の条件のどれかが満たされていなければエラー・メッセージが生成され、ファイル状況 39 が戻され、オープン操作は失敗に終わります。

SHARE(*YES) が指定されているデータベース・ファイルに対してオープン操作を実行しようとした場合、そのファイルがすでにオープンされていても現行のオープン操作とは異なるレコード長のものであるなら、エラー・メッセージが生成され、ファイル状況 90 が戻されます。

可変長レコードの DISK ファイルの読み書き

可変長レコードに対して READ、WRITE、または REWRITE ステートメントが実行される場合、そのレコードのサイズは *data-name-1* の内容で定義されます。

可変長レコードの処理方法の詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の形式 2 の RECORD 文節の部分を参照してください。

データベース・ファイルから可変長レコードを読み込むには、READ ステートメントを使います。 READ 操作が正常に実行された場合、*data-name-1* が指定されているなら、読み取ったレコードの文字位置数がそれに入れられます。 READ 操作が失敗した場合、*data-name-1* は READ 操作が試行される前の値のままです。

READ ステートメントに INTO 句を指定した場合、現行レコードの中で、暗黙の MOVE ステートメントの中の送り出し項目として加わっている文字位置数は、以下のようにして決められます。

- *data-name-1* が指定されている場合は、*data-name-1* の内容。または、
- *data-name-1* が指定されていない場合は、読み取ったレコードの中の文字位置数。

READ ステートメント実行時に、読み込んだレコードの中の文字位置の数がそのファイルのレコード記述項目で指定されている最小レコード長より小さい場合は、レコード域のうち読み込まれた有効な最後の文字の右の部分にブランクが埋め込まれます。読み込んだレコードの中の文字位置の数がそのファイルのレコード記述項目で指定されている最大レコード長より大きい場合は、レコードは、レコード記述項目で指定されている最大レコード・サイズの右側で切り捨てられます。読み込んだレコードの長さが、ファイルのファイル記述項目の RECORD 文節で定義されている最小レコード長と最大レコード長の間にない場合には、ファイル状況 04 が戻されます。

データベース・ファイルに可変長レコードを書き込むには、WRITE ステートメントまたは REWRITE ステートメントを使います。書き込むレコードの長さは、*data-name-1* に指定します。 *data-name-1* を指定しない場合、書き込むレコードの長さは以下のようにして決められます。

- レコードに OCCURS...DEPENDING ON 項目が含まれている場合には、固定部分と、WRITE ステートメント実行時のオカレンスの数によって記述されるテーブル部分との合計。
- レコードに OCCURS...DEPENDING ON 項目が含まれていない場合には、レコード定義の中での文字位置数。

DISK ファイルおよび DATABASE ファイルの処理の例

以下のサンプル・プログラムは、i5/OS ファイル編成の各タイプごとに対応する基本的なプログラミング・テクニックを示したものです。これらの例はもっぱら学習を目的としたものであり、特定のアクセス方式に必要な入出力ステートメントについて示すものです。その他の ILE COBOL 機能 (たとえば PERFORM ステートメントの使用) は、たまたま使用されているにすぎません。 このプログラムには、以下のことが示されています。

- 順次ファイルの作成
- 順次ファイルの更新および拡張
- 相対ファイルの作成
- 相対ファイルの更新
- 相対ファイルの検索
- 索引付きファイルの作成
- 索引付きファイルの更新

順次ファイルの作成

このプログラムでは、従業員給与レコードからなる順次ファイルを作成します。入力レコードは、従業員番号の昇順に並べます。出力ファイルも同じ順序です。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/CRTSEQ         ISERIES1   06/02/15 14:46:40        ページ    2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.  CRTSEQ.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600   SOURCE-COMPUTER. IBM-ISERIES
    6     000700   OBJECT-COMPUTER. IBM-ISERIES.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
    9     001000     SELECT INPUT-FILE ASSIGN TO DISK-FILEA
   11     001100         FILE STATUS IS INPUT-FILE-STATUS.
   12     001200     SELECT OUTPUT-FILE ASSIGN TO DISK-FILEB
   14     001300         FILE STATUS IS OUTPUT-FILE-STATUS.
   15     001400 DATA DIVISION.
   16     001500 FILE SECTION.
   17     001600 FD  INPUT-FILE.
   18     001700 01  INPUT-RECORD.
   19     001800     05  INPUT-EMPLOYEE-NUMBER      PICTURE 9(6).
   20     001900     05  INPUT-EMPLOYEE-NAME        PICTURE X(28).
   21     002000     05  INPUT-EMPLOYEE-CODE        PICTURE 9.
   22     002100     05  INPUT-EMPLOYEE-SALARY      PICTURE 9(6)V99.
   23     002200 FD  OUTPUT-FILE.
   24     002300 01  OUTPUT-RECORD.
   25     002400     05  OUTPUT-EMPLOYEE-NUMBER     PICTURE 9(6).
   26     002500     05  OUTPUT-EMPLOYEE-NAME       PICTURE X(28).
   27     002600     05  OUTPUT-EMPLOYEE-CODE       PICTURE 9.
   28     002700     05  OUTPUT-EMPLOYEE-SALARY     PICTURE 9(6)V99.
          002800
   29     002900 WORKING-STORAGE SECTION.
   30     003000 77  INPUT-FILE-STATUS              PICTURE XX.
   31     003100 77  OUTPUT-FILE-STATUS             PICTURE XX.
   32     003200 77  OP-NAME                        PICTURE X(7).
   33     003300 01  INPUT-END                      PICTURE X VALUE SPACE.
   34     003400     88 THE-END-OF-INPUT            VALUE "E".
          003500
   35     003600 PROCEDURE DIVISION.
   36     003700 DECLARATIVES.
          003800 INPUT-ERROR SECTION.
          003900      USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
          004000 INPUT-ERROR-PARA.
   37     004100      DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, "FOR INPUT-FILE".
   38     004200      DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
   39     004300      DISPLAY "PROCESSING ENDED".
   40     004400      STOP RUN.
          004500
          004600 OUTPUT-ERROR SECTION.
          004700      USE AFTER STANDARD ERROR PROCEDURE ON OUTPUT-FILE.
          004800 OUTPUT-ERROR-PARA.
   41     004900      DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, "FOR OUTPUT-FILE".
   42     005000      DISPLAY "FILE STATUS IS ", OUTPUT-FILE-STATUS.
   43     005100      DISPLAY "PROCESSING ENDED".
   44     005200      STOP RUN.
          005300 END DECLARATIVES.
```

図 122. 従業員給与レコードの順次ファイルの例 (1/2)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/CRTSEQ          ISERIES1  06/02/15 14:46:40        ページ    3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
          005400
          005500 MAIN-PROGRAM SECTION.
          005600 MAINLINE.
   45     005700     MOVE "OPEN" TO OP-NAME.
   46     005800     OPEN INPUT INPUT-FILE
          005900          OUTPUT OUTPUT-FILE.
          006000
   47     006100     MOVE "READ" TO OP-NAME.
   48     006200     READ INPUT-FILE INTO OUTPUT-RECORD
   49     006300         AT END SET THE-END-OF-INPUT TO TRUE
          006400     END-READ.
          006500
   50     006600     PERFORM UNTIL THE-END-OF-INPUT
   51     006700         MOVE "WRITE" TO OP-NAME
   52     006800         WRITE OUTPUT-RECORD
   53     006900         MOVE "READ" TO OP-NAME
   54     007000         READ INPUT-FILE INTO OUTPUT-RECORD
   55     007100             AT END SET THE-END-OF-INPUT TO TRUE
          007200                 END-READ
          007300     END-PERFORM.
          007400
   56     007500     MOVE "CLOSE" TO OP-NAME.
   57     007600     CLOSE INPUT-FILE
          007700           OUTPUT-FILE.
   58     007800     STOP RUN.
                         * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 122. 従業員給与レコードの順次ファイルの例 (2/2)

順次ファイルの更新および拡張

このプログラムでは、CRTSEQ プログラムによって作成されたファイルを更新および拡張します。 INPUT-FILE と MASTER-FILE をそれぞれ読み込みます。
INPUT-EMPLOYEE-NUMBER と MST-EMPLOYEE-NUMBER が一致していれば、元のレコードが入力レコードによって置き換えられます。 MASTER-FILE を処理した後、新しい従業員レコードをファイルの末尾に追加します。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/UPDTSEQ        ISERIES1   06/02/15 14:48:09        ページ    2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.  UPDTSEQ.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600   SOURCE-COMPUTER. IBM-ISERIES
    6     000700   OBJECT-COMPUTER. IBM-ISERIES.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
    9     001000     SELECT INPUT-FILE ASSIGN TO DISK-FILES
   11     001100         FILE STATUS IS INPUT-FILE-STATUS.
   12     001200     SELECT MASTER-FILE ASSIGN TO DISK-MSTFILEB
   14     001300         FILE STATUS IS MASTER-FILE-STATUS.
          001400
   15     001500 DATA DIVISION.
   16     001600 FILE SECTION.
   17     001700 FD  INPUT-FILE.
   18     001800 01  INPUT-RECORD.
   19     001900     05  INPUT-EMPLOYEE-NUMBER      PICTURE 9(6).
   20     002000     05  INPUT-EMPLOYEE-NAME        PICTURE X(28).
   21     002100     05  INPUT-EMPLOYEE-CODE        PICTURE 9.
   22     002200     05  INPUT-EMPLOYEE-SALARY      PICTURE 9(6)V99.
   23     002300 FD  MASTER-FILE.
   24     002400 01  MASTER-RECORD.
   25     002500     05 MST-EMPLOYEE-NUMBER         PICTURE 9(6).
   26     002600     05 MST-EMPLOYEE-NAME           PICTURE X(28).
   27     002700     05 MST-EMPLOYEE-CODE           PICTURE 9.
   28     002800     05 MST-EMPLOYEE-SALARY         PICTURE 9(6)V99.
   29     002900 WORKING-STORAGE SECTION.
   30     003000 77  INPUT-FILE-STATUS             PICTURE XX.
   31     003100 77  MASTER-FILE-STATUS            PICTURE XX.
   32     003200 77  OP-NAME                       PICTURE X(12).
   33     003300 01  INPUT-END                     PICTURE X VALUE SPACE.
   34     003400     88 THE-END-OF-INPUT           VALUE "E".
   35     003500 01  MASTER-END                    PICTURE X VALUE SPACE.
   36     003600     88 THE-END-OF-MASTER          VALUE "E".
   37     003700 PROCEDURE DIVISION.
   38     003800 DECLARATIVES.
          003900 INPUT-ERROR SECTION.
          004000      USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
          004100 INPUT-ERROR-PARA.
   39     004200      DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, "FOR INPUT-FILE".
   40     004300      DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
   41     004400      DISPLAY "PROCESSING ENDED".
   42     004500      STOP RUN.
          004600
          004700 I-O-ERROR SECTION.
          004800      USE AFTER STANDARD ERROR PROCEDURE ON MASTER-FILE.
          004900 I-O-ERROR-PARA.
   43     005000      DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, "FOR MASTER-FILE".
   44     005100      DISPLAY "FILE STATUS IS ", MASTER-FILE-STATUS.
   45     005200      DISPLAY "PROCESSING ENDED".
   46     005300      STOP RUN.
```

図 123. 順次ファイル更新プログラムの例 (1/2)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/UPDTSEQ        ISERIES1   06/02/15 14:48:09        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
         005400 END DECLARATIVES.
         005500
         005600 MAIN-PROGRAM SECTION.
         005700 MAINLINE.
   47    005800        MOVE "OPEN" TO OP-NAME.
   48    005900        OPEN INPUT INPUT-FILE
         006000             I-O   MASTER-FILE.
         006100
   49    006200       PERFORM READ-INPUT-FILE.
   50    006300       PERFORM READ-MASTER-FILE.
   51    006400       PERFORM PROCESS-FILES UNTIL THE-END-OF-INPUT.
         006500
   52    006600       MOVE "CLOSE" TO OP-NAME.
   53    006700       CLOSE MASTER-FILE
         006800             INPUT-FILE.
   54    006900       STOP RUN.
         007000
         007100 READ-INPUT-FILE.
   55    007200       MOVE "READ" TO OP-NAME.
   56    007300       READ INPUT-FILE
   57    007400           AT END SET THE-END-OF-INPUT TO TRUE
         007500       END-READ.
         007600
         007700 READ-MASTER-FILE.
   58    007800       MOVE "READ" TO OP-NAME.
   59    007900       READ MASTER-FILE
         008000           AT END
   60    008100              SET THE-END-OF-MASTER TO TRUE
   61    008200              MOVE "AT END CLOSE" TO OP-NAME
   62    008300              CLOSE MASTER-FILE
   63    008400              MOVE "OPEN EXTEND" TO OP-NAME
   64    008500              OPEN EXTEND MASTER-FILE
         008600       END-READ.
         008700
         008800 PROCESS-FILES.
   65    008900     IF THE-END-OF-MASTER THEN
   66    009000         WRITE MASTER-RECORD FROM INPUT-RECORD
   67    009100         PERFORM READ-INPUT-FILE
         009200     ELSE
   68    009300         IF MST-EMPLOYEE-NUMBER
         009400            LESS THAN INPUT-EMPLOYEE-NUMBER THEN
   69    009500              PERFORM READ-MASTER-FILE
         009600        ELSE
   70    009700           IF MST-EMPLOYEE-NUMBER EQUAL INPUT-EMPLOYEE-NUMBER THEN
   71    009800              MOVE "REWRITE" TO OP-NAME
   72    009900              REWRITE MASTER-RECORD FROM INPUT-RECORD
   73    010000              PERFORM READ-INPUT-FILE
   74    010100              PERFORM READ-MASTER-FILE
         010200           ELSE
   75    010300               DISPLAY "ERROR RECORD -> ", INPUT-EMPLOYEE-NUMBER
   76    010400               PERFORM READ-INPUT-FILE
         010500           END-IF
         010600        END-IF
         010700     END-IF.
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 123. 順次ファイル更新プログラムの例 (2/2)

相対ファイルの作成

このプログラムでは、順次アクセスを使用して販売実績レコードの相対ファイルを作成します。各レコードには、ある年の週ごとの販売数量と販売高の実績が、5 年間分含まれています。ファイルには 52 個のレコードが入っており、各レコードはそれぞれ 1 週間を表しています。

各入力レコードは、ある年の週ごとの販売実績を表しています。最近 5 年間 (昇順) の各年の最初の週のレコードが、最初の 5 つの入力レコードとなり、最近 5 年間の第 2 週のレコードが、次の 5 つの入力レコードとなり、.... というようになります。このように、5 つの入力レコードによって 1 つの出力レコードが作り出されます。

RELATIVE-FILE の RELATIVE KEY は指定されていません。 START ステートメントが使用されていない限り、順次アクセスでは必要ないからです。 (しかし更新において、キーは INPUT-WEEK です。)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/UPDTSEQ         ISERIES1   06/02/15 14:48:09        ページ    2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.   CRTREL.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600   SOURCE-COMPUTER. IBM-ISERIES
    6     000700   OBJECT-COMPUTER. IBM-ISERIES.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
     9     001000     SELECT RELATIVE-FILE ASSIGN TO DISK-FILED
    11     001100         ORGANIZATION IS RELATIVE
   12     001200          ACCESS IS SEQUENTIAL
   13     001300         FILE STATUS RELATIVE-FILE-STATUS.
   14     001400     SELECT INPUT-FILE ASSIGN TO DISK-FILEC
   16     001500         ORGANIZATION IS SEQUENTIAL
   17     001600         ACCESS IS SEQUENTIAL
   18     001700         FILE STATUS INPUT-FILE-STATUS.
          001800
   19     001900 DATA DIVISION.
   20     002000 FILE SECTION.
   21     002100 FD  RELATIVE-FILE.
   22     002200 01  RELATIVE-RECORD-01.
   23     002300     05  RELATIVE-RECORD OCCURS 5 TIMES INDEXED BY REL-INDEX.
   24     002400         10  RELATIVE-YEAR          PICTURE 99.
   25     002500         10  RELATIVE-WEEK          PICTURE 99.
   26     002600         10  RELATIVE-UNIT-SALES    PICTURE S9(6).
   27     002700         10  RELATIVE-DOLLAR-SALES  PICTURE S9(9)V99.
   28     002800 FD  INPUT-FILE.
   29     002900 01  INPUT-RECORD.
   30     003000     05  INPUT-YEAR                PICTURE 99.
   31     003100     05  INPUT-WEEK                PICTURE 99.
   32     003200     05  INPUT-UNIT-SALES          PICTURE S9(6).
   33     003300     05  INPUT-DOLLAR-SALES        PICTURE S9(9)V99.
          003400
   34     003500 WORKING-STORAGE SECTION.
   35     003600 77  RELATIVE-FILE-STATUS          PICTURE XX.
   36     003700 77  INPUT-FILE-STATUS             PICTURE XX.
   37     003800 77  OP-NAME                       PICTURE X(5).
   38     003900 01  WORK-RECORD.
   39     004000     05  WORK-YEAR                 PICTURE 99 VALUE 00.
   40     004100     05  WORK-WEEK                 PICTURE 99.
   41     004200     05  WORK-UNIT-SALES           PICTURE S9(6).
   42     004300     05  WORK-DOLLAR-SALES         PICTURE S9(9)V99.
   43     004400 01  INPUT-END                     PICTURE X VALUE SPACE.
   44     004500     88  THE-END-OF-INPUT          VALUE "E".
          004600
   45     004700 PROCEDURE DIVISION.
   46     004800 DECLARATIVES.
          004900 INPUT-ERROR SECTION.
          005000       USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
          005100 INPUT-ERROR-PARA.
   47     005200     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INPUT-FILE ".
   48     005300     DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
```

図 124. 相対ファイル・プログラムの例 (1/2)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/CRTREL          ISERIES1  06/02/15 14:49:23        ページ    3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   49     005400     DISPLAY "PROCESSING ENDED"
   50     005500     STOP RUN.
          005600
          005700 OUTPUT-ERROR SECTION.
          005800       USE AFTER STANDARD ERROR PROCEDURE ON RELATIVE-FILE.
          005900 OUTPUT-ERROR-PARA.
   51     006000     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR RELATIVE-FILE".
   52     006100     DISPLAY "FILE STATUS IS ", RELATIVE-FILE-STATUS.
   53     006200     DISPLAY "PROCESSING ENDED"
   54     006300     STOP RUN.
          006400 END DECLARATIVES.
          006500
          006600 MAIN-PROGRAM SECTION.
          006700 MAINLINE.
   55     006800     MOVE "OPEN" TO OP-NAME.
   56     006900     OPEN INPUT INPUT-FILE
          007000          OUTPUT RELATIVE-FILE.
          007100
   57     007200     SET REL-INDEX TO 1.
   58     007300     MOVE "READ" TO OP-NAME.
   59     007400     READ INPUT-FILE
   60     007500         AT END SET THE-END-OF-INPUT TO TRUE
          007600     END-READ.
          007700
   61     007800     PERFORM UNTIL THE-END-OF-INPUT
   62     007900         MOVE INPUT-RECORD TO RELATIVE-RECORD (REL-INDEX)
   63     008000         IF REL-INDEX NOT = 5
   64     008100             SET REL-INDEX UP BY 1
          008200         ELSE
   65     008300             SET REL-INDEX TO 1
   66     008400             MOVE "WRITE" TO OP-NAME
   67     008500             WRITE RELATIVE-RECORD-01
          008600         END-IF
          008700
   68     008800         MOVE "READ" TO OP-NAME
   69     008900         READ INPUT-FILE
   70     009000             AT END SET THE-END-OF-INPUT TO TRUE
          009100        END-READ
          009200     END-PERFORM.
          009300
   71     009400     CLOSE RELATIVE-FILE
          009500           INPUT-FILE.
   72     009600     STOP RUN.
          009700
                         * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 124. 相対ファイル・プログラムの例 (2/2)

相対ファイルの更新

このプログラムでは、CRTREL プログラムで作成された販売実績レコードのファイルを、順次アクセスを使って更新します。更新プログラムは、新しい年のレコードを追加し、最も古い年のレコードを RELATIVE-FILE から削除します。

入力レコードは、前年の週ごとの販売実績レコードを表しています。
RELATIVE-FILE の RELATIVE KEY は、入力レコードの中に INPUT-WEEK として含まれています。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/UPDTREL        ISERIES1   06/02/15 14:50:35        ページ    2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.  UPDTREL.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600   SOURCE-COMPUTER. IBM-ISERIES
    6     000700   OBJECT-COMPUTER. IBM-ISERIES.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
     9     001000     SELECT RELATIVE-FILE ASSIGN TO DISK-FILED
    11     001100         ORGANIZATION IS RELATIVE
   12     001200          ACCESS IS SEQUENTIAL
    13     001300         RELATIVE KEY INPUT-WEEK
   14     001400         FILE STATUS RELATIVE-FILE-STATUS.
   15     001500     SELECT INPUT-FILE ASSIGN TO DISK-FILES2
   17     001600         ORGANIZATION IS SEQUENTIAL
   18     001700         ACCESS IS SEQUENTIAL
   19     001800         FILE STATUS INPUT-FILE-STATUS.
          001900
   20     002000 DATA DIVISION.
   21     002100 FILE SECTION.
   22     002200 FD  RELATIVE-FILE.
   23     002300 01  RELATIVE-RECORD               PICTURE X(105).
   24     002400 FD  INPUT-FILE.
   25     002500 01  INPUT-RECORD.
   26     002600     05  INPUT-YEAR                PICTURE 99.
   27     002700     05  INPUT-WEEK                PICTURE 99.
   28     002800     05  INPUT-UNIT-SALES          PICTURE S9(6).
   29     002900     05  INPUT-DOLLAR-SALES        PICTURE S9(9)V99.
          003000
   30     003100 WORKING-STORAGE SECTION.
   31     003200 77  RELATIVE-FILE-STATUS          PICTURE XX.
   32     003300 77  INPUT-FILE-STATUS             PICTURE XX.
   33     003400 77  OP-NAME                       PICTURE X(7).
   34     003500 01  WORK-RECORD.
   35     003600     05  FILLER                    PICTURE X(21).
   36     003700     05  CURRENT-WORK-YEARS        PICTURE X(84).
   37     003800     05  NEW-WORK-YEAR.
   38     003900     10  WORK-YEAR                 PICTURE 99.
   39     004000     10  WORK-WEEK                 PICTURE 99.
   40     004100     10  WORK-UNIT-SALES           PICTURE S9(6).
   41     004200     10  WORK-DOLLAR-SALES         PICTURE S9(9)V99.
   42     004300 66  WORK-OUT-RECORD RENAMES
          004400     CURRENT-WORK-YEARS THROUGH NEW-WORK-YEAR.
   43     004500 01  INPUT-END                     PICTURE X VALUE SPACE.
   44     004600     88  THE-END-OF-INPUT          VALUE "E".
          004700
   45     004800 PROCEDURE DIVISION.
   46     004900 DECLARATIVES.
          005000 INPUT-ERROR SECTION.
          005100       USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
          005200 INPUT-ERROR-PARA.
   47     005300     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INPUT-FILE ".
```

図 125. 相対ファイル更新プログラムの例 (1/2)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/UPDTREL        ISERIES1   06/02/15 14:50:35        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   48     005400     DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
   49     005500     DISPLAY "PROCESSING ENDED"
   50     005600     STOP RUN.
          005700
          005800 I-O-ERROR SECTION.
          005900       USE AFTER STANDARD ERROR PROCEDURE ON RELATIVE-FILE.
          006000 I-O-ERROR-PARA.
   51     006100     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR RELATIVE-FILE".
   52     006200     DISPLAY "FILE STATUS IS ", RELATIVE-FILE-STATUS.
   53     006300     DISPLAY "PROCESSING ENDED"
   54     006400     STOP RUN.
          006500 END DECLARATIVES.
          006600
          006700 MAIN-PROGRAM SECTION.
          006800 MAINLINE.
   55     006900     MOVE "OPEN" TO OP-NAME.
   56     007000     OPEN INPUT INPUT-FILE
          007100          I-O RELATIVE-FILE.
          007200
   57     007300     MOVE "READ" TO OP-NAME.
   58     007400     READ RELATIVE-FILE INTO WORK-RECORD
   59     007500         AT END SET THE-END-OF-INPUT TO TRUE
          007600     END-READ.
   60     007700     READ INPUT-FILE INTO NEW-WORK-YEAR
   61     007800              AT END SET THE-END-OF-INPUT TO TRUE
          007900     END-READ.
          008000
   62     008100     PERFORM UNTIL THE-END-OF-INPUT
   63     008200        MOVE "REWRITE" TO OP-NAME
   64     008300        REWRITE RELATIVE-RECORD FROM WORK-OUT-RECORD
          008400
   65     008500        MOVE "READ" TO OP-NAME
   66     008600        READ RELATIVE-FILE INTO WORK-RECORD
   67     008700            AT END SET THE-END-OF-INPUT TO TRUE
          008800     END-READ
   68     008900        READ INPUT-FILE INTO NEW-WORK-YEAR
   69     009000            AT END SET THE-END-OF-INPUT TO TRUE
          009100        END-READ
          009200     END-PERFORM.
          009300
   70     009400     MOVE "CLOSE" TO OP-NAME.
   71     009500     CLOSE INPUT-FILE
          009600           RELATIVE-FILE.
   72     009700     STOP RUN.
          009800
                         * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 125. 相対ファイル更新プログラムの例 (2/2)

相対ファイルの検索

このプログラムは、CRTREL プログラムで作成された販売実績ファイルを、動的アクセスを使って検索します。

INPUT-FILE のレコードには 1 つの必要フィールド (INPUT-WEEK) が含まれており、これが RELATIVE-FILE の RELATIVE KEY です。また 1 つのオプショナル・フィールド (END-WEEK) も含まれています。 INPUT-WEEK にデータが含まれていて、END-WEEK にスペースが含まれている入力レコードは、その 1 つの特定の RELATIVE-RECORD の印刷出力を要求します。そのレコードはランダム・アクセスによって検索されます。 INPUT-WEEK と END-WEEK の両方にデータが含まれている入力レコードは、RELATIVE KEY が INPUT-WEEK から END-WEEK までの範囲にある RELATIVE-FILE レコードのすべての印刷出力を要求します。それらのレコードは、順次アクセスによって取り出されます。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/RTRVREL        ISERIES1   06/02/15 14:51:40        ページ     2
                           ソ ー ス
STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   1     000100 IDENTIFICATION DIVISION.
   2     000200 PROGRAM-ID.  RTRVREL.
         000300
   3     000400 ENVIRONMENT DIVISION.
   4     000500 CONFIGURATION SECTION.
   5     000600   SOURCE-COMPUTER. IBM-ISERIES
   6     000700   OBJECT-COMPUTER. IBM-ISERIES.
   7     000800 INPUT-OUTPUT SECTION.
   8     000900 FILE-CONTROL.
   9     001000     SELECT RELATIVE-FILE ASSIGN TO DISK-FILED
  11     001100         ORGANIZATION IS RELATIVE
  12     001200         ACCESS IS DYNAMIC
  13     001300         RELATIVE KEY INPUT-WEEK
  14     001400         FILE STATUS IS RELATIVE-FILE-STATUS.
  15     001500     SELECT INPUT-FILE ASSIGN TO DISK-FILEF
  17     001600         FILE STATUS IS INPUT-FILE-STATUS.
  18     001700     SELECT PRINT-FILE ASSIGN TO PRINTER-QSYSPRT
  20     001800         FILE STATUS IS PRINT-FILE-STATUS.
         001900
  21     002000 DATA DIVISION.
  22     002100 FILE SECTION.
  23     002200 FD  RELATIVE-FILE.
  24     002300 01  RELATIVE-RECORD-01.
  25     002400     05  RELATIVE-RECORD OCCURS 5 TIMES INDEXED BY REL-INDEX.
  26     002500         10  RELATIVE-YEAR         PICTURE 99.
  27     002600         10  RELATIVE-WEEK         PICTURE 99.
  28     002700         10  RELATIVE-UNIT-SALES   PICTURE S9(6).
  29     002800         10  RELATIVE-DOLLAR-SALES PICTURE S9(9)V99.
  30     002900 FD  INPUT-FILE.
  31     003000 01  INPUT-RECORD.
  32     003100     05 INPUT-WEEK                 PICTURE 99.
  33     003200     05  END-WEEK                  PICTURE 99.
  34     003300 FD  PRINT-FILE.
  35     003400 01  PRINT-RECORD.
  36     003500     05 PRINT-WEEK                 PICTURE 99.
  37     003600     05 FILLER                     PICTURE X(5).
  38     003700     05 PRINT-YEAR                 PICTURE 99.
  39     003800     05 FILLER                     PICTURE X(5).
  40     003900     05 PRINT-UNIT-SALES           PICTURE ZZZ,ZZ9.
  41     004000     05 FILLER                     PICTURE X(5).
  42     004100     05 PRINT-DOLLAR-SALES         PICTURE $$$$,$$$,$$$.99.
         004200
  43     004300 WORKING-STORAGE SECTION.
  44     004400 77  RELATIVE-FILE-STATUS         PICTURE XX.
  45     004500 77  INPUT-FILE-STATUS            PICTURE XX.
  46     004600 77  PRINT-FILE-STATUS            PICTURE XX.
  47     004700 77  HIGH-WEEK                    PICTURE 99 VALUE 53.
  48     004800 77  OP-NAME                      PICTURE X(9).
  49     004900 01  INPUT-END                    PICTURE X(9).
  50     005000     88  THE-END-OF-INPUT         VALUE "E".
         005100
  51     005200 PROCEDURE DIVISION.
  52     005300 DECLARATIVES.
```

図 126. 相対ファイル検索プログラムの例 (1/3)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/RTRVREL         ISERIES1   06/02/15 14:51:40        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
         005400 RELATIVE-FILE-ERROR SECTION.
         005500          USE AFTER STANDARD ERROR PROCEDURE ON RELATIVE-FILE.
         005600 RELATIVE-ERROR-PARA.
   53    005700     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR RELATIVE-FILE".
   54    005800     DISPLAY "FILE STATUS IS ", RELATIVE-FILE-STATUS.
   55    005900     DISPLAY "PROCESSING ENDED"
   56    006000     STOP RUN.
         006100
         006200 INPUT-FILE-ERROR SECTION.
         006300          USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
         006400 INPUT-ERROR-PARA.
   57    006500     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INPUT-FILE ".
   58    006600     DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
   59    006700     DISPLAY "PROCESSING ENDED"
   60    006800     STOP RUN.
         006900
         007000 PRINT-FILE-ERROR SECTION.
         007100          USE AFTER STANDARD ERROR PROCEDURE ON PRINT-FILE.
         007200 PRINT-ERROR-MSG.
   61    007300     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR PRINT-FILE ".
   62    007400     DISPLAY "FILE STATUS IS ", PRINT-FILE-STATUS.
   63    007500     DISPLAY "PROCESSING ENDED"
   64    007600     STOP RUN.
         007700 END DECLARATIVES.
         007800
         007900 MAIN-PROGRAM SECTION.
         008000 MAINLINE.
   65    008100     MOVE "OPEN" TO OP-NAME.
   66    008200     OPEN INPUT INPUT-FILE
         008300                RELATIVE-FILE
         008400          OUTPUT PRINT-FILE.
         008500
   67    008600     MOVE SPACES TO PRINT-RECORD.
   68    008700     PERFORM READ-INPUT-FILE.
   69    008800     PERFORM CONTROL-PROCESS THRU READ-INPUT-FILE
         008900                             UNTIL THE-END-OF-INPUT.
         009000
   70    009100     MOVE "CLOSE" TO OP-NAME.
   71    009200     CLOSE RELATIVE-FILE
         009300           INPUT-FILE
         009400           PRINT-FILE.
   72    009500     STOP RUN.
         009600
         009700 CONTROL-PROCESS.
   73    009800     IF (END-WEEK = SPACES OR END-WEEK = 00)
   74    009900         MOVE "READ" TO OP-NAME
   75    010000         READ RELATIVE-FILE
   76    010100         PERFORM PRINT-SUMMARY VARYING REL-INDEX FROM 1 BY 1
         010200                               UNTIL REL-INDEX > 5
         010300     ELSE
   77    010400         MOVE "READ" TO OP-NAME
   78    010500         READ RELATIVE-FILE
   79    010600         PERFORM READ-REL-SEQ
         010700              UNTIL RELATIVE-WEEK(1) GREATER THAN END-WEEK
         010800         END-IF.
```

図 126. 相対ファイル検索プログラムの例 (2/3)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/RTRVREL        ISERIES1   06/02/15 14:51:40        ページ    4
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
         010900
         011000 READ-INPUT-FILE.
   80    011100     MOVE "READ" TO OP-NAME.
   81    011200     READ INPUT-FILE
   82    011300         AT END SET THE-END-OF-INPUT TO TRUE
         011400     END-READ.
         011500
         011600 READ-REL-SEQ.
   83    011700     PERFORM PRINT-SUMMARY VARYING REL-INDEX FROM 1 BY 1
         011800                           UNTIL REL-INDEX > 5.
   84    011900     MOVE "READ NEXT" TO OP-NAME.
   85    012000     READ RELATIVE-FILE NEXT RECORD
   86    012100         AT END MOVE HIGH-WEEK TO RELATIVE-WEEK(1)
         012200     END-READ.
         012300
         012400 PRINT-SUMMARY.
   87    012500     MOVE RELATIVE-YEAR (REL-INDEX) TO PRINT-YEAR.
   88    012600     MOVE RELATIVE-WEEK (REL-INDEX) TO PRINT-WEEK.
   89    012700     MOVE RELATIVE-UNIT-SALES (REL-INDEX) TO PRINT-UNIT-SALES.
   90    012800     MOVE RELATIVE-DOLLAR-SALES(REL-INDEX) TO PRINT-DOLLAR-SALES.
   91    012900     MOVE "WRITE" TO OP-NAME.
   92    013000     WRITE PRINT-RECORD AFTER ADVANCING 2 LINES
         013100     END-WRITE.
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 126. 相対ファイル検索プログラムの例 (3/3)

索引付きファイルの作成

このプログラムでは、銀行預金者の記録の索引付きファイルを作成します。各索引付きファイル・レコード内のキーは INDEX-KEY (預金者の口座番号) です。入力レコードは、このキーの昇順に並べられています。レコードは入力ファイルから読み取られて、索引付きファイル・レコード域に転送されます。その後、索引付きファイルのレコードが書き込まれます。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/CRTIND          ISERIES1   06/02/15 14:52:38        ページ     2
                                   ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.  CRTIND.
           000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600   SOURCE-COMPUTER. IBM-ISERIES
    6     000700   OBJECT-COMPUTER. IBM-ISERIES.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
    9     001000     SELECT INDEXED-FILE ASSIGN TO DISK-INDEXFILE
   11     001100         ORGANIZATION IS INDEXED
   12     001200          ACCESS IS SEQUENTIAL
   13     001300         RECORD KEY IS INDEX-KEY
   14     001400         FILE STATUS IS INDEXED-FILE-STATUS.
   15     001500     SELECT INPUT-FILE ASSIGN TO DISK-FILEG
   17     001600         FILE STATUS IS INPUT-FILE-STATUS.
           001700
   18     001800 DATA DIVISION.
   19     001900 FILE SECTION.
   20     002000 FD  INDEXED-FILE.
   21     002100 01  INDEX-RECORD.
   22     002200     05  INDEX-KEY                    PICTURE X(10).
   23     002300     05  INDEX-FLD1                   PICTURE X(10).
   24     002400     05  INDEX-NAME                   PICTURE X(20).
   25     002500     05  INDEX-BAL                    PICTURE S9(5)V99.
   26     002600 FD  INPUT-FILE.
   27     002700 01  INPUT-RECORD.
   28     002800     05  INPUT-KEY                    PICTURE X(10).
   29     002900     05  INPUT-NAME                   PICTURE X(20).
   30     003000     05  INPUT-BAL                    PICTURE S9(5)V99.
   31     003100 WORKING-STORAGE SECTION.
   32     003200 77  INDEXED-FILE-STATUS              PICTURE XX.
   33     003300 77  INPUT-FILE-STATUS                PICTURE XX.
   34     003400 77  OP-NAME                          PICTURE X(7).
   35     003500 01  INPUT-END                        PICTURE X VALUE SPACES.
   36     003600     88  THE-END-OF-INPUT             VALUE "E".
           003700
   37     003800 PROCEDURE DIVISION.
   38     003900 DECLARATIVES.
          004000 INPUT-ERROR SECTION.
          004100       USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
          004200 INPUT-ERROR-PARA.
   39     004300     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INPUT-FILE ".
   40     004400     DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
   41     004500     DISPLAY "PROCESSING ENDED"
   42     004600     STOP RUN.
          004700
          004800 OUTPUT-ERROR SECTION.
          004900       USE AFTER STANDARD ERROR PROCEDURE ON INDEXED-FILE.
          005000 OUTPUT-ERROR-PARA.
   43     005100     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INDEXED-FILE ".
   44     005200     DISPLAY "FILE STATUS IS ", INDEXED-FILE-STATUS.
   45     005300     DISPLAY "PROCESSING ENDED"
```

図 127. 索引付きファイル・プログラムの例 (1/2)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/CRTIND         ISERIES1   06/02/15 14:52:38        ページ    3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   46     005400     STOP RUN.
          005500 END DECLARATIVES.
          005600
          005700 MAIN-PROGRAM SECTION.
          005800 MAINLINE.
   47     005900     MOVE "OPEN" TO OP-NAME.
   48     006000     OPEN INPUT INPUT-FILE
          006100         OUTPUT INDEXED-FILE.
          006200
   49     006300     MOVE "READ" TO OP-NAME.
   50     006400     READ INPUT-FILE
   51     006500          AT END SET THE-END-OF-INPUT TO TRUE
          006600     END-READ.
          006700
   52     006800     PERFORM UNTIL THE-END-OF-INPUT
   53     006900         MOVE INPUT-KEY TO INDEX-KEY
   54     007000         MOVE INPUT-NAME TO INDEX-NAME
   55     007100         MOVE INPUT-BAL TO INDEX-BAL
   56     007200         MOVE SPACES TO INDEX-FLD1
   57     007300         MOVE "WRITE" TO OP-NAME
   58     007400         WRITE INDEX-RECORD
          007500
   59     007600         MOVE "READ" TO OP-NAME
   60     007700         READ INPUT-FILE
   61     007800              AT END SET THE-END-OF-INPUT TO TRUE
          007900         END-READ
          008000     END-PERFORM.
          008100
   62     008200     MOVE "CLOSE" TO OP-NAME.
   63     008300     CLOSE INPUT-FILE
          008400           INDEXED-FILE.
   64     008500     STOP RUN.
          008600
                         * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 127. 索引付きファイル・プログラムの例 (2/2)

索引付きファイルの更新

このプログラムでは、CRTIND プログラムで作成された索引付きファイルを、動的アクセスを使って更新します。

入力レコードには、レコードのキー、預金者の名前、および取引き (トランザクション) 額が含まれています。

プログラムは、入力レコードを読み込んでから、次のことを判別します。

- それがトランザクション・レコードかどうか (その場合、レコードのすべてのフィールドにはデータが記入されている)
- 特定の総称クラスの順次検索を要求しているレコードかどうか (その場合、入力レコードの INPUT-GEN-FLD フィールドにのみデータが記入されている)

トランザクション・レコードを更新して印刷するのには、ランダム・アクセスを使用します。 1 つの総称クラスの中のすべてのレコードの検索と印刷には、順次アクセスを使用します。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/UPDTIND         ISERIES1   06/02/15 14:54:04        ページ    2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.  UPDTIND.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600   SOURCE-COMPUTER. IBM-ISERIES
    6     000700   OBJECT-COMPUTER. IBM-ISERIES.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
    9     001000     SELECT INDEXED-FILE ASSIGN TO DISK-INDXFILE
   11     001100         ORGANIZATION IS INDEXED
   12     001200         ACCESS IS DYNAMIC
   13     001300         RECORD KEY IS INDEX-KEY
   14     001400         FILE STATUS IS INDEXED-FILE-STATUS.
   15     001500     SELECT INPUT-FILE ASSIGN TO DISK-FILEH
   17     001600         FILE STATUS IS INPUT-FILE-STATUS.
   18     001700     SELECT PRINT-FILE ASSIGN TO PRINTER-OSYSPRT
   20     001800         FILE STATUS IS PRINT-FILE-STATUS.
          001900
   21     002000 DATA DIVISION.
   22     002100 FILE SECTION.
   23     002200 FD  INDEXED-FILE.
   24     002300 01  INDEX-RECORD.
   25     002400     05  INDEX-KEY.
   26     002500       10    INDEX-GEN-FLD    PICTURE X(5).
   27     002600       10    INDEX-DET-FLD    PICTURE X(5).
   28     002700     05  INDEX-FLD1           PICTURE X(10).
   29     002800     05  INDEX-NAME           PICTURE X(20).
   30     002900     05  INDEX-BAL            PICTURE S9(5)V99.
   31     003000 FD  INPUT-FILE.
   32     003100 01  INPUT-REC.
   33     003200     05  INPUT-KEY.
   34     003300       10  INPUT-GEN-FLD      PICTURE X(5).
   35     003400       10  INPUT-DET-FLD      PICTURE X(5).
   36     003500     05  INPUT-NAME           PICTURE X(20).
   37     003600     05  INPUT-AMT            PICTURE S9(5)V99.
   38     003700 FD  PRINT-FILE
          003800     LINAGE 12 LINES FOOTING AT 9.
   39     003900 01  PRINT-RECORD-1.
   40     004000     05  PRINT-KEY            PICTURE X(10).
   41     004100     05  FILLER               PICTURE X(5).
   42     004200     05  PRINT-NAME           PICTURE X(20).
   43     004300     05  FILLER               PICTURE X(5).
   44     004400     05  PRINT-BAL            PICTURE $$$,$$9.99-.
   45     004500     05  FILLER               PICTURE X(7).
   46     004600     05  PRINT-AMT            PICTURE $$$,$$9.99-.
   47     004700     05  FILLER               PICTURE X(5).
   48     004800     05  PRINT-NEW-BAL        PICTURE $$$,$$9.99-.
   49     004900 01  PRINT-RECORD-2           PICTURE X(89).
          005000
   50     005100 WORKING-STORAGE SECTION.
   51     005200 77  INDEXED-FILE-STATUS      PICTURE XX.
   52     005300 77  INPUT-FILE-STATUS        PICTURE XX.
```

図 128. 索引付きファイル更新プログラムの例 (1/4)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/UPDTIND        ISERIES1   06/02/15 14:54:04        ページ    3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   53     005400 77  PRINT-FILE-STATUS        PICTURE XX.
   54     005500 77  OP-NAME                  PICTURE X(9).
   55     005600 77  LINES-TO-FOOT            PICTURE 99.
   56     005700 01  PAGE-HEAD.
   57     005800     05  FILLER               PICTURE X(38) VALUE SPACES.
   58     005900     05  FILLER               PICTURE X(13) VALUE "UPDATE REPORT".
   59     006000     05  FILLER               PICTURE X(38) VALUE SPACES.
   60     006100 01  COLUMN-HEAD.
   61     006200     05  FILLER               PICTURE X(6)  VALUE "KEY ID".
   62     006300     05  FILLER               PICTURE X(9)  VALUE SPACES.
   63     006400     05  FILLER               PICTURE X(4)  VALUE "NAME".
   64     006500     05  FILLER               PICTURE X(21) VALUE SPACES.
   65     006600     05  FILLER               PICTURE X(11) VALUE "CUR BALANCE".
   66     006700     05  FILLER               PICTURE X(6)  VALUE SPACES.
   67     006800     05  FILLER               PICTURE X(13) VALUE "UPDATE AMOUNT".
   68     006900     05  FILLER               PICTURE X(4)  VALUE SPACES.
   69     007000     05  FILLER               PICTURE X(11) VALUE "NEW BALANCE".
   70     007100     05  FILLER               PICTURE X(4)  VALUE SPACES.
   71     007200 01  PAGE-FOOT.
   72     007300     05  FILLER               PICTURE X(81)  VALUE SPACES.
   73     007400     05  FILLER               PICTURE A(6)   VALUE "PAGE  ".
   74     007500     05  PG-NUMBER            PICTURE 99     VALUE 00.
          007600
   75     007700 01  INPUT-END               PICTURE X VALUE SPACE.
   76     007800     88  THE-END-OF-INPUT              VALUE "E".
          007900
   77     008000 PROCEDURE DIVISION.
   78     008100 DECLARATIVES.
          008200 INPUT-ERROR SECTION.
          008300       USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
          008400 INPUT-ERROR-PARA.
   79     008500     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INPUT-FILE ".
   80     008600     DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
   81     008700     DISPLAY "PROCESSING ENDED"
   82     008800     STOP RUN.
          008900
          009000 I-O-ERROR SECTION.
          009100       USE AFTER STANDARD ERROR PROCEDURE ON INDEXED-FILE.
          009200 I-O-ERROR-PARA.
   83     009300     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INDEXED-FILE ".
   84     009400     DISPLAY "FILE STATUS IS ", INDEXED-FILE-STATUS.
   85     009500     DISPLAY "PROCESSING ENDED"
   86     009600     STOP RUN.
          009700
          009800 OUTPUT-ERROR SECTION.
          009900       USE AFTER STANDARD ERROR PROCEDURE ON PRINT-FILE.
          010000 OUTPUT-ERROR-PARA.
   87     010100     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR PRINT-FILE ".
   88     010200     DISPLAY "FILE STATUS IS ", PRINT-FILE-STATUS.
   89     010300     DISPLAY "PROCESSING ENDED"
   90     010400     STOP RUN.
          010500 END DECLARATIVES.
          010600
          010700 MAIN-PROGRAM SECTION.
          010800 MAINLINE.
```

図 128. 索引付きファイル更新プログラムの例 (2/4)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/UPDTIND         ISERIES1   06/02/15 14:54:04        ページ     4
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   91     010900     MOVE "OPEN" TO OP-NAME.
   92     011000     OPEN INPUT INPUT-FILE
          011100         I-O INDEXED-FILE
          011200         OUTPUT PRINT-FILE.
          011300
   93     011400     PERFORM PAGE-START.
   94     011500     PERFORM READ-INPUT-FILE.
   95     011600     PERFORM PROCESS-DATA THRU READ-INPUT-FILE
          011700                           UNTIL THE-END-OF-INPUT.
   96     011800     PERFORM PAGE-END.
          011900
   97     012000     MOVE "CLOSE" TO OP-NAME.
   98     012100     CLOSE INPUT-FILE
          012200           INDEXED-FILE
          012300           PRINT-FILE.
   99     012400     STOP RUN.
          012500
          012600 PROCESS-DATA.
  100     012700     IF INPUT-DET-FLD EQUAL SPACES
  101     012800         MOVE INPUT-GEN-FLD TO INDEX-GEN-FLD
  102     012900         MOVE "START" TO OP-NAME
  103     013000         START INDEXED-FILE
          013100              KEY IS NOT LESS THAN INDEX-GEN-FLD
          013200         END-START
  104     013300         PERFORM SEQUENTIAL-PROCESS
          013400             UNTIL INPUT-GEN-FLD NOT EQUAL INDEX-GEN-FLD
          013500     ELSE
  105     013600         MOVE INPUT-KEY TO INDEX-KEY
  106     013700         MOVE "READ" TO OP-NAME
  107     013800         READ INDEXED-FILE
  108     013900         IF INPUT-GEN-FLD EQUAL INDEX-GEN-FLD THEN
  109     014000             MOVE INDEX-KEY TO PRINT-KEY
  110     014100             MOVE INDEX-NAME TO PRINT-NAME
  111     014200             MOVE INDEX-BAL TO PRINT-BAL
  112     014300             MOVE INPUT-AMT TO PRINT-AMT
  113     014400             ADD INPUT-AMT TO INDEX-BAL
  114     014500             MOVE INDEX-BAL TO PRINT-NEW-BAL
  115     014600             PERFORM PRINT-DETAIL
  116     014700             MOVE "REWRITE" TO OP-NAME
  117     014800             REWRITE INDEX-RECORD
          014900         END-IF
          015000     END-IF.
          015100
          015200 READ-INPUT-FILE.
  118     015300     MOVE "READ" TO OP-NAME.
  119     015400     READ INPUT-FILE
  120     015500         AT END SET THE-END-OF-INPUT TO TRUE
          015600     END-READ.
          015700
          015800 SEQUENTIAL-PROCESS.
  121     015900     MOVE "READ NEXT" TO OP-NAME.
  122     016000     READ INDEXED-FILE NEXT RECORD
  123     016100         AT END MOVE HIGH-VALUE TO INDEX-GEN-FLD
          016200     END-READ.
  124     016300     IF INPUT-GEN-FLD EQUAL INDEX-GEN-FLD THEN
```

図 128. 索引付きファイル更新プログラムの例 (3/4)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/UPDTIND        ISERIES1   06/02/15 14:54:04        ページ    5
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
  125     016400         MOVE INDEX-KEY TO PRINT-KEY
  126     016500         MOVE INDEX-NAME TO PRINT-NAME
  127     016600         MOVE INDEX-BAL TO PRINT-NEW-BAL
  128     016700         PERFORM PRINT-DETAIL
          016800     END-IF.
          016900
          017000 PRINT-DETAIL.
  129     017100     MOVE "WRITE" TO OP-NAME.
  130     017200     WRITE PRINT-RECORD-1
          017300          AT END-OF-PAGE
  131     017400          PERFORM PAGE-END THROUGH PAGE-START
          017500     END-WRITE.
  132     017600     MOVE SPACES TO PRINT-RECORD-1.
          017700
          017800 PAGE-END.
  133     017900     MOVE "WRITE" TO OP-NAME.
  134     018000     ADD 1 TO PG-NUMBER.
  135     018100     SUBTRACT LINAGE-COUNTER OF PRINT-FILE FROM 12
          018200          GIVING LINES-TO-FOOT.
  136     018300     MOVE SPACES TO PRINT-RECORD-1.
  137     018400     WRITE PRINT-RECORD-1
          018500          AFTER ADVANCING LINES-TO-FOOT
          018600     END-WRITE.
  138     018700     WRITE PRINT-RECORD-2 FROM PAGE-FOOT
          018800          BEFORE ADVANCING PAGE
          018900     END-WRITE.
          019000
          019100 PAGE-START.
  139     019200     WRITE PRINT-RECORD-2 FROM PAGE-HEAD
          019300          AFTER ADVANCING 1 LINE
          019400     END-WRITE.
  140     019500     MOVE SPACES TO PRINT-RECORD-2.
  141     019600     WRITE PRINT-RECORD-2 FROM COLUMN-HEAD
          019700          AFTER ADVANCING 1 LINE
          019800     END-WRITE.
  142     019900     MOVE SPACES TO PRINT-RECORD-2.
                          * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 128. 索引付きファイル更新プログラムの例 (4/4)

IBM i システム・ファイル

IBM i システムには、次の 4 つのカテゴリーのファイルがあります。

- データベース・ファイル
- 装置ファイル
- DDM ファイル
- 保管ファイル

データベース・ファイルを使用することにより、情報をシステムに永続的に保管できます。データベース・ファイルは、メンバーと呼ばれるレコードのグループに分割されます。データベース・ファイルには、次の 2 つのタイプがあります。

- **物理ファイル**は、データ・レコードが入るファイルです (ほかのシステムのディスク・ファイルに似ています)。
- **論理ファイル**はデータベース・ファイルの 1 つであり、それを通して 1 つまたは複数の物理ファイルのデータにアクセスできます。このデータの形式や編成は、物理ファイルのデータの場合と異なります。各論理ファイルは、物理ファイルのデータに異なるアクセス・パス (索引) を定義し、かつ物理ファイルに定義したフィールドを除外したり順序を変えたりすることができます。

データベース物理ファイルは、1 つの IBM i システムに置くことも、複数の IBM i システムに置くこともできます。データベース物理ファイルが、複数の IBM i シ

ステムに置かれた場合は、**分散物理ファイル**または**分散ファイル**と呼びます。論理ファイルは、1 つまたは複数の物理ファイルを基本としているため、基礎となるファイルが分散されれば、論理ファイルも分散ファイルになります。

ILE COBOL プログラムから分散ファイルにアクセスする場合は、分散ファイルをオープン (OPEN) します。他の中間ファイルは必要なく、かつ分散ファイルの一部が含まれる IBM i システムについて知る必要もありません。

これを、リモート・システムに存在するデータベース・ファイルの名前を識別する、分散データ管理 (DDM) ファイルと比べてみます。 ILE COBOL では、リモート・データベースをオープン (OPEN) するために、実際にはローカル DDM ファイルをオープンします。したがって、DDM ファイルは装置ファイルとデータベース・ファイルの特性を結び付けたものになります。 DDM ファイルは、装置ファイルとして、リモート・ロケーション名、ローカル・ロケーション名、装置名、モード、リモート・ネットワーク ID を参照して、リモート・システムをターゲット・システムとして識別します。 DDM ファイルは、アプリケーション・プログラムから見ればデータベース・ファイルであって、ILE COBOL プログラムとリモート・ファイル間のアクセス装置として使用されます。

DDM ファイルはリモート・データベース・ファイルを識別し、データベース・ファイルは分散ファイルになることもあるため、DDM ファイルは、分散ファイルを参照することができます。

DDM ファイルおよび分散ファイルの詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベース」カテゴリーの中の『*DB2 Universal Database for AS/400*』セクションの中の『*DB2 Universal Database for AS/400*』セクションを参照してください。

分散データ管理 (DDM) ファイル

DISK または DATABASE の装置に割り当てられた ILE COBOL ファイルは、DDM ファイルを参照することができます。

DDM ファイルは、ターゲット・システムのデータ・ファイルにアクセスする際に必要な情報が入っている、ローカル (またはソース) システム上のファイルです。プログラムがデータベース操作を行うためにアクセスできるデータ・ファイルではありません。ソース・システムで実行する ILE COBOL プログラムが DDM ファイルをオープンする際、そのファイル情報は、DDM がデータのアクセスを行うリモート・ファイルを見つけるのに使用されます。

DDM ファイルは、DDM ファイル作成 (CRTDDMF) コマンドによって作成されます。 DDM ファイルは、ほかのファイルまたはオブジェクトと同じく、ファイル・オブジェクトとしてライブラリーに保管されます。

ILE COBOL プログラムが DDM ファイルをオープンすると、DDM とターゲット・システムとの会話が確立されます。さらに、プログラムが DDM ファイルをオープンして、リモート・ファイルのレコードにアクセスする場合は、リモート・ファイルへのオープン・データ・パス (ODP) も確立されます。

DDM は、アーキテクチャーが異なるシステム間で通信を行う場合に使用することができます。たとえば、IBM i システムとシステム/36 では、アーキテクチャーが異なりますが、2 つのシステムは、DDM を用いてお互いのデータベースのファイルにアクセスすることができます。

以下のセクションで、DDM ファイルに固有の性質と DDM ファイルを用いたデータベース・ファイルのアクセスについて説明します。データベース・ファイルの他のトピックについては、本章の別の個所で説明します。

非 IBM i システムでの DDM ファイルの使用

IBM i システム間だけでなく、システム /38 システムまたはシステム/36 システム間で DDM を使用する場合は、リモート・ファイルを指す方法を除いては、両方のタイプの概念が類似していることに注意してください。

- IBM i システムとシステム /38 は、別個の DDM ファイルを用いてアクセスする各リモート・ファイルを参照します。
- システム/36 システムは、アクセスするリモート・ファイルごとに 1 つのネットワーク・リソース・ディレクトリー記入項目を含むネットワーク・リソース・ディレクトリーを使用します。

DDM プログラミングに関する考慮事項

一般に、i5/OS と非 i5/OS のどちらのターゲット・システムの場合も、ILE COBOL での DDM ファイル名の指定は、データベース・ファイルを指定できるところであればどこででも行うことができます。 この部分では、DDM ファイルについての ILE COBOL プログラミングに関する考慮事項を取りまとめます。

- DDM ファイル名は、バインド COBOL プログラムの作成 (CRTBNDCBL) コマンド、COBOL モジュールの作成 (CRTCBLMOD) コマンド、および COBOL プログラム作成 (CRTCBLPGM) コマンド上で指定することができます。
  - i5/OS または非 i5/OS システム上で、ソース・ステートメントが入っているリモート・ファイルにアクセスするには、SRCFILE パラメーターで DDM ファイル名を指定し、SRCMBR パラメーターでメンバー名を指定することができます。
  - IBM i またはシステム /38 ターゲット・システムの場合は、ローカルのソース・ファイルおよびメンバーと同じ方法で、リモートの IBM i またはシステム /38 のソース・ファイル (およびオプションで、メンバー) にアクセスすることができます。
  - 非 IBM i ターゲット・システムの場合は、CRTBNDCBL または CRTCBLPGM コマンドで、PGM と SRCMBR の両パラメーターにデフォルトが使用された場合は、リモート・ソース・ファイルにアクセスすることができます。すなわち、メンバー名を指定する場合は、SRCFILE パラメーターで指定される DDM ファイル名と同じでなければなりません。 CRTCBLMOD コマンドが従う規則は、PGM および SRCMBR パラメーターが MODULE および SRCMBR パラメーターによって置き換えられる点を除き、類似しています。

- コンパイラー・リストをターゲット・システムのデータベース・ファイルに入れるには、CRTCBLPGM コマンドの PRTFILE パラメーターで DDM ファイル名を指定することができます。

- DDM ファイル名は、ILE COBOL SORT および MERGE 操作の入出力ファイルとして指定することができます。
- COPY ステートメントの DDS オプションを用いて、外部記述レコード様式の 1 つまたはすべてを、DDM ファイルが参照するリモート・ファイルからコンパイルされるプログラムにコピーするときは、ILE COBOL のそのステートメントで DDM ファイルを使用することができます。リモート・ファイルが IBM i システムまたはシステム /38 上にないときにこれを行っても、レコード記述のフィールド宣言の名前は無意味となります。代わりに、フィールド名はすべて F*nnnnn* と宣言され、キー・フィールドは K*nnnnn* と宣言されます。

  ターゲットが IBM i システムでもシステム /38 でもないときに、リモート・ファイルを記述する方法としてお勧めするのは、ローカル・システム上にデータ記述仕様 (DDS) を持って、物理ファイル作成 (CRTPF) コマンドまたは論理ファイル作成 (CRTLF) コマンドをそのローカル・システムで入力することです。ローカル・ファイル名を用いて、プログラムをコンパイルします。リモート・システムのファイルが、対応するフィールド・タイプとフィールド長を持つようにしてください。リモート・ファイルにアクセスするには、たとえば次のように、プログラムの前にデータベース・ファイル指定変更 (OVRDBF) コマンドを使用してください。

  ```
  OVRDBF FILE(PGMFIL) TOFILE(DDMFIL) LVLCHK(*NO)
  ```

- DDM ファイル名は、COPY ステートメントで指定することができます。
  - ライブラリー名をファイル名と一緒に指定しない場合は、ユーザーのライブラリー・リストのそのファイル名で検出された最初のファイルが、組み込みファイルとして使用されます。
  - ターゲット・システムが IBM i システムでもシステム /38 でもない場合は、COPY ステートメント上で DDM ファイル名を組み込みファイルとして指定できますが、メンバー名は DDM ファイル名と同じでなければなりません。
- ターゲット・システムがシステム/36 の場合は、関連するリモート・ファイルの論理ファイルがその上に作成されていると、ILE COBOL を用いて出力用に DDM ファイルをオープンすることはできません。論理ファイルのシステム/36 の場合は、ILE COBOL のプログラミング言語がファイルを使用する前にクリアしようとするので、オープン操作 (出力のオープン) は失敗します。
- ILE COBOL プログラムでソース・システム上の DDM ファイルをオープンするときは、IBM i および非 IBM i の両方のターゲットの場合、CLOSE、DELETE、OPEN、READ、REWRITE、START、および WRITE のステートメントを用いて、ターゲット・システムのリモート・ファイルに対して入出力操作を行うことができます。

DDM 直接 (相対) ファイル・サポート

IBM i システムがサポートするファイル・タイプには、直接ファイルは含まれていません。 (IBM i システムは、直接ファイルを順次ファイルとして作成します。) しかし、IBM i システム上の ILE COBOL プログラムでは、SELECT ステートメントで RELATIVE の編成を指定して、ファイルを直接ファイルとしてアクセスするように指定することができます。

i5/OS で直接ファイルを処理するときは、以下の点に留意してください。

- ファイルを、非 IBM i システムのプログラムまたはユーザーが、ローカルの IBM i システム上で直接ファイルとして作成した場合は、ILE COBOL プログラムが、リモートの非 IBM i ソース・システムから、直接ファイルとしてファイルにアクセスすることができます。
- ファイルが IBM i システム上のプログラムまたはユーザーによって、同じローカルの IBM i システム上に作成された場合は、非 i5/OS がそのファイルに直接ファイルとしてアクセスすることはできません。その理由は、この場合、IBM i ターゲット・システムは、そのファイルが直接ファイルであるのか順次ファイルであるのかを判別できないからです。
- リモート・システムが作成したファイルは、いずれもローカルで使用することができます。

分散ファイル

分散ファイルを使用すると、単一データベースとしての外観および機能性を保ったまま、1 つのデータベース・ファイルを複数の IBM i システム上に分散させることができます。データベースの要求を複数のシステムに分割することで、大型照会プログラムのパフォーマンスを向上させることができます。分散ファイルの働きは、DATABASE ファイルとほとんど変わりません。しかし、ファイルは複数システム全体に分散されるため、到着順や相対番号は信頼できず、かつリモート・システムにアクセスするたびに、データ・リンクがシステム間でデータをやり取りするのに余計な時間が必要になります。

分散ファイルは、物理ファイルの作成 (CRTPF) コマンドを使用して、他のデータベース・ファイルと同じように作成されます。このコマンドには、以下のように、分散ファイルに関連する新しいパラメーターが 2 つあります。

- ノード・グループ (NODGRP)
- 区分化キー (PTNKEY)

最初のパラメーターが持つ値は、通常のファイルに対しては *NONE、分散ファイルに対してはノード・グループの名前です。ノード・グループとは、ファイルのレコードが入っているリレーショナル・データベースの名前を指定する、新しいシステム・オブジェクト・タイプ (タイプ *NODGRP) です。ノード・グループは、ノード・グループ作成 (CRTNODGRP) コマンドを使用して作成します。

分散ファイルのレコードは、区分化キーに基づいて、さまざまなリレーショナル・データベース間で分割されます。区分化キーは、分散ファイルのフィールド (もしくはフィールドのセット) で、この分散ファイルの値によって各レコードが保管されるリレーショナル・データベースを判別します。

既存の物理ファイルは、物理ファイル変更 (CHGPF) コマンドを用いて分散ファイルに変更することができます。 CRTPF コマンドに追加された 2 つの新しいパラメーターである、ノード・グループおよび区分化キーも CHGPF コマンドに追加されました。

データ処理の場合のオープンに関する考慮事項

分散ファイルのデータへのアクセスは、バッファー付きまたは非バッファー付きの方法で行うことができます。このレコードのバッファー方式は、SEQONLY 処理のように、他のバッファー方式に追加されるものです。

データベース・ファイル指定変更 (OVRDBF) コマンドには、以下の 3 つの値を持った分散データ (DSTDTA) という新しいパラメーターがあります。

***BUFFERED**
データがバッファー中に保持される。

***PROTECTED**
*BUFFERED に似ているが、ファイルは他のジョブから更新できないようにロックされている。

***CURRENT**
データは、バッファーに入っていない。

以降の部分で、分散データ処理が指定変更された場合と、指定変更されなかった場合の、オープンに関する考慮事項を説明します。

分散データ処理が指定変更される場合

以下の考慮事項は、分散データ処理が指定変更される 場合の、分散ファイルのオープンに適用されます。

- 分散ファイルのオープン操作が入力専用処理であって、ファイルがオープンされている間に、削除、挿入、もしくは更新されるレコードがプログラムにより直接処理される場合は、OVRDBF (データベース・ファイル指定変更) コマンドを用いて分散ファイルを非バッファー付き検索 (*CURRENT) に指定変更しなければなりません。非バッファー付き検索は、バッファー付き検索とはパフォーマンスが違いますが、データの保全性は保証され、分散ファイルをオープンしている間のレコードの並行性は最大です。
- ファイルのオープン操作が更新もしくは削除操作のためならば、OVRDBF (データベース・ファイル指定変更) コマンドを用いて、分散データ処理を、保護付きバッファー検索 (*PROTECTED) かバッファー付き検索 (*BUFFERED) のどちらかに指定変更することができます。

  保護付きバッファー検索には、以下のような利点と欠点があります。
  - 達成するパフォーマンスがバッファー付き検索と同じです。
  - プログラムがレコードの削除、挿入、もしくは更新を行わなければ、データ保全性が保証されます。
  - 別の処理の分散ファイルのオープンが、入力専用処理以外の処理用であって、同じく保護付きバッファー検索を組み込まない場合は、使用できません。

  バッファー付き検索には、以下のような利点と欠点があります。

– 達成するパフォーマンスが保護付きバッファー検索と同じです。
– 分散ファイルをオープンしている間の最大のレコード並行性を考慮しています。
– オープン後のレコードが、分散ファイルで削除、挿入、もしくは更新されても、それが見えていなかった場合があります。このため、プログラムが誤ったレコードを更新もしくは削除する場合があります。

分散データ処理が指定変更されない場合

以下の考慮事項は、分散データ処理が指定変更されない 場合の、分散ファイルのオープンに適用されます。

- システムは、バッファー付き検索 (*BUFFERED) コマンドを用いて入力専用にオープンされた分散ファイルを処理します。バッファー付き検索は、最大のレコード並行性とともに最上のパフォーマンスが得られますが、ファイルに変更を加えても、それがすべて見えない場合があります。詳細については『分散ファイルの入出力に関する考慮事項』を参照してください。
- システムは、出力専用にオープンされた分散ファイルを、一度に 1 レコードずつ処理します。分散ファイルが出力専用にオープンされている場合は、DSTDTA パラメーターはなにも作用しません。また、SEQONLY(*YES) 処理が要求されていると、SEQONLY(*NO) に変更されます。 SEQONLY(*NO) 処理では、レコードがファイルに挿入された時点でレコードごとにフィードバックがあります。
- システムは、非バッファー付き検索 (*CURRENT) を用いて、更新もしくは削除を組み込むオプションを指定してオープンされた分散ファイルを処理します。非バッファー付き検索の場合、更新もしくは削除するレコードは、分散ファイル・データがすべて非分散データベース・ファイルに入れられている場合に更新もしくは削除されるレコードと必ず同じになります。非バッファー付き検索が使用されるため、分散ファイルに関して最高のパフォーマンスにはなりませんが、最善のデータ保全性と最大のレコード並行性が保証されます。

  注: 到着順分散ファイルの場合は、レコードは、最初のノードから始まって、次が 2 番目のノードというように到着順に検索されます。複写キーに関する考慮事項については『分散ファイルの入出力に関する考慮事項』を参照してください。
- システムは、非バッファー付き検索 (*CURRENT) を用いて (更新と削除の両オプションを組み込んでいるため)、すべての操作 (*INP、*OUT、*UPD、*DLT) でオープンになった分散ファイルを処理します。

分散ファイルの入出力に関する考慮事項

以下の考慮事項は、分散ファイルの入出力操作に適用されます。

- 到着順分散ファイルと、キー順アクセス・パスがオープン時に無視されたキー順分散ファイルの入力の場合、レコードは以下のように検索されます。
  1. ファイルの作成時にノード・グループによって定義された、最初のノードからのレコードは、すべて最初のノードからの到着順に検索されます。
  2. 最初のノードからのすべてのレコードが検索されると、2 番目のノードからのすべてのレコードが、2 番目のノードからの到着順に検索されます。

3. 2 番目のノードからのすべてのレコードが検索されると、3 番目のノードからのすべてのレコードが、3 番目のノードからの到着順に検索されます。
4. ファイル作成時にノード・グループによって定義された最後のノードに到達するまで、これが続きます。
5. 最後のノードからのすべてのレコードが到着順に検索された後、ファイルの終わりに到達します。

したがって、到着順に処理された分散ファイルは、分散ファイルの異なるノードを通しては到着順に処理されません。

- キー順アクセス・パスがオープン時に無視されたキー順分散ファイルの入力の場合、レコードは以下のように検索されます。
  - 重複キー値を指定したレコードの、先に変更先出し法 (FCFO)、先入れ先出し法 (FIFO)、または後入れ先出し法 (LIFO) の順序が有効なのは、同じノードからのレコードの場合に限られます。
  - ファイルの作成時にノード・グループによって定義された最初のノードからの、重複キー値を指定したすべてのレコードは、指定したアクセス・パスの順序で検索されます。
  - 最初のノードからの、重複キー値を指定したすべてのレコードが検索されると、2 番目のノードからの重複キー値を指定したすべてのレコードが、指定したアクセス・パスの順序で検索されます。
  - 2 番目のノードからの、重複キー値を指定したすべてのレコードが検索されると、3 番目のノードからの重複キー値を指定したすべてのレコードが、指定したアクセス・パスの順序で検索されます。
  - ファイル作成時にノード・グループによって定義された最後のノードに到達するまで、これが続きます。
  - 重複キー値を指定したすべてのレコードが、指定したアクセス・パスの順序で最後のノードから検索されると、次の非重複キー値が検索されます。

  したがって、重複キー値を持つ分散ファイルは、分散ファイルの異なるノードを通しては、指定したアクセス・パスの順序では処理されません。
- バッファー付き検索 (*BUFFERED) もしくは保護付きバッファー検索 (*PROTECTED) が使用される場合は、次のようになります。
  - オープン後に分散ファイルで挿入もしくは更新されるレコードは、そのキー値が、最後のレコードがプログラムに戻された後にきた場合でも、レコードの検索中は見えなかった可能性があります。これは、各ノードに、最後のキー別検索要求に基づくその独自のキー位置があるためです。 571 ページの『レコードを挿入、更新、および削除用に検索する例』で、重複キー・レコードを挿入もしくは更新用に検索する方法の例を示します。
  - オープン後に分散ファイルから削除するレコードは、ファイルからレコードを検索する間も見えていた可能性があります。
  - バッファー付き検索と保護付きバッファー検索の唯一の違いは、保護付きバッファー検索では、分散ファイルの中のレコードの削除、挿入、および更新がそのジョブに限定されるという点です。
- 分散ファイルの出力の場合、システムは挿入要求を一度に 1 レコードずつ処理します。分散ファイルのオープン要求が、出力専用の SEQONLY(*YES) 処理用で

ある場合は、SEQONLY(*NO) に変更されます。単一のレコード出力処理では、レコードがファイルに挿入された時点でレコードごとにフィードバックがあります。

レコードを挿入、更新、および削除用に検索する例

図 129 で、バッファー付き検索の最初のキー別検索要求の後の、分散ファイル内の異なるレコード位置を示します。このキー別検察要求で、分散ファイルは各ノードの最初のレコードの位置に付けられています。

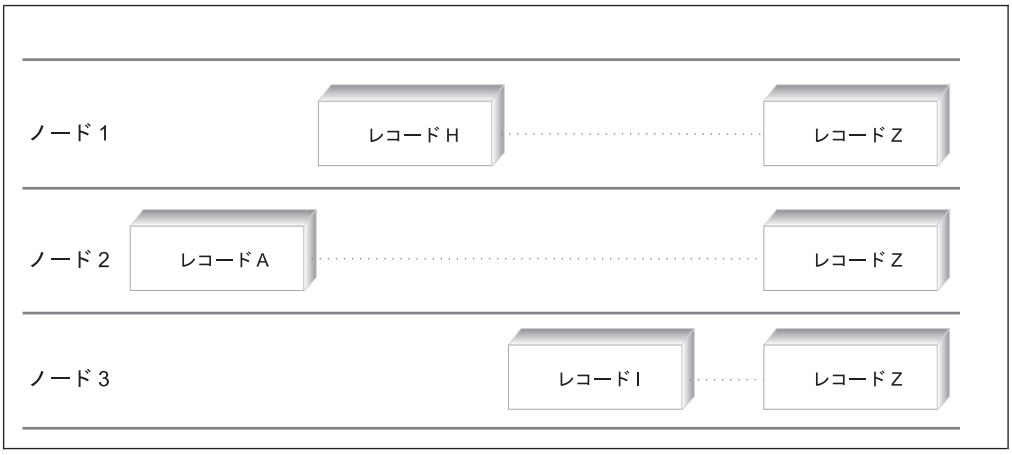


図 129. 分散ファイル内のノード全体での最初の重複レコード・キーの位置

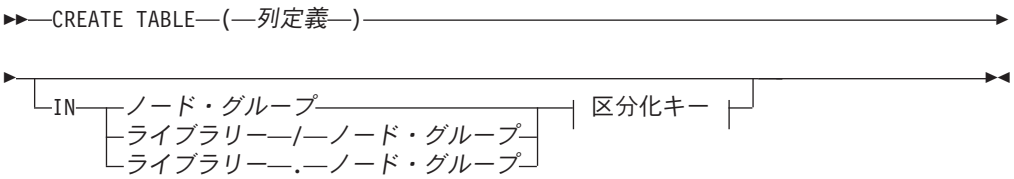
この例では、最初のキー別検索の要求でレコード A がプログラムに戻されています。レコード位置はノードによってまちまちであるため、それに続くキー別次検索の要求によって、ノード 1 のレコード H かノード 3 のレコード I のどちらかより前にあって、ノード 1 で挿入もしくは更新されていたレコードは戻されません。プログラムに戻された最後のレコードの後であっても、特定ノードの現行のキー位置の前にきている、挿入もしくは更新済みレコードは、レコードを読み取る方向を変えない限りプログラムには見えません。

削除されたレコードは、すでに特定のノードに位置が決められ、かつそこから検索されている場合は、プログラムから見える場合もあります。たとえば、ノード 2 のレコード A がプログラムに戻されている場合は、ノード 3 のレコード I がプログラムに戻されます (たとえ、それが、その検索に設定された次のキー別次検索の要求を出す前に削除されている場合であっても)。

非バッファー付き検索 (*CURRENT) が使用されているときは、オープン後に分散ファイルで挿入もしくは更新されたレコードは、ノードにまたがる重複キー値の場合を除き、非分散データベース・ファイルの場合と同じ方法で検索されます。非バッファー付き検索用にオープンされている後に、分散ファイルで挿入もしくは更新されるレコードも、プログラムに戻された最後のレコードの前にそのキー値がきている場合は、見えない可能性があります。分散ファイルへのキー順入力で、ノードにまたがる重複キー値の場合を除き、非分散データベース・ファイル用に検索された場合と同じレコードを検索するように求める場合は、キー付き分散ファイルのオープンを非バッファー付き検索に指定変更する必要があります。

分散データ・ファイル用 SQL ステートメントの追加

分散ファイルを SQL によって作成できるようにするため、SQL CREATE TABLE ステートメントに新しい文節が追加されました。これらの追加部分を以下に示します。



区分化キー:

ILE COBOL プログラムでの SQL コマンドの使用に関する詳細については、『ILE COBOL プログラムへの SQL ステートメントの組み込み』を参照してください。

分散ファイルの処理例

分散ファイルを作成するには、分散ファイルの一部が置かれているシステムごとに以下のことを行う必要があります。

1. ローカル・システムにリレーショナル・データベースのディレクトリー項目、およびファイルの一部が入る他のあらゆるシステムに、1 つのリレーショナル・データベースのディレクトリー項目を追加する必要があります。
2. 分散ファイルが入るライブラリーを作成しなければなりません。

1 次システムには、以下のことを行う必要があります。

1. リレーショナル・データベースのすべての名前が入ったノード・グループを作成する。
2. 物理ファイル用の DDS を定義する。
3. 物理ファイルを作成して、ノード・グループ (NODGRP) および区分化キー (PTNKEY) パラメーターを指定する。
4. 分散物理ファイルの上に論理ファイルを作成すると、分散論理ファイルになります。

たとえば、2 つのシステムがあって、それぞれに分散ファイルの一部を入れたいとします。以下を前提とします。

- 1 次システムを OS400SYS1 と呼び、他のシステムを OS400SYS2 と呼びます。
- 分散ファイルが存在するライブラリーは DISTRIBUTE です。

システム OS400SYS1 にリレーショナル・データベースのディレクトリー項目を作成するために、次のコマンドを入力します。

```
ADDRDBDIRE RDB(OS400SYS1) RMTLOCNAME(*LOCAL)
           TEXT('local database RDB directory entry')
ADDRDBDIRE RDB(OS400SYS2) RMTLOCNAME(AS400SYS2)
           TEXT('remote database RDB directory entry')
```

OS400SYS1 にライブラリー DISTRIBUTE を作成するために、CRTLIB コマンドを入力します。

システム OS400SYS2 にリレーショナル・データベースのディレクトリー項目を作成するために、次のコマンドを入力します。

```
ADDRDBDIRE RDB(OS400SYS2) RMTLOCNAME(*LOCAL)
           TEXT('local database RDB directory entry')
ADDRDBDIRE RDB(OS400SYS1) RMTLOCNAME(AS400SYS1)
           TEXT('remote database RDB directory entry')
```

OS400SYS2 にライブラリー DISTRIBUTE を作成するため、CRTLIB コマンドを入力します。

1 次システムでは、以下を前提とします。

- 分散ファイルのレコードを入れるリレーショナル・データベースを指定するノード・グループの名前は、NODEGROUP。
- 分散物理ファイルの名前は CUSTMAST。

次に、システム OS400SYS1 上にノード・グループを作成するために、次のコマンドを使用します。

```
CRTNODGRP NODGRP(DISTRIBUTE/NODEGROUP) RDB(OS400SYS1 AS400SYS2)
          TEXT('node group for distributed file')
```

物理ファイル作成 (CRTPF) コマンドの DDS (ライブラリー DISTRIBUTE のソース・ファイル QDDSSRC に入っている) を、次に示します。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R CUSTREC
     A            CUSTOMERNO     9S 0       ALIAS(CUSTOMER_NUMBER)
     A            FIRSTNAME     15A         ALIAS(CUSTOMER_FIRST_NAME)
     A            LASTNAME      15A         ALIAS(CUSTOMER_LAST_NAME)
     A            ADDRESS       20A         ALIAS(CUSTOMER_ADDRESS)
     A            ACCOUNTNO      9S 0       ALIAS(CUSTOMER_ACCOUNT_NUMBER)
```

以下のように、DDS フィールド CUSTOMERNO が、分散ファイルの区分化キーとして使用されます。

```
CRTPF FILE(DISTRIBUTE/CUSTMAST)
      SRCFILE(DISTRIBUTE/QDDSSRC) SRCMBR(CUSTMAST)
      NODGRP(DISTRIBUTE/NODEGROUP)
      PTNKEY(CUSTOMERNO)
```

1 次システム上で物理ファイルの作成 (CRTPF) コマンドが完了すると、1 次システムに加えて、ノード・グループの他のすべてのリレーショナル・データベース上にもファイルが作成されます。ファイルが作成された後は、ノード・グループを変更しても分散ファイルには影響しません。

制約付きファイルの処理

データベース物理ファイル (SQL TABLE) のフィールド内のデータは、制約関係を加えることで、特定の値に限定することができます。制約には、以下の 4 つのタイプがあります。
- 参照制約
- 固有制約
- 1 次キー (固有制約の特殊例)
- 検査制約

制約関係を用いて、ファイル間の依存関係を定義することができます。定義した関係は、ファイル内の情報に変更が生じたときにシステムにより強制されます。制約関係を定義すると、処理されるデータの**参照保全**を制御することになります。

検査制約は、データベースの物理ファイル (SQL テーブル) のフィールド (列) に対して行う妥当性検査で、データの保全性を増加させます。

制約付きフィールドで挿入もしくは更新が行われるときは、データはまずそのフィールドに対して行われる妥当性検査を満足させなければ、挿入もしくは更新操作を完了することはできません。制約のすべてが満たされなかった場合は、その入出力要求は実行されずに、メッセージがプログラムに戻され、制約違反があったことを示します。 COBOL の入出力ステートメントの実行中に検査制約の違反があると、ファイル状況 9W が設定されます。参照制約の違反があると、ファイル状況 9R が設定されます。

制約を持てるのは物理ファイルのみですが、制約付きの物理ファイル上に作成された論理ファイルで入出力を行う場合も、制約が実施されます。検査制約は、1 つまたは多数のフィールドに使用することができますが、フィールドとフィールドの比較、またはフィールドとリテラルの比較に使用することができます。

制約の詳細については、 Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベース」カテゴリーの中の『*DB2 Universal Database for AS/400*』セクションを参照してください。

制約事項

以下の制約事項は、ファイルもしくはテーブルに制約を加えるときに適用されます。ファイルには、以下の制約があります。
- データベース物理ファイルでなければならない
- 最大 1 つのメンバーを持つことができる
- プログラム記述ファイルにはなれない
- ソース・ファイルにはなれない
- QTEMP には常駐できない
- オープンできない
- コミットされなかった入出力変更を持てない

参照制約および検査制約には、以下の 4 つの状態があります。
- 定義済みで使用可能
- 定義済みで使用不可
- 設定済みで使用可能
- 設定済みで使用不可

定義済みは、制約の定義がファイルに加えられているが、必ずしもファイルのすべての部分に制約が実施されるわけではないことを意味します。たとえば、ファイルのメンバーが存在していません。

設定済みは、制約の定義がファイルに加えられ、かつファイルのすべての部分に制約が実施されることを意味します。

使用可能は、制約も設定されていれば、検査制約が実施されることを意味します。制約が定義されていれば、制約を実施するファイル・メンバー構造がまだ存在していません。

使用不可は、制約が設定されているか、あるいは定義されているかに関係なく、制約定義がまだ実施されないことを意味します。

参照制約を定義もしくは設定するには、親ファイルと従属ファイルが存在していなければなりません。しかし、親ファイルもしくは従属ファイルにメンバーがない場合は、制約の定義だけが行われます (設定は行われません)。

制約の追加、修正および除去

制約は、以下を使用して追加、修正、または除去することができます。

- SQL
- CL コマンド

制約は、SQL を介し、CREATE TABLE ステートメントを用いて、テーブルの列に加えることができます。すでにテーブルが存在している場合は、ALTER TABLE ステートメントを用いて制約を追加することができます。制約を DROP する場合は、ALTER TABLE ステートメントを使用することもできます。

CL コマンドを使用すると、PF 制約の追加 (ADDPFCST) コマンドを用いて制約の追加または変更を行い、PF 制約の除去 (RMVPFCST) コマンドを用いて制約を除去することができます。

制約が正常に追加または除去されたかの検査

ファイル記述の検索 (QDBRTVFD) API は、ILE COBOL プログラムから使用することができます。ファイル記述の表示 (DSPFD) コマンドは、外部のオペレーティング・システムから使用することができます。システム相互参照ファイルの QUERY (QADBFCST) も、ファイルに制約が加えられたかどうかを示すことができます。

ファイル記述の検索 (QDBRTVFD) API およびファイル記述の表示 (DSPFD) コマンドは両方とも、追加されたすべての制約とともにファイル定義を検索します。

操作の順序

以下に、コミットメント制御が開始されなかった ファイルの操作の順序を示します。

- BEFORE トリガーが発生
- *RESTRICT について参照制約が処理される
- 検査制約が処理される
- 入出力操作が処理される
- AFTER トリガーが発生

- *RESTRICT 以外の参照制約が処理される

以下に、コミットメント制御が開始された ファイルの操作の順序を示します。
- BEFORE トリガーが発生
- *RESTRICT について参照制約が処理される
- 入出力操作が処理される
- AFTER トリガーが発生
- *RESTRICT 以外の参照制約が処理される
- 検査制約が処理される

検査制約付きヌル・フィールドの処理

フィールドがヌル可能で検査制約に使用される場合は、フィールドの値によって制約が影響したり、しなかったりします。

- レコード (行) 内のフィールド (列) がヌル値ではない 場合、フィールドは、検査制約の妥当性検査処理に使用され、有効か検査保留のいずれかの状況を返します。
- フィールド (列) がヌル の場合、検査制約で特にヌル値のテストを行う場合を除き、そのフィールド (列) 値は、制約の妥当性検査に使用されません。すなわち、ヌル・フィールドが検査制約にどう影響したかは分かりません。

制約違反の処理

制約には、**検査保留**の状況を持たせることができます。検査保留の状況とは、レコード (行) 内のデータが制約に違反したことを意味します。 COBOL の入出力ステートメントの実行時は、システムが、制約違反の原因になる、レコードの挿入もしくは更新ができないようにします。挿入もしくは更新が試みられるとファイル状況 9W (検査制約障害) かファイル状況 9R (参照制約障害) になります。しかし、データがすでに存在しているところに制約を追加したり、古いデータを復元すると、制約違反を起こし、したがって、検査保留の状況になる可能性があります。

設定検査制約と使用可能検査制約の違反が生じると (状況は検査保留)、そのファイルからデータを読み取ることはできません。更新のための読み取りが必要な、これらの挿入、更新、または削除操作の場合は、入出力操作は行われません。それ以外では、挿入、更新、および削除操作が行われます。検査制約違反が起きた後で再度ファイルから読み取るには、PF 制約変更 (CHGPFCST) コマンドを用いて、検査制約を使用不可にしなければなりません。

設定参照制約および使用可能参照制約の状況が検査保留になると、次のようになります。
- 従属ファイルに対するファイル入出力は行えない
- 親ファイルに対するファイル入出力 (READ/INSERT) は限定される

制約が使用不可になった後で制約違反の原因を見つけるには、以下のいずれかの方法を使用することができます。

- CHKPND 制約表示 (DSPCPCST) コマンドを用いて、違反の原因となるレコードを検査します。
- PF 制約の処理 (WRKPFCST) コマンドを用いて、検査保留状態の制約を見つけます。

- PF 制約の除去 (RMVPFCST) コマンドを用いて制約を除去し、続いて、 PF 制約の追加 (ADDPFCST) コマンドを用いて制約を戻します。こうすると、違反の原因となっている制約の、最初の 20 レコードがリストされます。

参照制約または検査制約をサポートするデータベース機能

以下のデータベース機能は、参照制約および検査制約をサポートします。
- ジャーナル処理
- コミットメント制御
- 分散データ管理 (DDM) ファイル
- 分散 (マルチ・システム) ファイル

ジャーナル処理

参照制約または検査制約付きファイルは、ジャーナル処理できますが、必要ではありません。検査制約に関連付けられた特殊なジャーナル処理項目はありません。

コミットメント制御

コミットメント制御が活動状態にある場合にファイル入出力機能が従う規則は、コミットメント制御が活動状態にないときに適用される規則と同じです。すなわち、制約付きファイルで入出力を行うときは、制約規則違反が起こりそうな場所での、挿入、更新、または削除は行えません。違反が起こりそうになると、通知メッセージが出されます。入出力操作が正常に完了すれば、COMMIT または ROLLBACK を実行することができます。

分散データ管理 (DDM)

検査制約は、分散データ管理 (DDM) ファイルに対してサポートされます。 V4R2 と V4R2 よりも前のシステム間で DDM が使用されていても、V4R2 システム上に存在している可能性がある検査制約情報は、V4R2 よりも前のシステムには渡されません。

V4R2 と V4R2 よりも前のシステム間で DDM ファイルについての検査制約を伝えようとした場合、以下の操作を行っても、検査制約は伝わらないか失敗します。
- ファイル作成操作またはテーブル作成操作は作動しますが、検査制約は伝わりません
- ファイル定義抽出操作は作動しますが、検査制約は伝わりません
- ALTER TABLE ステートメントは失敗します
- 物理ファイルの変更 (CHGPF) CL コマンドは失敗します

分散ファイル

検査制約は、分散 (多重システム) ファイルに対してサポートされます。 V4R2 と V4R2 よりも前のシステム間で分散ファイルが使用されていても、V4R2 システム上に存在している可能性がある検査制約情報は、V4R2 よりも前のシステムには渡されません。

V4R2 と V4R2 よりも前のシステム間で分散ファイルについての検査制約を伝えようとした場合、以下の操作を行っても失敗します。
- ファイル作成操作またはテーブル作成操作
- PF 制約の追加 (ADDPFCST) CL コマンド

- ALTER TABLE ステートメント
- 物理ファイルの変更 (CHGPF) CL コマンド

第 21 章 トランザクション・ファイルの使用

この章では、ワークステーションおよびプログラム間通信をサポートする ILE COBOL 言語拡張機能について説明します。

TRANSACTION ファイル編成を使用すると、ILE COBOL プログラムで、以下のものと相互に通信を行えるようになります。
- 1 つまたは複数のワークステーション
- リモート・システム上にある 1 つまたは複数のプログラム
- リモート・システム上にある 1 つまたは複数の装置

AS/400 システムでは、リモート・システム上のプログラムや装置 (非同期通信タイプのものなど) との通信が可能です。それらの装置については、「*ICF Programming*」を参照してください。

ILE COBOL TRANSACTION ファイルは、通常、外部的に記述されます。それらのファイルがプログラムによって記述されている場合は、単純な表示のフォーマット設定のみが行えます。プログラムによって記述されたデータベースの使用については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベースおよびファイル・システム」カテゴリーを参照してください。

ILE COBOL TRANSACTION ファイルは、通常、ファイル情報とレコード内のフィールドについての記述が含まれている外部記述ファイルを使用します。そのファイルの中のレコードは、形式 2 の COPY ステートメントを使用することによって、ILE COBOL プログラムの中に記述することができます。形式 2 の COPY ステートメントの詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

パック・データ、2 進データ、および浮動小数点データ (COMP、COMP-1、COMP-2、COMP-3、COMP-4、COMP-5) は、出力データとしてディスプレイ装置に送らないでください。そのようなデータの中にはディスプレイ装置の制御文字が含まれていることがあり、予測不能の結果になる可能性があります。

データ記述仕様を使用してトランザクション・ファイルを定義する

データ記述仕様 (DDS) は、外部記述 TRANSACTION ファイルの記述に使用します。

フィールド記述 (フィールド名や属性など) のほかに、ディスプレイ装置ファイルのデータ記述仕様 (DDS) は、以下のことを行います。
- レコードをディスプレイ装置に配置するためのフォーマット設定を行うために、各フィールドおよび固定情報について、行番号や位置番号の項目を指定する。
- 下線付きおよび強調表示フィールド、反転表示、または明滅カーソルなどのアテンション機能を指定する。
- ディスプレイ装置に入力されたデータの妥当性検査を指定する。

- 新しいデータを表示する場合に、フィールドの消去、オーバーレイ、または保存をいつ行うかなどの、ディスプレイ装置管理機能を制御する。
- タイプ CA または CF として指定されているファンクション・キーに、標識 01～99 を関連付ける。ファンクション・キーが CF と指定されている場合、変更データ・レコードと応答標識の両方がプログラムに戻されます。 ファンクション・キーが CA と指定されている場合、応答標識はプログラムに戻されますが、通常、データ・レコードには、入力専用フィールドのデフォルト値および非表示入出力フィールドの様式に書き込まれる値は入れられます。 CF および CA ファンクション・キーの詳細については、 Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「*データベースおよびファイル・システム*」カテゴリーを参照してください。
- フィールドの値をどのように表示するかを指定するために、編集コード (EDTCDE キーワード) または編集ワード (EDTWRD キーワード) をフィールドに割り当てる。
- サブファイルを指定する。

表示形式データは、ディスプレイ装置を定義または記述します。ディスプレイ装置のレコード様式には、以下の 3 つのタイプのフィールドが含まれています。

- *入力フィールド:* 入力フィールドは、プログラムがレコードを読み込むときに、装置からプログラムに渡されます。入力フィールドはデフォルト値で初期設定することができます。デフォルト値が変更されていない場合、そのデフォルト値がプログラムに渡されます。初期設定されていない入力フィールドはブランクとして表示され、ワークステーションのユーザーはそこにデータを入力できます。
- *出力フィールド:* 出力フィールドは、プログラムがディスプレイ装置にレコードを書き込むときに、プログラムから装置に渡されます。出力フィールドは、プログラムまたは装置ファイル中のレコード様式で指定できます。
- *入出力フィールド:* 入出力フィールドは、入力フィールドにも変更可能な出力フィールドです。入出力フィールドは、プログラムがディスプレイ装置にレコードを書き込むときにはプログラムから 渡され、プログラムがディスプレイ装置からレコードを読み込むときにはプログラムへと 渡されます。出力 / 入力フィールドは、ユーザーがプログラムから表示装置に書き出されるデータを変更または更新する時に使用されます。

データ通信ファイルの詳細については、「*ICF Programming*」を参照してください。外部定義ディスプレイ・ファイルおよび有効なデータ記述仕様 (DDS) キーワードの詳細については、 Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「*データベースおよびファイル・システム*」カテゴリーを参照してください。

図 130 に、ディスプレイ装置ファイルの DDS の例を示します。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
    A* CUSTOMER MASTER INQUIRY FILE ** CUSMINQ
    A*
    A                                      REF(CUSMSTP)  1
    A          R CUSPMT                    TEXT('CUSTOMER PROMPT')
    A                                      CA01(15 'END OF PROGRAM')  2
    A                                    1  3'CUSTOMER MASTER INQUIRY'
    A                                    3  3'CUSTOMER NUMBER'
    A            CUST      R        I  3 20
    A  99                                  ERRMSG('CUSTOMER NUMBER NOT FOUND +  3
    A                                      PRESS RESET, THEN ENTER A VALID NU+
    A                                      MBER' 99)
    A                                    5  3'USE CF1 TO END PROGRAM, USE ENTE+
    A                                       R TO RETURN TO PROMPT SCREEN'
    A          R CUSFLDS                   TEXT('CUSTOMER DISPLAY')
    A                                      CA01(15 'END OF PROGRAM')
    A                                      OVERLAY  4
    A                                    8  3'NAME'
    A            NAME      R             8 11
    A                                    9  3'ADDRESS'
    A            ADDR      R             9 11
    A                                   10  3'CITY'  5
    A            CITY      R            10 11
    A                              6     11  3'STATE'
    A            STATE     R            11 11
    A                                   11 21'ZIP CODE'
    A            ZIP       R            11 31
    A                                   12  3'A/R BALANCE'
    A            ARBAL     R            12 17
```

図 130. ディスプレイ装置ファイルのデータ記述仕様の例

このディスプレイ装置ファイルには、CUSPMT と CUSFLDS という 2 つのレコード様式が含まれています。

1 このファイル中のフィールドの属性は、CUSMSTP フィールド参照ファイルで定義されています。たとえば、ARBAL というフィールドの CUSMSTP では EDTCDE(J) が定義されています。

2 F1 キーは標識 15 と関連付けられており、ユーザーがプログラムを終了するためのものです。

3 ERRMSG キーワードは、このレコード様式を使用しているプログラム中で標識 99 がオンに設定された場合に表示されるエラー・メッセージを指定します。

4 レコード様式 CUSFLDS に対して OVERLAY キーワードが使用されています。これは、CUSFLDS レコードがディスプレイ装置に書き込まれたときに、ディスプレイ装置上の CUSPMT レコードが消去されないようにするためです。

5 'NAME'、'ADDRESS'、および 'CITY' などの定数が、プログラムによって書き込まれるフィールドに記述されます。

6 行と位置の項目によって、フィールドや定数がディスプレイ装置上のどこに書き込まれるかが識別されます。

外部記述トランザクション・ファイルの処理

外部記述 TRANSACTION ファイルを処理する場合、オペレーティング・システムは、データを ILE COBOL プログラムから、そのファイルに指定されている形式に変換して、データを表示します。データが ILE COBOL プログラムに渡される時点で、そのデータは ILE COBOL プログラムで使用される形式に変換されます。

オペレーティング・システムは、装置に対する入出力操作を実行するための装置制御情報を提供します。 ILE COBOL プログラムが装置からの入力レコードを要求すると、オペレーティング・システムはその要求を発行し、データから装置制御情報を除去した後、データをプログラムに渡します。それに加えて、オペレーティング・システムは ILE COBOL プログラムに、レコードに変更が加えられた場合に、どのフィールドに変更が加えられたかを示す標識を渡すことができます。

ILE COBOL プログラムが出力操作を要求すると、オペレーティング・システムに出力レコードが渡されます。オペレーティング・システムは、そのレコードを表示するのに必要な装置制御情報を提供します。また、レコード様式に固定情報が指定されていれば、システムはレコードの表示時にそれも追加します。

レコードが ILE COBOL プログラムに渡されると、フィールドは DDS に指定されている順番に配置されます。フィールドが表示される順番は、DDS 中でフィールドに割り当てられている表示位置 (行番号と位置) に基づいています。そのため、DDS 中で指定されているフィールドの順番と、ディスプレイ装置上でフィールドが表示される順番は必ずしも同じではありません。

トランザクション・ファイルを使用したプログラムの作成

一般には、TRANSACTION ファイルを使用することによって、ディスプレイ装置から 1 レコードを読み込んだり、ディスプレイ装置に 1 レコードを書き込んだりします。 ILE COBOL プログラムで TRANSACTION ファイルを使用するには、次のようにする必要があります。

- 環境部 (ENVIRONMENT DIVISION) の FILE-CONTROL 段落のファイル制御記入項目を使用してファイルの名前を指定する。
- データ部 (DATA DIVISION) のファイル記述項目を使用してファイルを記述する。
- トランザクション処理をサポートするための、PROCEDURE DIVISION のステートメントに対する拡張機能を使用する。

注: 拡張 ACCEPT/DISPLAY ステートメントと TRANSACTION ファイルを同じプログラムの中で使うことはお勧めできません。拡張 ACCEPT/DISPLAY ステートメントと TRANSACTION ファイルが同じプログラムの中で使用されている場合、拡張 ACCEPT/DISPLAY ステートメントの実行時には、TRANSACTION ファイルをクローズしておいてください。 TRANSACTION ファイルがオープンしているときに拡張 ACCEPT/DISPLAY ステートメントを実行すると、結果は予測不能になります。重大エラーが生じたり、ワークステーション上のデータが重ね書きされたり混ざり合ってしまったりする危険性があります。

トランザクション・ファイルの名前の指定

ILE COBOL プログラムで TRANSACTION ファイルを使用するには、FILE-CONTROL 段落のファイル制御項目を使用してそのファイルの名前を指定する必要があります。 FILE-CONTROL 段落の説明については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

TRANSACTION ファイルの名前は、FILE-CONTROL 段落の中で以下のようにして指定します。

```
FILE-CONTROL.
    SELECT transaction-file-name
        ASSIGN TO WORKSTATION-display_file_name
        ORGANIZATION IS TRANSACTION
        ACCESS MODE IS SEQUENTIAL
        CONTROL AREA IS control-area-data-item.
```

ファイルを選択するには SELECT 文節を使用します。そのファイルは、DATA DIVISION の FD 項目で指定されているものでなければなりません。

TRANSACTION ファイルをディスプレイ・ファイルまたは ICF ファイルと関連付けるには、ASSIGN 文節を使用します。 TRANSACTION ファイルを使用するには、ASSIGN 文節に装置タイプ WORKSTATION を指定しなければなりません。この TRANSACTION ファイルのために別個の標識域を使用する場合は、ASSIGN 文節に -SI 属性を含める必要があります。別個の標識域の使い方については 595 ページの『トランザクション・ファイルでの標識の使用』を参照してください。

TRANSACTION ファイルを使用するためには、ファイル制御記入項目の中に ORGANIZATION IS TRANSACTION を指定してください。この文節は、ILE COBOL プログラムに対して、ワークステーション・ユーザーや別のシステムとの対話が行われることを知らせるものです。

TRANSACTION ファイルには順次アクセスを行います。 TRANSACTION ファイルへのアクセス方法を ILE COBOL プログラムに知らせるには、ファイル制御記入項目に ACCESS MODE 文節を使用します。 TRANSACTION ファイルへの順次読み込みまたは書き込みを行うには、ACCESS MODE IS SEQUENTIAL と指定します。 ACCESS MODE 文節を指定しなければ、順次アクセスと見なされます。

TRANSACTION ファイルを参照する入出力要求の状況をフィードバックしたい場合は、FILE STATUS 文節を使用してファイル制御項目の中に状況キー・データ項目を定義します。 FILE STATUS 文節を指定すると、システムは、明示的にまたは暗黙のうちに TRANSACTION ファイルを参照する各入出力要求の後で、状況キー・データ項目に値を移動します。この値は、入出力ステートメントの実行状況を示します。

TRANSACTION ファイルの入出力操作を制御するのに使用される特定の装置依存情報やシステム依存情報を取得するには、CONTROL-AREA 文節を使用して制御域データ項目を指定します。 CONTROL-AREA 文節で指定されるデータ項目は、以下の形式で LINKAGE SECTION または WORKING-STORAGE SECTION の中で定義します。

```
01  control-area-data-item.
    05  function-key          PIC X(2).
    05  device-name           PIC X(10).
    05  record-format         PIC X(10).
```

制御域の長さは、2 バイトか 12 バイトか 22 バイトです。したがって、必要とする情報の種類に応じて、指定できる要素は、最初の 05 レベル要素、最初の 2 つの 05 レベル要素、または 3 つの 05 要素すべてのいずれかになります。

制御域データ項目を使用することによって、以下のものを指定することができます。

- トランザクションを開始するためにオペレーターが押したファンクション・キー。
- 使用されているプログラム装置の名前。
- 最後の入出力ステートメントで参照された DDS レコード様式の名前。

トランザクション・ファイルの記述

ILE COBOL プログラムの中で TRANSACTION ファイルを使用するには、DATA DIVISION のファイル記述項目を使用してそのファイルを記述する必要があります。ファイル記述項目の説明については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。 TRANSACTION ファイルを記述するには、形式 6 のファイル記述項目を使用します。

DATA DIVISION の中で TRANSACTION ファイルを記述するファイル記述項目は、以下のようなものになります。

```
FD  CUST-DISPLAY.
01  DISP-REC.
    COPY DDS-ALL-FORMATS OF CUSMINQ.
```

ILE COBOL において TRANSACTION ファイルは、通常、外部的に記述されます。使用する TRANSACTION ファイルの DDS を作成してください。 DDS の作成方法については 579 ページの『データ記述仕様を使用してトランザクション・ファイルを定義する』を参照してください。 その後、TRANSACTION ファイルを作成します。

TRANSACTION ファイルの DDS と TRANSACTION ファイルを作成したなら、形式 2 の COPY ステートメントを使用して、TRANSACTION ファイルのデータ・レコードのレイアウトを記述してください。 ILE COBOL プログラムをコンパイルすると、形式 2 の COPY により、TRANSACTION ファイルを記述するためのデータ部 (DATA DIVISION) ステートメントが作成されます。すべての様式に対して 1 つのストレージ域を生成するためには、形式 2 の COPY ステートメントの DDS-ALL-FORMATS オプションを使用してください。

トランザクション・ファイルの処理

PROCEDURE DIVISION ステートメントのうち、特に ILE COBOL プログラムで TRANSACTION ファイルを処理するための拡張機能があるものをすべて以下に示します。これらの各ステートメントに関する詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

- ACCEPT ステートメント - 形式 6
- ACQUIRE ステートメント
- CLOSE ステートメント - 形式 1
- DROP ステートメント
- OPEN ステートメント - 形式 3

- READ ステートメント - 形式 4 (非サブファイル)
- WRITE ステートメント - 形式 4 (非サブファイル)

トランザクション・ファイルのオープン

PROCEDURE DIVISION で TRANSACTION ファイルを処理するためには、まずそのファイルをオープンする必要があります。 TRANSACTION ファイルをオープンするには、形式 3 の OPEN ステートメントを使用します。 TRANSACTION ファイルは、I-O モードでオープンしてください。

```
OPEN I-O file-name.
```

プログラム装置の獲得

TRANSACTION ファイル用のプログラム装置を獲得する必要があります。それを獲得したなら、入出力操作のためにそのプログラム装置を使用することができます。プログラム装置は、暗黙のうちに獲得するか、または明示的に獲得します。

TRANSACTION ファイルをオープンすると、暗黙のうちに 1 つのプログラム装置を獲得します。ファイルが ICF ファイルの場合、暗黙のうちに獲得される単一のプログラム装置は、ICF ファイルの作成 (CRTICFF) コマンドの ACQPGMDEV パラメーターによって決定されます。ファイルがディスプレイ・ファイルの場合、暗黙のうちに獲得される単一のプログラム装置は、CRTDSPF コマンドの DEV パラメーターの中の最初の項目によって決定されます。それ以外のプログラム装置は、明示的に獲得しなければなりません。

プログラム装置を明示的に獲得するには、ACQUIRE ステートメントを使用します。 ICF ファイルの場合は、ファイルがオープンされる前に ADDICFDEVE または OVRICFDEVE の CL コマンドを使用して、その装置をそのファイルに対して定義しておく必要があります。ディスプレイ・ファイルの場合、このような必要はありません。すなわち、ACQUIRE ステートメントで指定する装置は、CRTDSPF コマンド、CHGDSPF コマンド、または OVRDSPF コマンドの DEV パラメーターに指定する必要はありません。 しかし、ディスプレイ・ファイルを作成する場合には、獲得される装置の数 (デフォルトは 1) を指定する必要があります。 ディスプレイ・ファイルの場合、プログラム装置の名前はディスプレイ装置に一致していなければなりません。

```
ACQUIRE program-device-name FOR transaction-file-name.
```

トランザクション・ファイルへの書き込み

TRANSACTION ファイルをオープンして、そのためのプログラム装置を獲得すれば、それに対する入出力操作を実行するための準備は完了です。

TRANSACTION ファイルに対して実行する最初の入出力操作は、多くの場合、ディスプレイ装置にレコードを書き出すことです。そのレコードは、ユーザーに対して応答や何らかのデータを入力するよう求めるプロンプトとして使用されます。

TRANSACTION ファイルに論理レコードを書き出すには、形式 4 の WRITE ステートメントを使用します。 WRITE ステートメントは、次のように簡単にコーディングできます。

```
WRITE record-name FORMAT IS format-name.
```

TRANSACTION ファイルに関して、様式の異なる複数のデータ・レコードを活動状態にしたい場合があります。このような場合に TRANSACTION ファイルに書き出す出力データ・レコードの形式を指定するには、形式 4 の WRITE ステートメントの FORMAT 句を使用する必要があります。

TRANSACTION ファイル用に複数のプログラム装置を明示的に獲得した場合、出力レコードの送り先のプログラム装置を指定するには、形式 4 の WRITE ステートメントの TERMINAL 句を使う必要があります。

形式 4 の WRITE ステートメントの STARTING 句および ROLLING 句を指定することによって、WRITE ステートメントが出力レコードを書き込む、ディスプレイ装置上の行番号を制御することができます。 STARTING 句は、可変レコード開始行キーワードを使用するレコード様式の開始行番号を指定します。 ROLLING 句を使うと、ワークステーション画面に表示された行を移動させることができます。画面上のすべての行または一部の行を、上下に送ることができます。

```
WRITE record-name FORMAT IS format-name
                  TERMINAL IS program-device-name
                  STARTING AT LINE start-line-no
                  AFTER ROLLING LINES first-line-no THRU last-line-no
                                DOWN no-of-lines LINES
END-WRITE.
```

トランザクション・ファイルからの読み取り

TRANSACTION ファイルから論理レコードを読み込むには、形式 4 の READ ステートメントを使用します。 READ ステートメントが実行されるときにデータが使用可能である場合には、レコード域の中にデータが戻されます。レコード様式とプログラム装置の名前が、それぞれ I-O-FEEDBACK 域および CONTROL-AREA 域に戻されます。

READ ステートメントを使用する前に、TRANSACTION ファイルのために少なくとも 1 つのプログラム装置を獲得しておかなければなりません。 READ ステートメントが実行される時にプログラム装置が獲得されていない場合は、ファイル状況が 92 に設定されることによって論理エラーが報告されます。

READ ステートメントは、次のような最も簡単な形で使用できます。

```
READ record-name RECORD.
```

プログラム装置を 1 つだけ獲得している場合にこの簡単な形の READ ステートメントを使用すると、データが利用可能になるまで常に待機することになります。ジョブが制御付き取り消しを受け取っても、あるいはディスプレイ・ファイルまたは ICF ファイルの WAITRCD パラメーターに待ち時間が指定されていたとしても、プログラムが READ ステートメントから制御を再び得ることはできなくなります。

複数のプログラム装置を獲得した場合、この簡単な形の READ ステートメントは、最初にデータが利用可能になって送信勧誘されたプログラム装置からデータを受け取ります。複数のプログラム装置を獲得した場合、送信勧誘された装置がなく待ち時間が指定されていないか、またはジョブの制御付き取り消しが発生したか、または指定された待ち時間が経過すると、この簡単な形の READ ステートメントはデータを戻すことなく完了する場合があります。

READ 操作の実行方法の詳細な説明については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の『READ ステートメント』の部分を参照してください。

複数のプログラム装置を獲得した場合、READ ステートメントの TERMINAL 句の中で、どのプログラム装置からデータを読み取るかを明示的に指定することができます。

特定の様式でデータを受け取りたい場合には、その様式を READ ステートメントの FORMAT 句に指定することができます。使用可能なデータが、要求されたレコード様式に合致しない場合、ファイル状況 9K が設定されます。

次に示すのは、TERMINAL 句および FORMAT 句を指定した READ ステートメントの例です。

```
READ record-name RECORD
     FORMAT IS record-format
END-READ
READ record-name RECORD
     TERMINAL IS program-device-name
END-READ
READ record-name RECORD
     FORMAT IS record-format
     TERMINAL IS program-device-name
END-READ
```

この READ ステートメントが実行されると、次の条件が発生する場合があります。

1. データはただちに利用可能で、AT END 条件が存在しない。 AT END 条件は、送信勧誘されたプログラム装置がなく、待ち時間が指定されていない場合に発生します。
2. データがただちに利用可能ではない。
3. AT END 条件が存在している。

NO DATA 句、AT END 句、または NOT AT END 句を指定することによって、READ ステートメントの実行結果の条件に基づいて、READ ステートメントから、ILE COBOL プログラム内のさまざまなステートメントに制御を移すことができます。

READ ステートメントが正常に完了した場合に一群のステートメントを実行するには、READ ステートメントの NOT AT END 句を指定します。

データがただちに利用可能でない場合に一群のステートメントを実行するには、READ ステートメントの NO DATA 句を指定します。 NO DATA 句によって、READ ステートメントはデータが使用可能になるのを待つ必要がなくなります。

AT END 条件が存在する場合に一群のステートメントを実行するには、READ ステートメントの AT END 句を指定します。

次に示すのは、READ ステートメントに NO DATA、NOT AT END、および AT END 句を指定した例です。

```
READ record-name RECORD
     TERMINAL IS program-device-name
     NO DATA imperative-statement-1
END-READ
```

```
READ record-name RECORD
     TERMINAL IS program-device-name
     AT END imperative-statement-2
     NOT AT END imperative-statement-3
END-READ
```

プログラム装置のドロップ

TRANSACTION ファイル用に獲得したプログラム装置を使用し終えた時点で、それをドロップする必要があります。プログラム装置をドロップするとは、その装置が TRANSACTION ファイルを使用した入出力操作では使えなくなるということです。プログラム装置をドロップすると、その装置は他のアプリケーションから使用できるようになります。 プログラム装置は、暗黙のうちにドロップしたり明示的にドロップしたりすることができます。

TRANSACTION ファイルをクローズすると、そのファイルに付加されていたすべてのプログラム装置も暗黙のうちにドロップされます。

DROP ステートメントで指定することによって、プログラム装置を明示的にドロップすることができます。ドロップされた装置は、必要に応じてそれを再獲得することができます。

```
DROP program-device-name FROM transaction-file-name.
```

TRANSACTION ファイルのクローズ

TRANSACTION ファイルの使用を終了したならば、それをクローズしてください。TRANSACTION ファイルをクローズするには、形式 1 の CLOSE ステートメントを使用します。 ファイルをクローズすると、そのファイルは、それを再びオープンするまで処理できなくなります。

```
CLOSE transaction-file-name.
```

トランザクション・ファイルを使用した基本的な照会プログラムの例

図 131 に、ILE COBOL TRANSACTION ファイルを使用した基本的な照会プログラムに関連する DDS を示します。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* CUSTOMER MASTER INQUIRY FILE ** CUSMINQ
     A*
     A                                       REF(CUSMSTP)
     A          R CUSPMT                     TEXT('CUSTOMER PROMPT')
     A                                       CA01(15 'END OF PROGRAM')
     A                                     1  3'CUSTOMER MASTER INQUIRY'
     A                                     3  3'CUSTOMER NUMBER'
     A            CUST      R        I  3 20
     A  99                                   ERRMSG('CUSTOMER NUMBER NOT FOUND +
     A                                       PRESS RESET, THEN ENTER A VALID NU+
     A                                       MBER' 99)
     A  98                                   ERRMSG('EOF CONDITION IN READ, +
     A                                       PROGRAM ENDED' 98)
     A                                     5  3'USE F1 TO END PROGRAM, USE ENTE+
     A                                       R TO RETURN TO PROMPT SCREEN'
     A          R CUSFLDS                    TEXT('CUSTOMER DISPLAY')
     A                                       CA01(15 'END OF PROGRAM')
     A                                       OVERLAY
     A                                     8  3'NAME'
     A            NAME      R           8 11
     A                                     9  3'ADDRESS'
     A            ADDR      R           9 11
     A                                    10  3'CITY'
     A            CITY      R          10 11
     A                                    11  3'STATE'
     A            STATE     R          11 11
     A                                    11 21'ZIP CODE'
     A            ZIP       R          11 31
     A                                    12  3'A/R BALANCE'
     A            ARBAL     R          12 17
```

図 131. 単一のディスプレイ装置を使用した TRANSACTION 照会プログラムの例

このプログラムが使うディスプレイ装置ファイル (CUSMINQ) のデータ記述仕様 (DDS) は、2 つのレコード様式 (CUSPMT および CUSFLDS) を記述します。

CUSPMT レコード様式には、ディスプレイ装置を識別する 'CUSTOMER MASTER INQUIRY' (得意先マスター照会) という固定情報が含まれています。また、'CUSTOMER NUMBER' (得意先番号) というプロンプトと、得意先番号を入力する入力フィールド (CUST) も含まれています。ディスプレイ装置において得意先番号を入力する CUST 入力フィールドの下に、5 つの下線が表示されます。以下のエラー・メッセージも、このレコード様式に含まれています。

```
Customer number not found
```

このメッセージは、プログラムによって、標識 99 が **ON** に設定されると表示されます。さらに、このレコード様式では、プログラムを終了させるために押すことのできるファンクション・キーを定義しています。ファンクション・キー F1 を押すと、ILE COBOL プログラム中で標識 15 が **ON** に設定されます。次いでこの標識が、プログラムを終了させるために使用されます。

CUSFLDS レコード様式には、次の固定情報が含まれています。

- NAME
- ADDRESS
- CITY
- STATE
- ZIP CODE

- A/R BALANCE

これらの定数は、プログラムから書き込まれるフィールドを識別します。このレコード様式では、これらの定数に対応するフィールドも記述されています。それらのフィールドにはプログラムがデータを入れるため、すべて出力フィールドとして記述されています (位置 38 がブランク)。したがって、これらのフィールドにはデータを記入できません。別の得意先番号を入力するには、このレコードに応答して実行キーを押します。 CUSFLDS レコードは CUSPMT レコードに OVERLAY していることに注意してください。そのため、CUSFLDS レコードがディスプレイ装置に書き出されるとき、CUSPMT レコードはディスプレイ装置上にそのまま残ります。

ディスプレイ装置用に固定情報、フィールド、および属性を記述することに加えて、レコード様式は、定数やフィールドが表示される水平位置と行番号も定義します。

注: フィールド属性は、ディスプレイ・ファイルの DDS 中で定義されるのではなく、フィールド参照の目的で使用される物理ファイル (CUSMSTP) 中で定義されます。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
    A* THIS IS THE CUSTOMER MASTER FILE ** CUSMSTP
    A
    A
    A                                     UNIQUE
    A          R CUSMST                   TEXT('CUSTOMER MASTER RECORD')
    A            CUST           5         TEXT('CUSTOMER NUMBER')
    A            NAME          25         TEXT('CUSTOMER NAME')
    A            ADDR          20         TEXT('CUSTOMER ADDRESS')
    A            CITY          20         TEXT('CUSTOMER CITY')
    A            STATE          2         TEXT('STATE')
    A            ZIP            5 00      TEXT('ZIP CODE')
    A            SRHCOD         6         TEXT('CUSTOMER NUMBER SEARCH CODE')
    A            CUSTYP         1 00      TEXT('CUSTOMER TYPE 1=GOV 2=SCH +
    A                                     3=BUS 4=PVT 5=OT')
    A            ARBAL          8 02      TEXT('ACCOUNTS REC. BALANCE')
    A            ORDBAL         8 02      TEXT('A/R AMT. IN ORDER FILE')
    A            LSTAMT         8 02      TEXT('LAST AMT. PAID IN A/R')
    A            LSTDAT         6 00      TEXT('LAST DATE PAID IN A/R')
    A            CRDLMT         8 02      TEXT('CUSTOMER CREDIT LIMIT')
    A            SLSYR         10 02      TEXT('CUSTOMER SALES THIS YEAR')
    A            SLSLYR        10 02      TEXT('CUSTOMER SALES LAST YEAR')
                 K CUST
```

図 132. レコード様式 CUSMST のデータ記述仕様

このプログラムが使うデータベース・ファイルのデータ記述仕様 (DDS) は、1 つのレコード様式 CUSMST を記述しています。レコード様式中の各フィールドが記述されており、CUST ファイルがレコード様式のキー・フィールドとして指定されています。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/INQUIRY         ISERIES1   06/02/15 14:57:34        ページ    2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     INQUIRY.
          000300*    SAMPLE TRANSACTION INQUIRY PROGRAM USING 1 DISPLAY DEVICE
          000400
    3     000500 ENVIRONMENT DIVISION.
    4     000600   CONFIGURATION SECTION.
    5     000700 SOURCE-COMPUTER. IBM-ISERIES.
    6     000800 OBJECT-COMPUTER. IBM-ISERIES.
    7     000900   INPUT-OUTPUT SECTION.
    8     001000   FILE-CONTROL.
    9     001100     SELECT CUST-DISPLAY
   10     001200         ASSIGN TO WORKSTATION-CUSMINQ
   11     001300       ORGANIZATION IS TRANSACTION
   12     001400         CONTROL-AREA IS WS-CONTROL.
   13     001500     SELECT CUST-MASTER
   14     001600         ASSIGN TO DATABASE-CUSMSTP
   15     001700         ORGANIZATION IS INDEXED
   16     001800         ACCESS IS RANDOM
   17     001900             RECORD KEY IS CUST OF CUSMST
   18     002000         FILE STATUS IS CM-STATUS.
          002100
   19     002200 DATA DIVISION.
   20     002300 FILE SECTION.
   21     002400 FD  CUST-DISPLAY.
   22     002500 01  DISP-REC.
          002600     COPY DDS-ALL-FORMATS OF CUSMINQ.
   23    +000001       05  CUSMINQ-RECORD PIC X(80).                                    <-ALL-FMTS
         +000002*  INPUT FORMAT:CUSPMT     FROM FILE CUSMINQ    OF LIBRARY CBLGUIDE     <-ALL-FMTS
         +000003*                          CUSTOMER PROMPT                              <-ALL-FMTS
   24    +000004       05  CUSPMT-I      REDEFINES CUSMINQ-RECORD.                     <-ALL-FMTS
   25    +000005           06 CUSPMT-I-INDIC.                                          <-ALL-FMTS
   26    +000006                07 IN15            PIC 1  INDIC 15.                   <-ALL-FMTS
         +000007*                         END OF PROGRAM                               <-ALL-FMTS
   27    +000008                07 IN99            PIC 1  INDIC 99.                   <-ALL-FMTS
         +000009*                         CUSTOMER NUMBER NOT FOUND PRESS RESET, THEN ENT <-ALL-FMTS
   28    +000010                07 IN98            PIC 1  INDIC 98.                   <-ALL-FMTS
         +000011*                         EOF CONDITION IN READ, PROGRAM ENDED         <-ALL-FMTS
   29    +000012           06 CUST                 PIC X(5).                          <-ALL-FMTS
         +000013*                    CUSTOMER NUMBER                                   <-ALL-FMTS
         +000014* OUTPUT FORMAT:CUSPMT     FROM FILE CUSMINQ    OF LIBRARY CBLGUIDE     <-ALL-FMTS
         +000015*                          CUSTOMER PROMPT                              <-ALL-FMTS
   30    +000016       05  CUSPMT-O      REDEFINES CUSMINQ-RECORD.                     <-ALL-FMTS
   31    +000017           06 CUSPMT-O-INDIC.                                          <-ALL-FMTS
   32    +000018                07 IN99            PIC 1  INDIC 99.                   <-ALL-FMTS
         +000019*                         CUSTOMER NUMBER NOT FOUND PRESS RESET, THEN ENT <-ALL-FMTS
   33    +000020                07 IN98            PIC 1  INDIC 98.                   <-ALL-FMTS
         +000021*                         EOF CONDITION IN READ, PROGRAM ENDED         <-ALL-FMTS
         +000022*  INPUT FORMAT:CUSFLDS     FROM FILE CUSMINQ    OF LIBRARY CBLGUIDE     <-ALL-FMTS
         +000023*                          CUSTOMER DISPLAY                             <-ALL-FMTS
   34    +000024       05  CUSFLDS-I     REDEFINES CUSMINQ-RECORD.                     <-ALL-FMTS
   35    +000025           06 CUSFLDS-I-INDIC.                                         <-ALL-FMTS
   36    +000026                07 IN15            PIC 1  INDIC 15.                   <-ALL-FMTS
         +000027*                         END OF PROGRAM                               <-ALL-FMTS
```

図 133. 単一のディスプレイ装置を使用した TRANSACTION 照会プログラムのソース・リスト (1/3)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/INQUIRY         ISERIES1   06/02/15 14:57:34        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
        +000028* OUTPUT FORMAT:CUSFLDS    FROM FILE CUSMINQ    OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000029*                          CUSTOMER DISPLAY                                  <-ALL-FMTS
  37    +000030       05  CUSFLDS-O     REDEFINES CUSMINQ-RECORD.                         <-ALL-FMTS
  38    +000031           06 NAME                  PIC X(25).                             <-ALL-FMTS
        +000032*                 CUSTOMER NAME                                             <-ALL-FMTS
  39    +000033           06 ADDR                  PIC X(20).                             <-ALL-FMTS
        +000034*                 CUSTOMER ADDRESS                                          <-ALL-FMTS
  40    +000035           06 CITY                  PIC X(20).                             <-ALL-FMTS
        +000036*                 CUSTOMER CITY                                             <-ALL-FMTS
  41    +000037           06 STATE                 PIC X(2).                              <-ALL-FMTS
        +000038*                 STATE                                                     <-ALL-FMTS
  42    +000039           06 ZIP                   PIC S9(5).                             <-ALL-FMTS
        +000040*                 ZIP CODE                                                  <-ALL-FMTS
  43    +000041           06 ARBAL                 PIC S9(6)V9(2).                        <-ALL-FMTS
        +000042*                 ACCOUNTS REC. BALANCE                                     <-ALL-FMTS
         002700
  44     002800 FD  CUST-MASTER.
  45     002900 01  CUST-REC.
         003000     COPY DDS-CUSMST OF CUSMSTP.
        +000001*    I-O FORMAT:CUSMST     FROM FILE CUSMSTP    OF LIBRARY CBLGUIDE          CUSMST
        +000002*                          CUSTOMER MASTER RECORD                            CUSMST
        +000003*          USER SUPPLIED KEY BY RECORD KEY CLAUSE                            CUSMST
  46    +000004       05  CUSMST.                                                          CUSMST
  47    +000005           06 CUST                  PIC X(5).                              CUSMST
        +000006*                 CUSTOMER NUMBER                                           CUSMST
  48    +000007           06 NAME                  PIC X(25).                             CUSMST
        +000008*                 CUSTOMER NAME                                             CUSMST
  49    +000009           06 ADDR                  PIC X(20).                             CUSMST
        +000010*                 CUSTOMER ADDRESS                                          CUSMST
  50    +000011           06 CITY                  PIC X(20).                             CUSMST
        +000012*                 CUSTOMER CITY                                             CUSMST
  51    +000013           06 STATE                 PIC X(2).                              CUSMST
        +000014*                 STATE                                                     CUSMST
  52    +000015           06 ZIP                   PIC S9(5)         COMP-3.              CUSMST
        +000016*                 ZIP CODE                                                  CUSMST
  53    +000017           06 SRHCOD                PIC X(6).                              CUSMST
        +000018*                 CUSTOMER NUMBER SEARCH CODE                               CUSMST
  54    +000019           06 CUSTYP                PIC S9(1)         COMP-3.              CUSMST
        +000020*                 CUSTOMER TYPE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT                CUSMST
  55    +000021           06 ARBAL                 PIC S9(6)V9(2)    COMP-3.              CUSMST
        +000022*                 ACCOUNTS REC. BALANCE                                     CUSMST
  56    +000023           06 ORDBAL                PIC S9(6)V9(2)    COMP-3.              CUSMST
        +000024*                 A/R AMT. IN ORDER FILE                                    CUSMST
  57    +000025           06 LSTAMT                PIC S9(6)V9(2)    COMP-3.              CUSMST
        +000026*                 LAST AMT. PAID IN A/R                                     CUSMST
  58    +000027           06 LSTDAT                PIC S9(6)         COMP-3.              CUSMST
        +000028*                 LAST DATE PAID IN A/R                                     CUSMST
  59    +000029           06 CRDLMT                PIC S9(6)V9(2)    COMP-3.              CUSMST
        +000030*                 CUSTOMER CREDIT LIMIT                                     CUSMST
  60    +000031           06 SLSYR                 PIC S9(8)V9(2)    COMP-3.              CUSMST
        +000032*                 CUSTOMER SALES THIS YEAR                                  CUSMST
  61    +000033           06 SLSLYR                PIC S9(8)V9(2)    COMP-3.              CUSMST
        +000034*                 CUSTOMER SALES LAST YEAR                                  CUSMST
         003100
  62     003200 WORKING-STORAGE SECTION.
```

図 133. 単一のディスプレイ装置を使用した TRANSACTION 照会プログラムのソース・リスト (2/3)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/INQUIRY        ISERIES1   06/02/15 14:57:34       ページ    4
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   63     003300 01  ONE                              PIC 1 VALUE B"1".
   64     003400 01  CM-STATUS                        PIC X(2).
   65     003500 01  WS-CONTROL.
   66     003600     02  WS-IND                       PIC X(2).
   67     003700     02  WS-FORMAT                    PIC X(10).
          003800
   68     003900 PROCEDURE DIVISION.
   69     004000 DECLARATIVES.
          004100 DISPLAY-ERR-SECTION SECTION.
          004200     USE AFTER STANDARD EXCEPTION PROCEDURE ON CUST-DISPLAY.
          004300 DISPLAY-ERR-PARAGRAPH.
   70     004400     MOVE ONE TO IN98 OF CUSPMT-O
   71     004500     WRITE DISP-REC FORMAT IS "CUSPMT"
          004600     END-WRITE
   72     004700     CLOSE CUST-MASTER
          004800           CUST-DISPLAY.
   73     004900     STOP RUN.
          005000 END DECLARATIVES.
          005100
          005200 MAIN-PROGRAM SECTION.
          005300 MAINLINE.
   74     005400     OPEN INPUT CUST-MASTER
          005500          I-O CUST-DISPLAY.
          005600
   75     005700     MOVE ZERO TO IN99 OF CUSPMT-O
   76     005800     WRITE DISP-REC FORMAT IS "CUSPMT"  1
          005900     END-WRITE
   77     006000     READ CUST-DISPLAY RECORD
          006100     END-READ
          006200
   78     006300     PERFORM UNTIL IN15 OF CUSPMT-I IS EQUAL TO ONE
          006400
   79     006500         MOVE CUST OF CUSPMT-I TO CUST OF CUSMST
   80     006600         READ CUST-MASTER RECORD  2
          006700             INVALID KEY  3
   81     006800                 MOVE ONE TO IN99 OF CUSPMT-O
   82     006900                 WRITE DISP-REC FORMAT IS "CUSPMT"
          007000                 END-WRITE
   83     007100                 READ CUST-DISPLAY RECORD
          007200                 END-READ
          007300             NOT INVALID KEY
   84     007400                 MOVE CORRESPONDING CUSMST TO CUSFLDS-O
                 *** CORRESPONDING items for statement 84:
                 ***     NAME
                 ***     ADDR
                 ***     CITY
                 ***     STATE
                 ***     ZIP
                 ***     ARBAL
                 *** End of CORRESPONDING items for statement 84
   85     007500                 WRITE DISP-REC FORMAT IS "CUSFLDS"
          007600                 END-WRITE
   86     007700                 READ CUST-DISPLAY RECORD
          007800                 END-READ
   87     007900                 IF  IN15 OF CUSPMT-I IS NOT EQUAL TO ONE
   88     008000                     MOVE ZERO TO IN99 OF CUSPMT-O
   89     008100                     WRITE DISP-REC FORMAT IS "CUSPMT"
          008200                     END-WRITE
   90     008300                     READ CUST-DISPLAY RECORD
          008400                     END-READ
          008500                 END-IF
          008600         END-READ
          008700
          008800     END-PERFORM
          008900
   91     009000     CLOSE CUST-MASTER
          009100           CUST-DISPLAY.
   92     009200     GOBACK.
                         * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 133. 単一のディスプレイ装置を使用した TRANSACTION 照会プログラムのソース・リスト (3/3)

このプログラム例のソース・リスト全体がこの図に示されています。特に、FILE-CONTROL と FD 項目、および形式 2 の COPY ステートメントによって生成されたデータ構造に注目してください。

1 の WRITE 操作では、CUSPMT 様式をディスプレイ装置に書き込みます。このレコードは、得意先番号の入力を要求するものです。得意先番号を入力して実行キーを押すと、次の READ 操作によってそのレコードがプログラムの中に読み戻されます。

2 の READ 操作では、得意先番号 (CUST) フィールドを使用して、CUSMSTP ファイルから対応する CUSMST レコードを取り出します。 CUSMSTP ファイルの中でレコードが見つからなければ、**3** の INVALID KEY 命令ステートメントが実行されます。標識 99 がオンに設定され、

```
Customer number not found
```

というメッセージが様式書き込み時に表示されます。メッセージは、そのファイルの DDS の標識 99 によって条件付けられています。 このメッセージを受け取ると、キーボードはロックされます。このメッセージに応答してキーボードをアンロックするには、リセット・キーを押さなければなりません。その後、別の得意先番号を入力できるようになります。

READ 操作によって CUSMSTP ファイルからレコードが取り出されると、WRITE 操作によって CUSFLDS レコードがディスプレイ・ワークステーションに書き込まれます。このレコードには得意先の名前 (NAME)、所在地 (ADDRESS)、および売掛金残高 (A/R BALANCE) が含まれています。

次いで実行キーを押すと、プログラムは最初に戻ります。 別の得意先番号を入力するか、またはプログラムを終了することができます。プログラムを終了するには、F1 キーを押します。これによりプログラムの中で標識 15 がオンに設定されます。

標識 15 がオンになると、プログラムはすべてのファイルをクローズし、GOBACK ステートメントを処理します。その後、制御をこの ILE COBOL プログラムの呼び出し側に戻します。

1 の WRITE 操作によって書き出される最初の表示は、次のようなものです。

```
Customer Master Inquiry

Customer Number  ________

Use F3 to end program, use enter key to return to prompt screen
```

最初の表示画面への応答として入力された得意先番号に対するレコードが CUSMSTP ファイルの中に見つかれば、次の画面が表示されます。

```
Customer Master Inquiry

Customer Number   1000

Use F3 to end program, use enter key to return to prompt screen


Name    EXAMPLE WHOLESALERS LTD.
Address ANYWHERE STREET
City    ACITY
State   IL        Zipcode  12345
A/R balance       137.02
```

最初の表示画面への応答として入力された得意先番号に対するレコードが CUSMSTP ファイルの中にない場合には、次の画面が表示されます。

```
Customer Master Inquiry

Customer Number

Use F3 to end program, use enter key to return to prompt screen




Customer number not found, press reset, then enter valid number
```

トランザクション・ファイルでの標識の使用

標識は、B"0" または B"1" という値をもつブール・データ項目です。

DDS を使ってレコード様式を定義する場合、標識を使ってオプションの条件付けをすることができます。また、標識を使って特定の応答を反映させることができます。これらの標識は、それぞれオプション標識および応答標識と呼ばれます。

オプション標識は、行送り、下線付け、および ILE COBOL プログラムからプリンターやディスプレイ装置へのデータ転送を許可したり要求するためのオプションを提供します。応答標識は、ワークステーション・ユーザーが押したファンクション・キーや、データが入力されたかどうかなど、装置から ILE COBOL プログラムへの応答情報を提供します。

標識は、レコード域内のデータ・レコードとともに渡したり、レコード域以外の別個の標識域で渡したりすることができます。

別個の標識域で標識を渡す

DDS 中でファイル・レベル・キーワード INDARA を指定すると、そのファイルのレコード様式で定義されているすべての標識を ILE COBOL プログラムとの間で受け渡しすることは、レコード域ではなく別個の標識域で行われます。 INDARA キーワードの指定方法については、Web サイト http://www.ibm.com/systems/i/

infocenter/ にある **i5/OS Information Center** の「データベースおよびファイル・システム」カテゴリーを参照してください。

DDS 中で INDARA が指定されているファイルのファイル制御項目は、別個の標識域属性 SI が割り当て名の一部になっていなければなりません。たとえば、DSPFILE という名前のファイルに対する割り当ては、次のようになります。

```
FILE-CONTROL.
    SELECT DISPFILE
           ASSIGN TO WORKSTATION-DSPFILE-SI
           ORGANIZATION IS TRANSACTION
           ACCESS IS SEQUENTIAL.
```

別個の標識域を使用することには、次のような利点があります。

- ファイルの中のレコード様式の入出力ステートメントで使われている標識の数や順序が、そのレコード様式の DDS で指定されている標識の数や順序と一致している必要がない。
- データ記述項目中の標識番号は、プログラムによって適切な標識に関連付けられる。

レコード域で標識を渡す

ファイルの DDS で INDARA というキーワードが使われていない場合、標識はレコード域の中に作成されます。ファイルのレコード様式で標識が定義されると、それらの標識はレコード域内のデータとともに読み取られたり、再書き込みされたり、書き込まれたりします。

ファイルのレコード様式に関する DDS で定義された標識の数と順序によって、そのレコード様式の標識に関するデータ記述項目を ILE COBOL プログラムでコーディングすべき数と順序が定められます。

関連付けられた DDS 中で INDARA キーワードが指定されていないファイルのファイル制御項目では、別個の標識域属性 SI を割り当て名の一部としてはなりません。

標識を ILE COBOL プログラムにコピーするために形式 2 の COPY ステートメントを使用する場合、標識が定義される順番は、そのファイルに対する DDS 中で指定されている順番になります。

ILE COBOL プログラムでの標識の使用例

ここでは、レコード域または別個の標識域のいずれかで標識を使用する ILE COBOL プログラムの例を示します。

すべての ILE COBOL プログラムは、次のことを行います。

1. 現在の日付を調べる。
2. それが月の最初の日である場合、出力フィールドを表示して明滅させるオプション標識をオンに変える。
3. ユーザーがプログラムを終了させるためにファンクション・キーを押せるようにするか、または応答標識をオンに変えて日報または月報を作成するプログラムを呼び出す。

598 ページの図 135 には、レコード域内の標識は使用するが、入出力ステートメントの中で INDICATORS 句を使わない ILE COBOL プログラムを示します。 図 134 には、そのファイルに対する関連付けられた DDS を示します。

601 ページの図 136 には、レコード域の中の標識を使用し、入出力ステートメントで INDICATORS 句を使用する ILE COBOL プログラムが示されています。 図 136 に対する関連付けられた DDS は、図 134 です。

604 ページの図 138 には、形式 2 の COPY ステートメントを使用して WORKING-STORAGE SECTION 中で定義された別個の標識域で標識を使う ILE COBOL プログラムが示されています。 603 ページの図 137 には、そのファイルに対する関連付けられた DDS を示します。

607 ページの図 139 で、WORKING-STORAGE SECTION のテーブルで定義され、個別の標識域の標識を使用する、ILE COBOL プログラムを示します。そのファイルに関連する DDS は 603 ページの図 137 と同じです。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A* DISPLAY FILE DDS FOR INDICATOR EXAMPLES - INDICATORS IN RECORD AREA
     A* DSPFILEX                                 1
     A      2   R FORMAT1                    3 CF01(99 'END OF PROGRAM')
     A                                          CF05(51 'DAILY REPORT')
     A                                          CF09(52 'MONTHLY REPORT')
     A*
     A                                4     10 10'DEPARTMENT NUMBER:'
     A            DEPTNO             5   I 10 32
     A   5 01                               20 26'PRODUCE MONTHLY REPORTS'
     A                                          DSPATR(BL)
     A*
     A                                 6    24 01'F5 = DAILY REPORT'
     A                                      24 26'F9 = MONTHLY REPORT'
     A                                      24 53'F1 = TERMINATE'
     A          R ERRFMT
     A     98                                6  5'INPUT-OUTPUT ERROR'
```

図 134. 入出力ステートメント中の INDICATORS 句を使わずにレコード域の標識を使用するプログラムの例 - DDS

1 INDARA キーワードは使用しません。標識はデータ・フィールドとともにレコード域に保管されます。

2 レコード様式 FORMAT1 を指定しています。

3 3 つの標識が 3 つのファンクション・キーに関連付けられています。 F1 キーを押すと標識 99 がオンに設定されます。その他も同様です。

4 入力用に 1 つのフィールドが定義されます。

5 標識がオンの場合には関連付けられた固定情報フィールドが明滅するように、標識 01 が定義されます。

6 機能 (F) キー定義が、ワークステーションのディスプレイ装置上に表示されるようにします。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/INDIC1         ISERIES1   06/02/15 14:59:29        ページ     2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200   PROGRAM-ID.     INDIC1.
          000300*    SAMPLE PROGRAM WITH INDICATORS IN RECORD AREA.
          000400
    3     000500 ENVIRONMENT DIVISION.
    4     000600   CONFIGURATION SECTION.
    5     000700 SOURCE-COMPUTER. IBM-ISERIES
    6     000800 OBJECT-COMPUTER. IBM-ISERIES
    7     000900   INPUT-OUTPUT SECTION.
    8     001000   FILE-CONTROL.
    9     001100     SELECT DISPFILE
   10     001200       ASSIGN TO WORKSTATION-DSPFILEX  1
   11     001300       ORGANIZATION IS TRANSACTION
   12     001400         ACCESS IS SEQUENTIAL.
          001500
   13     001600 DATA DIVISION.
   14     001700 FILE SECTION.
   15     001800 FD  DISPFILE.
   16     001900 01  DISP-REC.
          002000       COPY DDS-ALL-FORMATS OF DSPFILEX.  2
   17    +000001       05  DSPFILEX-RECORD PIC X(8).                                    <-ALL-FMTS
         +000002*  INPUT FORMAT:FORMAT1    FROM FILE DSPFILEX   OF LIBRARY CBLGUIDE    <-ALL-FMTS
         +000003*                                                                      <-ALL-FMTS
   18    +000004       05  FORMAT1-I     REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
   19    +000005           06 FORMAT1-I-INDIC.                                        <-ALL-FMTS
   20    +000006                07 IN99             PIC 1  INDIC 99.  3              <-ALL-FMTS
         +000007*                            END OF PROGRAM                            <-ALL-FMTS
   21    +000008                07 IN51             PIC 1  INDIC 51.                 <-ALL-FMTS
         +000009*                            DAILY REPORT                              <-ALL-FMTS
   22    +000010                07 IN52             PIC 1  INDIC 52.                 <-ALL-FMTS
         +000011*                            MONTHLY REPORT                            <-ALL-FMTS
   23    +000012           06 DEPTNO               PIC X(5).                         <-ALL-FMTS
         +000013* OUTPUT FORMAT:FORMAT1    FROM FILE DSPFILEX   OF LIBRARY CBLGUIDE    <-ALL-FMTS
         +000014*                                                                      <-ALL-FMTS
   24    +000015       05  FORMAT1-O     REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
   25    +000016           06 FORMAT1-O-INDIC.                                        <-ALL-FMTS
   26    +000017                07 IN01             PIC 1  INDIC 01.                 <-ALL-FMTS
         +000018*  INPUT FORMAT:ERRFMT     FROM FILE DSPFILEX   OF LIBRARY CBLGUIDE    <-ALL-FMTS
         +000019*                                                                      <-ALL-FMTS
         +000020*      05  ERRFMT-I      REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
         +000021* OUTPUT FORMAT:ERRFMT     FROM FILE DSPFILEX   OF LIBRARY CBLGUIDE    <-ALL-FMTS
         +000022*                                                                      <-ALL-FMTS
   27    +000023       05  ERRFMT-O      REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
   28    +000024           06 ERRFMT-O-INDIC.                                         <-ALL-FMTS
   29    +000025                07 IN98             PIC 1  INDIC 98.                 <-ALL-FMTS
          002100
   30     002200 WORKING-STORAGE SECTION.
   31     002300 01  CURRENT-DATE.
   32     002400     05  CURR-YEAR                PIC 9(2).
   33     002500     05  CURR-MONTH               PIC 9(2).
   34     002600     05  CURR-DAY                 PIC 9(2).
   35     002700 01  INDIC-AREA.  4
   36     002800    05  IN01                      PIC 1.
```

図 135. 入出力ステートメント中の INDICATORS 句を使わずにレコード域の標識を使用するプログラムの例 - COBOL ソース・プログラム (1/2)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/INDIC1         ISERIES1   06/02/15 14:59:29        ページ    3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   37     002900       88  NEW-MONTH              VALUE B"1".  5
   38     003000    05  IN51                      PIC 1.
   39     003100       88  WANT-DAILY             VALUE B"1".
   40     003200    05  IN52                      PIC 1.
   41     003300       88  WANT-MONTHLY           VALUE B"1".
   42     003400    05  IN98                      PIC 1.
   43     003500       88  IO-ERROR               VALUE B"1".
   44     003600    05  IN99                      PIC 1.
   45     003700       88  NOT-END-OF-JOB         VALUE B"0".
   46     003800       88  END-OF-JOB             VALUE B"1".
          003900
   47     004000 PROCEDURE DIVISION.
   48     004100 DECLARATIVES.
          004200 DISPLAY-ERR-SECTION SECTION.
          004300     USE AFTER STANDARD EXCEPTION PROCEDURE ON DISPFILE.
          004400 DISPLAY-ERR-PARAGRAPH.
   49     004500     SET IO-ERROR TO TRUE
   50     004600     MOVE CORR INDIC-AREA TO ERRFMT-O-INDIC
                 *** CORRESPONDING items for statement 50:
                 ***     IN98
                 *** End of CORRESPONDING items for statement 50
   51     004700     WRITE DISP-REC FORMAT IS "ERRFMT"
          004800     END-WRITE
   52     004900     CLOSE DISPFILE.
   53     005000     STOP RUN.
          005100 END DECLARATIVES.
          005200
          005300 MAIN-PROGRAM SECTION.
          005400 MAINLINE.
   54     005500     OPEN I-O DISPFILE.
   55     005600     ACCEPT CURRENT-DATE FROM DATE.
   56     005700     SET NOT-END-OF-JOB TO TRUE.
   57     005800     PERFORM UNTIL END-OF-JOB
          005900
   58     006000         MOVE ZEROS TO INDIC-AREA  6
   59     006100         IF CURR-DAY = 01 THEN
   60     006200            SET NEW-MONTH TO TRUE  7
          006300         END-IF
   61     006400         MOVE CORR INDIC-AREA TO FORMAT1-O-INDIC  8
                 *** CORRESPONDING items for statement 61:
                 ***     IN01
                 *** End of CORRESPONDING items for statement 61
   62     006500         WRITE DISP-REC FORMAT IS "FORMAT1"  9
          006600         END-WRITE
          006700
   63     006800         MOVE ZEROS TO INDIC-AREA
   64     006900         READ DISPFILE FORMAT IS "FORMAT1"  10
          007000         END-READ
   65     007100         MOVE CORR FORMAT1-I-INDIC TO INDIC-AREA  11
                 *** CORRESPONDING items for statement 65:
                 ***     IN99
                 ***     IN51
                 ***     IN52
                 *** End of CORRESPONDING items for statement 65
   66     007200         IF WANT-DAILY THEN
   67     007300            CALL "DAILY" USING DEPTNO
          007400         ELSE
   68     007500            IF WANT-MONTHLY THEN
   69     007600               CALL "MONTHLY" USING DEPTNO  12
          007700             END-IF
          007800         END-IF
          007900
          008000     END-PERFORM.
   70     008100     CLOSE DISPFILE.
   71     008200     STOP RUN.
                         * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 135. 入出力ステートメント中の INDICATORS 句を使わずにレコード域の標識を使用するプログラムの例 - COBOL ソース・プログラム (2/2)

1 別個の標識域属性 SI は、ASSIGN 文節中にはコーディングしません。その結果、標識はレコード域の一部となります。

2 形式 2 の COPY ステートメントで、レコード域の中のデータ・フィールドと標識を定義します。

3 ファイル標識がレコード域の一部となっているので、DDS 中で使用されている順序で応答およびオプション標識を定義し、標識番号が文書として扱われます。

4 プログラムが使うすべての標識は、WORKING-STORAGE SECTION のデータ記述項目の中で意味をもつ名前を付けて定義されています。標識番号は影響はないので、ここでは省略されています。

5 各標識ごとに、その標識の値と、意味のあるレベル 88 条件名とが関連付けられています。

6 グループ・レベルをゼロに初期設定します。

7 月の最初の日である場合、WORKING-STORAGE SECTION の IN01 をオンに設定します。

8 FORMAT1 の出力に該当する標識を、レコード域にコピーします。

9 FORMAT1 が、レコード域のデータおよび標識値の両方とともに、ワークステーションのディスプレイ装置に書き込まれます。

別個の標識域はなく、これ以前の MOVE CORRESPONDING ステートメントでレコード域内に標識値が設定されているので、INDICATORS 句は不要です。

10 FORMAT1 (データと標識の両方を含む) がディスプレイ装置から読み込まれます。

11 FORMAT1 に対する応答標識を、レコード域から WORKING-STORAGE SECTION のデータ記述項目にコピーします。

12 F5 が押されている場合、プログラム呼び出しを処理します。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/INDIC1          ISERIES1   06/02/15 15:00:29        ページ    2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200   PROGRAM-ID.     INDIC2.
          000300*    SAMPLE PROGRAM - FILE WITH INDICATORS IN RECORD AREA
          000400
    3     000500 ENVIRONMENT DIVISION.
    4     000600   CONFIGURATION SECTION.
    5     000700 SOURCE-COMPUTER. IBM-ISERIES
    6     000800 OBJECT-COMPUTER. IBM-ISERIES
    7     000900   INPUT-OUTPUT SECTION.
    8     001000   FILE-CONTROL.
    9     001100     SELECT DISPFILE
   10     001200       ASSIGN TO WORKSTATION-DSPFILEX  1
   11     001300       ORGANIZATION IS TRANSACTION
   12     001400         ACCESS IS SEQUENTIAL.
          001500
   13     001600 DATA DIVISION.
   14     001700 FILE SECTION.
   15     001800 FD  DISPFILE.
   16     001900 01  DISP-REC.
          002000       COPY DDS-ALL-FORMATS OF DSPFILEX.  2
   17    +000001       05  DSPFILEX-RECORD PIC X(8).                                    <-ALL-FMTS
         +000002*  INPUT FORMAT:FORMAT1    FROM FILE DSPFILEX   OF LIBRARY CBLGUIDE    <-ALL-FMTS
         +000003*                                                                      <-ALL-FMTS
   18    +000004       05  FORMAT1-I     REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
   19    +000005           06 FORMAT1-I-INDIC.                                        <-ALL-FMTS
   20    +000006                07 IN99            PIC 1  INDIC 99.  3               <-ALL-FMTS
         +000007*                          END OF PROGRAM                            <-ALL-FMTS
   21    +000008                07 IN51            PIC 1  INDIC 51.                  <-ALL-FMTS
         +000009*                          DAILY REPORT                              <-ALL-FMTS
   22    +000010                07 IN52            PIC 1  INDIC 52.                  <-ALL-FMTS
         +000011*                          MONTHLY REPORT                            <-ALL-FMTS
   23    +000012           06 DEPTNO              PIC X(5).                         <-ALL-FMTS
         +000013* OUTPUT FORMAT:FORMAT1    FROM FILE DSPFILEX   OF LIBRARY CBLGUIDE    <-ALL-FMTS
         +000014*                                                                      <-ALL-FMTS
   24    +000015       05  FORMAT1-O     REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
   25    +000016           06 FORMAT1-O-INDIC.                                        <-ALL-FMTS
   26    +000017                07 IN01            PIC 1  INDIC 01.                  <-ALL-FMTS
         +000018*  INPUT FORMAT:ERRFMT     FROM FILE DSPFILEX   OF LIBRARY CBLGUIDE    <-ALL-FMTS
         +000019*                                                                      <-ALL-FMTS
         +000020*      05  ERRFMT-I      REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
         +000021* OUTPUT FORMAT:ERRFMT     FROM FILE DSPFILEX   OF LIBRARY CBLGUIDE    <-ALL-FMTS
         +000022*                                                                      <-ALL-FMTS
   27    +000023       05  ERRFMT-O      REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
   28    +000024           06 ERRFMT-O-INDIC.                                         <-ALL-FMTS
   29    +000025                07 IN98            PIC 1  INDIC 98.                  <-ALL-FMTS
          002100
   30     002200 WORKING-STORAGE SECTION.
   31     002300 01  CURRENT-DATE.
   32     002400     05  CURR-YEAR               PIC 9(2).
   33     002500     05  CURR-MONTH              PIC 9(2).
   34     002600     05  CURR-DAY                PIC 9(2).
          002700
   35     002800 77  IND-OFF                    PIC 1      VALUE B"0".
```

図 136. レコード域の標識と入出力ステートメントの INDICATORS 句を使用するプログラムの例 - COBOL ソース・プログラム (1/2)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/INDIC1         ISERIES1   06/02/15 15:00:29        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   36     002900 77  IND-ON                        PIC 1      VALUE B"1".
          003000
   37     003100 01  RESPONSE-INDICS.
   38     003200     05  END-OF-PROGRAM            PIC 1.  4
   39     003300     05  DAILY-REPORT              PIC 1.
   40     003400     05  MONTHLY-REPORT            PIC 1.
   41     003500 01  OPTION-INDICS.
   42     003600     05  NEW-MONTH                 PIC 1.
   43     003700 01  ERROR-INDICS.
   44     003800     05  IO-ERROR                  PIC 1.
          003900
   45     004000 PROCEDURE DIVISION.
   46     004100 DECLARATIVES.
          004200 DISPLAY-ERR-SECTION SECTION.
          004300     USE AFTER STANDARD EXCEPTION PROCEDURE ON DISPFILE.
          004400 DISPLAY-ERR-PARAGRAPH.
   47     004500     MOVE IND-ON TO IO-ERROR
   48     004600     WRITE DISP-REC FORMAT IS "ERRFMT"
          004700                    INDICATORS ARE ERROR-INDICS
          004800     END-WRITE
   49     004900     CLOSE DISPFILE.
   50     005000     STOP RUN.
          005100 END DECLARATIVES.
          005200
          005300 MAIN-PROGRAM SECTION.
          005400 MAINLINE.
   51     005500     OPEN I-O DISPFILE.
   52     005600     ACCEPT CURRENT-DATE FROM DATE.
   53     005700     MOVE IND-OFF TO END-OF-PROGRAM.
   54     005800     PERFORM UNTIL END-OF-PROGRAM = IND-ON
   55     005900         MOVE ZEROS TO OPTION-INDICS
   56     006000         IF CURR-DAY = 01 THEN  5
   57     006100            MOVE IND-ON TO NEW-MONTH
          006200         END-IF
   58     006300         WRITE DISP-REC FORMAT IS "FORMAT1"  6
          006400                        INDICATORS ARE OPTION-INDICS
          006500         END-WRITE
          006600
   59     006700         MOVE ZEROS TO RESPONSE-INDICS
   60     006800         READ DISPFILE FORMAT IS "FORMAT1"  7
          006900                       INDICATORS ARE RESPONSE-INDICS  8
          007000         END-READ
   61     007100         IF DAILY-REPORT = IND-ON THEN
   62     007200            CALL "DAILY" USING DEPTNO  9
          007300         ELSE
   63     007400            IF MONTHLY-REPORT = IND-ON THEN
   64     007500               CALL "MONTHLY" USING DEPTNO
          007600            END-IF
          007700             END-IF
          007800
          007900     END-PERFORM
   65     008000     CLOSE DISPFILE.
   66     008100     STOP RUN.
          008200
                         * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 136. レコード域の標識と入出力ステートメントの INDICATORS 句を使用するプログラムの例 - COBOL ソース・プログラム (2/2)

1 別個の標識域属性 SI は、ASSIGN 文節中にはコーディングしません。

2 形式 2 の COPY ステートメントで、レコード域の中のデータ・フィールドと標識を定義します。

3 ファイルに別個の標識域がないので、応答標識とオプション標識は DDS 中で使われている順番で定義され、標識番号が文書として処理されます。

4 プログラムが使うすべての標識は、WORKING-STORAGE SECTION のデータ記述項目の中で意味をもつ名前を付けて定義されています。標識番号は影響はないので、ここでは省略されています。ディスプレイ・ファイルで必要となる順番で標識を定義する必要があります。

5 月の最初の日である場合、WORKING-STORAGE SECTION の IN01 をオンに設定します。

6 FORMAT1 が以下のようにしてワークステーションのディスプレイ装置に書き込まれます。

- INDICATORS 句によって、変数 OPTION-INDICS の内容がレコード域の先頭にコピーされます。
- 日付および標識値が、ワークステーションのディスプレイ装置に書き込まれます。

7 FORMAT1 (データと標識の両方を含む) がワークステーションのディスプレイ装置から読み込まれます。

8 INDICATORS 句によって、レコード域の先頭からのバイトが RESPONSE-INDICS にコピーされます。

9 F5 が押されている場合、プログラム呼び出しを処理します。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A* DISPLAY FILE FOR INDICATOR EXAMPLES - INDICATORS IN SI AREA
     A* DSPFILE
     A                                      INDARA 1
     A          R FORMAT1                   CF01(99 'END OF PROGRAM')
     A                                      CF05(51 'DAILY REPORT')
     A                                      CF09(52 'MONTHLY REPORT')
     A*
     A                                   10 10'DEPARTMENT NUMBER:'
     A            DEPTNO         5   I 10 32
     A     01                            20 26'PRODUCE MONTHLY REPORTS'
     A                                        DSPATR(BL)
     A*
     A                                   24 01'F5 = DAILY REPORT'
     A                                   24 26'F9 = MONTHLY REPORT'
     A                                   24 53'F1 = TERMINATE'
     A          R ERRFMT
     A     98                             6  5'INPUT-OUTPUT ERROR'
```

図 137. COPY ステートメントを使用して WORKING-STORAGE 内で定義される別個の標識域中の標識を使用するプログラムの例 - DDS

1 INDARA キーワードが指定されています。標識はレコード域ではなく、別個の標識域に保管されます。この指定以外については、このファイルの DDS は 597 ページの図 134 に示されているものと同じです。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/INDIC1         ISERIES1   06/02/15 15:01:36        ページ     2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     INDIC3.
          000300*    SAMPLE PROGRAM - FILE WITH SEPERATE INDICATORS AREA
          000400
    3     000500 ENVIRONMENT DIVISION.
    4     000600   CONFIGURATION SECTION.
    5     000700 SOURCE-COMPUTER. IBM-ISERIES
    6     000800 OBJECT-COMPUTER. IBM-ISERIES
    7     000900   INPUT-OUTPUT SECTION.
    8     001000   FILE-CONTROL.
    9     001100     SELECT DISPFILE
   10     001200         ASSIGN TO WORKSTATION-DSPFILE-SI  1
   11     001300       ORGANIZATION IS TRANSACTION
   12     001400         ACCESS IS SEQUENTIAL.
          001500
   13     001600 DATA DIVISION.
   14     001700 FILE SECTION.
   15     001800 FD  DISPFILE.
   16     001900 01  DISP-REC.
          002000     COPY DDS-ALL-FORMATS OF DSPFILE.  2
   17    +000001       05  DSPFILE-RECORD PIC X(5).                                  <-ALL-FMTS
         +000002*  INPUT FORMAT:FORMAT1    FROM FILE DSPFILE    OF LIBRARY CBLGUIDE  <-ALL-FMTS
         +000003*                                                                   <-ALL-FMTS
   18    +000004       05  FORMAT1-I     REDEFINES DSPFILE-RECORD.                  <-ALL-FMTS
   19    +000005           06 DEPTNO               PIC X(5).                       <-ALL-FMTS
         +000006* OUTPUT FORMAT:FORMAT1    FROM FILE DSPFILE    OF LIBRARY CBLGUIDE  <-ALL-FMTS
         +000007*                                                                   <-ALL-FMTS
         +000008*      05  FORMAT1-O     REDEFINES DSPFILE-RECORD.                  <-ALL-FMTS
         +000009*  INPUT FORMAT:ERRFMT     FROM FILE DSPFILE    OF LIBRARY CBLGUIDE  <-ALL-FMTS
         +000010*                                                                   <-ALL-FMTS
         +000011*      05  ERRFMT-I      REDEFINES DSPFILE-RECORD.                  <-ALL-FMTS
         +000012* OUTPUT FORMAT:ERRFMT     FROM FILE DSPFILE    OF LIBRARY CBLGUIDE  <-ALL-FMTS
         +000013*                                                                   <-ALL-FMTS
         +000014*      05  ERRFMT-O      REDEFINES DSPFILE-RECORD.                  <-ALL-FMTS
          002100
   20     002200 WORKING-STORAGE SECTION.
   21     002300 01  CURRENT-DATE.
   22     002400    05  CURR-YEAR                 PIC 9(2).
   23     002500    05  CURR-MONTH                PIC 9(2).
   24     002600    05  CURR-DAY                  PIC 9(2).
          002700
   25     002800    77 IND-OFF                    PIC 1 VALUE B"0".
   26     002900    77 IND-ON                     PIC 1 VALUE B"1".
          003000
   27     003100 01  DISPFILE-INDICS.
          003200     COPY DDS-ALL-FORMATS-INDIC OF DSPFILE.  3
   28    +000001       05  DSPFILE-RECORD.                                           <-ALL-FMTS
         +000002*  INPUT FORMAT:FORMAT1    FROM FILE DSPFILE    OF LIBRARY CBLGUIDE  <-ALL-FMTS
         +000003*                                                                   <-ALL-FMTS
   29    +000004           06 FORMAT1-I-INDIC.                                      <-ALL-FMTS
   30    +000005                07 IN51             PIC 1  INDIC 51.  4           <-ALL-FMTS
         +000006*                          DAILY REPORT                             <-ALL-FMTS
   31    +000007                07 IN52             PIC 1  INDIC 52.              <-ALL-FMTS
```

図 138. 別個の標識域の中の標識を使用する COBOL のリスト (1/2)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/INDIC1          ISERIES1   06/02/15 15:01:36        ページ    3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
        +000008*                          MONTHLY REPORT                                    <-ALL-FMTS
   32   +000009                07 IN99             PIC 1  INDIC 99.                        <-ALL-FMTS
        +000010*                          END OF PROGRAM                                    <-ALL-FMTS
        +000011* OUTPUT FORMAT:FORMAT1    FROM FILE DSPFILE    OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000012*                                                                            <-ALL-FMTS
   33   +000013           06 FORMAT1-O-INDIC.                                              <-ALL-FMTS
   34   +000014                07 IN01             PIC 1  INDIC 01.                        <-ALL-FMTS
        +000015*  INPUT FORMAT:ERRFMT     FROM FILE DSPFILE    OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000016*                                                                            <-ALL-FMTS
        +000017*          06 ERRFMT-I-INDIC.                                                <-ALL-FMTS
        +000018* OUTPUT FORMAT:ERRFMT     FROM FILE DSPFILE    OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000019*                                                                            <-ALL-FMTS
   35   +000020           06 ERRFMT-O-INDIC.                                               <-ALL-FMTS
   36   +000021                07 IN98             PIC 1  INDIC 98.                        <-ALL-FMTS
         003300
   37    003400 PROCEDURE DIVISION.
   38    003500 DECLARATIVES.
         003600 DISPLAY-ERR-SECTION SECTION.
         003700     USE AFTER STANDARD EXCEPTION PROCEDURE ON DISPFILE.
         003800 DISPLAY-ERR-PARAGRAPH.
   39    003900     MOVE IND-ON TO IN98 IN ERRFMT-O-INDIC
   40    004000     WRITE DISP-REC FORMAT IS "ERRFMT"
         004100                    INDICATORS ARE ERRFMT-O-INDIC
         004200     END-WRITE
   41    004300     CLOSE DISPFILE.
   42    004400     STOP RUN.
         004500 END DECLARATIVES.
         004600
         004700 MAIN-PROGRAM SECTION.
         004800 MAINLINE.
         004900
   43    005000     OPEN I-O DISPFILE.
   44    005100     ACCEPT CURRENT-DATE FROM DATE.
   45    005200     MOVE IND-OFF TO IN99 IN FORMAT1-I-INDIC.
   46    005300     PERFORM UNTIL IN99 IN FORMAT1-I-INDIC = IND-ON
         005400
   47    005500         MOVE ZEROS TO FORMAT1-O-INDIC
   48    005600         IF CURR-DAY = 01 THEN
   49    005700            MOVE IND-ON TO IN01 IN FORMAT1-O-INDIC  5
         005800         END-IF
   50    005900         WRITE DISP-REC FORMAT IS "FORMAT1"
         006000                        INDICATORS ARE FORMAT1-O-INDIC  6
         006100         END-WRITE
         006200
   51    006300         MOVE ZEROS TO FORMAT1-I-INDIC
   52    006400         READ DISPFILE FORMAT IS "FORMAT1"
         006500                       INDICATORS ARE FORMAT1-I-INDIC  7
         006600         END-READ
   53    006700         IF IN51 IN FORMAT1-I-INDIC = IND-ON THEN
   54    006800             CALL "DAILY" USING DEPTNO
         006900         ELSE
   55    007000             IF IN52 IN FORMAT1-I-INDIC = IND-ON THEN
   56    007100                 CALL "MONTHLY" USING DEPTNO  8
         007200             END-IF
         007300         END-IF
         007400
         007500     END-PERFORM
   57    007600     CLOSE DISPFILE.
   58    007700     STOP RUN.
         007800
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 138. 別個の標識域の中の標識を使用する COBOL のリスト (2/2)

1 別個の標識域属性 SI は、ASSIGN 文節の中に指定します。

2 形式 2 の COPY ステートメントは、データ・フィールドのみのレコード域の中にデータ記述を生成します。そのファイルに対して別個の標識域が指定されているので、標識のデータ記述項目は生成されません。

3 INDICATOR 属性 INDIC が指定されている形式 2 の COPY ステートメントにより、ファイルのレコード様式の DDS 内で使用されるすべての標識のデータ記述項目を WORKING-STORAGE SECTION の中で定義します。

4 ファイルに別個の標識域があるので、データ記述項目内で使われている標識番号は、文書としては扱われません。

5 FORMAT1 のための別個の標識域の IN01 は、月の最初の日である場合に、オンに設定されます。

6 標識値をワークステーションのディスプレイ装置へ送るため、INDICATORS 句が必要です。

7 標識値をワークステーションのディスプレイ装置から受け取るため、INDICATORS 句が必要です。 F5 キーを押した場合、IN51 がオンに設定されます。

8 IN51 がオンに設定されている場合、プログラム呼び出しが処理されます。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/INDIC4         ISERIES1   06/02/15 15:02:22        ページ    2
                                  ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     INDIC4.
          000300*    SAMPLE PROGRAM
          000400*    FILE WITH SEPERATE INDICATORS AREA IN WORKING STORAGE
          000500
    3     000600 ENVIRONMENT DIVISION.
    4     000700 CONFIGURATION SECTION.
    5     000800 SOURCE-COMPUTER. IBM-ISERIES
    6     000900 OBJECT-COMPUTER. IBM-ISERIES
    7     001000 INPUT-OUTPUT SECTION.
    8     001100 FILE-CONTROL.
    9     001200     SELECT DISPFILE
   10     001300         ASSIGN TO WORKSTATION-DSPFILE-SI  1
   11     001400         ORGANIZATION IS TRANSACTION
   12     001500         ACCESS IS SEQUENTIAL.
          001600
   13     001700 DATA DIVISION.
   14     001800 FILE SECTION.
   15     001900 FD  DISPFILE.
   16     002000 01  DISP-REC.
          002100     COPY DDS-ALL-FORMATS OF DSPFILE.  2
   17    +000001       05  DSPFILE-RECORD PIC X(5).                                          <-ALL-FMTS
         +000002*  INPUT FORMAT:FORMAT1    FROM FILE DSPFILE    OF LIBRARY CBLGUIDE        <-ALL-FMTS
         +000003*                                                                           <-ALL-FMTS
   18    +000004       05  FORMAT1-I     REDEFINES DSPFILE-RECORD.                        <-ALL-FMTS
   19    +000005           06 DEPTNO               PIC X(5).                               <-ALL-FMTS
         +000006* OUTPUT FORMAT:FORMAT1    FROM FILE DSPFILE    OF LIBRARY CBLGUIDE        <-ALL-FMTS
         +000007*                                                                           <-ALL-FMTS
         +000008*      05  FORMAT1-O     REDEFINES DSPFILE-RECORD.                        <-ALL-FMTS
         +000009*  INPUT FORMAT:ERRFMT     FROM FILE DSPFILE    OF LIBRARY CBLGUIDE        <-ALL-FMTS
         +000010*                                                                           <-ALL-FMTS
         +000011*      05  ERRFMT-I      REDEFINES DSPFILE-RECORD.                        <-ALL-FMTS
         +000012* OUTPUT FORMAT:ERRFMT     FROM FILE DSPFILE    OF LIBRARY CBLGUIDE        <-ALL-FMTS
         +000013*                                                                           <-ALL-FMTS
         +000014*      05  ERRFMT-O      REDEFINES DSPFILE-RECORD.                        <-ALL-FMTS
          002200
   20     002300 WORKING-STORAGE SECTION.
   21     002400 01  CURRENT-DATE.
   22     002500    05  CURR-YEAR                 PIC 9(2).
   23     002600    05  CURR-MONTH                PIC 9(2).
   24     002700    05  CURR-DAY                  PIC 9(2).
          002800
   25     002900 01  INDIC-AREA.
   26     003000    05 INDIC-TABLE  OCCURS 99      PIC 1    INDICATOR 1.  3
   27     003100      88  IND-OFF                          VALUE B"0".
   28     003200      88  IND-ON                           VALUE B"1".
          003300
   29     003400 01  DISPFILE-INDIC-USAGE.
   30     003500    05  IND-NEW-MONTH              PIC 9(2) VALUE 01.
   31     003600    05  IND-DAILY                  PIC 9(2) VALUE 51.  4
   32     003700    05  IND-MONTHLY                PIC 9(2) VALUE 52.
   33     003800    05  IND-IO-ERROR               PIC 9(2) VALUE 98.
   34     003900    05  IND-EOJ                    PIC 9(2) VALUE 99.
```

図 139. WORKING-STORAGE の表で定義され、個別の標識域の標識を使用するプログラムの例 (1/2)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/INDIC4          ISERIES1   06/02/15 15:02:22        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
        004000
   35   004100 PROCEDURE DIVISION.
   36   004200 DECLARATIVES.
        004300 DISPLAY-ERR-SECTION SECTION.
        004400     USE AFTER STANDARD EXCEPTION PROCEDURE ON DISPFILE.
        004500 DISPLAY-ERR-PARAGRAPH.
   37   004600     SET IND-ON (IND-IO-ERROR) TO TRUE
   38   004700     WRITE DISP-REC FORMAT IS "ERRFMT"
        004800                      INDICATORS ARE INDIC-TABLE
        004900     END-WRITE
   39   005000     CLOSE DISPFILE.
   40   005100     STOP RUN.
        005200 END DECLARATIVES.
        005300
        005400 MAIN-PROGRAM SECTION.
        005500 MAINLINE.
   41   005600     OPEN I-O DISPFILE.
   42   005700     ACCEPT CURRENT-DATE FROM DATE.
   43   005800     SET IND-OFF (IND-EOJ) TO TRUE.
   44   005900     PERFORM UNTIL IND-ON (IND-EOJ)
        006000
   45   006100         MOVE ZEROS TO INDIC-AREA
   46   006200         IF CURR-DAY = 01 THEN
   47   006300            SET IND-ON (IND-NEW-MONTH) TO TRUE  5
        006400         END-IF
   48   006500         WRITE DISP-REC FORMAT IS "FORMAT1"
        006600                        INDICATORS ARE INDIC-TABLE  6
        006700         END-WRITE
        006800
   49   006900         READ DISPFILE FORMAT IS "FORMAT1"
        007000                        INDICATORS ARE INDIC-TABLE  7
        007100         END-READ
   50   007200         IF IND-ON (IND-DAILY) THEN
   51   007300             CALL "DAILY" USING DEPTNO  8
        007400         ELSE
   52   007500             IF IND-ON (IND-MONTHLY) THEN
   53   007600                 CALL "MONTHLY" USING DEPTNO
        007700             END-IF
        007800         END-IF
        007900
        008000     END-PERFORM
   54   008100     CLOSE DISPFILE.
   55   008200     STOP RUN.
        008300
                       * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 139. WORKING-STORAGE の表で定義され、個別の標識域の標識を使用するプログラムの例 (2/2)

1 別個の標識域属性 SI は、ASSIGN 文節の中に指定します。

2 形式 2 の COPY ステートメントは、データ・フィールドについてのみレコード域の中にフィールドを生成します。

3 99 個のブール・データ項目の表を、WORKING-STORAGE SECTION の中に定義します。このデータ記述項目に対する INDICATOR 文節によって、これらのデータ項目がそれぞれ標識 1 ～ 99 に関連付けられます。このような表を使うことによって、個々の標識に対する複数の従属項目をもつグループ項目を使うより、パフォーマンスが向上する場合があります。

4 標識の表を参照するために使う添え字に意味のある名前を与えるために、WORKING-STORAGE SECTION に一連のデータ項目が定義されています。このようなデータ項目を使うことは必要ではありません。

5 FORMAT1 のための別個の標識域の INDIC-TABLE (01) は、月の最初の日である場合に、オンに設定されます。

6 標識値をワークステーションのディスプレイ装置へ送るため、INDICATOR 句が必要です。

7 標識値をワークステーションのディスプレイ装置から受け取るため、INDICATOR 句が必要です。 F5 キーが押された場合、INDIC-TABLE (51) はオンに設定されます。

8 INDIC-TABLE (51) がオンに設定されている場合、プログラム DAILY が呼び出されます。

サブファイル・トランザクション・ファイルの使用

サブファイルは、ディスプレイ装置との間で読み書きする一群のレコードのことです。プログラムは一度に 1 レコードだけを処理しますが、オペレーティング・システムやワークステーションは複数レコードのブロックを送ったり受け取ったりします。一度にディスプレイ装置で表示できるより多くのレコードが送信された場合、ワークステーションのオペレーターは、プログラムに制御を戻すことなく、複数レコードのブロックをページ送りすることができます。

サブファイルの使用は、多数の同じようなレコードをディスプレイ装置に書き込んだりディスプレイ装置から読み取ったりするときに便利です。サブファイルは、レコードを順次アクセスしたり、相対キー値によってランダム・アクセスしたりすることのできるディスプレイ・ファイルです。

たとえば、昨年 $5000 以上を購入した得意先のすべてを表示したいとします。データベースの QUERY を実行して、そのような得意先のすべての名前を取得し、WRITE SUBFILE 操作をサブファイルに対して実行することによってそれらを特別のファイル (サブファイル) に入れることができます。このようにすれば、サブファイル制御レコードに対して WRITE 操作を実行することによって、サブファイルの内容全体をディスプレイ装置に書き込むことができます。その後、サブファイル制御レコードに対して READ 操作を使用することにより、ユーザーの修正した得意先リストを読み込んでから、READ SUBFILE 操作を使用してサブファイルから個々のレコードを取り出すことができます。

ディスプレイ装置上で同じタイプの複数のレコードを取り扱うことができるようにするためのサブファイルは、ディスプレイ・ファイルの DDS 中で指定できます。サブファイル表示の例については 610 ページの図 140 を参照してください。

サブファイル中に含められるレコード様式は、そのファイルの DDS 中で定義されます。サブファイルに含めることのできるレコード数も、DDS 中で指定しなければなりません。 1 つのファイルの中に、複数のサブファイルを含めることができます。しかし、1 つの装置については、同時に活動状態にできるのは 12 個のサブファイルまでです。

データ記述仕様を使用したサブファイルの定義

サブファイルの DDS は、サブファイル・レコード様式とサブファイル制御レコード様式という 2 つのレコード様式から構成されています。

サブファイル・レコード様式には、そのサブファイル中のレコードのフィールド記述が含まれています。 READ、WRITE、または REWRITE でサブファイル・レコード様式を指定すると、指定されたサブファイル・レコードが処理されますが、表示されるデータは直接影響を受けません。

READ または WRITE のステートメントにサブファイル制御レコード様式を指定すると、サブファイルの物理的な読み取り、書き込み、またはセットアップ操作が行われます。 612 ページの図 141 には、サブファイル・レコード様式の DDS と、サブファイル制御レコード様式の例が示されています。

サブファイル内のレコードの表示方法、およびサブファイルに指定可能なキーワードについては、 Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベースおよびファイル・システム」カテゴリーを参照してください。

```
Customer Name Search
Search Code  _____

Number Name                 Address              City                 State

XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
XXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX
```

図 140. サブファイルの表示

ディスプレイ・ファイルのためのサブファイルの使用

ILE COBOL プログラムのディスプレイ・ファイルのためのサブファイルを使うには、入出力操作で SUBFILE 句を指定する必要があります。有効なサブファイル操作は、次のとおりです。

- READ SUBFILE ファイル名 RECORD
- WRITE SUBFILE レコード名
- REWRITE SUBFILE レコード名

サブファイルは、READ SUBFILE NEXT MODIFIED ステートメントを使って順次処理したり、相対キー値を指定してランダム処理したりすることができます。相対キーは、ファイル中のレコードを検索するためにシステムが直接使用することのできる符号なし数値です。

TRANSACTION ファイルは、外部的に記述されたファイルでなければなりません。ILE COBOL において、サブファイルへのアクセスは、READ SUBFILE NEXT MODIFIED を使用した場合を除き、相対レコード番号によってなされます。TRANSACTION ファイルで SUBFILE 句を使用した場合、ENVIRONMENT DIVISION の SELECT ステートメントにおいて、ACCESS MODE IS DYNAMIC と記述し、RELATIVE KEY を指定しなければなりません。

ディスプレイ・ファイルが複数のディスプレイ装置を獲得している場合、各ディスプレイ装置ごとに別個のサブファイルが存在します。 TRANSACTION ファイルによって獲得された特定のディスプレイ装置のためにサブファイルが作成されている場合、そのサブファイルのレコード様式を参照するすべての入力操作は、その装置に属しているサブファイルに対して実行されます。サブファイルとして指定されていないレコード様式名を参照するすべての操作は、ディスプレイ装置に対する直接的な入出力操作として処理されます。

サブファイルの一般的な使用方法について表 32 にまとめています。

表 32. サブファイルの使用方法

用途	**意味**
表示専用	ワークステーション・ユーザーが表示画面を見る。
選択して表示	表示画面上の 1 つの項目の詳細情報をユーザーが要求する。
修正	ユーザーが 1 つまたは複数のレコードを修正する。
入力専用 (妥当性検査なし)	サブファイルはデータ入力機能として使われる。
入力専用 (妥当性検査あり)	サブファイルはデータ入力機能として使われ、同時にレコードが検査される。
複数のタスクの組み合わせ	サブファイルを修正可能な表示画面として使える。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A* THIS IS THE DISPLAY DEVICE FILE FOR PAYUPDT ** PAYUPDTD
     A* ACCOUNTS RECEIVABLE INTERACTIVE PAYMENT UPDATE
     A*
     A
     A          R SUBFILE1                  SFL  1
     A                                      TEXT('SUBFILE FOR CUSTOMER PAYMENT'
     A*
     A            ACPPMT         4A  I  5  4TEXT('ACCEPT PAYMENT')
     A                                  2   VALUES('*YES' '*NO')  3
     A  51                                  DSPATR(RI MDT)
     A N51                                  DSPATR(ND PR)
     A*
     A            CUST           5   B  5 15TEXT('CUSTOMER NUMBER')
     A  52                          4       DSPATR(RI)
     A  53                                  DSPATR(ND)
     A  54                                  DSPATR(PR)
     A*
     A            AMPAID         8 02B  5 24TEXT('AMOUNT PAID')
     A                                      CHECK(FE)  5
     A                                      AUTO(RAB)  6
     A                                      CMP(GT 0)  7
     A  52                                  DSPATR(RI)
     A  53                                  DSPATR(ND)
     A  54                                  DSPATR(PR)
     A*
     A            ECPMSG        31A  O  5 37TEXT('EXCEPTION MESSAGE')
     A  52                                  DSPATR(RI)
     A  53                                  DSPATR(ND)
     A  54                                  DSPATR(BL)
     A*
     A            OVRPMT         8Y 2O  5 70TEXT('OVERPAYMENT')
     A                                      EDTCDE(1)  8
     A  55                                  DSPATR(BL)  9
     A N56                                  DSPATR(ND)
     A*
     A            STSCDE         1A  H      TEXT('STATUS CODE')
     A          R CONTROL1                  TEXT('SUBFILE CONTROL')
     A                                      SFLCTL(SUBFILE1)  10
     A                                      SFLSIZ(17)  11
     A                                      SFLPAG(17)  12
     A  61                                  SFLCLR  13
     A  62                                  SFLDSP  14
     A  62                                  SFLDSPCTL  15
     A                                      OVERLAY
     A                                      LOCK  16
     A*
     A                                      HELP(99 'HELP KEY')  17
     A                                      CA12(98 'END PAYMENT UPDATE')
     A                                      CA11(97 'IGNORE INPUT')
     A*                                      18
     A  99                                  SFLMSG(' F11 - IGNORE INVALID INPUT+
     A                                                   F12 - END PAYMENT +
     A                                      UPDATE')
```

図 141. サブファイル・レコード様式のデータ記述仕様 (1/2)

```
     A*
     A                                    1  2'CUSTOMER PAYMENT UPDATE PROMPT'
     A                                    1 65'DATE'
     A                                    1 71DATE EDTCDE(Y)
     A  63                                3  2'ACCEPT'
     A  63                                4  2'PAYMENT'
     A                                    3 14'CUSTOMER'
     A                                    3 26'PAYMENT'
     A  64                                3 37'EXCEPTION MESSAGE'
     A*
     A          R MESSAGE1                    TEXT('MESSAGE RECORD')
     A                                        OVERLAY
     A                                        LOCK
     A*
     A  71                               24  2' ACCEPT PAYMENT VALUES: (*NO
*YES)
DSPATR(RI)
```

図 141. サブファイル・レコード様式のデータ記述仕様 (2/2)

サブファイル・レコード様式のデータ記述仕様 (DDS) は、サブファイルの中のレコードを次のように記述します。

1 SFL キーワードによって、そのレコード様式がサブファイルとして識別されます。

2 行および位置の項目によって、ディスプレイ装置上でのフィールドの位置が識別されます。

3 VALUES キーワードは、ACPPMT フィールドの値としてユーザーが指定できるのが *YES または *NO だけであることを指定します。

4 使用法項目は、名前の指定されたフィールドが出力 (O)、入力 (I)、入出力 (B)、または隠し (H) フィールドになることを定義します。

5 項目 CHECK(FE) は、いずれかのフィールド終了キーを押さなければ、ユーザーが次の入力フィールドにスキップできないように指定します。

6 項目 AUTO(RAB) は、AMPAID というフィールドに入力されたデータが自動的に右寄せされ、先行文字がブランクで埋められるように指定します。

7 項目 CMP(GT 0) は、AMPAID フィールドに入力されたデータがゼロと比較され、それがゼロよりも大きいようにすることを指定します。

8 EDTCDE キーワードは、出力フィールド OVRPMT のための編集を指定します。 EDTCDE(1) は、フィールド OVRPMT 印刷時にコンマと小数点を使用し、符号は付けないで印刷することを示します。また、残高ゼロも印刷され、先行ゼロは消去されます。

9 DSPATR キーワードは、名前の指定されたフィールドに対応する標識の状況が真であるときに、そのフィールドの表示属性を指定するために使用されています。指定される属性は、次のとおりです。
- BL (明滅)
- RI (反転表示)
- PR (保護)
- MDT (変更データ・タグ設定)
- ND (非表示)

サブファイル制御レコード様式は、サブファイル、検索入力フィールド、定数、およびコマンド・キーの属性を定義します。使用されるキーワードは、次のものを示します。

10 SFLCTL は、このレコードがサブファイル制御レコードであることを識別し、関連付けられたサブファイル・レコード (SUBFILE1) の名前を指定します。

11 SFLSIZ は、サブファイルに含まれるレコードの合計数を示します (17)。

12 SFLPAG は、1 ページ中のレコードの合計数を示します (17)。

13 SFLCLR は、サブファイルがクリアされる場合 (標識 61 がオンになる場合) を示します。

14 SFLDSP は、サブファイルを表示する場合 (標識 62 がオンになる場合) を示します。

15 SFLDSPCTL は、サブファイル制御レコードを表示する場合 (標識 62 がオンになる場合) を示します。

16 LOCK キーワードは、CONTROL1 レコード様式が最初に表示された時点でワークステーション・ユーザーがキーボードを使えないようにします。

17 HELP を指定すると、ユーザーはヘルプ・キーを押して標識 99 をオンに設定できるようになります。

18 SFLMSG は、標識 99 がオンである場合に表示されるメッセージとして定数を指定します。

サブファイル制御レコード様式は、制御情報に加えて、サブファイル・レコード様式の列見出しとして使用される固定情報を定義します。 サブファイル制御レコード様式の例については 612 ページの図 141 を参照してください。

単一装置ファイルおよび複数装置ファイルへのアクセス

単一装置ファイルとは、それに対して定義されているプログラム装置が 1 つだけである装置ファイルです。プリンター・ファイル、ディスケット・ファイル、およびテープ・ファイルは単一装置ファイルです。プログラム装置の最大数を 1 に指定して作成されたディスプレイ・ファイルやシステム間通信機能 (ICF) ファイルも、単一装置ファイルです。

複数装置ファイルとは、ディスプレイ・ファイルまたはシステム間通信機能 (ICF) ファイルのいずれかです。複数装置ファイルは、複数のプログラム装置を獲得することができます。複数装置ファイルの使用例については 618 ページの図 145 を参照してください。

CRTDSPF コマンドの MAXDEV パラメーターが 1 より大きい場合、ディスプレイ・ファイルに複数のプログラム装置が可能です。このコマンドの DEV パラメーターに対して *NONE を指定した場合、このファイルに関連付けられているフィールドを使用する*前*に、ディスプレイ装置の名前を指定しなければなりません。

ディスプレイ・ファイルの作成および使用方法の詳細については、Web サイト
http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベースおよびファイル・システム」カテゴリーを参照してください

CRTICFF コマンドの MAXPGMDEV パラメーターが 1 より大きい場合、ICF ファイルには複数のプログラム装置が可能です。 ICF ファイルの作成と使用法の詳細については、「*ICF Programming*」を参照してください。

ILE COBOL では、あるファイルが単一装置ファイルか複数装置ファイルかを、そのファイルに複数の装置が*可能*かどうか に基づいて、実行時に判別します。実際に獲得された装置の数は、そのファイルが単一装置ファイルか複数装置ファイルかについての判断に影響を及ぼしません。ファイルが単一装置ファイルか複数装置ファイルかは、コンパイル時には判別*されません*。この判別は、ディスプレイ装置または ICF ファイルの現行記述に基づきます。

複数装置ファイルの場合、特定のプログラム装置を入出力ステートメントで使用するつもりなら、その装置を TERMINAL 句で指定します。単一装置ファイルについても、TERMINAL 句を指定することができます。

以下のページにおいて、複数装置ファイルの使用方法を説明するための例を示します。このプログラムではディスプレイ・ファイルを使用し、バッチ・モードで実行されるようになっています。このプログラムは端末装置を獲得し、サインオン画面を使ってそれらの端末装置を送信勧誘します。端末装置が送信勧誘された後、それらはポーリングされます。 待ち時間が終わるまでにサインオンした人がいない場合、プログラムは終了します。 有効なパスワードを入力すると、別の ILE COBOL プログラムを呼び出して従業員ファイルを更新することができるようになります。更新が完了したなら、再び装置が送信勧誘され、再び端末がポーリングされます。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
    A*            THIS IS THE MULTIPLE DEVICE DISPLAY FILE
    A*
    A*                  DDS FOR THE MIXED FILE MULT
    A*
    A
    A          R SIGNON                    INVITE 1
    A                                 O  5 20'                    '
    A                                        DSPATR(RI)
    A                                 O  6 20'  '
    A                                        DSPATR(RI)
    A                                 O  6 38'  '
    A                                        DSPATR(RI)
    A                                 O  7 20'  '
    A                                        DSPATR(RI)
    A                                 O  7 27'M D F'
    A                                        DSPATR(HI BL)
    A                                 O  7 38'  '
    A                                        DSPATR(RI)
    A                                 O  9 20'                    '
    A                                        DSPATR(RI)
    A                                 O 20 20'PLEASE LOGON'
    A                                        DSPATR(HI)
    A            PASSWORD      10A  I 20 43DSPATR(PC ND)
    A            WRONG         20A  O 21 43
    A
    A          R UPDATE
    A                                 O  3  5'UPDATE OF PERSONNEL FILE'
    A                                        DSPATR(BL)
    A                                 O  7  5'TYPE IN EMPLOYEE NUMBER TO BE +
    A                                          UPDATED'
    A            NUM            7A  I  7 44DSPATR(RI PC)
    A
    A          R EMPLOYEE
    A                                 O  3  5'EMPLOYEE NUMBER'
    A            NUM            7A  B  3 25DSPATR(PC)
    A                                 O  5  5'EMPLOYEE NAME'
    A            NAME          30A  B  5 25DSPATR(PC)
    A                                 O  7  5'EMPLOYEE ADDRESS'
    A                                 O  9  5'STREET'
    A            STREET        30A  B  9 25DSPATR(PC)
    A                                 O 11  5'APARTMENT NUMBER'
    A            APTNO          5A  B 11 25DSPATR(PC)
    A                                 O 13  5'CITY'
    A            CITY          20A  B 13 25DSPATR(PC)
    A                                 O 15  5'PROVINCE'
    A            PROV          20A  B 15 25DSPATR(PC)
    A
    A          R RECOVERY
    A                                 O  3  5'THE EMPLOYEE NUMBER YOU HAVE GIVEN
    A                                          IS INVALID'
    A                                 O  6  5'TYPE Y TO RETRY'
    A                                 O  8  5'TYPE N TO EXIT'
    A            ANSWER         1X  I 10  5DSPATR(RI PC)
    A                                        VALUES('Y' 'N')
```

*図 142. 複数装置ファイルの使用例 ** ディスプレイ・ファイル*

1 様式 SIGNON には、キーワード INVITE が関連付けられています。このことは、WRITE ステートメントで様式 SIGNON が使用されるとその書き込み先装置が送信勧誘されるということを意味します。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
    A** DDS FOR THE PHYSICAL FILE PASSWORD
    A*
    A*
    A                                       UNIQUE
    A          R PASSWORDS
    A            PASSKEY       10
    A            PASSWORD      10
    A          K PASSKEY
    A
```

*図 143. 複数装置ファイルの使用例 ** 物理ファイル PASSWORD*

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
    A*  DDS FOR THE PHYSICAL FILE TERM
    A*  WHICH CONTAINS THE LIST OF TERMINALS
    A*
    A
    A          R TERM
    A            TERM          10
```

*図 144. 複数装置ファイルの使用例 ** 物理ファイル TERM*

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/SAMPMDF         ISERIES1   06/02/15 15:04:02        ページ     2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     SAMPMDF.
          000300
          000400*************************************************************
          000500*  THE FOLLOWING PROGRAM DEMONSTRATES SOME OF THE FUNCTIONS  *
          000600*  AVAILABLE WITH MULTIPLE DEVICE FILE SUPPORT.              *
          000700*************************************************************
          000800
    3     000900 ENVIRONMENT DIVISION.
    4     001000 CONFIGURATION SECTION.
    5     001100 SOURCE-COMPUTER. IBM-ISERIES
    6     001200 OBJECT-COMPUTER. IBM-ISERIES
    7     001300 SPECIAL-NAMES.  ATTRIBUTE-DATA IS ATTR.  1
    9     001400 INPUT-OUTPUT SECTION.
   10     001500 FILE-CONTROL.
   11     001600     SELECT MULTIPLE-FILE
   12     001700            ASSIGN TO WORKSTATION-MULT
   13     001800            ORGANIZATION IS TRANSACTION  2
   14     001900            ACCESS MODE IS SEQUENTIAL
   15     002000            FILE STATUS IS MULTIPLE-FS1, MULTIPLE-FS2  3
   16     002100            CONTROL-AREA IS MULTIPLE-CONTROL-AREA.  4
          002200
   17     002300     SELECT TERMINAL-FILE
   18     002400            ASSIGN TO DATABASE-TERM
   19     002500            ORGANIZATION IS SEQUENTIAL
   20     002600            ACCESS IS SEQUENTIAL
   21     002700            FILE STATUS IS TERMINAL-FS1.
          002800
   22     002900     SELECT PASSWORD-FILE
   23     003000            ASSIGN TO DATABASE-PASSWORD
   24     003100            ORGANIZATION IS INDEXED
   25     003200            RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
   26     003300            ACCESS MODE IS RANDOM
   27     003400            FILE STATUS IS PASSWORD-FS1.
          003500
   28     003600     SELECT PRINTER-FILE
   29     003700            ASSIGN TO PRINTER-QPRINT.
          003800
   30     003900 DATA DIVISION.
   31     004000 FILE SECTION.
   32     004100 FD  MULTIPLE-FILE.
   33     004200 01  MULTIPLE-REC.
          004200                COPY DDS-SIGNON OF MULT.  5
   34    +000001       05  MULT-RECORD PIC X(20).                                  SIGNON
         +000002*  INPUT FORMAT:SIGNON     FROM FILE MULT       OF LIBRARY CBLGUIDE  SIGNON
         +000003*                                                                  SIGNON
   35    +000004       05  SIGNON-I      REDEFINES MULT-RECORD.                   SIGNON
   36    +000005           06 PASSWORD              PIC X(10).  6                  SIGNON
         +000006* OUTPUT FORMAT:SIGNON     FROM FILE MULT       OF LIBRARY CBLGUIDE  SIGNON
         +000007*                                                                  SIGNON
   37    +000008       05  SIGNON-O      REDEFINES MULT-RECORD.                   SIGNON
   38    +000009           06 WRONG                 PIC X(20).                     SIGNON
         004300
```

図 145. 複数装置ファイル・サポートの ILE COBOL ソース・リスト (1/5)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/SAMPMDF         ISERIES1   06/02/15 15:04:02        ページ    3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   39     004400 FD  TERMINAL-FILE.
   40     004500 01  TERMINAL-REC.
          004500                  COPY DDS-ALL-FORMATS OF TERM.
   41    +000001       05  TERM-RECORD PIC X(10).                                        <-ALL-FMTS
         +000002*    I-O FORMAT:TERM       FROM FILE TERM       OF LIBRARY CBLGUIDE      <-ALL-FMTS
         +000003*                                                                       <-ALL-FMTS
   42    +000004       05  TERM          REDEFINES TERM-RECORD.                        <-ALL-FMTS
   43    +000005           06 TERM                 PIC X(10).                          <-ALL-FMTS
          004600
   44     004700 FD  PASSWORD-FILE.
   45     004800 01  PASSWORD-REC.
          004800                  COPY DDS-ALL-FORMATS OF PASSWORD.
   46    +000001       05  PASSWORD-RECORD PIC X(20).                                    <-ALL-FMTS
         +000002*    I-O FORMAT:PASSWORDS  FROM FILE PASSWORD   OF LIBRARY CBLGUIDE      <-ALL-FMTS
         +000003*                                                                       <-ALL-FMTS
         +000004*THE KEY DEFINITIONS FOR RECORD FORMAT  PASSWORDS                       <-ALL-FMTS
         +000005*  NUMBER              NAME               RETRIEVAL       ALTSEQ      <-ALL-FMTS
         +000006*   0001   PASSKEY                        ASCENDING         NO         <-ALL-FMTS
   47    +000007       05  PASSWORDS     REDEFINES PASSWORD-RECORD.                    <-ALL-FMTS
   48    +000008           06 PASSKEY              PIC X(10).                          <-ALL-FMTS
   49    +000009           06 PASSWORD             PIC X(10).                          <-ALL-FMTS
          004900
   50     005000 FD  PRINTER-FILE.
   51     005100 01  PRINTER-REC.
   52     005200    05 PRINTER-RECORD        PIC X(132).
          005300
   53     005400 WORKING-STORAGE SECTION.
          005500
          005600*****************************************************************
          005700*       DECLARE THE FILE STATUS FOR EACH FILE                   *
          005800*****************************************************************
          005900
   54     006000 01  MULTIPLE-FS1           PIC X(2)    VALUE SPACES.
   55     006100 01  MULTIPLE-FS2.  7
   56     006200    05  MULTIPLE-MAJOR       PIC X(2)    VALUE SPACES.
   57     006300    05  MULTIPLE-MINOR       PIC X(2)    VALUE SPACES.
   58     006400 01  TERMINAL-FS1            PIC X(2)    VALUE SPACES.
   59     006500 01  PASSWORD-FS1            PIC X(2)    VALUE SPACES.
          006600
          006700*****************************************************************
          006800*       DECLARE STRUCTURE FOR HOLDING FILE ATTRIBUTES           *
          006900*****************************************************************
          007000
   60     007100 01  STATION-ATTR.
   61     007200    05 STATION-TYPE          PIC X(1).  8
   62     007300    05 STATION-SIZE          PIC X(1).
   63     007400    05 STATION-LOC           PIC X(1).
   64     007500    05 FILLER                PIC X(1).
   65     007600    05 STATION-ACQUIRE       PIC X(1).
   66     007700    05 STATION-INVITE        PIC X(1).
   67     007800    05 STATION-DATA          PIC X(1).
   68     007900    05 STATION-STATUS        PIC X(1).
   69     008000    05 STATION-DISPLAY       PIC X(1).
   70     008100    05 STATION-KEYBOARD      PIC X(1).
   71     008200    05 STATION-SIGNON        PIC X(1).
```

図 145. 複数装置ファイル・サポートの ILE COBOL ソース・リスト (2/5)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/SAMPMDF         ISERIES1   06/02/15 15:04:02        ページ     4
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
   72     008300    05 FILLER               PIC X(5).
          008400
          008500*************************************************************
          008600*       DECLARE THE CONTROL AREA FOR MULTIPLE-FILE           *
          008700*************************************************************
          008800
   73     008900 01  MULTIPLE-CONTROL-AREA.
   74     009000    05 MULTIPLE-KEY-FEEDBACK PIC X(2)    VALUE SPACES.
   75     009100    05 MULTIPLE-DEVICE-NAME  PIC X(10)   VALUE SPACES.
   76     009200    05 MULTIPLE-FORMAT-NAME  PIC X(10)   VALUE SPACES.
          009300
          009400*************************************************************
          009500*               DECLARE ERROR REPORT VARIABLES              *
          009600*************************************************************
          009700
   77     009800 01  HEADER-LINE.
   78     009900    05 FILLER                PIC X(60)   VALUE SPACES.
   79     010000    05 FILLER                PIC X(72)
          010100                              VALUE "MDF ERROR REPORT".
   80     010200 01  DETAIL-LINE.
   81     010300    05 FILLER                PIC X(15)   VALUE SPACES.
   82     010400    05 DESCRIPTION           PIC X(25)   VALUE SPACES.
   83     010500    05 DETAIL-VALUE          PIC X(92)   VALUE SPACES.
          010600
          010700*************************************************************
          010800*        DECLARE COUNTERS, FLAGS AND STORAGE VARIABLES       *
          010900*************************************************************
          011000
   84     011100 01  CURRENT-TERMINAL        PIC X(10)   VALUE SPACES.
   85     011200 01  TERMINAL-ARRAY.
   86     011300    05 LIST-OF-TERMINALS OCCURS 250 TIMES.
   87     011400      07 DEVICE-NAME         PIC X(10).
   88     011500 01  COUNTER                 PIC 9(3)    VALUE IS 1.
   89     011600 01  NO-OF-TERMINALS         PIC 9(3)    VALUE IS 1.
   90     011700 01  TERMINAL-LIST-FLAG      PIC 1.
   91     011800    88 END-OF-TERMINAL-LIST              VALUE IS B"1".
   92     011900    88 NOT-END-OF-TERMINAL-LIST          VALUE IS B"0".
   93     012000 01  NO-DATA-FLAG            PIC 1.
   94     012100    88 NO-DATA-AVAILABLE                 VALUE IS B"1".
   95     012200    88 DATA-AVAILABLE                    VALUE IS B"0".
          012300
   96     012400 PROCEDURE DIVISION.
          012500
   97     012600 DECLARATIVES.
          012700
          012800 MULTIPLE-SECTION SECTION.
          012900     USE AFTER STANDARD EXCEPTION PROCEDURE ON MULTIPLE-FILE.
          013000
          013100 MULTIPLE-PARAGRAPH.
   98     013200     WRITE PRINTER-REC FROM HEADER-LINE AFTER ADVANCING PAGE.
   99     013300     MOVE "FILE NAME IS:" TO DESCRIPTION OF DETAIL-LINE.
  100     013400     MOVE "MULTIPLE FILE" TO DETAIL-VALUE OF DETAIL-LINE.
  101     013500     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 5 LINES.
  102     013600     MOVE "FILE STATUS IS:" TO DESCRIPTION OF DETAIL-LINE.
  103     013700     MOVE MULTIPLE-FS1 TO DETAIL-VALUE OF DETAIL-LINE.
```

図 145. 複数装置ファイル・サポートの ILE COBOL ソース・リスト (3/5)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/SAMPMDF         ISERIES1   06/02/15 15:04:02        ページ    5
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
  104     013800     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
  105     013900     MOVE "EXTENDED STATUS IS:" TO DESCRIPTION OF DETAIL-LINE.  9
  106     014000     MOVE MULTIPLE-FS2 TO DETAIL-VALUE OF DETAIL-LINE.
  107     014100     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
  108     014200     ACCEPT STATION-ATTR FROM ATTR.  10
  109     014300     MOVE "FILE ATTRIBUTES ARE:" TO DESCRIPTION OF DETAIL-LINE.
  110     014400     MOVE STATION-ATTR TO DETAIL-VALUE OF DETAIL-LINE.
  111     014500     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
  112     014600     STOP RUN.
          014700
          014800 TERMINAL-SECTION SECTION.
          014900     USE AFTER STANDARD EXCEPTION PROCEDURE ON TERMINAL-FILE.
          015000 TERMINAL-PARAGRAPH.
  113     015100     WRITE PRINTER-REC FROM HEADER-LINE AFTER ADVANCING PAGE.
  114     015200     MOVE "FILE NAME IS:" TO DESCRIPTION OF DETAIL-LINE.
  115     015300     MOVE "TERMINAL FILE" TO DETAIL-VALUE OF DETAIL-LINE.
  116     015400     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 5 LINES.
  117     015500     MOVE "FILE STATUS IS:" TO DESCRIPTION OF DETAIL-LINE.
  118     015600     MOVE TERMINAL-FS1 TO DETAIL-VALUE OF DETAIL-LINE.
  119     015700     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
  120     015800     STOP RUN.
          015900
          016000 PASSWORD-SECTION SECTION.
          016100     USE AFTER STANDARD EXCEPTION PROCEDURE ON PASSWORD-FILE.
          016200 PASSWORD-PARAGRAPH.
  121     016300     WRITE PRINTER-REC FROM HEADER-LINE AFTER ADVANCING PAGE.
  122     016400     MOVE "FILE NAME IS:" TO DESCRIPTION OF DETAIL-LINE.
  123     016500     MOVE "PASSWORD FILE" TO DETAIL-VALUE OF DETAIL-LINE.
  124     016600     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 5 LINES.
  125     016700     MOVE "FILE STATUS IS:" TO DESCRIPTION OF DETAIL-LINE.
  126     016800     MOVE PASSWORD-FS1 TO DETAIL-VALUE OF DETAIL-LINE.
  127     016900     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
  128     017000     STOP RUN.
          017100
          017200 END DECLARATIVES.
          017300
          017400*****************************************************************
          017500*        MAIN PROGRAM LOGIC BEGINS HERE                         *
          017600*****************************************************************
          017700
          017800 MAIN-PROGRAM SECTION.
          017900 MAINLINE.
  129     018000     OPEN I-O    MULTIPLE-FILE  11
          018100          INPUT  TERMINAL-FILE
          018200          I-O    PASSWORD-FILE
          018300          OUTPUT PRINTER-FILE.
          018400
  130     018500       MOVE 1 TO COUNTER.
  131     018600       SET NOT-END-OF-TERMINAL-LIST TO TRUE.
          018700******
          018800*      Fill Terminal List
          018900******
  132     019000       PERFORM UNTIL END-OF-TERMINAL-LIST
  133     019100          READ TERMINAL-FILE RECORD
          019200             INTO LIST-OF-TERMINALS(COUNTER)
```

図 145. 複数装置ファイル・サポートの ILE COBOL ソース・リスト (4/5)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/SAMPMDF         ISERIES1   06/02/15 15:04:02        ページ     6
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
         019300             AT END
  134    019400                SET END-OF-TERMINAL-LIST TO TRUE
  135    019500                SUBTRACT 1 FROM COUNTER
  136    019600                MOVE COUNTER TO NO-OF-TERMINALS
         019700          END-READ
  137    019800          ADD 1 TO COUNTER
         019900       END-PERFORM.
         020000******
         020100*      Acquire and invite terminals
         020200******
  138    020300       PERFORM VARYING COUNTER FROM 1 BY 1
         020400               UNTIL COUNTER GREATER THAN NO-OF-TERMINALS
  139    020500          ACQUIRE LIST-OF-TERMINALS(COUNTER) FOR MULTIPLE-FILE   12
  140    020600          WRITE MULTIPLE-REC   13
         020700             FORMAT IS "SIGNON"
         020800             TERMINAL IS LIST-OF-TERMINALS(COUNTER)
         020900          END-WRITE
         021000       END-PERFORM.
         021100
  141    021200       MOVE 1 TO COUNTER.
  142    021300       SET DATA-AVAILABLE TO TRUE.
         021400******
         021500*      Poll terminals
         021600******
  143    021700       PERFORM UNTIL NO-DATA-AVAILABLE
  144    021800          READ MULTIPLE-FILE RECORD   14
  145    021900          IF MULTIPLE-FS2 EQUAL "310" THEN
  146    022000             SET NO-DATA-AVAILABLE TO TRUE   15
         022100          END-IF
  147    022200          IF DATA-AVAILABLE THEN
  148    022300             MOVE MULTIPLE-DEVICE-NAME TO CURRENT-TERMINAL
         022400******
         022500*            Validate Password   16
         022600******
  149    022700             MOVE CURRENT-TERMINAL TO PASSKEY OF PASSWORD-REC
  150    022800             READ PASSWORD-FILE RECORD
  151    022900             IF PASSWORD OF SIGNON-I EQUAL
         023000                PASSWORD OF PASSWORD-REC THEN
  152    023100                   CALL "UPDT" USING CURRENT-TERMINAL
  153    023200                   MOVE SPACES TO WRONG OF SIGNON-O
         023300             ELSE
  154    023400                   MOVE "INVALID PASSWORD" TO WRONG OF SIGNON-O
         023500          END-IF
  155    023600             WRITE MULTIPLE-REC
         023700                FORMAT IS "SIGNON"
         023800                TERMINAL IS CURRENT-TERMINAL
         023900             END-WRITE
         024000          END-IF
         024100       END-PERFORM.
         024200******
         024300*      Drop terminals
         024400******
  156    024500       PERFORM VARYING COUNTER FROM 1 BY 1
         024600               UNTIL COUNTER GREATER THAN NO-OF-TERMINALS
  157    024700          DROP LIST-OF-TERMINALS(COUNTER) FROM MULTIPLE-FILE   17
         024800       END-PERFORM.
         024900
  158    025000       CLOSE   MULTIPLE-FILE
         025100               TERMINAL-FILE
         025200               PASSWORD-FILE
         025300               PRINTER-FILE.
  159    025400       STOP RUN.
         025500
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 145. 複数装置ファイル・サポートの ILE COBOL ソース・リスト (5/5)

1 ATTR は関数名 ATTRIBUTE-DATA に関連付けられた簡略名です。 ATTR は、TRANSACTION ファイル MULTIPLE-FILE の属性データを入手するための ACCEPT ステートメントで使用されます。項目 **10** を参照してください。

2 ファイル MULT は CRTDSPF コマンドを使って作成されていなければなりません。この場合、DEV パラメーターの値は *NONE、MAXDEV パラメ

ーターは 1 よりも大きな値です。 WAITRCD パラメーターは、ファイルの READ 操作の待ち時間を指定します。 WAITRCD パラメーターは 0 より大きな値でなければなりません。

3 MULTIPLE-FS2 は、TRANSACTION ファイルの MULTIPLE-FILE の拡張ファイル状況です。この変数はプログラムの WORKING-STORAGE SECTION で宣言されています。項目 **7** を参照してください。

4 MULTIPLE-CONTROL-AREA は、TRANSACTION ファイル MULTIPLE-FILE のための制御域です。この変数は、サインオンするのにどのプログラム装置が使用されたかを判別するために使用されます。項目 **15** を参照してください。

5 MULTIPLE-REC のデータ記述は、COPY DDS ステートメントを使用して定義されています。

注: 名前の指定されているフィールドは、コピーされたフィールドだけです。使用された DDS に関するコメントについては、この例の DDS を参照してください。

6 様式 SIGNON は、INVITE キーワードが指定されている様式です。この様式は、WRITE ステートメントを使って装置を送信勧誘するのに使用されます。

7 これは、拡張ファイル状況 MULTIPLE-FS2 の宣言です。メジャー戻りコード (最初の 2 バイト) とマイナー戻りコード (最後の 2 バイト) とで構成される 4 バイトのフィールドです。

8 STATION-ATTR は、ACCEPT ステートメントが TRANSACTION ファイル MULTIPLE-FILE の属性データを格納する場所です。項目 **10** を参照してください。

9 このステートメントで、拡張ファイル状況 MULTIPLE-FS2 が書き込まれています。

10 このステートメントで、TRANSACTION ファイル MULTIPLE-FILE の属性データを受け入れています。この ACCEPT ステートメントに FOR 句が指定されていないため、最後のプログラム装置が使用されます。

11 このステートメントによって TRANSACTION ファイル MULTIPLE-FILE がオープンされます。 CRTDSPF コマンドの ACQPGMDEV パラメーターの値が *NONE なので、このファイルがオープンされても、暗黙のうちに獲得されるプログラム装置はありません。

12 このステートメントは、変数 LIST-OF-TERMINALS (COUNTER) 中に含まれているプログラム装置を TRANSACTION ファイル MULTIPLE-FILE のために獲得します。

13 この WRITE ステートメントは、TERMINAL 句で指定されているプログラム装置を送信勧誘します。様式 SIGNON には、DDS キーワード INVITE が関連付けられています。項目 **14** を参照してください。

14 この READ ステートメントによって、送信勧誘されたプログラム装置からの読み込みが行われます。項目 **13** を参照してください。送信勧誘された装置に入力される前に待ち時間が終ったなら、拡張ファイル状況は “0310” に設定され、処理が続行されます。項目 **15** を参照してください。

15 このステートメントでは、待ち時間が終ったかどうかを調べるために、MULTIPLE-FILE の拡張ファイル状況を調べています。

16 制御域に保管されているプログラム装置名によって、サインオンするのにどのプログラム装置が使われたかを調べます。項目 **4** を参照してください。

17 この DROP ステートメントによって、変数 LIST-OF-TERMINALS に含まれているプログラム装置が、TRANSACTION ファイル MULTIPLE-FILE から切り離されます。

サブファイル・トランザクション・ファイルを使用したプログラムの作成

多くの場合、一群のレコードをディスプレイ装置から読み込んだり、一群のレコードをディスプレイ装置に書き込んだりするには、サブファイル TRANSACTION ファイルを使用します。 ILE COBOL プログラムの中でサブファイル TRANSACTION ファイルを使うには、次のようにする必要があります。

- 環境部 (ENVIRONMENT DIVISION) の FILE-CONTROL 段落のファイル制御記入項目を使用してファイルの名前を指定する。
- データ部 (DATA DIVISION) のファイル記述項目を使用してファイルを記述する。
- トランザクション処理をサポートするための、PROCEDURE DIVISION のステートメントに対する拡張機能を使用する。

サブファイル・トランザクション・ファイルの名前の指定

ILE COBOL プログラムでサブファイル TRANSACTION ファイルを使用するには、FILE-CONTROL 段落のファイル制御項目を使用してそのファイルの名前を指定する必要があります。 FILE-CONTROL 段落の説明については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

TRANSACTION ファイルの名前は、FILE-CONTROL 段落の中で以下のようにして指定します。

```
FILE-CONTROL.
    SELECT transaction-file-name
        ASSIGN TO WORKSTATION-display_file_name
        ORGANIZATION IS TRANSACTION
        ACCESS MODE IS DYNAMIC
               RELATIVE KEY IS relative-key-data-item
        CONTROL AREA IS control-area-data-item.
```

ファイルを選択するには SELECT 文節を使用します。そのファイルは、DATA DIVISION の FD 項目で指定されているものでなければなりません。

TRANSACTION ファイルをディスプレイ・ファイルと関連付けるには、ASSIGN 文節を使用します。 TRANSACTION ファイルを使用するには、ASSIGN 文節に装置タイプ WORKSTATION を指定しなければなりません。 この TRANSACTION ファイルのために別個の標識域を使用する場合は、ASSIGN 文節に -SI 属性を含める必要があります。別個の標識域の使い方については 595 ページの『トランザクション・ファイルでの標識の使用』を参照してください。

TRANSACTION ファイルを使用するためには、ファイル制御記入項目の中に ORGANIZATION IS TRANSACTION を指定してください。この文節は、ILE COBOL プログラムに対して、ワークステーション・ユーザーや別のシステムとの対話が行われることを知らせるものです。

サブファイル TRANSACTION ファイルには動的にアクセスします。動的アクセスを使うなら、特定の入出力要求の形式に応じて、ファイルとの間のレコードの読み書きを、順次アクセスでもランダム・アクセスでも行えるようになります。 ランダムにアクセスできる TRANSACTION ファイルはサブファイルだけです。TRANSACTION ファイルへのアクセス方法を ILE COBOL プログラムに知らせるには、ファイル制御記入項目に ACCESS MODE 文節を使用します。サブファイル TRANSACTION ファイルとの間の読み書きを行うには、ACCESS MODE IS DYNAMIC と指定する必要があります。

サブファイルを使用する場合、相対キーを提供する必要があります。相対キー・データ項目を指定するには、RELATIVE KEY 文節を使用します。相対キー・データ項目は、サブファイル中の特定のレコードに対する相対レコード番号を指定します。

TRANSACTION ファイルを参照する入出力要求の状況をフィードバックしたい場合は、FILE STATUS 文節を使用してファイル制御項目の中に状況キー・データ項目を定義します。 FILE STATUS 文節を指定すると、システムは、明示的にまたは暗黙のうちに TRANSACTION ファイルを参照する各入出力要求の後で、状況キー・データ項目に値を移動します。この値は、入出力ステートメントの実行状況を示します。

TRANSACTION ファイルの入出力操作を制御するのに使用される特定の装置依存情報やシステム依存情報を取得するには、CONTROL-AREA 文節を使用して制御域データ項目を指定します。 CONTROL-AREA 文節で指定されるデータ項目は、以下の形式で LINKAGE SECTION または WORKING-STORAGE SECTION の中で定義します。

```
01  control-area-data-item.
    05  function-key          PIC X(2).
    05  device-name           PIC X(10).
    05  record-format         PIC X(10).
```

制御域の長さは、2 バイトか 12 バイトか 22 バイトです。したがって、必要とする情報の種類に応じて、指定できる要素は、最初の 05 レベル要素、最初の 2 つの 05 レベル要素、または 3 つの 05 要素すべてのいずれかになります。

制御域データ項目を使用することによって、以下のものを指定することができます。

- トランザクションを開始するためにオペレーターが押したファンクション・キー。
- 使用されているプログラム装置の名前。
- 最後の入出力ステートメントで参照された DDS レコード様式の名前。

サブファイル・トランザクション・ファイルの記述

ILE COBOL プログラムで TRANSACTION ファイルを使用するには、DATA DIVISION のファイル記述項目を使用してそのファイルを記述する必要があります。ファイル記述項目の詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。 TRANSACTION ファイルを記述するには、形式 6 のファイル記述項目を使用します。

DATA DIVISION の中で TRANSACTION ファイルを記述するファイル記述項目は、以下のようなものになります。

```
FD  CUST-DISPLAY.
01  DISP-REC.
    COPY DDS-ALL-FORMATS OF CUSMINQ.
```

ILE COBOL において TRANSACTION ファイルは、通常、外部的に記述されます。使用する TRANSACTION ファイルの DDS を作成してください。 DDS の作成方法については 579 ページの『データ記述仕様を使用してトランザクション・ファイルを定義する』を参照してください。 その後、TRANSACTION ファイルを作成します。

TRANSACTION ファイルの DDS と TRANSACTION ファイルを作成したなら、形式 2 の COPY ステートメントを使用して、TRANSACTION ファイルのデータ・レコードのレイアウトを記述してください。 ILE COBOL プログラムをコンパイルすると、形式 2 の COPY により、TRANSACTION ファイルを記述するためのデータ部 (DATA DIVISION) ステートメントが作成されます。すべての様式に対して 1 つのストレージ域を生成するためには、形式 2 の COPY ステートメントの DDS-ALL-FORMATS オプションを使用してください。

サブファイル・トランザクション・ファイルの処理

PROCEDURE DIVISION ステートメントのうち、特に ILE COBOL プログラムで TRANSACTION ファイルを処理するための拡張機能があるものをすべて以下に示します。これらの各ステートメントに関する詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

- ACCEPT ステートメント - 形式 6
- ACQUIRE ステートメント
- CLOSE ステートメント - 形式 1
- DROP ステートメント
- OPEN ステートメント - 形式 3
- READ ステートメント - 形式 5 (サブファイル)
- REWRITE ステートメント - 形式 2 (サブファイル)
- WRITE ステートメント - 形式 5 (サブファイル)

サブファイル・トランザクション・ファイルのオープン

PROCEDURE DIVISION で TRANSACTION ファイルを処理するためには、まずそのファイルをオープンする必要があります。 TRANSACTION ファイルをオープンするには、形式 3 の OPEN ステートメントを使用します。 TRANSACTION ファイルは、I-O モードでオープンしてください。

```
OPEN I-O file-name.
```

プログラム装置の獲得

TRANSACTION ファイル用のプログラム装置を獲得する必要があります。それを獲得したなら、入出力操作のためにそのプログラム装置を使用することができます。プログラム装置は、暗黙のうちに獲得するか、または明示的に獲得します。

TRANSACTION ファイルをオープンすると、暗黙のうちに 1 つのプログラム装置を獲得します。ファイルがディスプレイ・ファイルの場合、暗黙のうちに獲得される単一のプログラム装置は、CRTDSPF コマンドの DEV パラメーターの中の最初の項目によって決定されます。それ以外のプログラム装置は、明示的に獲得しなければなりません。

プログラム装置を明示的に獲得するには、ACQUIRE ステートメントを使用します。ディスプレイ・ファイルの場合、ACQUIRE ステートメントに指定する装置は、CRTDSPF コマンド、CHGDSPF コマンド、または OVRDSPF コマンドの DEV パラメーターで指定する必要はありません。 しかし、ディスプレイ・ファイルを作成する場合には、獲得される装置の数 (デフォルトは 1) を指定する必要があります。 ディスプレイ・ファイルの場合、プログラム装置の名前はディスプレイ装置に一致していなければなりません。

```
ACQUIRE program-device-name FOR transaction-file-name.
```

サブファイル・トランザクション・ファイルへの書き込み

TRANSACTION ファイルをオープンして、そのためのプログラム装置を獲得すれば、それに対する入出力操作を実行するための準備は完了です。

TRANSACTION ファイルに対して実行する最初の入出力操作は、多くの場合、ディスプレイ装置にレコードを書き出すことです。そのレコードは、ユーザーに対して応答や何らかのデータを入力するよう求めるプロンプトとして使用されます。

サブファイル TRANSACTION ファイルに論理レコードを書き込むには、形式 5 の WRITE ステートメントを使用します。 WRITE ステートメントは、次のように簡単にコーディングできます。

```
WRITE SUBFILE record-name FORMAT IS format-name.
```

TRANSACTION ファイルに関して、様式の異なる複数のデータ・レコードを活動状態にしたい場合があります。そのような場合に TRANSACTION ファイルに書き込む出力データ・レコードの様式を指定するには、形式 5 の WRITE ステートメントの FORMAT 句を使用しなければなりません。

TRANSACTION ファイル用に複数のプログラム装置を明示的に獲得した場合、出力レコードの送り先のプログラム装置のサブファイルを指定するには、形式 5 の WRITE ステートメントの TERMINAL 句を使う必要があります。

```
WRITE SUBFILE record-name
      FORMAT IS format-name
      TERMINAL IS program-device-name
END-WRITE.
```

形式 5 の WRITE ステートメントを使ってサブファイル TRANSACTION ファイルにレコードを入れる前後に、形式 4 の WRITE ステートメントを使用することによって、サブファイル制御レコードをプログラム装置に書き込むことができます。形式 4 の WRITE ステートメントを使って TRANSACTION ファイルに書き込む方法

の説明については 585 ページの『トランザクション・ファイルへの書き込み』を参照してください。サブファイル制御レコードに書き込むと、サブファイル制御レコード、サブファイル・レコード、あるいはその両方のいずれかが表示される場合があります。

サブファイル・トランザクション・ファイルからの読み取り

サブファイル制御レコードを読み込むには、形式 4 の READ ステートメントを使用します。形式 4 の READ ステートメントを使って TRANSACTION ファイルを読み込む方法の説明については 586 ページの『トランザクション・ファイルからの読み取り』を参照してください。サブファイル制御レコードを読み込むと、プログラム装置からのレコードが物理的に転送されて、それらがサブファイルで利用できるようになります。

サブファイルからレコードを使用できるようになったら、形式 5 の READ ステートメントを使用することによって、サブファイル TRANSACTION ファイルからの指定したレコードを読み込みます。形式 5 の READ ステートメントを使用できるのは、サブファイル・レコードである様式を読み込む場合だけです。通信装置の場合、それは使用できません。

READ ステートメントを使用する前に、TRANSACTION ファイルのために少なくとも 1 つのプログラム装置を獲得しておかなければなりません。 READ ステートメントが実行される時にプログラム装置が獲得されていない場合、ファイル状況が 92 に設定されることによって論理エラーが報告されます。

サブファイルは順次でもランダムでも読み込むことができます。

サブファイルを順次に読み取るには、形式 5 の READ ステートメントで
NEXT MODIFIED 句を指定する必要があります。 NEXT MODIFIED 句を指定した
場合、使用可能になるレコードは、サブファイル内で変更されている最初のレコードです。 サブファイル・レコードが変更されたことを示すマークを付ける方法については、 Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「データベースおよびファイル・システム」カテゴリーを参照してください。

利用可能な修正されたサブファイル・レコードとして次のものがない場合、AT END 条件が生じ、ファイル状況は 12 に設定され、RELATIVE KEY データ項目の値がサブファイル中の最後のレコードのキーの値に設定されます。

サブファイルを順次に読み込む場合、形式 5 の READ ステートメントで AT END 句も指定する必要があります。 AT END 句では、AT END 条件の発生時に実行する命令ステートメントを指定することかできます。

```
READ SUBFILE subfile-name NEXT MODIFIED RECORD
     AT END imperative-statement
END-READ
```

サブファイルをランダムに読み込むには、RELATIVE KEY データ項目で、読み込むサブファイル・レコードの相対レコード番号を指定する必要があります。また、形式 5 READ ステートメントで NEXT MODIFIED 句を指定しないでください。NEXT MODIFIED 句を指定していない場合は、次に利用できるレコードは、サブファイルのレコードのうち、RELATIVE KEY データ項目の値に対応する相対キー・

レコード番号のレコードです。 READ ステートメントの実行時に、RELATIVE KEY データ項目の値がサブファイルの相対レコード番号に対応していないものであるなら、INVALID KEY 条件が生じます。

サブファイルをランダムに読み込む場合、形式 5 の READ ステートメントで INVALID KEY 句も指定する必要があります。 INVALID KEY 句では、INVALID KEY 条件の発生時に実行する命令ステートメントを指定することかできます。

```
READ SUBFILE subfile-name RECORD
     INVALID KEY imperative-statement
END-READ
```

READ 操作の実行方法の詳細な説明については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の『READ ステートメント』の部分を参照してください。

複数のプログラム装置を獲得した場合、READ ステートメントの TERMINAL 句の中で、どのプログラム装置からデータを読み取るかを明示的に指定することができます。

特定の様式でデータを受け取りたい場合には、その様式を READ ステートメントの FORMAT 句に指定することができます。使用可能なデータが、要求されたレコード様式に合致しない場合、ファイル状況 9K が設定されます。

次に示すのは、TERMINAL 句および FORMAT 句を指定した READ ステートメントの例です。

```
READ SUBFILE subfile-name RECORD
     FORMAT IS record-format
END-READ
READ SUBFILE subfile-name RECORD
     TERMINAL IS program-device-name
END-READ
READ SUBFILE subfile-name RECORD
     FORMAT IS record-format
     TERMINAL IS program-device-name
END-READ
```

サブファイル・レコードの置換 (再作成)

サブファイル・レコードを読み込んで修正したら、REWRITE ステートメントを使用することによって、サブファイルの中でそれを置換することができます。

```
REWRITE SUBFILE record-name
        FORMAT IS record-format
        TERMINAL IS program-device-name
END-REWRITE
```

サブファイル内の置換されるレコードは、サブファイル中のレコードのうち、それ以前に正常に実行された READ 操作でアクセスされたレコードです。

プログラム装置のドロップ

TRANSACTION ファイル用に獲得したプログラム装置を使用し終えた時点で、それをドロップする必要があります。プログラム装置をドロップするとは、その装置が TRANSACTION ファイルを使用した入出力操作では使えなくなるということです。

プログラム装置をドロップすると、その装置は他のアプリケーションから使用できるようになります。 プログラム装置は、暗黙のうちにドロップしたり明示的にドロップしたりすることができます。

TRANSACTION ファイルをクローズすると、そのファイルに付加されていたすべてのプログラム装置も暗黙のうちにドロップされます。

DROP ステートメントで指定することによって、プログラム装置を明示的にドロップすることができます。ドロップされた装置は、必要に応じてそれを再獲得することができます。

```
DROP program-device-name FROM transaction-file-name.
```

サブファイル・トランザクション・ファイルのクローズ

サブファイル TRANSACTION ファイルの使用が終ったら、それをクローズしてください。 TRANSACTION ファイルをクローズするには、形式 1 の CLOSE ステートメントを使用します。 ファイルをクローズすると、そのファイルは、それを再びオープンするまで処理できなくなります。

```
CLOSE transaction-file-name.
```

発注照会プログラムで WRITE SUBFILE を使用する例

634 ページの図 149 に示すのは、サブファイルを使用する発注照会プログラムの例 ORDINQ です。対応する DDS も示します (得意先マスター・ファイル CUSMSTP の DDS を除く)。 CUSMSTP の DDS については 590 ページの図 132 を参照してください。

ORDINQ は、要求された注文番号に対応するすべての明細注文レコードを表示します。このプログラムは、表示する注文番号を入力するようプロンプトを出します。その注文番号を、注文ヘッダー・ファイルの ORDHDRP と比較します。その注文番号が存在していれば、その注文ヘッダー・ファイルからアクセスされた得意先番号が、得意先マスター・ファイル CUSMSTP のものと比較されます。要求された注文に対する ORDDTLP 中のすべての明細注文レコードが読み取られ、サブファイルに書き込まれます。サブファイル制御レコード様式に対する書き込みが処理され、サブファイル中の明細注文レコードが表示されます。このプログラムは、F12 を押すと終了します。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A** PHYSICAL ORDDTLP     ORDER DETAIL FILE
     A
     A                                      UNIQUE
     A*
     A          R ORDDTL                    TEXT('ORDER DETAIL RECORD')
     A*
     A            CUST           5          CHECK(MF)
     A                                      COLHDG('CUSTOMER' 'NUMBER')
     A*
     A            ORDERN         5  0       COLHDG('ORDER' 'NUMBER')
     A*
     A            LINNUM         3  0
     A                                      COLHDG('LINE' 'NO')
     A                                      TEXT('LINE NUMBER OF LINE IN ORDER'
     A                                      )
     A*
     A            ITEM           5  0       CHECK(M10)
     A                                      COLHDG('ITEM' 'NUMBER')
     A            QTYORD         3  0
     A                                      COLHDG('QUANTITY' 'ORDERED')
     A                                      TEXT('QUANTITY ORDERED')
     A*
     A            DESCRP        30          COLHDG('ITEM' 'DESCRIPTION')
     A*
     A            PRICE          6  2       CMP(GT 0)
     A                                      COLHDG('PRICE')
     A                                      TEXT('SELLING PRICE')
     A                                      EDTCDE(J)
     A            EXTENS         8  2       COLHDG('EXTENSION')
     A                                      TEXT('EXTENSION AMOUNT OF QTYORD X
     A                                      PRICE')
     A*
     A            WHSLOC         3          CHECK(MF)
     A                                      COLHDG('BIN' 'NO.')
     A*
     A            ORDDAT         6  0       TEXT('DATE ORDER WAS ENTERED')
     A*
     A            CUSTYP         1  0       RANGE(1 5)
     A                                      COLHDG('CUST' 'TYPE')
     A                                      TEXT('CUSTOMER TYPE 1=GOV 2=SCH   +
     A                                      3=BUS 4=PVT 5=OT')
     A*
     A            STATE          2          CHECK(MF)
     A                                      COLHDG('STATE')
     A*
     A            ACTMTH         2  0       COLHDG('ACCT' 'MTH')
     A                                      TEXT('ACCOUNTING MONTH OF SALE')
     A*
     A            ACTYR          2  0       COLHDG('ACCT' 'YEAR')
     A                                      TEXT('ACCOUNTING YEAR OF SALE')
     A
     A          K ORDERN
     A          K LINNUM
```

図 146. 発注照会プログラムのデータ記述仕様 - 明細注文ファイル

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A*   ORDINQD            EXISTING ORDER REVIEW DISPLAY FILE
     A
     A*
     A          R SUB1                        SFL
     A            ITEM           5  0  10  2TEXT('ITEM NUMBER')
     A            QTYORD         3  0  10  9TEXT('QUANTITY ORDERED')
     A            DESCRP        30     10 14TEXT('ITEM DESCRIPTION')
     A            PRICE          6  2  10 46TEXT('SELLING PRICE')
     A            EXTENS         8  2  10 56EDTCDE(J)
     A                                      TEXT('EXTENSION AMOUNT OF QTYORD +
     A                                      X PRICE')
     A
     A          R SUBCTL1                     SFLCTL(SUB1)
     A  58                                    SFLCLR
     A  57                                    SFLDSP
     A N58                                    SFLDSPCTL
     A                                        SFLSIZ(57)
     A                                        SFLPAG(14)
     A  57                                    SFLEND
     A                                        OVERLAY
     A                                        LOCK
     A N45
     AON47                                    ROLLUP(97 'CONTINUE DISPLAY')
     A                                        CA12(98 'END OF PROGRAM')
     A                                        SETOFF(57 'DISPLAY SUBFILE')
     A                                        SETOFF(58 'OFF = DISPLAY SUBCTL1 O+
     A                                        N = CLEAR SUBFILE')
     A                                   1  2'EXISTING ORDER INQUIRY'
     A                                   3  2'ORDER'
     A            ORDERN         5Y 0B  3  8TEXT('ORDER NUMBER')
     A  61                                    ERRMSG('ORDER NUMBER NOT FOUND' 61)
     A  47                                    ERRMSG('NO LINE FOR THIS ORDER' 47)
     A  62                                    ERRMSG('NO CUSTOMER RECORD' 62)
     A                                   4  2'DATE'
     A            ORDDAT         6  0   4  7TEXT('DATE ORDER WAS ENTERED')
     A                                   5  2'CUST #'
     A            CUST           5      5  9TEXT('CUSTOMER NUMBER')
     A            NAME          25      3 16TEXT('CUSTOMER NAME')
     A            ADDR          20      4 16TEXT('CUSTOMER ADDRESS')
     A            CITY          20      5 16TEXT('CUSTOMER CITY')
     A            STATE          2      6 16TEXT('CUSTOMER STATE')
     A            ZIP            5  0   6 31TEXT('ZIP CODE')
     A                                   1 44'TOTAL'
     A            ORDAMT         8  2   1 51TEXT('TOTAL AMOUNT OF ORDER')
     A                                   2 44'STATUS'
     A            STSORD        12      2 51
     A                                   3 44'OPEN'
     A            STSOPN        12      3 51
     A                                   4 44'CUSTOMER ORDER'
     A            CUSORD        15      4 59TEXT('CUSTOMER PURCHASE ORDER +
     A                                      NUMBER')
     A                                   5 44'SHIP VIA'
     A            SHPVIA        15      5 59TEXT('SHIPPING INSTRUCTIONS')
     A                                   6 44'PRINTED DATE'
     A            PRTDAT         6  0   6 57TEXT('DATE ORDER WAS PRINTED')
     A                                   7 29'INVOICE'
```

図 147. 発注照会プログラムのデータ記述仕様 - 注文検討ファイル (1/2)

```
     A            INVNUM         5  0   7 38TEXT('INVOICE NUMBER')
     A                                   7 64'MTH'
     A            ACTMTH         2  0   7 68TEXT('ACCOUNTING MONTH OF SALE')
     A                                   7 72'YEAR'
     A            ACTYR          2  0   7 77TEXT('ACCOUNTING YEAR OF SALE')
     A                                   8  2'ITEM'
     A                                   8  8'QTY'
     A                                   8 14'ITEM DESCRIPTION'
     A                                   8 46'PRICE'
     A                                   8 55'EXTENSION'
```

図 147. 発注照会プログラムのデータ記述仕様 - 注文検討ファイル (2/2)

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A* THIS IS THE ORDER HEADER FILE ** ORDHDRP
     A
     A
     A                                     UNIQUE
     A          R ORDHDR                   TEXT('ORDER HEADER RECORD')
     A            CUST           5         TEXT('CUSTOMER NUMBER')
     A            ORDERN         5 00      TEXT('ORDER NUMBER')
     A            ORDDAT         6 00      TEXT('DATE ORDER ENTERED')
     A            CUSORD        15         TEXT('CUSTOMER PURCHASE ORDER +
     A                                     NUMBER')
     A            SHPVIA        15         TEXT('SHIPPING INSTRUCTIONS')
     A            ORDSTS         1 00      TEXT('ORDER SATAUS 1PCS 2CNT  + 3CHK 4RDY 5PRT 6PCK')
     A            OPRNAM        10         TEXT('OPERATOR WHO ENTERED ORD')
     A            ORDAMT         8 02      TEXT('DOLLAR AMOUNT OF ORDER')
     A            CUSTYP         1 00      TEXT('CUSTOMER TYPE 1=GOV 2=SCH +
     A                                     3=BUS 4=PVT 5=OT')
     A            INVNUM         5 00      TEXT('INVOICE NUMBER')
     A            PRTDAT         6 00      TEXT('DATE ORDER WAS PRINTED')
     A            OPNSTS         1 00      TEXT('ORDER OPEN STATUS 1=OPEN + 2= CLOSE 3=CANCEL')
     A            TOTLIN         3 00      TEXT('TOTAL LINE ITEMS IN ORDER')
     A            ACTMTH         2 00      TEXT('ACCOUNTING MONTH OF SALE')
     A            ACTYR          2 00      TEXT('ACCOUNTING YEAR OF SALE')
     A            STATE          2         TEXT('STATE')
     A            AMPAID         8 02      TEXT('AMOUNT PAID')
                K ORDERN
```

図 148. 発注照会プログラムのデータ記述仕様 - 注文ヘッダー・ファイル

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/ORDINQ          ISERIES1   06/02/15 15:06:50        ページ     2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     ORDINQ.
          000300*    SAMPLE ORDER INQUIRY PROGRAM
          000400
    3     000500 ENVIRONMENT DIVISION.
    4     000600   CONFIGURATION SECTION.
    5     000700 SOURCE-COMPUTER. IBM-ISERIES
    6     000800 OBJECT-COMPUTER. IBM-ISERIES
    7     000900   INPUT-OUTPUT SECTION.
    8     001000   FILE-CONTROL.
    9     001100     SELECT ORDER-HEADER-FILE
   10     001200         ASSIGN TO DATABASE-ORDHDRP
   11     001300         ORGANIZATION IS INDEXED
   12     001400         ACCESS MODE IS RANDOM
   13     001500         RECORD KEY IS ORDERN OF ORDER-HEADER-RECORD.
   14     001600     SELECT ORDER-DETAIL-FILE
   15     001700         ASSIGN TO DATABASE-ORDDTLP
   16     001800         ORGANIZATION IS INDEXED
   17     001900         ACCESS IS DYNAMIC
   18     002000         RECORD KEY IS ORDER-DETAIL-RECORD-KEY.
   19     002100     SELECT CUSTOMER-MASTER-FILE
   20     002200         ASSIGN TO DATABASE-CUSMSTP
   21     002300         ORGANIZATION IS INDEXED
   22     002400         ACCESS IS RANDOM
   23     002500         RECORD KEY IS CUST OF CUSTOMER-MASTER-RECORD.
   24     002600     SELECT EXISTING-ORDER-DISPLAY-FILE
   25     002700         ASSIGN TO WORKSTATION-ORDINQD
   26     002800         ORGANIZATION IS TRANSACTION
   27     002900         ACCESS IS DYNAMIC
   28     003000         RELATIVE KEY IS SUBFILE-RECORD-NUMBER
   29     003100         FILE STATUS IS STATUS-CODE-ONE.
          003200
   30     003300 DATA DIVISION.
   31     003400 FILE SECTION.
   32     003500 FD  ORDER-HEADER-FILE.
   33     003600 01  ORDER-HEADER-RECORD.
          003700     COPY DDS-ORDHDR OF ORDHDRP.
         +000001*    I-O FORMAT:ORDHDR     FROM FILE ORDHDRP    OF LIBRARY CBLGUIDE          ORDHDR
         +000002*                          ORDER HEADER RECORD                               ORDHDR
         +000003*          USER SUPPLIED KEY BY RECORD KEY CLAUSE                            ORDHDR
   34    +000004       05  ORDHDR.                                                          ORDHDR
   35    +000005           06 CUST                 PIC X(5).                               ORDHDR
         +000006*                 CUSTOMER NUMBER                                          ORDHDR
   36    +000007           06 ORDERN                PIC S9(5)         COMP-3.               ORDHDR
         +000008*                 ORDER NUMBER                                             ORDHDR
   37    +000009           06 ORDDAT                PIC S9(6)         COMP-3.               ORDHDR
         +000010*                 DATE ORDER ENTERED                                       ORDHDR
   38    +000011           06 CUSORD                PIC X(15).                              ORDHDR
         +000012*                 CUSTOMER PURCHASE ORDER NUMBER                           ORDHDR
   39    +000013           06 SHPVIA                PIC X(15).                              ORDHDR
         +000014*                 SHIPPING INSTRUCTIONS                                    ORDHDR
   40    +000015           06 ORDSTS                PIC S9(1)         COMP-3.               ORDHDR
         +000016*                 ORDER SATAUS 1PCS 2CNT  3CHK 4RDY 5PRT 6PCK              ORDHDR
```

図 149. 発注照会プログラムの例 (1/10)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                  CBLGUIDE/ORDINQ          ISERIES1   06/02/15 15:06:50        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
   41    +000017           06 OPRNAM                PIC X(10).                                ORDHDR
         +000018*                 OPERATOR WHO ENTERED ORD                                    ORDHDR
   42    +000019           06 ORDAMT                PIC S9(6)V9(2)    COMP-3.                 ORDHDR
         +000020*                 DOLLAR AMOUNT OF ORDER                                      ORDHDR
   43    +000021           06 CUSTYP                PIC S9(1)         COMP-3.                 ORDHDR
         +000022*                 CUSTOMER TYPE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT                  ORDHDR
   44    +000023           06 INVNUM                PIC S9(5)         COMP-3.                 ORDHDR
         +000024*                 INVOICE NUMBER                                              ORDHDR
   45    +000025           06 PRTDAT                PIC S9(6)         COMP-3.                 ORDHDR
         +000026*                 DATE ORDER WAS PRINTED                                      ORDHDR
   46    +000027           06 OPNSTS                PIC S9(1)         COMP-3.                 ORDHDR
         +000028*                 ORDER OPEN STATUS 1=OPEN 2= CLOSE 3=CANCEL                  ORDHDR
   47    +000029           06 TOTLIN                PIC S9(3)         COMP-3.                 ORDHDR
         +000030*                 TOTAL LINE ITEMS IN ORDER                                   ORDHDR
   48    +000031           06 ACTMTH                PIC S9(2)         COMP-3.                 ORDHDR
         +000032*                 ACCOUNTING MONTH OF SALE                                    ORDHDR
   49    +000033           06 ACTYR                 PIC S9(2)         COMP-3.                 ORDHDR
         +000034*                 ACCOUNTING YEAR OF SALE                                     ORDHDR
   50    +000035           06 STATE                 PIC X(2).                                 ORDHDR
         +000036*                 STATE                                                       ORDHDR
   51    +000037           06 AMPAID                PIC S9(6)V9(2)    COMP-3.                 ORDHDR
         +000038*                 AMOUNT PAID                                                 ORDHDR
          003800
   52     003900 FD  ORDER-DETAIL-FILE.
   53     004000 01  ORDER-DETAIL-RECORD.
          004100     COPY DDS-ORDDTL OF ORDDTLP.
         +000001*    I-O FORMAT:ORDDTL     FROM FILE ORDDTLP    OF LIBRARY CBLGUIDE          ORDDTL
         +000002*                          ORDER DETAIL RECORD                                 ORDDTL
         +000003*          USER SUPPLIED KEY BY RECORD KEY CLAUSE                              ORDDTL
   54    +000004       05  ORDDTL.                                                            ORDDTL
   55    +000005           06 CUST                  PIC X(5).                                 ORDDTL
         +000006*                 CUSTOMER NUMBER                                             ORDDTL
   56    +000007           06 ORDERN                PIC S9(5)         COMP-3.                 ORDDTL
         +000008*                 ORDER NUMBER                                                ORDDTL
   57    +000009           06 LINNUM                PIC S9(3)         COMP-3.                 ORDDTL
         +000010*                 LINE NUMBER OF LINE IN ORDER                                ORDDTL
   58    +000011           06 ITEM                  PIC S9(5)         COMP-3.                 ORDDTL
         +000012*                 ITEM NUMBER                                                 ORDDTL
   59    +000013           06 QTYORD                PIC S9(3)         COMP-3.                 ORDDTL
         +000014*                 QUANTITY ORDERED                                            ORDDTL
   60    +000015           06 DESCRP                PIC X(30).                                ORDDTL
         +000016*                 ITEM DESCRIPTION                                            ORDDTL
   61    +000017           06 PRICE                 PIC S9(4)V9(2)    COMP-3.                 ORDDTL
         +000018*                 SELLING PRICE                                               ORDDTL
   62    +000019           06 EXTENS                PIC S9(6)V9(2)    COMP-3.                 ORDDTL
         +000020*                 EXTENSION AMOUNT OF QTYORD X PRICE                          ORDDTL
   63    +000021           06 WHSLOC                PIC X(3).                                 ORDDTL
         +000022*                 BIN NO.                                                     ORDDTL
   64    +000023           06 ORDDAT                PIC S9(6)         COMP-3.                 ORDDTL
         +000024*                 DATE ORDER WAS ENTERED                                      ORDDTL
   65    +000025           06 CUSTYP                PIC S9(1)         COMP-3.                 ORDDTL
         +000026*                 CUSTOMER TYPE 1=GOV 2=SCH   3=BUS 4=PVT 5=OT                ORDDTL
   66    +000027           06 STATE                 PIC X(2).                                 ORDDTL
         +000028*                 STATE                                                       ORDDTL
   67    +000029           06 ACTMTH                PIC S9(2)         COMP-3.                 ORDDTL
```

図 149. 発注照会プログラムの例 (2/10)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/ORDINQ          ISERIES1   06/02/15 15:06:50        ページ     4
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
        +000030*                ACCOUNTING MONTH OF SALE                                    ORDDTL
   68   +000031           06 ACTYR                 PIC S9(2)         COMP-3.                ORDDTL
        +000032*                ACCOUNTING YEAR OF SALE                                     ORDDTL
   69    004200 66  ORDER-DETAIL-RECORD-KEY RENAMES ORDERN THRU LINNUM.
         004300
   70    004400 FD  CUSTOMER-MASTER-FILE.
   71    004500 01  CUSTOMER-MASTER-RECORD.
         004600     COPY DDS-CUSMST OF CUSMSTP.
        +000001*    I-O FORMAT:CUSMST     FROM FILE CUSMSTP    OF LIBRARY CBLGUIDE           CUSMST
        +000002*                          CUSTOMER MASTER RECORD                             CUSMST
        +000003*          USER SUPPLIED KEY BY RECORD KEY CLAUSE                             CUSMST
   72   +000004       05  CUSMST.                                                           CUSMST
   73   +000005           06 CUST                  PIC X(5).                               CUSMST
        +000006*                CUSTOMER NUMBER                                             CUSMST
   74   +000007           06 NAME                  PIC X(25).                              CUSMST
        +000008*                CUSTOMER NAME                                               CUSMST
   75   +000009           06 ADDR                  PIC X(20).                              CUSMST
        +000010*                CUSTOMER ADDRESS                                            CUSMST
   76   +000011           06 CITY                  PIC X(20).                              CUSMST
        +000012*                CUSTOMER CITY                                               CUSMST
   77   +000013           06 STATE                 PIC X(2).                               CUSMST
        +000014*                STATE                                                       CUSMST
   78   +000015           06 ZIP                   PIC S9(5)         COMP-3.               CUSMST
        +000016*                ZIP CODE                                                    CUSMST
   79   +000017           06 SRHCOD                PIC X(6).                               CUSMST
        +000018*                CUSTOMER NUMBER SEARCH CODE                                 CUSMST
   80   +000019           06 CUSTYP                PIC S9(1)         COMP-3.               CUSMST
        +000020*                CUSTOMER TYPE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT                  CUSMST
   81   +000021           06 ARBAL                 PIC S9(6)V9(2)    COMP-3.               CUSMST
        +000022*                ACCOUNTS REC. BALANCE                                       CUSMST
   82   +000023           06 ORDBAL                PIC S9(6)V9(2)    COMP-3.               CUSMST
        +000024*                A/R AMT. IN ORDER FILE                                      CUSMST
   83   +000025           06 LSTAMT                PIC S9(6)V9(2)    COMP-3.               CUSMST
        +000026*                LAST AMT. PAID IN A/R                                       CUSMST
   84   +000027           06 LSTDAT                PIC S9(6)         COMP-3.               CUSMST
        +000028*                LAST DATE PAID IN A/R                                       CUSMST
   85   +000029           06 CRDLMT                PIC S9(6)V9(2)    COMP-3.               CUSMST
        +000030*                CUSTOMER CREDIT LIMIT                                       CUSMST
   86   +000031           06 SLSYR                 PIC S9(8)V9(2)    COMP-3.               CUSMST
        +000032*                CUSTOMER SALES THIS YEAR                                    CUSMST
   87   +000033           06 SLSLYR                PIC S9(8)V9(2)    COMP-3.               CUSMST
        +000034*                CUSTOMER SALES LAST YEAR                                    CUSMST
         004700
   88    004800 FD  EXISTING-ORDER-DISPLAY-FILE.
   89    004900 01  EXISTING-ORDER-DISPLAY-RECORD.
         005000     COPY DDS-ALL-FORMATS OF ORDINQD.
   90   +000001       05  ORDINQD-RECORD PIC X(171).                                        <-ALL-FMTS
        +000002*    I-O FORMAT:SUB1       FROM FILE ORDINQD    OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000003*                                                                            <-ALL-FMTS
   91   +000004       05  SUB1          REDEFINES ORDINQD-RECORD.                          <-ALL-FMTS
   92   +000005           06 ITEM                  PIC S9(5).                              <-ALL-FMTS
        +000006*                ITEM NUMBER                                                 <-ALL-FMTS
   93   +000007           06 QTYORD                PIC S9(3).                              <-ALL-FMTS
        +000008*                QUANTITY ORDERED                                            <-ALL-FMTS
   94   +000009           06 DESCRP                PIC X(30).                              <-ALL-FMTS
```

図 149. 発注照会プログラムの例 (3/10)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/ORDINQ          ISERIES1   06/02/15 15:06:50        ページ     5
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
        +000010*                ITEM DESCRIPTION                                          <-ALL-FMTS
  95    +000011           06 PRICE                PIC S9(4)V9(2).                         <-ALL-FMTS
        +000012*                SELLING PRICE                                             <-ALL-FMTS
  96    +000013           06 EXTENS               PIC S9(6)V9(2).                         <-ALL-FMTS
        +000014*                EXTENSION AMOUNT OF QTYORD X PRICE                        <-ALL-FMTS
        +000015*  INPUT FORMAT:SUBCTL1    FROM FILE ORDINQD    OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000016*                                                                          <-ALL-FMTS
  97    +000017       05  SUBCTL1-I     REDEFINES ORDINQD-RECORD.                       <-ALL-FMTS
  98    +000018           06 SUBCTL1-I-INDIC.                                            <-ALL-FMTS
  99    +000019                07 IN97            PIC 1  INDIC 97.                      <-ALL-FMTS
        +000020*                       CONTINUE DISPLAY                                  <-ALL-FMTS
 100    +000021                07 IN98            PIC 1  INDIC 98.                      <-ALL-FMTS
        +000022*                       END OF PROGRAM                                    <-ALL-FMTS
 101    +000023                07 IN57            PIC 1  INDIC 57.                      <-ALL-FMTS
        +000024*                       DISPLAY SUBFILE                                   <-ALL-FMTS
 102    +000025                07 IN58            PIC 1  INDIC 58.                      <-ALL-FMTS
        +000026*                       OFF = DISPLAY SUBCTL1 ON = CLEAR SUBFILE          <-ALL-FMTS
 103    +000027                07 IN61            PIC 1  INDIC 61.                      <-ALL-FMTS
        +000028*                       ORDER NUMBER NOT FOUND                            <-ALL-FMTS
 104    +000029                07 IN47            PIC 1  INDIC 47.                      <-ALL-FMTS
        +000030*                       NO LINE FOR THIS ORDER                            <-ALL-FMTS
 105    +000031                07 IN62            PIC 1  INDIC 62.                      <-ALL-FMTS
        +000032*                       NO CUSTOMER RECORD                                <-ALL-FMTS
 106    +000033           06 ORDERN               PIC S9(5).                              <-ALL-FMTS
        +000034*                ORDER NUMBER                                              <-ALL-FMTS
        +000035* OUTPUT FORMAT:SUBCTL1    FROM FILE ORDINQD    OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000036*                                                                          <-ALL-FMTS
 107    +000037       05  SUBCTL1-O     REDEFINES ORDINQD-RECORD.                       <-ALL-FMTS
 108    +000038           06 SUBCTL1-O-INDIC.                                            <-ALL-FMTS
 109    +000039                07 IN58            PIC 1  INDIC 58.                      <-ALL-FMTS
        +000040*                       OFF = DISPLAY SUBCTL1 ON = CLEAR SUBFILE          <-ALL-FMTS
 110    +000041                07 IN57            PIC 1  INDIC 57.                      <-ALL-FMTS
        +000042*                       DISPLAY SUBFILE                                   <-ALL-FMTS
 111    +000043                07 IN45            PIC 1  INDIC 45.                      <-ALL-FMTS
 112    +000044                07 IN47            PIC 1  INDIC 47.                      <-ALL-FMTS
        +000045*                       NO LINE FOR THIS ORDER                            <-ALL-FMTS
 113    +000046                07 IN61            PIC 1  INDIC 61.                      <-ALL-FMTS
        +000047*                       ORDER NUMBER NOT FOUND                            <-ALL-FMTS
 114    +000048                07 IN62            PIC 1  INDIC 62.                      <-ALL-FMTS
        +000049*                       NO CUSTOMER RECORD                                <-ALL-FMTS
 115    +000050           06 ORDERN               PIC S9(5).                              <-ALL-FMTS
        +000051*                ORDER NUMBER                                              <-ALL-FMTS
 116    +000052           06 ORDDAT               PIC S9(6).                              <-ALL-FMTS
        +000053*                DATE ORDER WAS ENTERED                                    <-ALL-FMTS
 117    +000054           06 CUST                 PIC X(5).                               <-ALL-FMTS
        +000055*                CUSTOMER NUMBER                                           <-ALL-FMTS
 118    +000056           06 NAME                 PIC X(25).                              <-ALL-FMTS
        +000057*                CUSTOMER NAME                                             <-ALL-FMTS
 119    +000058           06 ADDR                 PIC X(20).                              <-ALL-FMTS
        +000059*                CUSTOMER ADDRESS                                          <-ALL-FMTS
 120    +000060           06 CITY                 PIC X(20).                              <-ALL-FMTS
        +000061*                CUSTOMER CITY                                             <-ALL-FMTS
 121    +000062           06 STATE                PIC X(2).                               <-ALL-FMTS
        +000063*                CUSTOMER STATE                                            <-ALL-FMTS
 122    +000064           06 ZIP                  PIC S9(5).                              <-ALL-FMTS
```

図 149. 発注照会プログラムの例 (4/10)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/ORDINQ          ISERIES1   06/02/15 15:06:50        ページ     6
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
        +000065*                ZIP CODE                                                    <-ALL-FMTS
  123   +000066           06 ORDAMT               PIC S9(6)V9(2).                          <-ALL-FMTS
        +000067*                TOTAL AMOUNT OF ORDER                                       <-ALL-FMTS
  124   +000068           06 STSORD               PIC X(12).                               <-ALL-FMTS
  125   +000069           06 STSOPN               PIC X(12).                               <-ALL-FMTS
  126   +000070           06 CUSORD               PIC X(15).                               <-ALL-FMTS
        +000071*                CUSTOMER PURCHASE ORDER NUMBER                              <-ALL-FMTS
  127   +000072           06 SHPVIA               PIC X(15).                               <-ALL-FMTS
        +000073*                SHIPPING INSTRUCTIONS                                       <-ALL-FMTS
  128   +000074           06 PRTDAT               PIC S9(6).                               <-ALL-FMTS
        +000075*                DATE ORDER WAS PRINTED                                      <-ALL-FMTS
  129   +000076           06 INVNUM               PIC S9(5).                               <-ALL-FMTS
        +000077*                INVOICE NUMBER                                              <-ALL-FMTS
  130   +000078           06 ACTMTH               PIC S9(2).                               <-ALL-FMTS
        +000079*                ACCOUNTING MONTH OF SALE                                    <-ALL-FMTS
  131   +000080           06 ACTYR                PIC S9(2).                               <-ALL-FMTS
        +000081*                ACCOUNTING YEAR OF SALE                                     <-ALL-FMTS
        005100
  132   005200 WORKING-STORAGE SECTION.
  133   005300 01  EXISTING-ORDER-DISPLAY-KEY.
  134   005400    05  SUBFILE-RECORD-NUMBER              PIC 9(2)
        005500                                       VALUE ZERO.
        005600
  135   005700 01  ORDER-STATUS-COMMENT-VALUES.
  136   005800    05  FILLER                             PIC X(12)
        005900                                       VALUE "1-IN PROCESS".
  137   006000    05  FILLER                             PIC X(12)
        006100                                       VALUE "2-CONTINUED ".
  138   006200    05  FILLER                             PIC X(12)
        006300                                       VALUE "3-CREDIT CHK".
  139   006400    05  FILLER                             PIC X(12)
        006500                                       VALUE "4-READY PRT ".
  140   006600    05  FILLER                             PIC X(12)
        006700                                       VALUE "5-PRINTED   ".
  141   006800    05  FILLER                             PIC X(12)
        006900                                       VALUE "6-PICKED    ".
  142   007000    05  FILLER                             PIC X(12)
        007100                                       VALUE "7-INVOICED  ".
  143   007200    05  FILLER                             PIC X(12)
        007300                                       VALUE "8-INVALID   ".
  144   007400    05  FILLER                             PIC X(12)
        007500                                       VALUE "9-CANCELED  ".
        007600
  145   007700 01  ORDER-STATUS-COMMENT-TABLE
        007800         REDEFINES ORDER-STATUS-COMMENT-VALUES.
  146   007900    05  ORDER-STATUS OCCURS 9 TIMES.
  147   008000       10 ORDER-STATUS-COMMENT              PIC X(12).
        008100
  148   008200 01  OPEN-STATUS-COMMENT-VALUES.
  149   008300    05  FILLER                             PIC X(12)
        008400                                       VALUE "1-OPEN      ".
  150   008500    05  FILLER                             PIC X(12)
        008600                                       VALUE "2-CLOSED    ".
  151   008700    05  FILLER                             PIC X(12)
        008800                                       VALUE "3-CANCELED  ".
```

図 149. 発注照会プログラムの例 (5/10)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/ORDINQ          ISERIES1   06/02/15 15:06:50        ページ    7
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
        008900
  152     009000 01  OPEN-STATUS-COMMENT-TABLE
          009100         REDEFINES OPEN-STATUS-COMMENT-VALUES.
  153     009200    05  OPEN-STATUS OCCURS 3 TIMES.
  154     009300       10  OPEN-STATUS-COMMENT              PIC X(12).
          009400
  155     009500 01  ERRHDL-PARAMETERS.
  156     009600    05  STATUS-CODE-ONE                     PIC X(2).
  157     009700       88  SUBFILE-IS-FULL                VALUE "0M".
          009800
  158     009900 01  ERRPGM-PARAMETERS.
  159     010000    05  DISPLAY-PARAMETER                   PIC X(8)
          010100                                          VALUE "ORD220D ".
  160     010200    05  DUMMY-ONE                           PIC X(6)
          010300                                          VALUE SPACES.
  161     010400    05  DUMMY-TWO                           PIC X(8)
          010500                                          VALUE SPACES.
  162     010600    05  STATUS-CODE-TWO.
  163     010700       10  PRIMARY                         PIC X(1).
  164     010800       10  SECONDARY                       PIC X(1).
  165     010900       10  FILLER                          PIC X(5)
          011000                                          VALUE SPACES.
          011100
  166     011200 01  SWITCH-AREA.
  167     011300    05  SW01                                PIC 1.
  168     011400       88  NO-MORE-DETAIL-LINE-ITEMS      VALUE B"1".
  169     011500       88  MORE-DETAIL-LINE-ITEMS-EXIST   VALUE B"0".
  170     011600    05  SW02                                PIC 1.
  171     011700       88  WRITE-DISPLAY                  VALUE B"1".
  172     011800       88  READ-DISPLAY                   VALUE B"0".
  173     011900    05  SW03                                PIC 1.
  174     012000       88  SUBCTL1-FORMAT                 VALUE B"1".
  175     012100       88  NOT-SUBCTL1-FORMAT             VALUE B"0".
  176     012200    05  SW04                                PIC 1.
  177     012300       88  SUB1-FORMAT                    VALUE B"1".
  178     012400       88  NOT-SUB1-FORMAT                VALUE B"0".
          012500
  179     012600 01  INDICATOR-AREA.
  180     012700    05  IN98                                PIC 1 INDIC 98.
  181     012800       88  END-OF-EXISTING-ORDER-INQUIRY   VALUE B"1".
  182     012900    05  IN97                                PIC 1 INDIC 97.
  183     013000       88  CONTINUE-DETAIL-LINES-DISPLAY   VALUE B"1".
  184     013100    05  IN62                                PIC 1 INDIC 62.
  185     013200       88  CUSTOMER-NOT-FOUND              VALUE B"1".
  186     013300       88  CUSTOMER-EXIST                  VALUE B"0".
  187     013400    05  IN61                                PIC 1 INDIC 61.
  188     013500       88  ORDER-NOT-FOUND                 VALUE B"1".
  189     013600       88  ORDER-EXIST                     VALUE B"0".
  190     013700    05  IN58                                PIC 1 INDIC 58.
  191     013800       88  CLEAR-SUBFILE                   VALUE B"1".
  192     013900       88  DISPLAY-SUBFILE-CONTROL         VALUE B"0".
  193     014000    05  IN57                                PIC 1 INDIC 57.
  194     014100       88  DISPLAY-SUBFILE                 VALUE B"1".
  195     014200    05  IN47                                PIC 1 INDIC 47.
  196     014300       88  NO-DETAIL-LINES-FOR-ORDER       VALUE B"1".
```

図 149. 発注照会プログラムの例 (6/10)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/ORDINQ          ISERIES1   06/02/15 15:06:50        ページ     8
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
  197     014400       88  DETAIL-LINES-FOR-ORDER-EXIST   VALUE B"0".
  198     014500    05  IN45                                 PIC 1 INDIC 45.
  199     014600       88  END-OF-ORDER                   VALUE B"1".
          014700
  200     014800 PROCEDURE DIVISION.
          014900
  201     015000 DECLARATIVES.
          015100 TRANSACTION-ERROR SECTION.
          015200     USE AFTER STANDARD ERROR PROCEDURE
          015300         EXISTING-ORDER-DISPLAY-FILE.
          015400 WORK-STATION-ERROR-HANDLER.
  202     015500     IF NOT (SUBFILE-IS-FULL) THEN
  203     015600         DISPLAY "WORK-STATION ERROR"   STATUS-CODE-ONE
          015700     END-IF.
          015800 END DECLARATIVES.
          015900
          016000 MAIN-PROGRAM SECTION.
          016100 MAINLINE.
  204     016200     OPEN INPUT  ORDER-HEADER-FILE
          016300                 ORDER-DETAIL-FILE
          016400                 CUSTOMER-MASTER-FILE
          016500           I-O   EXISTING-ORDER-DISPLAY-FILE.
  205     016600     MOVE SPACES TO   CUST   OF SUBCTL1-O
          016700                      NAME   OF SUBCTL1-O
          016800                      ADDR   OF SUBCTL1-O
          016900                      CITY   OF SUBCTL1-O
          017000                      STATE  OF SUBCTL1-O
          017100                      STSORD OF SUBCTL1-O
          017200                      STSOPN OF SUBCTL1-O
          017300                      CUSORD OF SUBCTL1-O.
  206     017400     MOVE ZEROS TO    ORDERN OF SUBCTL1-O
          017500                      ORDDAT OF SUBCTL1-O
          017600                      ZIP    OF SUBCTL1-O
          017700                      ORDAMT OF SUBCTL1-O
          017800                      PRTDAT OF SUBCTL1-O
          017900                      INVNUM OF SUBCTL1-O
          018000                      ACTMTH OF SUBCTL1-O
          018100                      ACTYR  OF SUBCTL1-O.
  207     018200     MOVE B"0" TO INDICATOR-AREA.
  208     018300     SET READ-DISPLAY
          018400         NOT-SUBCTL1-FORMAT
          018500         NOT-SUB1-FORMAT TO TRUE.
  209     018600     MOVE CORR INDICATOR-AREA TO SUBCTL1-O-INDIC.
                *** CORRESPONDING items for statement 209:
                ***     IN62
                ***     IN61
                ***     IN58
                ***     IN57
                ***     IN47
                ***     IN45
                *** End of CORRESPONDING items for statement 209
  210     018700     WRITE EXISTING-ORDER-DISPLAY-RECORD FORMAT IS "SUBCTL1"
          018800     END-WRITE
  211     018900     READ EXISTING-ORDER-DISPLAY-FILE RECORD.
  212     019000     MOVE CORR SUBCTL1-I-INDIC TO INDICATOR-AREA.
```

図 149. 発注照会プログラムの例 (7/10)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/ORDINQ          ISERIES1   06/02/15 15:06:50        ページ    9
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
               *** CORRESPONDING items for statement 212:
               ***     IN97
               ***     IN98
               ***     IN57
               ***     IN58
               ***     IN61
               ***     IN47
               ***     IN62
               *** End of CORRESPONDING items for statement 212
         019100
  213    019200     PERFORM EXISTING-ORDER-INQUIRY
         019300         UNTIL END-OF-EXISTING-ORDER-INQUIRY.
         019400
  214    019500      CLOSE     ORDER-HEADER-FILE
         019600                ORDER-DETAIL-FILE
         019700                CUSTOMER-MASTER-FILE
         019800                EXISTING-ORDER-DISPLAY-FILE.
  215    019900      STOP RUN.
         020000
         020100 EXISTING-ORDER-INQUIRY.
  216    020200     IF CONTINUE-DETAIL-LINES-DISPLAY THEN
  217    020300         PERFORM READ-NEXT-ORDER-DETAIL-RECORD
  218    020400         IF MORE-DETAIL-LINE-ITEMS-EXIST THEN
  219    020500             IF ORDERN OF ORDER-DETAIL-RECORD IS NOT EQUAL TO
         020600                ORDERN OF ORDER-HEADER-RECORD THEN
  220    020700                    SET DISPLAY-SUBFILE TO TRUE
  221    020800                    SET NO-DETAIL-LINES-FOR-ORDER TO TRUE
         020900             ELSE
  222    021000                 PERFORM SUBFILE-SET-UP
         021100             END-IF
         021200         ELSE
  223    021300             SET DISPLAY-SUBFILE TO TRUE
  224    021400             SET NO-DETAIL-LINES-FOR-ORDER TO TRUE
         021500         END-IF
         021600     ELSE
  225    021700         PERFORM ORDER-NUMBER-VALIDATION
         021800     END-IF
  226    021900     MOVE CORR INDICATOR-AREA TO SUBCTL1-O-INDIC.
               *** CORRESPONDING items for statement 226:
               ***     IN62
               ***     IN61
               ***     IN58
               ***     IN57
               ***     IN47
               ***     IN45
               *** End of CORRESPONDING items for statement 226
  227    022000     SET WRITE-DISPLAY TO TRUE.
  228    022100     SET SUBCTL1-FORMAT TO TRUE.
  229    022200     WRITE EXISTING-ORDER-DISPLAY-RECORD FORMAT IS "SUBCTL1".
  230    022300     READ EXISTING-ORDER-DISPLAY-FILE RECORD.
  231    022400     MOVE CORR SUBCTL1-I-INDIC TO INDICATOR-AREA.
               *** CORRESPONDING items for statement 231:
               ***     IN97
               ***     IN98
               ***     IN57
```

図 149. 発注照会プログラムの例 (8/10)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/ORDINQ          ISERIES1   06/02/15 15:06:50        ページ   10
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
                ***     IN58
                ***     IN61
                ***     IN47
                ***     IN62
                *** End of CORRESPONDING items for statement 231
          022500
          022600 ORDER-NUMBER-VALIDATION.
  232     022700     PERFORM READ-ORDER-HEADER-FILE.
  233     022800     IF ORDER-EXIST THEN
  234     022900         PERFORM READ-CUSTOMER-MASTER-FILE
  235     023000         IF CUSTOMER-EXIST THEN
  236     023100             PERFORM READ-FIRST-ORDER-DETAIL-RECORD
  237     023200             IF DETAIL-LINES-FOR-ORDER-EXIST THEN
  238     023300                 PERFORM SUBFILE-SET-UP
          023400             END-IF
          023500         END-IF
          023600     END-IF.
          023700
          023800 READ-ORDER-HEADER-FILE.
  239     023900     MOVE ORDERN OF SUBCTL1-I OF EXISTING-ORDER-DISPLAY-RECORD
          024000         TO ORDERN OF ORDER-HEADER-RECORD.
  240     024100     READ ORDER-HEADER-FILE
  241     024200         INVALID KEY SET ORDER-NOT-FOUND TO TRUE
          024300     END-READ.
          024400
          024500 READ-CUSTOMER-MASTER-FILE.
  242     024600     MOVE CUST OF ORDER-HEADER-RECORD
          024700         TO CUST OF CUSTOMER-MASTER-RECORD.
  243     024800     READ CUSTOMER-MASTER-FILE
  244     024900         INVALID KEY SET CUSTOMER-NOT-FOUND TO TRUE
          025000     END-READ.
          025100
          025200 READ-FIRST-ORDER-DETAIL-RECORD.
  245     025300     MOVE ORDERN OF ORDER-HEADER-RECORD
          025400         TO ORDERN OF ORDER-DETAIL-RECORD.
  246     025500     MOVE 1 TO LINNUM OF ORDER-DETAIL-RECORD.
  247     025600     READ ORDER-DETAIL-FILE
  248     025700         INVALID KEY SET NO-DETAIL-LINES-FOR-ORDER TO TRUE
          025800     END-READ.
          025900
          026000 SUBFILE-SET-UP.
  249     026100     SET CLEAR-SUBFILE TO TRUE.
  250     026200     MOVE CORR INDICATOR-AREA TO SUBCTL1-O-INDIC.
                *** CORRESPONDING items for statement 250:
                ***     IN62
                ***     IN61
                ***     IN58
                ***     IN57
                ***     IN47
                ***     IN45
                *** End of CORRESPONDING items for statement 250
  251     026300     SET WRITE-DISPLAY TO TRUE.
  252     026400     SET SUBCTL1-FORMAT TO TRUE.
  253     026500     WRITE EXISTING-ORDER-DISPLAY-RECORD FORMAT IS "SUBCTL1"
          026600     END-WRITE
```

図 149. 発注照会プログラムの例 (9/10)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/ORDINQ          ISERIES1   06/02/15 15:06:50        ページ   11
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
  254     026700     SET DISPLAY-SUBFILE-CONTROL TO TRUE.
  255     026800     PERFORM BUILD-DISPLAY-SUBFILE
          026900         UNTIL NO-MORE-DETAIL-LINE-ITEMS OR SUBFILE-IS-FULL.
  256     027000     MOVE CORR ORDHDR OF ORDER-HEADER-RECORD
          027100         TO SUBCTL1-O OF EXISTING-ORDER-DISPLAY-RECORD.
                *** CORRESPONDING items for statement 256:
                ***     CUST
                ***     ORDERN
                ***     ORDDAT
                ***     CUSORD
                ***     SHPVIA
                ***     ORDAMT
                ***     INVNUM
                ***     PRTDAT
                ***     ACTMTH
                ***     ACTYR
                ***     STATE
                *** End of CORRESPONDING items for statement 256
  257     027200     MOVE CORR CUSMST OF CUSTOMER-MASTER-RECORD
          027300         TO SUBCTL1-O OF EXISTING-ORDER-DISPLAY-RECORD.
                *** CORRESPONDING items for statement 257:
                ***     CUST
                ***     NAME
                ***     ADDR
                ***     CITY
                ***     STATE
                ***     ZIP
                *** End of CORRESPONDING items for statement 257
  258     027400     MOVE ORDER-STATUS(ORDSTS) TO STSORD.
  259     027500     MOVE OPEN-STATUS(OPNSTS) TO STSOPN.
  260     027600     SET MORE-DETAIL-LINE-ITEMS-EXIST TO TRUE.
  261     027700     MOVE ZEROS TO SUBFILE-RECORD-NUMBER.
          027800
          027900 BUILD-DISPLAY-SUBFILE.
  262     028000     MOVE CORR ORDDTL OF ORDER-DETAIL-RECORD
          028100         TO SUB1 OF EXISTING-ORDER-DISPLAY-RECORD.
                *** CORRESPONDING items for statement 262:
                ***     ITEM
                ***     QTYORD
                ***     DESCRP
                ***     PRICE
                ***     EXTENS
                *** End of CORRESPONDING items for statement 262
  263     028200     SET WRITE-DISPLAY TO TRUE.
  264     028300     SET SUB1-FORMAT TO TRUE.
  265     028400     ADD 1 TO SUBFILE-RECORD-NUMBER.
  266     028500     WRITE SUBFILE EXISTING-ORDER-DISPLAY-RECORD FORMAT IS "SUB1"
          028600     END-WRITE
  267     028700     IF SUBFILE-IS-FULL THEN
  268     028800         SET DISPLAY-SUBFILE TO TRUE
          028900     ELSE
  269     029000         PERFORM READ-NEXT-ORDER-DETAIL-RECORD
  270     029100         IF MORE-DETAIL-LINE-ITEMS-EXIST THEN
  271     029200             IF ORDERN OF ORDER-DETAIL-RECORD IS NOT EQUAL TO
          029300                ORDERN OF ORDER-HEADER-RECORD THEN
  272     029400                   SET DISPLAY-SUBFILE TO TRUE
  273     029500                   SET NO-MORE-DETAIL-LINE-ITEMS TO TRUE
          029600             END-IF
          029700         END-IF
          029800     END-IF.
          029900
          030000 READ-NEXT-ORDER-DETAIL-RECORD.
  274     030100     READ ORDER-DETAIL-FILE NEXT RECORD
  275     030200         AT END SET DISPLAY-SUBFILE TO TRUE
  276     030300                SET NO-MORE-DETAIL-LINE-ITEMS TO TRUE
          030400     END-READ.
                         * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 149. 発注照会プログラムの例 (10/10)

これは、ワークステーションに表示される最初の注文入力プロンプトの表示画面です。

```
Existing Order Entry                    Total  000000000
                                        Status
Order 12400                             Open
Date 000000                             Customer order
Cust #                                  Ship via
                            00000       Printed date 000000
                          Invoice 00000                     Mth 00  Year 00
Item  Qty   Item Description              Price   Extension
```

最初の表示画面に入力された注文番号に対応する得意先に対する詳細注文レコードが存在する場合、次の表示画面が表示されます。

```
Existing Order Entry                    Total  007426656
                                        Status 7-INVOICED
Order 17924   ABC HARDWARE LTD.         Open   2-CLOSED
Date 110896   123 ANYWHERE AVE.         Customer order TESTCS179330001I
Cust # 11200  TORONTO                   Ship via       TRUCKCO
              ONT            M4K 0A0    Printed date 110896
                          Invoice 17924                     Mth 12  Year 88
Item  Qty   Item Description              Price   Extension
33001  003  TORQUE WRENCH 75LB 14 INCH    009115       273.45
33100  001  TORQUE WRENCH W/GAUGE 200 LB  015777       650.95
44529  004  WOOD CHISEL - 3 1/4           006840        56.87
44958  002  POWER DRILL 1/2 REV           008200       797.50
46102  001  WROUGHT IRON RAILING 4FTX6FT  007930       237.75
46201  001  WROUGHT IRON HAND RAIL 6FT    007178        77.35
47902  002  ESCUTCHEON BRASS 15X4 INCHES  044488       213.00
```

次の表示画面は、最初の表示画面に入力された注文番号に対応するレコードが ORDHDRP ファイルに含まれていないときに表示されるものです。

```
Existing Order Entry                    Total  000000000
                                        Status
Order 12400                             Open
Date 000000                             Customer order
Cust #                                  Ship via
                            00000       Printed date 000000
                          Invoice 00000                     Mth 00  Year 00
Item  Qty   Item Description              Price   Extension









Order number not found
```

支払更新プログラムで READ SUBFILE...NEXT MODIFIED および REWRITE SUBFILE を使用する例

648 ページの図 152 に示すのは、支払更新プログラムの例 PAYUPDT です。これに関連する DDS については 645 ページの図 150 および 646 ページの図 151 を参照してください。関連した表示画面の例については 660 ページの得意先支払い表示

画面を参照してください。得意先マスター・ファイル CUSMSTP の DDS については 590 ページの図 132 を参照してください。

この例では、得意先からの支払いが登録されます。オペレーターに対して、1 つまたは複数の得意先番号を入力して、各得意先の口座に入れる金額を入力するようにプロンプトが出されます。プログラムは得意先番号を調べて、送り状が未処理になっている既存の得意先の支払いであれば、無条件に受け入れます。得意先からの支払額によって過剰支払いが起きた場合、オペレーターはその支払いを受け取るか受け取らないかを選択できます。得意先番号に対する得意先レコードが存在していない場合は、エラー・メッセージが出されます。オペレーターが F12 を押してプログラムを終了するまで、支払いの入力を続けることができます。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
    A** THIS IS THE ORDER HEADER LOGICAL FILE ** ORDHDRL
    A
    A
    A                                       UNIQUE
    A          R ORDHDR                     PFILE(ORDHDRP)
    A*
    A            CUST
    A            INVNUM
    A            ORDERN
    A            ORDDAT
    A            CUSORD
    A            SHPVIA
    A            ORDSTS
    A            OPRNAM
    A            ORDAMT
    A            CUSTYP
    A            PRTDAT
    A            OPNSTS
    A            TOTLIN
    A            ACTMTH
    A            ACTYR
    A            STATE
    A            AMPAID
    A          K CUST
    A          K INVNUM
```

図 150. 支払更新プログラムのデータ記述仕様の例 - 論理注文ファイル

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
    A* THIS IS THE DISPLAY DEVICE FILE FOR PAYUPDT ** PAYUPDTD
    A* ACCOUNTS RECEIVABLE INTERACTIVE PAYMENT UPDATE
    A*
    A
    A          R SUBFILE1                  SFL
    A                                      TEXT('SUBFILE FOR CUSTOMER PAYMENT'
    A*
    A            ACPPMT         4A  I  5  4TEXT('ACCEPT PAYMENT')
    A                                      VALUES('*YES' '*NO')
    A  51                                  DSPATR(RI MDT)
    A N51                                  DSPATR(ND PR)
    A*
    A            CUST           5   B  5 15TEXT('CUSTOMER NUMBER')
    A  52                                  DSPATR(RI)
    A  53                                  DSPATR(ND)
    A  54                                  DSPATR(PR)
    A*
    A            AMPAID         8 02B  5 24TEXT('AMOUNT PAID')
    A                                      CHECK(FE)
    A                                      AUTO(RAB)
    A                                      CMP(GT 0)
    A  52                                  DSPATR(RI)
    A  53                                  DSPATR(ND)
    A  54                                  DSPATR(PR)
    A*
    A            ECPMSG        31A  O  5 37TEXT('EXCEPTION MESSAGE')
    A  52                                  DSPATR(RI)
    A  53                                  DSPATR(ND)
    A  54                                  DSPATR(BL)
    A*
    A            OVRPMT         8Y 2O  5 70TEXT('OVERPAYMENT')
    A                                      EDTCDE(1)
    A  55                                  DSPATR(BL)
    A N56                                  DSPATR(ND)
    A*
```

図 151. 支払更新プログラムのデータ記述仕様の例 - ディスプレイ装置ファイル (1/2)

```
A            STSCDE         1A  H      TEXT('STATUS CODE')
A          R CONTROL1                   TEXT('SUBFILE CONTROL')
A                                       SFLCTL(SUBFILE1)
A                                       SFLSIZ(17)
A                                       SFLPAG(17)
A  61                                   SFLCLR
A  62                                   SFLDSP
A  62                                   SFLDSPCTL
A                                       OVERLAY
A                                       LOCK
A*
A                                       HELP(99 'HELP KEY')
A                                             CA12(98 'END PAYMENT UPDATE')
A                                       CA11(97 'IGNORE INPUT')
A*
A  99                                   SFLMSG(' F11 - IGNORE INVALID INPUT+
A                                                    F12 - END PAYMENT +
A                                       UPDATE')
A*
A                                     1  2'CUSTOMER PAYMENT UPDATE PROMPT'
A                                     1 65'DATE'
A                                     1 71DATE EDTCDE(Y)
A  63                                 3  2'ACCEPT'
A  63                                 4  2'PAYMENT'
A                                     3 14'CUSTOMER'
A                                     3 26'PAYMENT'
A  64                                 3 37'EXCEPTION MESSAGE'
A*
A          R MESSAGE1                   TEXT('MESSAGE RECORD')
A                                       OVERLAY
A                                       LOCK
A*
A  71                                24  2' ACCEPT PAYMENT VALUES: (*NO *YES)
                                        DSPATR(RI)
```

図 151. 支払更新プログラムのデータ記述仕様の例 - ディスプレイ装置ファイル (2/2)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/PAYUPDT         ISERIES1   06/02/15 15:08:37        ページ     2
                                ソ ー ス
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
         000100 PROCESS APOST
    1    000200 IDENTIFICATION DIVISION.
    2    000300 PROGRAM-ID.  PAYUPDT.
         000400
    3    000500 ENVIRONMENT DIVISION.
    4    000600   CONFIGURATION SECTION.
    5    000700 SOURCE-COMPUTER. IBM-ISERIES
    6    000800 OBJECT-COMPUTER. IBM-ISERIES
    7    000900   INPUT-OUTPUT SECTION.
    8    001000   FILE-CONTROL.
    9    001100     SELECT CUSTOMER-INVOICE-FILE
   10    001200         ASSIGN TO DATABASE-ORDHDRL
   11    001300         ORGANIZATION IS INDEXED
   12    001400         ACCESS MODE IS SEQUENTIAL
   13    001500         RECORD KEY IS COMP-KEY
   14    001600         FILE STATUS IS STATUS-CODE-ONE.
   15    001700     SELECT CUSTOMER-MASTER-FILE
   16    001800         ASSIGN TO DATABASE-CUSMSTP
   17    001900         ORGANIZATION IS INDEXED
   18    002000         ACCESS IS RANDOM
   19    002100         RECORD KEY IS CUST OF CUSTOMER-MASTER-RECORD.
   20    002200     SELECT PAYMENT-UPDATE-DISPLAY-FILE
   21    002300         ASSIGN TO WORKSTATION-PAYUPDTD
   22    002400         ORGANIZATION IS TRANSACTION
   23    002500         ACCESS IS DYNAMIC
   24    002600         RELATIVE KEY IS REL-NUMBER
   25    002700         FILE STATUS IS STATUS-CODE-ONE
   26    002800         CONTROL-AREA IS WS-CONTROL.
         002900
   27    003000 DATA DIVISION.
   28    003100 FILE SECTION.
   29    003200 FD  CUSTOMER-INVOICE-FILE.
   30    003300 01  CUSTOMER-INVOICE-RECORD.
         003400     COPY DDS-ORDHDR OF ORDHDRL.
        +000001*    I-O FORMAT:ORDHDR     FROM FILE ORDHDRL    OF LIBRARY CBLGUIDE          ORDHDR
        +000002*                                                                            ORDHDR
        +000003*          USER SUPPLIED KEY BY RECORD KEY CLAUSE                            ORDHDR
   31   +000004       05  ORDHDR.                                                          ORDHDR
   32   +000005           06 CUST                 PIC X(5).                              ORDHDR
        +000006*                  CUSTOMER NUMBER                                         ORDHDR
   33   +000007           06 INVNUM               PIC S9(5)         COMP-3.              ORDHDR
        +000008*                  INVOICE NUMBER                                          ORDHDR
   34   +000009           06 ORDERN               PIC S9(5)         COMP-3.              ORDHDR
        +000010*                  ORDER NUMBER                                            ORDHDR
   35   +000011           06 ORDDAT               PIC S9(6)         COMP-3.              ORDHDR
        +000012*                  DATE ORDER ENTERED                                      ORDHDR
   36   +000013           06 CUSORD               PIC X(15).                             ORDHDR
        +000014*                  CUSTOMER PURCHASE ORDER NUMBER                          ORDHDR
   37   +000015           06 SHPVIA               PIC X(15).                             ORDHDR
        +000016*                  SHIPPING INSTRUCTIONS                                   ORDHDR
   38   +000017           06 ORDSTS               PIC S9(1)         COMP-3.              ORDHDR
        +000018*                  ORDER SATAUS 1PCS 2CNT  3CHK 4RDY 5PRT 6PCK           ORDHDR
   39   +000019           06 OPRNAM               PIC X(10).                             ORDHDR
```

図 152. 支払更新プログラム例のソース・リスト (1/13)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/PAYUPDT         ISERIES1   06/02/15 15:08:37        ページ     3
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
       +000020*                OPERATOR WHO ENTERED ORD                                    ORDHDR
  40   +000021           06 ORDAMT                PIC S9(6)V9(2)    COMP-3.               ORDHDR
       +000022*                DOLLAR AMOUNT OF ORDER                                      ORDHDR
  41   +000023           06 CUSTYP                PIC S9(1)         COMP-3.               ORDHDR
       +000024*                CUSTOMER TYPE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT                  ORDHDR
  42   +000025           06 PRTDAT                PIC S9(6)         COMP-3.               ORDHDR
       +000026*                DATE ORDER WAS PRINTED                                      ORDHDR
  43   +000027           06 OPNSTS                PIC S9(1)         COMP-3.               ORDHDR
       +000028*                ORDER OPEN STATUS 1=OPEN 2= CLOSE 3=CANCEL                  ORDHDR
  44   +000029           06 TOTLIN                PIC S9(3)         COMP-3.               ORDHDR
       +000030*                TOTAL LINE ITEMS IN ORDER                                   ORDHDR
  45   +000031           06 ACTMTH                PIC S9(2)         COMP-3.               ORDHDR
       +000032*                ACCOUNTING MONTH OF SALE                                    ORDHDR
  46   +000033           06 ACTYR                 PIC S9(2)         COMP-3.               ORDHDR
       +000034*                ACCOUNTING YEAR OF SALE                                     ORDHDR
  47   +000035           06 STATE                 PIC X(2).                               ORDHDR
       +000036*                STATE                                                       ORDHDR
  48   +000037           06 AMPAID                PIC S9(6)V9(2)    COMP-3.               ORDHDR
       +000038*                AMOUNT PAID                                                 ORDHDR
  49    003500 66  COMP-KEY RENAMES CUST THRU INVNUM.
        003600
  50    003700 FD  CUSTOMER-MASTER-FILE.
  51    003800 01  CUSTOMER-MASTER-RECORD.
        003900     COPY DDS-CUSMST OF CUSMSTP.
       +000001*    I-O FORMAT:CUSMST     FROM FILE CUSMSTP    OF LIBRARY CBLGUIDE          CUSMST
       +000002*                          CUSTOMER MASTER RECORD                            CUSMST
       +000003*          USER SUPPLIED KEY BY RECORD KEY CLAUSE                            CUSMST
  52   +000004       05  CUSMST.                                                          CUSMST
  53   +000005           06 CUST                  PIC X(5).                               CUSMST
       +000006*                CUSTOMER NUMBER                                             CUSMST
  54   +000007           06 NAME                  PIC X(25).                              CUSMST
       +000008*                CUSTOMER NAME                                               CUSMST
  55   +000009           06 ADDR                  PIC X(20).                              CUSMST
       +000010*                CUSTOMER ADDRESS                                            CUSMST
  56   +000011           06 CITY                  PIC X(20).                              CUSMST
       +000012*                CUSTOMER CITY                                               CUSMST
  57   +000013           06 STATE                 PIC X(2).                               CUSMST
       +000014*                STATE                                                       CUSMST
  58   +000015           06 ZIP                   PIC S9(5)         COMP-3.               CUSMST
       +000016*                ZIP CODE                                                    CUSMST
  59   +000017           06 SRHCOD                PIC X(6).                               CUSMST
       +000018*                CUSTOMER NUMBER SEARCH CODE                                 CUSMST
  60   +000019           06 CUSTYP                PIC S9(1)         COMP-3.               CUSMST
       +000020*                CUSTOMER TYPE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT                  CUSMST
  61   +000021           06 ARBAL                 PIC S9(6)V9(2)    COMP-3.               CUSMST
       +000022*                ACCOUNTS REC. BALANCE                                       CUSMST
  62   +000023           06 ORDBAL                PIC S9(6)V9(2)    COMP-3.               CUSMST
       +000024*                A/R AMT. IN ORDER FILE                                      CUSMST
  63   +000025           06 LSTAMT                PIC S9(6)V9(2)    COMP-3.               CUSMST
       +000026*                LAST AMT. PAID IN A/R                                       CUSMST
  64   +000027           06 LSTDAT                PIC S9(6)         COMP-3.               CUSMST
       +000028*                LAST DATE PAID IN A/R                                       CUSMST
  65   +000029           06 CRDLMT                PIC S9(6)V9(2)    COMP-3.               CUSMST
       +000030*                CUSTOMER CREDIT LIMIT                                       CUSMST
  66   +000031           06 SLSYR                 PIC S9(8)V9(2)    COMP-3.               CUSMST
```

図 152. 支払更新プログラム例のソース・リスト (2/13)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/PAYUPDT         ISERIES1   06/02/15 15:08:37        ページ     4
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
        +000032*                CUSTOMER SALES THIS YEAR                                    CUSMST
  67    +000033           06 SLSLYR               PIC S9(8)V9(2)    COMP-3.                CUSMST
        +000034*                CUSTOMER SALES LAST YEAR                                    CUSMST
         004000
  68     004100 FD  PAYMENT-UPDATE-DISPLAY-FILE.
  69     004200 01  PAYMENT-UPDATE-DISPLAY-RECORD.
         004300     COPY DDS-ALL-FORMATS OF PAYUPDTD.
  70    +000001       05  PAYUPDTD-RECORD PIC X(59).                                        <-ALL-FMTS
        +000002*  INPUT FORMAT:SUBFILE1   FROM FILE PAYUPDTD   OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000003*                         SUBFILE FOR CUSTOMER PAYMENT                       <-ALL-FMTS
  71    +000004       05  SUBFILE1-I    REDEFINES PAYUPDTD-RECORD.                         <-ALL-FMTS
  72    +000005           06 ACPPMT               PIC X(4).                                <-ALL-FMTS
        +000006*                ACCEPT PAYMENT                                             <-ALL-FMTS
  73    +000007           06 CUST                 PIC X(5).                                <-ALL-FMTS
        +000008*                CUSTOMER NUMBER                                            <-ALL-FMTS
  74    +000009           06 AMPAID               PIC S9(6)V9(2).                          <-ALL-FMTS
        +000010*                AMOUNT PAID                                                <-ALL-FMTS
  75    +000011           06 ECPMSG               PIC X(31).                               <-ALL-FMTS
        +000012*                EXCEPTION MESSAGE                                          <-ALL-FMTS
  76    +000013           06 OVRPMT               PIC S9(6)V9(2).                          <-ALL-FMTS
        +000014*                OVERPAYMENT                                                <-ALL-FMTS
  77    +000015           06 STSCDE               PIC X(1).                                <-ALL-FMTS
        +000016*                STATUS CODE                                                <-ALL-FMTS
        +000017* OUTPUT FORMAT:SUBFILE1   FROM FILE PAYUPDTD   OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000018*                         SUBFILE FOR CUSTOMER PAYMENT                       <-ALL-FMTS
  78    +000019       05  SUBFILE1-O    REDEFINES PAYUPDTD-RECORD.                         <-ALL-FMTS
  79    +000020           06 SUBFILE1-O-INDIC.                                              <-ALL-FMTS
  80    +000021                07 IN51            PIC 1  INDIC 51.                         <-ALL-FMTS
  81    +000022                07 IN52            PIC 1  INDIC 52.                         <-ALL-FMTS
  82    +000023                07 IN53            PIC 1  INDIC 53.                         <-ALL-FMTS
  83    +000024                07 IN54            PIC 1  INDIC 54.                         <-ALL-FMTS
  84    +000025                07 IN55            PIC 1  INDIC 55.                         <-ALL-FMTS
  85    +000026                07 IN56            PIC 1  INDIC 56.                         <-ALL-FMTS
  86    +000027           06 CUST                 PIC X(5).                                <-ALL-FMTS
        +000028*                CUSTOMER NUMBER                                            <-ALL-FMTS
  87    +000029           06 AMPAID               PIC S9(6)V9(2).                          <-ALL-FMTS
        +000030*                AMOUNT PAID                                                <-ALL-FMTS
  88    +000031           06 ECPMSG               PIC X(31).                               <-ALL-FMTS
        +000032*                EXCEPTION MESSAGE                                          <-ALL-FMTS
  89    +000033           06 OVRPMT               PIC S9(6)V9(2).                          <-ALL-FMTS
        +000034*                OVERPAYMENT                                                <-ALL-FMTS
  90    +000035           06 STSCDE               PIC X(1).                                <-ALL-FMTS
        +000036*                STATUS CODE                                                <-ALL-FMTS
        +000037*  INPUT FORMAT:CONTROL1   FROM FILE PAYUPDTD   OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000038*                         SUBFILE CONTROL                                    <-ALL-FMTS
  91    +000039       05  CONTROL1-I    REDEFINES PAYUPDTD-RECORD.                         <-ALL-FMTS
  92    +000040           06 CONTROL1-I-INDIC.                                              <-ALL-FMTS
  93    +000041                07 IN99            PIC 1  INDIC 99.                         <-ALL-FMTS
        +000042*                        HELP KEY                                           <-ALL-FMTS
  94    +000043                07 IN98            PIC 1  INDIC 98.                         <-ALL-FMTS
        +000044*                        END PAYMENT UPDATE                                 <-ALL-FMTS
  95    +000045                07 IN97            PIC 1  INDIC 97.                         <-ALL-FMTS
        +000046*                        IGNORE INPUT                                       <-ALL-FMTS
        +000047* OUTPUT FORMAT:CONTROL1   FROM FILE PAYUPDTD   OF LIBRARY CBLGUIDE          <-ALL-FMTS
        +000048*                         SUBFILE CONTROL                                    <-ALL-FMTS
```

図 152. 支払更新プログラム例のソース・リスト (3/13)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/PAYUPDT         ISERIES1   06/02/15 15:08:37        ページ    5
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
   96    +000049       05  CONTROL1-O    REDEFINES PAYUPDTD-RECORD.                          <-ALL-FMTS
   97    +000050           06 CONTROL1-O-INDIC.                                             <-ALL-FMTS
   98    +000051                07 IN61             PIC 1  INDIC 61.                       <-ALL-FMTS
   99    +000052                07 IN62             PIC 1  INDIC 62.                       <-ALL-FMTS
  100    +000053                07 IN99             PIC 1  INDIC 99.                       <-ALL-FMTS
         +000054*                           HELP KEY                                       <-ALL-FMTS
  101    +000055                07 IN63             PIC 1  INDIC 63.                       <-ALL-FMTS
  102    +000056                07 IN64             PIC 1  INDIC 64.                       <-ALL-FMTS
         +000057*  INPUT FORMAT:MESSAGE1   FROM FILE PAYUPDTD   OF LIBRARY CBLGUIDE         <-ALL-FMTS
         +000058*                          MESSAGE RECORD                                  <-ALL-FMTS
         +000059*      05  MESSAGE1-I    REDEFINES PAYUPDTD-RECORD.                       <-ALL-FMTS
         +000060* OUTPUT FORMAT:MESSAGE1   FROM FILE PAYUPDTD   OF LIBRARY CBLGUIDE         <-ALL-FMTS
         +000061*                          MESSAGE RECORD                                  <-ALL-FMTS
  103    +000062       05  MESSAGE1-O    REDEFINES PAYUPDTD-RECORD.                       <-ALL-FMTS
  104    +000063           06 MESSAGE1-O-INDIC.                                          <-ALL-FMTS
  105    +000064                07 IN71             PIC 1  INDIC 71.                       <-ALL-FMTS
          004400
  106     004500 WORKING-STORAGE SECTION.
          004600
  107     004700 01  REL-NUMBER                   PIC 9(05)
          004800                                    VALUE ZEROS.
          004900
  108     005000 01  WS-CONTROL.
  109     005100     05  WS-IND                   PIC X(02).
  110     005200     05  WS-FORMAT                PIC X(10).
  111     005300 01  SYSTEM-DATE.
  112     005400     05  SYSTEM-YEAR              PIC 99.
  113     005500     05  SYSTEM-MONTH             PIC 99.
  114     005600     05  SYSTEM-DAY               PIC 99.
  115     005700 01  PROGRAM-DATE.
  116     005800     05  PROGRAM-MONTH            PIC 99.
  117     005900     05  PROGRAM-DAY              PIC 99.
  118     006000     05  PROGRAM-YEAR             PIC 99.
  119     006100 01  FILE-DATE REDEFINES PROGRAM-DATE
          006200                                   PIC S9(6).
  120     006300 01  EXCEPTION-STATUS.
  121     006400     05  STATUS-CODE-ONE          PIC XX.
  122     006500         88  SUBFILE-IS-FULL      VALUE '0M'.
  123     006600 01  EXCEPTION-MESSAGES.
  124     006700     05  MESSAGE-ONE              PIC X(31)
          006800         VALUE 'CUSTOMER DOES NOT EXIST        '.
  125     006900     05  MESSAGE-TWO              PIC X(31)
          007000         VALUE 'NO INVOICES EXIST FOR CUSTOMER '.
  126     007100     05  MESSAGE-THREE            PIC X(31)
          007200         VALUE 'CUSTOMER HAS AN OVER PAYMENT OF'.
  127     007300 01  PROGRAM-VARIABLES.
  128     007400     05  AMOUNT-OWED              PIC S9(6)V99.
  129     007500     05  AMOUNT-PAID              PIC S9(6)V99.
  130     007600     05  INVOICE-BALANCE          PIC S9(6)V99.
  131     007700 01  ERRPGM-PARAMETERS.
  132     007800     05  DISPLAY-PARAMETER        PIC X(8)
          007900                                    VALUE 'PAYUPDTD'.
  133     008000     05  DUMMY-ONE                PIC X(6)
          008100                                    VALUE SPACES.
  134     008200     05  DUMMY-TWO                PIC X(6)
```

図 152. 支払更新プログラム例のソース・リスト (4/13)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/PAYUPDT         ISERIES1   06/02/15 15:08:37        ページ     6
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
        008300                                  VALUE SPACES.
  135   008400    05  STATUS-CODE-TWO.
  136   008500       10  PRIMARY               PIC X(1).
  137   008600       10  SECONDARY             PIC X(1).
  138   008700       10  FILLER                PIC X(5)
        008800                                  VALUE SPACES.
  139   008900    05  DUMMY-THREE              PIC X(10)
        009000                                  VALUE SPACES.
        009100
  140   009200 01  SWITCH-AREA.
  141   009300    05  SW01                     PIC 1.
  142   009400       88  WRITE-DISPLAY              VALUE B'1'.
  143   009500       88  READ-DISPLAY               VALUE B'0'.
  144   009600    05  SW02                     PIC 1.
  145   009700       88  SUBFILE1-FORMAT            VALUE B'1'.
  146   009800       88  NOT-SUBFILE1-FORMAT        VALUE B'0'.
  147   009900    05  SW03                     PIC 1.
  148   010000       88  CONTROL1-FORMAT            VALUE B'1'.
  149   010100       88  NOT-CONTROL1-FORMAT        VALUE B'0'.
  150   010200    05  SW04                     PIC 1.
  151   010300       88  NO-MORE-TRANSACTIONS-EXIST  VALUE B'1'.
  152   010400       88  TRANSACTIONS-EXIST          VALUE B'0'.
  153   010500    05  SW05                     PIC 1.
  154   010600       88  CUSTOMER-NOT-FOUND          VALUE B'1'.
  155   010700       88  CUSTOMER-EXIST              VALUE B'0'.
  156   010800    05  SW06                     PIC 1.
  157   010900       88  NO-MORE-INVOICES-EXIST      VALUE B'1'.
  158   011000       88  CUSTOMER-INVOICE-EXIST      VALUE B'0'.
  159   011100    05  SW07                     PIC 1.
  160   011200       88  NO-MORE-PAYMENT-EXIST       VALUE B'1'.
  161   011300       88  PAYMENT-EXIST               VALUE B'0'.
  162   011400    05  SW08                     PIC 1.
  163   011500       88  INPUT-ERRORS-EXIST          VALUE B'1'.
  164   011600       88  NO-INPUT-ERRORS-EXIST       VALUE B'0'.
  165   011700    05  SW09                     PIC 1.
  166   011800       88  OVER-PAYMENT-DISPLAYED-ONCE  VALUE B'1'.
  167   011900       88  OVER-PAYMENT-NOT-DISPLAYED   VALUE B'0'.
        012000
  168   012100 01  INDICATOR-AREA.
  169   012200    05  IN99                     PIC 1 INDIC 99.
  170   012300       88 HELP-IS-NEEDED             VALUE B'1'.
  171   012400       88 HELP-IS-NOT-NEEDED         VALUE B'0'.
  172   012500    05  IN98                     PIC 1 INDIC 98.
  173   012600       88 END-OF-PAYMENT-UPDATE      VALUE B'1'.
  174   012700    05  IN97                     PIC 1 INDIC 97.
  175   012800       88 IGNORE-INPUT               VALUE B'1'.
  176   012900    05  IN51                     PIC 1 INDIC 51.
  177   013000       88 DISPLAY-ACCEPT-PAYMENT     VALUE B'1'.
  178   013100       88 DO-NOT-DISPLAY-ACCEPT-PAYMENT  VALUE B'0'.
  179   013200    05  IN52                     PIC 1 INDIC 52.
  180   013300       88 REVERSE-FIELD-IMAGE        VALUE B'1'.
  181   013400       88 DO-NOT-REVERSE-FIELD-IMAGE  VALUE B'0'.
  182   013500    05  IN53                     PIC 1 INDIC 53.
  183   013600       88 DO-NOT-DISPLAY-FIELD       VALUE B'1'.
  184   013700       88 DISPLAY-FIELD              VALUE B'0'.
```

図 152. 支払更新プログラム例のソース・リスト (5/13)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/PAYUPDT         ISERIES1   06/02/15 15:08:37        ページ    7
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
  185     013800    05  IN54                      PIC 1 INDIC 54.
  186     013900       88 PROTECT-INPUT-FIELD            VALUE B'1'.
  187     014000       88 DO-NOT-PROTECT-INPUT-FIELD     VALUE B'0'.
  188     014100    05  IN55                      PIC 1 INDIC 55.
  189     014200       88 MAKE-FIELD-BLINK               VALUE B'1'.
  190     014300       88 DO-NOT-MAKE-FIELD-BLINK        VALUE B'0'.
  191     014400    05  IN56                      PIC 1 INDIC 56.
  192     014500       88 DISPLAY-OVER-PAYMENT           VALUE B'1'.
  193     014600       88 DO-NOT-DISPLAY-OVER-PAYMENT    VALUE B'0'.
  194     014700    05  IN61                      PIC 1 INDIC 61.
  195     014800       88 CLEAR-SUBFILE                  VALUE B'1'.
  196     014900       88 DO-NOT-CLEAR-SUBFILE           VALUE B'0'.
  197     015000    05  IN62                      PIC 1 INDIC 62.
  198     015100       88 DISPLAY-SCREEN                 VALUE B'1'.
  199     015200       88 DO-NOT-DISPLAY-SCREEN          VALUE B'0'.
  200     015300    05  IN63                      PIC 1 INDIC 63.
  201     015400       88 DISPLAY-ACCEPT-HEADING         VALUE B'1'.
  202     015500       88 DO-NOT-DISPLAY-ACCEPT-HEADING  VALUE B'0'.
  203     015600    05  IN64                      PIC 1 INDIC 64.
  204     015700       88 DISPLAY-EXCEPTION              VALUE B'1'.
  205     015800       88 DO-NOT-DISPLAY-EXCEPTION       VALUE B'0'.
  206     015900    05  IN71                      PIC 1 INDIC 71.
  207     016000       88 DISPLAY-ACCEPT-MESSAGE         VALUE B'1'.
  208     016100       88 DO-NOT-DISPLAY-ACCEPT-MESSAGE  VALUE B'0'.
          016200
  209     016300 PROCEDURE DIVISION.
          016400
  210     016500 DECLARATIVES.
          016600
          016700 TRANSACTION-ERROR SECTION.
          016800     USE AFTER STANDARD ERROR PROCEDURE
          016900         PAYMENT-UPDATE-DISPLAY-FILE.
          017000 WORK-STATION-ERROR-HANDLER.
  211     017100     IF NOT (SUBFILE-IS-FULL) THEN
  212     017200       DISPLAY 'ERROR IN PAYMENT-UPDATE' STATUS-CODE-ONE
          017300     END-IF.
          017400 END DECLARATIVES.
          017500
          017600 MAIN-PROGRAM SECTION.
          017700 MAINLINE.
  213     017800     OPEN I-O      CUSTOMER-INVOICE-FILE
          017900                   CUSTOMER-MASTER-FILE
          018000                   PAYMENT-UPDATE-DISPLAY-FILE.
          018100
  214     018200     MOVE ALL B'0' TO INDICATOR-AREA
          018300                      SWITCH-AREA.
  215     018400     ACCEPT SYSTEM-DATE FROM DATE
          018500     END-ACCEPT.
  216     018600     MOVE SYSTEM-YEAR  TO PROGRAM-YEAR.
  217     018700     MOVE SYSTEM-MONTH TO PROGRAM-MONTH.
  218     018800     MOVE SYSTEM-DATE  TO PROGRAM-DAY.
  219     018900     SET WRITE-DISPLAY
          019000         CONTROL1-FORMAT
          019100         DO-NOT-DISPLAY-OVER-PAYMENT
          019200         DO-NOT-PROTECT-INPUT-FIELD
```

図 152. 支払更新プログラム例のソース・リスト (6/13)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/PAYUPDT         ISERIES1   06/02/15 15:08:37        ページ     8
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
          019300         DO-NOT-REVERSE-FIELD-IMAGE
          019400         DO-NOT-MAKE-FIELD-BLINK
          019500         CLEAR-SUBFILE TO TRUE.
  220     019600     MOVE CORR INDICATOR-AREA TO CONTROL1-O-INDIC.
                 *** CORRESPONDING items for statement 220:
                 ***     IN99
                 ***     IN61
                 ***     IN62
                 ***     IN63
                 ***     IN64
                 *** End of CORRESPONDING items for statement 220
  221     019700     WRITE PAYMENT-UPDATE-DISPLAY-RECORD
          019800         FORMAT IS 'CONTROL1'
          019900     END-WRITE.
  222     020000     SET DO-NOT-CLEAR-SUBFILE TO TRUE.
  223     020100     PERFORM INITIALIZE-SUBFILE-RECORD 17 TIMES.
  224     020200     SET DISPLAY-SCREEN TO TRUE.
  225     020300     MOVE CORR INDICATOR-AREA TO CONTROL1-O-INDIC.
                 *** CORRESPONDING items for statement 225:
                 ***     IN99
                 ***     IN61
                 ***     IN62
                 ***     IN63
                 ***     IN64
                 *** End of CORRESPONDING items for statement 225
  226     020400     WRITE PAYMENT-UPDATE-DISPLAY-RECORD
          020500         FORMAT IS 'CONTROL1'
          020600     END-WRITE.
  227     020700     READ PAYMENT-UPDATE-DISPLAY-FILE RECORD
          020800         FORMAT IS 'CONTROL1'
          020900     END-READ.
  228     021000     MOVE CORR CONTROL1-I-INDIC TO INDICATOR-AREA.
                 *** CORRESPONDING items for statement 228:
                 ***     IN99
                 ***     IN98
                 ***     IN97
                 *** End of CORRESPONDING items for statement 228
          021100
  229     021200     PERFORM PROCESS-TRANSACTION-FILE
          021300       UNTIL END-OF-PAYMENT-UPDATE.
          021400
  230     021500     CLOSE CUSTOMER-INVOICE-FILE
          021600           CUSTOMER-MASTER-FILE
          021700           PAYMENT-UPDATE-DISPLAY-FILE.
  231     021800     STOP RUN.
          021900
          022000 PROCESS-TRANSACTION-FILE.
  232     022100     IF HELP-IS-NOT-NEEDED THEN
  233     022200        IF  IGNORE-INPUT THEN
  234     022300            SET WRITE-DISPLAY
          022400                CONTROL1-FORMAT
          022500                CLEAR-SUBFILE
          022600                DISPLAY-FIELD
          022700                DO-NOT-DISPLAY-OVER-PAYMENT
          022800                DO-NOT-PROTECT-INPUT-FIELD
```

図 152. 支払更新プログラム例のソース・リスト (7/13)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/PAYUPDT         ISERIES1   06/02/15 15:08:37        ページ    9
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
         022900                DO-NOT-REVERSE-FIELD-IMAGE
         023000                DO-NOT-DISPLAY-ACCEPT-PAYMENT
         023100                DO-NOT-DISPLAY-ACCEPT-HEADING
         023200                DO-NOT-DISPLAY-ACCEPT-MESSAGE
         023300                DO-NOT-MAKE-FIELD-BLINK TO TRUE
  235    023400             MOVE CORR INDICATOR-AREA TO CONTROL1-O-INDIC
               *** CORRESPONDING items for statement 235:
               ***     IN99
               ***     IN61
               ***     IN62
               ***     IN63
               ***     IN64
               *** End of CORRESPONDING items for statement 235
  236    023500             WRITE PAYMENT-UPDATE-DISPLAY-RECORD
         023600                 FORMAT IS 'CONTROL1'
         023700             END-WRITE
  237    023800             SET DO-NOT-CLEAR-SUBFILE TO TRUE
  238    023900             MOVE 0 TO REL-NUMBER
  239    024000             PERFORM INITIALIZE-SUBFILE-RECORD 17 TIMES
         024100          ELSE
  240    024200             SET TRANSACTIONS-EXIST
         024300                 DO-NOT-DISPLAY-ACCEPT-HEADING
         024400                 DO-NOT-DISPLAY-ACCEPT-MESSAGE
         024500                 DO-NOT-DISPLAY-EXCEPTION TO TRUE
  241    024600             PERFORM READ-MODIFIED-SUBFILE-RECORD
  242    024700             PERFORM TRANSACTION-VALIDATION
         024800                 UNTIL NO-MORE-TRANSACTIONS-EXIST
  243    024900             SET NO-INPUT-ERRORS-EXIST TO TRUE
  244    025000             PERFORM TEST-FOR-RECORD-INPUT-ERRORS
         025100                 VARYING REL-NUMBER
         025200                 FROM    1
         025300                 BY      1
         025400                 UNTIL   REL-NUMBER IS GREATER THAN 17
         025500                 OR      INPUT-ERRORS-EXIST
  245    025600             IF  NO-INPUT-ERRORS-EXIST THEN
  246    025700                 IF  OVER-PAYMENT-DISPLAYED-ONCE THEN
  247    025800                     SET WRITE-DISPLAY
         025900                         CONTROL1-FORMAT
         026000                         DO-NOT-DISPLAY-OVER-PAYMENT
         026100                         DO-NOT-PROTECT-INPUT-FIELD
         026200                         DO-NOT-REVERSE-FIELD-IMAGE
         026300                         DO-NOT-MAKE-FIELD-BLINK
         026400                         DO-NOT-DISPLAY-ACCEPT-PAYMENT
         026500                         DO-NOT-DISPLAY-ACCEPT-HEADING
         026600                         DO-NOT-DISPLAY-ACCEPT-MESSAGE
         026700                         DO-NOT-DISPLAY-EXCEPTION
         026800                         CLEAR-SUBFILE
         026900                         DISPLAY-FIELD
         027000                           TO TRUE
  248    027100                     MOVE CORR INDICATOR-AREA TO CONTROL1-O-INDIC
               *** CORRESPONDING items for statement 248:
               ***     IN99
               ***     IN61
               ***     IN62
               ***     IN63
```

図 152. 支払更新プログラム例のソース・リスト (8/13)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL AS400 用        CBLGUIDE/PAYUPDT         ISERIES1   06/02/15 15:08:37        ページ   10
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
               ***     IN64
               *** End of CORRESPONDING items for statement 248
  249     027200                    WRITE PAYMENT-UPDATE-DISPLAY-RECORD
          027300                        FORMAT IS 'CONTROL1'
          027400                    END-WRITE
  250     027500                    SET DO-NOT-CLEAR-SUBFILE TO TRUE
  251     027600                    MOVE 0 TO REL-NUMBER
  252     027700                    PERFORM INITIALIZE-SUBFILE-RECORD 17 TIMES
          027800                ELSE
  253     027900                    SET OVER-PAYMENT-DISPLAYED-ONCE TO TRUE
          028000                END-IF
          028100            END-IF
          028200         END-IF
          028300     END-IF.
  254     028400     SET WRITE-DISPLAY, DISPLAY-SCREEN TO TRUE.
  255     028500     MOVE CORR INDICATOR-AREA TO MESSAGE1-O-INDIC.
               *** CORRESPONDING items for statement 255:
               ***     IN71
               *** End of CORRESPONDING items for statement 255
  256     028600     WRITE PAYMENT-UPDATE-DISPLAY-RECORD
          028700         FORMAT IS 'MESSAGE1'
          028800     END-WRITE.
  257     028900     SET WRITE-DISPLAY, CONTROL1-FORMAT TO TRUE.
  258     029000     MOVE CORR INDICATOR-AREA TO CONTROL1-O-INDIC.
               *** CORRESPONDING items for statement 258:
               ***     IN99
               ***     IN61
               ***     IN62
               ***     IN63
               ***     IN64
               *** End of CORRESPONDING items for statement 258
  259     029100     WRITE PAYMENT-UPDATE-DISPLAY-RECORD
          029200         FORMAT IS 'CONTROL1'
          029300     END-WRITE.
  260     029400     READ PAYMENT-UPDATE-DISPLAY-FILE RECORD
          029500         FORMAT IS 'CONTROL1'
          029600     END-READ.
  261     029700     MOVE CORR CONTROL1-I-INDIC TO INDICATOR-AREA.
               *** CORRESPONDING items for statement 261:
               ***     IN99
               ***     IN98
               ***     IN97
               *** End of CORRESPONDING items for statement 261
          029800
          029900 READ-MODIFIED-SUBFILE-RECORD.
  262     030000     READ SUBFILE PAYMENT-UPDATE-DISPLAY-FILE
          030100         NEXT MODIFIED RECORD FORMAT IS 'SUBFILE1'
  263     030200         AT END SET NO-MORE-TRANSACTIONS-EXIST TO TRUE
          030300     END-READ.
          030400
          030500 TEST-FOR-RECORD-INPUT-ERRORS.
  264     030600     READ SUBFILE PAYMENT-UPDATE-DISPLAY-FILE RECORD
          030700         FORMAT IS 'SUBFILE1'
          030800     END-READ.
  265     030900     IF  STSCDE OF SUBFILE1-I IS EQUAL TO '1' THEN
```

図 152. 支払更新プログラム例のソース・リスト (9/13)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL               CBLGUIDE/PAYUPDT        ISERIES1   06/02/15 15:08:37       ページ   11
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
  266     031000         SET INPUT-ERRORS-EXIST TO TRUE
          031100     END-IF.
          031200
          031300 TRANSACTION-VALIDATION.
  267     031400     MOVE CUST OF SUBFILE1-I OF PAYMENT-UPDATE-DISPLAY-RECORD
          031500         TO CUST OF CUSTOMER-MASTER-RECORD.
  268     031600     SET CUSTOMER-EXIST TO TRUE.
  269     031700     READ CUSTOMER-MASTER-FILE
  270     031800         INVALID KEY SET CUSTOMER-NOT-FOUND TO TRUE
          031900     END-READ.
  271     032000     IF  CUSTOMER-EXIST THEN
  272     032100         MOVE CUST OF CUSMST TO CUST OF ORDHDR
  273     032200         MOVE ZEROES TO INVNUM
  274     032300         SET CUSTOMER-INVOICE-EXIST TO TRUE
  275     032400         PERFORM START-ON-CUSTOMER-INVOICE-FILE
  276     032500         IF  CUSTOMER-INVOICE-EXIST THEN
  277     032600             PERFORM READ-CUSTOMER-INVOICE-RECORD
  278     032700             IF CUSTOMER-INVOICE-EXIST THEN
  279     032800                 PERFORM CUSTOMER-MASTER-FILE-UPDATE
  280     032900                 MOVE AMPAID OF SUBFILE1-I TO AMOUNT-PAID
  281     033000                 SET PAYMENT-EXIST TO TRUE
  282     033100                 PERFORM PAYMENT-UPDATE
          033200                     UNTIL NO-MORE-INVOICES-EXIST
          033300                     OR NO-MORE-PAYMENT-EXIST
  283     033400                 IF  ARBAL OF CUSTOMER-MASTER-RECORD IS NEGATIVE
  284     033500                     SET MAKE-FIELD-BLINK
          033600                         DISPLAY-FIELD
          033700                         DO-NOT-REVERSE-FIELD-IMAGE
          033800                         OVER-PAYMENT-NOT-DISPLAYED
          033900                         DISPLAY-OVER-PAYMENT
          034000                         DISPLAY-EXCEPTION
          034100                         DO-NOT-DISPLAY-ACCEPT-PAYMENT
          034200                         PROTECT-INPUT-FIELD TO TRUE
  285     034300                     MOVE ARBAL TO OVRPMT OF SUBFILE1-O
  286     034400                     MOVE MESSAGE-THREE TO ECPMSG OF SUBFILE1-O
  287     034500                     MOVE '0' TO STSCDE OF SUBFILE1-O
  288     034600                     PERFORM REWRITE-DISPLAY-SUBFILE-RECORD
          034700                 ELSE
  289     034800                     SET DO-NOT-DISPLAY-FIELD
          034900                         DO-NOT-DISPLAY-OVER-PAYMENT
          035000                         DO-NOT-REVERSE-FIELD-IMAGE
          035100                         DO-NOT-MAKE-FIELD-BLINK
          035200                         DO-NOT-DISPLAY-ACCEPT-PAYMENT
          035300                         PROTECT-INPUT-FIELD TO TRUE
  290     035400                     MOVE SPACES TO ECPMSG OF SUBFILE1-O
  291     035500                     MOVE ZEROES TO OVRPMT OF SUBFILE1-O
  292     035600                     MOVE '0' TO STSCDE OF SUBFILE1-O
  293     035700                     PERFORM REWRITE-DISPLAY-SUBFILE-RECORD
          035800                 END-IF
          035900             ELSE
  294     036000                 PERFORM NO-CUSTOMER-INVOICE-ROUTINE
          036100             END-IF
          036200         ELSE
  295     036300             PERFORM NO-CUSTOMER-INVOICE-ROUTINE
          036400         END-IF
```

図 152. 支払更新プログラム例のソース・リスト (10/13)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/PAYUPDT         ISERIES1   06/02/15 15:08:37        ページ   12
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
          036500     ELSE
  296     036600         SET REVERSE-FIELD-IMAGE
          036700             DO-NOT-PROTECT-INPUT-FIELD
          036800             DISPLAY-FIELD
          036900             DO-NOT-DISPLAY-OVER-PAYMENT
          037000             DO-NOT-MAKE-FIELD-BLINK
          037100             DISPLAY-EXCEPTION
          037200             DO-NOT-DISPLAY-ACCEPT-PAYMENT
          037300             DO-NOT-PROTECT-INPUT-FIELD TO TRUE
  297     037400         MOVE ZEROES TO OVRPMT OF SUBFILE1-O
  298     037500         MOVE MESSAGE-ONE TO ECPMSG OF SUBFILE1-O
  299     037600         MOVE '1' TO STSCDE OF SUBFILE1-O
  300     037700         PERFORM REWRITE-DISPLAY-SUBFILE-RECORD
          037800     END-IF.
  301     037900     PERFORM READ-MODIFIED-SUBFILE-RECORD.
          038000
          038100 START-ON-CUSTOMER-INVOICE-FILE.
  302     038200     START CUSTOMER-INVOICE-FILE
          038300         KEY IS GREATER THAN COMP-KEY
  303     038400         INVALID KEY SET NO-MORE-INVOICES-EXIST TO TRUE
          038500     END-START.
          038600
          038700 READ-CUSTOMER-INVOICE-RECORD.
  304     038800     READ CUSTOMER-INVOICE-FILE NEXT RECORD
  305     038900         AT END SET NO-MORE-INVOICES-EXIST TO TRUE
          039000     END-READ.
  306     039100     IF  CUST OF CUSTOMER-MASTER-RECORD
          039200         IS NOT EQUAL TO CUST OF CUSTOMER-INVOICE-RECORD THEN
  307     039300         SET NO-MORE-INVOICES-EXIST TO TRUE
          039400     END-IF.
          039500
          039600 CUSTOMER-MASTER-FILE-UPDATE.
  308     039700     MOVE FILE-DATE TO LSTDAT OF CUSTOMER-MASTER-RECORD.
  309     039800     MOVE AMPAID OF SUBFILE1-I
          039900         TO LSTAMT OF CUSTOMER-MASTER-RECORD.
  310     040000     SUBTRACT AMPAID OF SUBFILE1-I
          040100         FROM ARBAL OF CUSTOMER-MASTER-RECORD.
  311     040200     REWRITE CUSTOMER-MASTER-RECORD
          040300         INVALID KEY
  312     040400             DISPLAY 'ERROR IN REWRITE OF CUSTOMER MASTER'
          040500     END-REWRITE.
          040600
          040700  REWRITE-DISPLAY-SUBFILE-RECORD.
  313     040800     MOVE AMPAID OF SUBFILE1-I TO AMPAID OF SUBFILE1-O.
  314     040900     MOVE CUST OF SUBFILE1-I TO CUST OF SUBFILE1-O.
  315     041000     SET WRITE-DISPLAY TO TRUE.
  316     041100     SET SUBFILE1-FORMAT TO TRUE.
  317     041200     MOVE CORR INDICATOR-AREA TO SUBFILE1-O-INDIC.
                *** CORRESPONDING items for statement 317:
                ***     IN51
                ***     IN52
                ***     IN53
                ***     IN54
                ***     IN55
                ***     IN56
```

図 152. 支払更新プログラム例のソース・リスト (11/13)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/PAYUPDT         ISERIES1   06/02/15 15:08:37        ページ   13
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
              *** End of CORRESPONDING items for statement 317
  318     041300     REWRITE SUBFILE PAYMENT-UPDATE-DISPLAY-RECORD
          041400         FORMAT IS 'SUBFILE1'
          041500     END-REWRITE.
          041600
          041700 NO-CUSTOMER-INVOICE-ROUTINE.
  319     041800     IF  STSCDE OF SUBFILE1-I IS EQUAL TO '1' THEN
  320     041900        IF  ACPPMT OF SUBFILE1-I IS EQUAL TO '*NO' THEN
  321     042000           SET DO-NOT-DISPLAY-FIELD
          042100               DO-NOT-DISPLAY-OVER-PAYMENT
          042200               DO-NOT-REVERSE-FIELD-IMAGE
          042300               DO-NOT-MAKE-FIELD-BLINK
          042400               DO-NOT-DISPLAY-ACCEPT-PAYMENT
          042500               PROTECT-INPUT-FIELD
          042600                 TO TRUE
  322     042700            MOVE SPACES TO ECPMSG OF SUBFILE1-O
  323     042800            MOVE ZEROES TO OVRPMT OF SUBFILE1-O
  324     042900            MOVE '0' TO STSCDE OF SUBFILE1-O
  325     043000            PERFORM REWRITE-DISPLAY-SUBFILE-RECORD
          043100        ELSE
  326     043200            PERFORM CUSTOMER-MASTER-FILE-UPDATE
  327     043300            SET MAKE-FIELD-BLINK
          043400                DISPLAY-FIELD
          043500                DO-NOT-REVERSE-FIELD-IMAGE
          043600                OVER-PAYMENT-NOT-DISPLAYED
          043700                DISPLAY-OVER-PAYMENT
          043800                DISPLAY-EXCEPTION
          043900                DO-NOT-DISPLAY-ACCEPT-PAYMENT
          044000                PROTECT-INPUT-FIELD
          044100                  TO TRUE
  328     044200            MOVE ARBAL TO OVRPMT OF SUBFILE1-O
  329     044300            MOVE MESSAGE-THREE TO ECPMSG OF SUBFILE1-O
  330     044400            MOVE '0' TO STSCDE OF SUBFILE1-O
  331     044500            PERFORM REWRITE-DISPLAY-SUBFILE-RECORD
          044600         END-IF
          044700     ELSE
  332     044800         SET REVERSE-FIELD-IMAGE
          044900             DISPLAY-FIELD
          045000             DO-NOT-PROTECT-INPUT-FIELD
          045100             DO-NOT-DISPLAY-OVER-PAYMENT
          045200             DISPLAY-EXCEPTION
          045300             DISPLAY-ACCEPT-PAYMENT
          045400             DISPLAY-ACCEPT-HEADING
          045500             DISPLAY-ACCEPT-MESSAGE
          045600             DO-NOT-MAKE-FIELD-BLINK
          045700               TO TRUE
  333     045800         MOVE ZEROS TO OVRPMT OF SUBFILE1-O
  334     045900         MOVE MESSAGE-TWO TO ECPMSG OF SUBFILE1-O
  335     046000         MOVE '1' TO STSCDE OF SUBFILE1-O
  336     046100         PERFORM REWRITE-DISPLAY-SUBFILE-RECORD
          046200     END-IF.
          046300
          046400 PAYMENT-UPDATE.
  337     046500     SUBTRACT AMPAID OF CUSTOMER-INVOICE-RECORD
          046600         FROM ORDAMT OF CUSTOMER-INVOICE-RECORD
```

図 152. 支払更新プログラム例のソース・リスト (12/13)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/PAYUPDT         ISERIES1   06/02/15 15:08:37        ページ   14
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付
         046700         GIVING AMOUNT-OWED.
  338    046800     SUBTRACT AMOUNT-PAID
         046900         FROM AMOUNT-OWED
         047000         GIVING INVOICE-BALANCE.
  339    047100     IF  INVOICE-BALANCE IS LESS THAN 0.01 THEN
  340    047200         MOVE 2 TO OPNSTS OF CUSTOMER-INVOICE-RECORD
  341    047300         MOVE ORDAMT OF CUSTOMER-INVOICE-RECORD
         047400           TO AMPAID OF CUSTOMER-INVOICE-RECORD
  342    047500         SUBTRACT AMOUNT-OWED
         047600             FROM AMOUNT-PAID
         047700     ELSE
  343    047800         ADD AMOUNT-PAID TO AMPAID OF CUSTOMER-INVOICE-RECORD
  344    047900         SET NO-MORE-PAYMENT-EXIST TO TRUE
         048000     END-IF.
  345    048100     REWRITE CUSTOMER-INVOICE-RECORD
         048200         INVALID KEY
  346    048300             DISPLAY 'ERROR IN REWRITE OF CUSTOMER INVOICE'
         048400     END-REWRITE.
  347    048500     IF  PAYMENT-EXIST THEN
  348    048600         PERFORM READ-CUSTOMER-INVOICE-RECORD
         048700     END-IF.
         048800
         048900 INITIALIZE-SUBFILE-RECORD.
  349    049000     MOVE SPACES TO CUST OF SUBFILE1-O.
  350    049100     MOVE SPACES TO ECPMSG OF SUBFILE1-O.
  351    049200     MOVE '0' TO STSCDE OF SUBFILE1-O.
  352    049300     MOVE ZEROS TO AMPAID OF SUBFILE1-O.
  353    049400     MOVE ZEROS TO OVRPMT OF SUBFILE1-O.
  354    049500     ADD 1 TO REL-NUMBER.
  355    049600     MOVE CORR INDICATOR-AREA TO SUBFILE1-O-INDIC.
               *** CORRESPONDING items for statement 355:
               ***     IN51
               ***     IN52
               ***     IN53
               ***     IN54
               ***     IN55
               ***     IN56
               *** End of CORRESPONDING items for statement 355
  356    049700     WRITE SUBFILE PAYMENT-UPDATE-DISPLAY-RECORD
         049800         FORMAT IS 'SUBFILE1'
         049900     END-WRITE.
                        * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 152. 支払更新プログラム例のソース・リスト (13/13)

これは、得意先番号と支払額の入力を要求するためにワークステーションに表示される最初の表示画面です。

```
Customer Payment Update Prompt                                  Date  11/08/96

            Customer    Payment


            ______   _________
            ______   _________
            ______   _________
            ______   _________
            ______   _________
            ______   _________
            ______   _________
            ______   _________
            ______   _________
            ______   _________

            ______   _________
            ______   _________
            ______   _________
            ______   _________
            ______   _________
```

次のように、得意先番号と支払額を入力します。

```
Customer Payment Update Prompt                                  Date  11/08/96

            Customer    Payment


             34500      2000
             40500     30000
             36000      2500
             12500       200
             22799      4500
             41900      7500
             10001      5000
             49500      2500
             13300      3500
             56900      4000
```

支払過剰になるものや得意先番号が正しくないものは、次に示すように画面に残り、そのことを示すメッセージが表示されます。

```
Customer Payment Update Prompt                                  Date  11/08/96
Accept      Customer    Payment    Exception Message
Payment

_____        40500       30000     NO INVOICES EXIST FOR CUSTOMER

_____        12500         200     NO INVOICES EXIST FOR CUSTOMER

_____        41900        7500     NO INVOICES EXIST FOR CUSTOMER
             10001        5000     CUSTOMER DOES NOT EXIST

_____        13300        3500     NO INVOICES EXIST FOR CUSTOMER









Accept payment values: (*NO *YES)
```

次のようにして、どの支払いを受け付けるかを指定します。

```
Customer Payment Update Prompt                                  Date  11/08/96

Accept      Customer    Payment    Exception Message
Payment

  *NO        40500       30000     NO INVOICES EXIST FOR CUSTOMER

  *YES       12500         200     NO INVOICES EXIST FOR CUSTOMER

  *NO        41900        7500     NO INVOICES EXIST FOR CUSTOMER
             10001        5000     CUSTOMER DOES NOT EXIST

  *NO        13300        3500     NO INVOICES EXIST FOR CUSTOMER









Accept payment values: (*NO *YES)
```

受け付けられた支払いは処理され、支払過剰が次にように表示されます。

```
Customer Payment Update Prompt                                  Date  11/08/96
Accept      Customer    Payment    Exception Message
Payment

            12500         200      CUSTOMER HAS AN OVERPAYMENT OF         58.50

            10001        5000      CUSTOMER DOES NOT EXIST
```

第 4 部 付録

付録 A. 言語サポートのレベル

COBOL 標準規格

標準 COBOL (xi ページの『本書について』で定義されているもの) は、12 の機能処理モジュールで構成されており、そのうちの 7 つは必須であり、5 つはオプションです。

7 つの必須モジュールは、中核、順次入出力、相対入出力、索引付き入出力、プログラム間連絡、ソート・マージ、およびソース・テキスト操作です。 5 つのオプショナルのモジュールは、組み込み関数、報告書作成機能、通信、デバッグ、セグメント化です。

モジュール内の言語エレメントは、レベル 1 エレメントとレベル 2 エレメントにクラス分けされます。 9 つのモジュール内のエレメントは、レベル 1 エレメントとレベル 2 エレメントに分けられます。 3 つのモジュール (SORT-MERGE、REPORT WRITER、および INTRINSIC FUNCTION) に含まれるものは、すべてレベル 1 エレメントです。たとえば、中核レベル 1 エレメントは、基本的な内部操作を実行します。中核レベル 2 エレメントは、より拡張された高度な内部処理を行います。

標準 COBOL には、上位サブセット、中位サブセット、および最小サブセットの 3 つのサブセットがあります。各サブセットは、7 つの必須モジュール (中核、順次入出力、相対入出力、索引付き入出力、プログラム間連絡、ソート・マージ、およびソース・テキスト操作) のレベルで構成されています。 5 つのオプショナルのモジュール (組み込み関数、報告書作成機能、通信、デバッグ、およびセグメント化) は、標準 COBOL の 3 つのサブセットでは必要とされません。

- 上位サブセットは、すべての必須モジュールの最上位レベルのすべての言語エレメントから構成されています。それには次のものが含まれます。
  - 中核、順次入出力、相対入出力、索引付き入出力、プログラム間連絡、およびソース・テキスト操作のレベル 2 エレメント。
  - ソート・マージのレベル 1 エレメント。
- 中位サブセットは、すべての必須モジュールのレベル 1 のすべての言語エレメントから構成されています。それには次のものが含まれます。
  - 中核、順次入出力、相対入出力、索引付き入出力、プログラム間連絡、ソート・マージ、およびソース・テキスト操作のレベル 1 エレメント。
- 最小サブセットは、中核、順次入出力、およびプログラム間連絡のモジュールのレベル 1 のすべての言語エレメントから構成されています。

5 つのオプショナルのモジュールは、どのサブセットでも、不可欠な部分ではありません。ただし、オプショナルのモジュールのすべて、または任意の組み合わせを、任意のサブセットに関連付けるか、あるいはまったく関連付けないこともできます。

ILE COBOL の言語サポートのレベル

ILE COBOL コンパイラーは、次のことをサポートしています。

- レベル 2 の順次入出力およびソース・テキスト操作
- レベル 1 の中核、相対入出力、索引付き入出力、プログラム間連絡、およびソート・マージのモジュール。

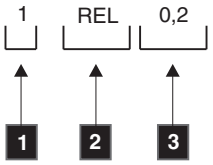
標準 COBOL の報告書作成機能、通信、デバッグ、およびセグメント化のモジュールは、ILE COBOL コンパイラーではサポートされていません。

ANSI X3.23a-1989 の組み込み関数モジュールは、ILE COBOL コンパイラーによって完全サポートされています。

ILE COBOL コンパイラーが提供するサポートのレベルを表 33 に示します。表には、次のことが示されています。

- 標準 COBOL の機能処理モジュールごとに、ILE COBOL コンパイラーがサポートしているレベル。
- 各モジュールの説明。

次に示すのは、この表で使われている表記法の説明です。



1 ILE COBOL コンパイラーでサポートされているこのモジュールのレベル。この例では、相対入出力モジュールのレベル 1 に対するサポートが提供されています。

2 モジュールを識別する 3 文字のコード。この例では、相対入出力モジュールです。

3 標準 COBOL で**定義されている**サポートのレベルの**範囲**。レベル 0 は、**最小**規格の COBOL では、標準規格に従うためにこのモジュールのサポートが必要ないことを表しています。

表 33. ILE COBOL コンパイラー・サポートのレベル

ILE COBOL でサポートされる言語のレベル	**モジュールの説明**
中核 1 NUC 1,2	プログラムの 4 つの基本的な部 (DIVISION) でのデータの内部処理、およびテーブルの定義とアクセスのために必要な言語エレメントが含まれています。
順次入出力 2 SEQ 1,2	ファイルに書き込まれている順序でファイル・レコードにアクセスすることができます。

表 33. ILE COBOL コンパイラー・サポートのレベル (続き)

ILE COBOL でサポートされる言語のレベル	モジュールの説明
相対入出力 1 REL 0,2	レコードに対するランダム・アクセスまたは順次アクセスが提供されます。各レコードは、ファイル内でのそのレコードの論理的な位置を表す整数によって、固有に識別されます。
索引付き入出力 1 INX 0,2	レコードに対するランダム・アクセスまたは順次アクセスが提供されます。索引付きファイル内の各レコードは、レコード・キーによって固有に識別されます。
プログラム間通信 1 IPC 1,2	一般的なデータ項目に対する制御およびアクセスの転送によって、COBOL プログラムが他のプログラムと通信できるようにします。
ソート・マージ 1 SRT 0,1	ユーザーが指定するキーに従って、1 つまたは複数のファイル内のレコードを並べ替えたり、同一規則で並べられた 2 つ以上のファイルを組み合わせたりします。
ソース・テキスト操作 2 STM 0,2	コンパイル時に、事前に定義された COBOL テキストをプログラム中に挿入することを可能にします。
報告書作成プログラム 0 RPW 0,1	印刷された報告書をほぼ自動的に作成する機能を提供します。
通信 0 COM 0,2	メッセージまたはメッセージの一部をアクセス、処理、および作成する機能を提供します。また、メッセージ制御システムによって、ローカルおよびリモートの通信装置との通信を可能にします。
デバッグ 0 DEB 0,2	デバッグのためのステートメントおよびプロシージャーを指定できます。
組み込み関数 1 ITR 0,1	オブジェクト・プログラムの実行中に参照された時点で、自動的に値が導かれるデータ項目を参照する機能を提供します。
セグメント化 0 SEG 0,2	手続き部 (PROCEDURE DIVISION) のセクションでのオブジェクト時オーバーレイ機能を提供します。

システム・アプリケーション体系 (SAA) 共通プログラミング・インターフェース (CPI) のサポート

プロダクト・ライブラリー QSYSINC 内のソース・ファイル QCBLLESRC には、複数の SAA 共通プログラミング・インターフェースの仕様が入っているメンバーが含まれています。これらの仕様には、パラメーター・インターフェースが記述されています。このファイルは IBM 所有のものであり、変更を加えることはできません。

仕様のいずれかをカスタマイズしたい場合は、変更したいメンバーをユーザーのライ
ブラリーのソース・ファイルにコピーする必要があります。これは、ファイル・

コピー (CPYF) コマンドを使用して行えます。 CPYF コマンドの詳細については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「プログラミング」カテゴリーの中の『*CL および API*』セクションを参照してください。

これらの仕様を自分のライブラリーにコピーした場合、プロダクトの新しいリリースがインストールされた時、またはプログラム一時修正 (PTF) を使用して変更が行われた時には、そのコピーを最新のものに更新する必要があります。 IBM は、配布ライブラリー内の仕様のみを維持管理します。

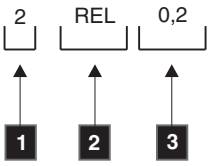
付録 B. 連邦情報処理標準 (FIPS) 標識機能

FIPS 標識機能を指定することにより、FIPS COBOL サブセット、すべてのオプショナル・モジュール、すべての古い言語エレメント、または FIPS COBOL やオプショナル・モジュールとすべての古いエレメントの組み合わせをモニターすることができます。

モニターとは、ソース・プログラムで使用されている構文を、ユーザーが選択した FIPS サブセットおよびオプショナル・モジュールに含まれている構文と比べる分析作業のことです。ソース・プログラムの中で、選択された FIPS COBOL サブセットとオプショナル・モジュールに適合しない構文が使用されているなら、それらの構文が識別されます。ソース・プログラム中で使用されている古い言語エレメントの構文についても識別されます (選択したコンパイラー・オプションに応じて異なります)。 FIPS 標識機能のパラメーターの詳細については 46 ページの『FLAGSTD パラメーター』を参照してください。

FIPS 21-4 COBOL 仕様は、標準 COBOL (xi ページの『本書について』で述べられているもの) の中に含まれている言語仕様です。 FIPS COBOL は、3 つのサブセットと 4 つのオプショナル・モジュールに分割されます。 3 つのサブセットは、最小、中位、および上位と呼ばれています。 4 つのオプショナル・モジュールは、報告書作成機能、通信、デバッグ、およびセグメント化です。これら 4 つのオプショナル・モジュールは、どのサブセットでも、不可欠な部分ではありません。しかし、オプショナル・モジュールのすべて、または任意の組み合わせを、サブセットに関連付けるか、あるいは全く関連付けないこともできます。 FIPS 21-4 COBOL 標準規格に準拠して作成されたプログラムは、FIPS 21-4 COBOL のいずれか 1 つのサブセットに準拠していなければなりません。 670 ページの表 34 に、FIPS 21-4 COBOL の各サブセットに含まれている ANSI 標準 COBOL 処理モジュールを示します。

次に示すのは、この表で使われている表記法の説明です。



1 FIPS 21-4 COBOL 標準規格でサポートされているこのモジュールのレベル。この例では、相対入出力モジュールのレベル 2 に対するサポートが提供されています。

2 モジュールを識別する 3 文字のコード。この例では、相対入出力モジュールです。

3 標準 COBOL で**定義されている**サポートのレベルの**範囲**。レベル 0 は、**最小**規格の COBOL では、標準規格に従うためにこのモジュールのサポートが必要ないことを表しています。

表 34. 標準 COBOL および FIPS 21-4 COBOL

ANSI モジュール名	上位 FIPS	中位 FIPS	最小 FIPS
中核	2 NUC 1,2	1 NUC 1,2	1 NUC 1,2
順次 I-O	2 SEQ 1,2	1 SEQ 1,2	1 SEQ 1,2
相対入出力	2 REL 0,2	1 REL 0,2	0 REL 0,2
索引付き入出力	2 INX 0,2	1 INX 0,2	0 INX 0,2
ソース・テキスト操作	2 STM 0,2	1 STM 0,2	0 STM 0,2
ソート・マージ	1 SRT 0,1	1 SRT 0,1	0 SRT 0,1
組み込み関数	1 ITR 0,1	0 ITR 0,1	0 ITR 0,1
プログラム間通信	2 IPC 1,2	1 IPC 1,2	1 IPC 1,2
報告書作成機能	0 または 1 RPW 0,1	0 または 1 RPW 0,1	0 または 1 RPW 0,1
セグメント化	0、1、または 2 SEG 0,2	0、1、または 2 SEG 0,2	0、1、または 2 SEG 0,2
デバッグ	0、1、または 2 DEB 0,2	0、1、または 2 DEB 0,2	0、1、または 2 DEB 0,2
通信	0、1、または 2 COM 0,2	0、1、または 2 COM 0,2	0、1、または 2 COM 0,2

ILE COBOL ソース・プログラムで指定されていて、FIPS 21-4 COBOL には含まれていないエレメントには 665 ページの『付録 A. 言語サポートのレベル』に示されているフラグが付けられます。

付録 C. ILE COBOL メッセージ

この付録には、ILE COBOL ライセンス・プログラムとともに IBM が提供するメッセージについての一般的な説明が記載されています。

COBOL メッセージについての説明

ILE COBOL ライセンス・プログラムのメッセージは、LNC、LNM、LNP、LNR、または LNT という接頭部から始まります。

- LNC メッセージは、ILE COBOL ソース・コードの入力に SEU が使われた場合に、COBOL 構文検査機能によって出されるものです。コンパイラー生成メッセージにも LNC メッセージがあります。
- LNM メッセージは、実行時 ILE COBOL 定様式ダンプのヘッディングとして使用されます。
- LNP メッセージは、ILE COBOL CL コマンドおよびメニューで使用されます。
- LNR メッセージは、実行時のシステム操作に関する追加情報を提供するものです。
- LNT メッセージは、ILE COBOL コンパイラー・リストのさまざまな部分のヘッディングとして使用されます。

メッセージ番号は、次のように割り当てられています。

エラー・メッセージ	説明
LNR7000 ～ LNR7199	エスケープ・メッセージ
LNR7200 ～ LNR7999	実行時メッセージ
LNR8000 ～ LNR8200	エスケープ・メッセージ
LNC0000 ～ LNC0999	重大度が 30 より小さいメッセージ
LNC1000 ～ LNC2999	重大度が 30 以上のメッセージ
LNC8000 ～ LNC8799	FIPS 標識機能メッセージ
LNC9001 ～ LNC9099	コンパイラー・メッセージ

重大度レベル

ILE COBOL ライセンス・プログラムのメッセージ重大度レベルには、以下のものがあります。

重大度	意味
00	通知: このレベルは、ユーザーに情報を伝えるために使用されます。エラーが発生したわけではありません。通知メッセージは、FLAG (00) オプションが指定されたときにのみ表示されます。
10	警告: このレベルは、エラーが検出されたが、プログラムの実行を妨げるほどのものではなかったことを示すものです。
20	エラー: このレベルは、エラーが発生したが、コンパイラーはリカバリー処置を取ったため、意図されたコードが作成されているかもしれないことを示すものです。

30 重大エラー: このレベルは、深刻なエラーが検出されたことを示すものです。コンパイルは完了しますが、モジュール・オブジェクトは作成されず、プログラムを実行することはできません。

40 リカバリー不能: 多くの場合、このレベルは、ユーザー・エラーのために処理を終了せざるを得なくなったことを示すものです。

50 リカバリー不能: 多くの場合、このレベルは、コンパイラー・エラーのために処理を終了せざるを得なくなったことを示すものです。

99 アクション: 応答の入力、印刷の用紙の変更、あるいはディスケットを入れ替えるなど、何らかの手動アクションが必要とされています。

注: PROCESS ステートメントの FLAG(30) オプションが使用されている場合、または CRTCBLMOD/CRTBNDCBL コマンドで FLAG(30) が指定されていて PROCESS ステートメントで指定変更されていない場合には、00、10、および 20 のメッセージは抑止されます。詳細については 60 ページの『PROCESS ステートメントを使用したコンパイラー・オプションの指定』を参照してください。

エラーが検出された場合でも、コンパイラーは必ずプログラムのソース・テキスト全体を診断しようと試みます。コンパイラーが、あるステートメントの診断を続けられない場合、コンパイラーがそのステートメントの診断を続けられないのでステートメントの残りの部分は無視される、というメッセージが出されます。このエラーが発生した場合、プログラマーはそのステートメント全体を調べる必要があります。

IBM i メッセージ機能は、すべてのメッセージを生成する場合に使用されます。ILE COBOL コンパイラー・メッセージはメッセージ・ファイル QLNCMSG に入れられており、実行時メッセージはメッセージ・ファイル QLNRMSG に入れられています。

置換変数および有効な応答値は、メッセージ・ファイル中に保管されているメッセージ記述によってではなく、そのメッセージを送るプログラムによって決められます。しかし、テキスト、重大度レベル、デフォルトの応答、あるいはダンプ・リストなど、メッセージ記述のうちの特定のエレメントは変更可能です。そのような変更を行うには、メッセージ記述の追加 (ADDMSGD) コマンドを使用して別のメッセージ記述を定義し、修正後の記述をユーザー作成メッセージ・ファイルに入れて、メッセージ・ファイル指定変更 (OVRMSGF) コマンドを使用してそのファイルを指定しておく必要があります。 OVRMSGF コマンドを使用すれば、コンパイラーは、指定されたファイルからメッセージを検索できるようになります。追加情報については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「*プログラミング*」カテゴリーの中の『*CL および API*』セクションの ADDMSGD コマンドおよび OVRMSGF コマンドの説明を参照してください。

注: IBM 提供のメッセージを、そのメッセージ・ファイルの中で変更および置換する必要がある場合には、サービス技術員にご連絡ください。

注意

IBM 提供のメッセージをユーザー作成のメッセージで置き換えると、予期しない結果が生じる場合があります。応答値が保存されていないと、プログラムが応答に反応しない場合があります。 *NOTIFY タイプのメッセージのデフォルト応答を変更すると、プログラムを不在モードで実行することができなくなる場合があります。重大度を変えると、以前には取り消されなかったジョブが取り消される可能性が生じます。 IBM 提供のメッセージをユーザー作成メッセージで置き換える場合には、十分な注意が必要です。

コンパイル・メッセージ

LNC メッセージは、プログラムのコンパイルの際にエラーが見つかった場合に、プログラム・リストの中に出力されます。 LNC メッセージには、連邦情報処理標準 (FIPS) 標識機能が要求された場合に出されるメッセージが含まれています。 FIPS メッセージの詳細については、この付録の 669 ページの『FIPS 標識機能』を参照してください。

プログラム・リスト

コンパイラー出力では、ソース・リストの後に ILE COBOL メッセージ・リストが出力されます。 ILE COBOL メッセージ・リストには、メッセージ ID、重大度、テキスト、通常はエラーの位置、およびメッセージの要約が含まれています。

CRTCBLMOD または CRTBNDCBL コマンドの OPTION パラメーターに *IMBEDERR という値が指定されている場合、第 1 レベル・メッセージ・テキストは、ソース・リストの中のエラーが検出された行のすぐ下にも出力されます。

プログラム・リストの詳細については 77 ページの『ソース・リスト』を参照してください。

対話式メッセージ

対話環境では、メッセージはワークステーションのディスプレイ装置に表示されます。それらは、プログラムの実行結果として、またはプロンプト、メニュー、コマンド入力画面、または IBM Rational Development Studio for System i のツールに対するキー入力への応答として、現在の画面に表示されます。メッセージは、DISPLAY コマンドやメニューのオプションの結果として、要求に応じて表示させることもできます。

ILE COBOL ライセンス・プログラムのメッセージは、LNC または LNR という接頭部から始まります。

LNC メッセージは、ILE COBOL ソース・コードの入力に原始ステートメント入力ユーティリティー (SEU) が使用された場合に、ILE COBOL 構文検査機能によって出されるものです。たとえば、PROGRAM-ID 段落に正しくないプログラム名を入力すると、次の画面が表示されます。

```
桁 . . . . . :   1  71          編集                          XMPLIB/QCBLLESRC
SEU==> _____________________________________________________          TESTPR
FMT CB ......-A+++B++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       *************** データの始め **********
0000.10        IDENTIFICATION DIVISION.
0000.20        PROGRAM-ID. #TESTPR.
0000.70        ENVIRONMENT DIVISION.
0000.90        SOURCE-COMPUTER. IBM-ISERIES.
       ***************** データの終わり ************************************
F3=終了    F4=プロンプト    F5=最新表示    F9=コマンドの複写    F10=カーソル F11= 切り替え
F16=検索の反復        F17=変更の反復          F24=キーの続き
 COBOL 文字が正しくない。20 桁目の '#' から無視されます。
```

図 153. ILE COBOL 構文検査機能のメッセージの例

LNC メッセージは、プログラムをコンパイルしているときにも出されます。そのことについては 673 ページの『コンパイル・メッセージ』を参照してください。

LNR メッセージは、実行時のシステム操作に関する追加情報を提供するものです。たとえば、実行時エラーがあると次のような画面が表示されます。

```
                          プログラム・メッセージの表示
QSYS のサブシステム QINTER のジョブ 008529/TESTLIB/QPADEV0003 が 94/04/08 15:
(C D F G) ライブラリー TESTLIB のプログラム・オブジェクト SAMPDUMP にメッセージ MCH1202


応答を入力して、実行キーを押してください。
  応答 . . .   ___________________________________________________________
_____________________________________________________________________________

F3=終了    F12=取り消し
```

図 154. 実行時エラー・メッセージ

メッセージ番号 MCH1202 が示されている行にカーソルを移動させて HELP キーまたは F1 キーのいずれかを押すと、次のような LNR メッセージ情報が表示されます。

```
                          追加のメッセージ情報
メッセージ ID. . . . . :   LNR7200       重大度 . . . . . :   50
メッセージ・タイプ. . :    照会
送信日付  . . . . . :   96/11/08       送信時刻 . . . . . :   15:33:31
メッセージ. . . . . :   (C D F G) ライブラリー TESTLIB のプログラム・オブ
    ジェクト SAMPDUMP にメッセージ MCH1202。
 原因 - ライブラリー TESTLIB のプログラム・オブジェクト SAMPDUMP の COBOL
    プログラム SAMPDUMP の COBOL ステートメント 42 でメッセージ MCH1202 が検出
    されました。
 回復手順 - 次の MI 命令からプログラムを続行するために G を入力するか、
    ダンプが必要な場合に C を入力するか、COBOL ID のダンプが必要な場合に D を
    入力するか、あるいは COBOL ID とファイル情報の両方をダンプするために F を
    入力してください。
    MCH1202 のメッセージ・テキストは次の通りです。10 進数データ・エラー
  メッセージに応答するための選択可能な項目:
   C -- 定様式ダンプは取られない。
   D -- COBOL ID のダンプが取られる。
                                                                  続く...
続行するには、実行キーを押してください。
F3= 終了   F6= 印刷     F9= メッセージ詳細の表示
F10= ジョブ・ログ中のメッセージの表示    F12= 取り消し   F21= 援助レベルの選択
```

```
                          追加のメッセージ情報
メッセージ ID. . . . . :   LNR7200       重大度 . . . . . :   50
メッセージ・. . . . . :   照会
   F -- COBOL ID とファイル情報のダンプが取られる。
   G -- 次の MI 命令からプログラムを続行する。

                                                                  終わり
続行するには、実行キーを押してください。

F3= 終了    F6= 印刷   F9= メッセージ詳細の表示
F10= ジョブ・ログ中のメッセージの表示   F12= 取り消し    F21= 援助レベルの選択
```

図 155. 実行時エラー・メッセージ - 2 次レベル・テキスト

『メッセージへの応答』では、2 次レベル・メッセージ・テキストを表示させたり、メッセージに応答したりする方法について説明します。

LNM メッセージ 0001～0050 は、ILE COBOL 定様式ダンプの際に出力される情報のヘッディングとして使用されます。

メッセージへの応答

対話環境では、メッセージは次の条件の 1 つまたは複数によって示されます。

- メッセージ行の短いメッセージ (第 1 レベル・テキスト)
- エラーのある入力フィールドの反転表示による強調表示
- ロックされたキーボード
- 警報音 (警報オプションがインストールされている場合)

以下の部分では、エラー・メッセージに応答するためのいくつかの方法について説明します。さらに詳しい情報については、IBM Rational Development Studio for System i の資料を参照してください。

最初の画面で必要な修正の方法が明らかであれば、キーボードがロックされている場合にはエラー・リセット・キーを押し、正しい情報を入力してから、作業を続行することができます。

応答を選択する必要があるメッセージ (たとえば、取り消しなら **C**、COBOL 識別名のダンプなら **D**、COBOL 識別名およびファイル情報のダンプなら **F**、次の

COBOL ステートメントの処理再開なら **G** など) では、応答オプションが第 1 レベル・メッセージ・テキストの括弧の中に示されます。例については 674 ページの図 154 を参照してください。

最初の情報表示画面の情報ではエラーを処理するために十分なデータが得られなかった場合は、(必要であればメッセージ行にカーソルを移動させた後) HELP キーを押すと、 2 次レベルの表示画面が表示され、それにはエラーを訂正する方法に関する追加情報が含まれています。最初の表示画面に戻るには、実行キーを押します。次いで、キーボードがロックされている場合にはエラー・リセット・キーを押してから、訂正や応答を行います。

プログラムをコンパイルしているときや実行しているときにエラーが起きたならば、ILE COBOL ソース・ステートメントまたは制御言語 (CL) コマンドを修正する必要があります。ステートメントを変更する方法については、「*AS/400 適用業務開発ツールセット AS/400 用 原始ステートメント入力ユーティリティー 使用者の手引きと参照*」を参照してください。

付録 D. 2 バイト文字セットを使用する国別言語のサポート

この付録では、COBOL プログラミング言語に対する拡張機能のうち、2 バイト文字を処理するプログラムの作成のためのものだけについて説明します。

特にこの付録では、COBOL プログラムの各部分の中で 2 バイト文字セット (DBCS) 文字を使用できる場所について、また ILE COBOL 言語で DBCS データを処理する際の考慮事項について説明します。

DBCS 文字を指定するには、次の 2 つの方法があります。
- シフト・コード付き DBCS
- DBCS グラフィック・データ

一般に COBOL では、シフト文字付き DBCS は英数字と同様に処理されます。**シフト文字付き DBCS** は、1 文字が 2 バイトで表される文字ストリングです。文字列はシフトアウト (SO) 文字で開始され、シフトイン (SI) 文字で終了します。どのデータ項目に DBCS 文字が含まれるかを知ること (または COBOL プログラムに検査させること)、およびプログラムが情報を正しく受け取って処理するようにすることは、プログラマーの責任です。

ILE COBOL プログラムで、DBCS グラフィック・データ・フィールドを定義する DDS 記述を使用できるようになりました。 **DBCS グラフィック**は、1 文字が 2 バイトで表される文字ストリングです。その文字ストリングには、シフトアウト文字またはシフトイン文字は含まれません。 ILE COBOL プログラムで外部記述ファイルに指定されたグラフィック・データ項目の処理についての情報は 504 ページの『DBCS グラフィック・フィールド』を参照してください。

リテラルの中で DBCS 文字を使用する

混合リテラルは、2 バイト文字セット (DBCS) と 1 バイト文字セット (SBCS) 文字とで構成されます。

PROCESS ステートメントの GRAPHIC オプションは、混合リテラルの中の DBCS 文字の処理に使用できます。 GRAPHIC オプションを指定する場合、混合リテラルは、16 進 0E および 16 進 0F が、それぞれシフトアウト文字およびシフトイン文字であると見なして処理され、それらで混合リテラルの DBCS 文字が囲まれます。NOGRAPHIC が明示的または暗黙のうちに指定されると、ILE COBOL コンパイラーは、16 進数 0E および 16 進数 0F を含む非数字リテラルを、SBCS 文字しか含まれていないものとして扱います。 16 進 0E および 16 進 0F はシフトイン文字およびシフトアウト文字として処理されず、SBCS 文字ストリングの一部であると見なされます。

DBCS リテラルは、2 バイト文字セットの文字だけで構成されているものであり、常に DBCS 文字ストリングとして扱われます。

注: PROCESS ステートメントの GRAPHIC オプションのことを、DDS 記述から 2 バイト・グラフィック・データを指定するために使用する PROCESS ステート

メントの CVTPICXGRAPHIC および CVTPICGGRAPHIC オプションや、CRTCBLMOD または CRTBNDCBL コマンドの CVTOPT パラメーターの *PICXGRAPHIC または *PICGGRAPHIC の値と混同しないようにしてください。グラフィック・データの指定の詳細については 504 ページの『DBCS グラフィック・フィールド』を参照してください。

DBCS 文字を含むリテラルの指定方法

DBCS 文字を含むリテラルを指定する場合、英数字リテラルを指定するときに適用されるのと同じ規則に従うとともに、混合および DBCS リテラルに特有の次の規則にも従ってください。

- 混合リテラルは、多くの異なる形式で作成可能です。次に示すのは、可能な 2 つの例にすぎません。

  "SINGLE0$_E$K1K2K30$_F$BYTES"

  "0$_E$K1K20$_F$"

- DBCS リテラルは、

  G"O$_E$ or N"0$_E$

  で始まり、その後に 1 文字以上の 2 バイト文字が続き、

  0$_F$"

  で終わります。たとえば、次のとおりです。

  G"0$_E$KIK20$_F$"
  N"0$_E$ 0$_F$"

- 混合リテラルには、暗黙のうちに USAGE DISPLAY が指定されます。 DBCS リテラルには、暗黙のうちに USAGE DISPLAY-1 が指定されます。
- 混合リテラル内にある任意の DBCS ストリングの前または後に EBCDIC 文字を入れることが可能です。
- すべての DBCS ストリングは、シフトイン文字とシフトアウト文字の間に現れます。**シフトアウト**文字は、2 バイト文字ストリングの開始を示す制御文字 (16 進数 0E) です。シフトアウト文字は、1 バイトを占めます。**シフトイン**文字は、2 バイト文字ストリングの終了を示す制御文字 (16 進数 0F) です。シフトイン文字は、1 バイトを占めます。
- 混合リテラル内にある SBCS 引用符はすべて二重にします。 G″ リテラル内の DBCS 引用符を二重にする必要はありませんが、N″ リテラル内の DBCS 引用符は二重にする必要があります。以下に例を示します。

  "Mixed ""0$_E$K1K2K30$_F$"" literal"
  G"0$_E$K1K2K3"K4"K5K60$_F$"
  N"0$_E$K1K2K3""K4""K5K60$_F$"

- ヌルの DBCS ストリング (シフトアウトおよびシフトイン文字だけで DBCS 文字がない) を混合リテラル内で使用できるのは、リテラルに最低 1 つの SBCS 文字が含まれる場合だけ です。

シフトアウトおよびシフトイン文字はネストできません。

シフト制御文字は混合リテラルの一部であり (純粋な DBCS リテラルではない)、すべての操作に影響します。

その他の考慮事項

引用符: 前述の説明ではリテラルを識別する文字を*引用符* としていましたが、実際に使用される文字は、CRTCBLMOD または CRTBNDCBL コマンド、または PROCESS ステートメント上に指定されるオプションによって異なる場合があります。 APOST オプションを指定した場合、アポストロフィ (') が使用されます。その他の場合、引用符 (") が使用されます。この付録では、*引用符* は、アポストロフィと引用符の両方を指します。どちらの文字を選択しても、リテラル指定に関する規則には影響しません。

シフト文字: シフトアウト文字とシフトイン文字は、EBCDIC 文字と DBCS 文字とを区切るものです。それらは混合リテラルの一部です。そのため、混合リテラル内に現れるシフト・コード文字は、すべての操作に影響します。 DBCS リテラルに存在する場合、操作には何の影響もありません。

COBOL コンパイラーが DBCS 文字を検査する方法

COBOL コンパイラーが DBCS ストリングを見つけると、DBCS を 1 文字ずつ走査することによりその DBCS ストリングを検査します。

次の条件があると、COBOL コンパイラーは DBCS を含むリテラルを*有効でない*と診断します。

- リテラルの構文が正しくない。
- 混合リテラルが 1 行を超えていて、非数値リテラルを継続するための規則に従っていない。 (詳細については『混合リテラルを次の行に継続する方法』を参照。)
- DBCS リテラルの長さが 1 行を超えている。

無効な DBCS リテラル、混合リテラル、または SBCS リテラルがあると、そのそれぞれについて、コンパイラーはエラー・メッセージを生成し、そのリテラルを受け入れるかまたは無視します。

混合リテラルを次の行に継続する方法

混合リテラルをソース・コードの次の行に継続するには、次のすべて を行います。

- 継続する行の 71 桁目または 72 桁目にシフトイン文字を入れる。 (71 桁目にシフトイン文字を入れた場合、72 桁目のブランクは無視されます。)
- 次の行の 7 桁目 (継続域) にハイフン (-) を入れる。
- 新しい行の区域 B に、引用符、次にシフトアウト文字、次にリテラルの残りを入れる。

以下に例を示します。

```
-A 1 B
   ⋮
 01  DBCS1             PIC X(12)                VALUE "0EK1K2K30F
-    "0EK4K50F".
   ⋮
```

DBCS1 の値は、"0_EK1K2K3K4K50_F" です。

行の継続に使用するシフトイン文字、引用符、およびシフトアウト文字は、混合リテラルの長さには数えません。最初のシフトアウト文字、および最後のシフトイン文字は数えます。

構文検査機能に関する考慮事項

構文検査機能には、リテラルを含む行を処理する際に、プログラムのコンパイル時のユーザーが GRAPHIC オプションを指定する意向かどうかを知る方法はありません。したがって、デフォルト・オプションの NOGRAPHIC が有効と見なします。すなわち、GRAPHIC オプションを指定してコンパイルした場合に有効な特定の混合リテラルが、構文エラーのフラグが付けられることを意味します。たとえば、次のとおりです。

```
"ABC0EK1K"0FDEF"
```

は、GRAPHIC オプションが指定されれば有効です。なぜなら、シフトアウト文字とシフトイン文字の間に置かれた二重引用符は DBCS 文字の 1 つのエレメントとして扱われるためです。しかし、構文検査機能は、この二重引用符をリテラルの終了文字と取り違え、残りの文字 (シフトイン文字 (SI) で始まる) にはエラーのフラグが付けられます。混合リテラルを、SBCS 非数値リテラルと純粋の DBCS リテラルの組み合わせに置き換えれば、これを避けることができます。

COBOL プログラム内で DBCS 文字を使用できる場所

一般的に、非数値リテラルが使用可能な場所なら、どこでも混合リテラルを使用できます。ただし、次の部分では、リテラルに 2 バイト文字を含めることができません。

- ALPHABET-名文節
- CURRENCY SIGN 文節
- ASSIGN 文節
- CLASS-名文節
- CALL ステートメント
- CANCEL ステートメント

非数値リテラルが使用可能な場所であれば、どこでも DBCS リテラルを使用できます。ただし、次の部分ではリテラルとして使用できません。

- ALPHABET 文節
- ASSIGN 文節
- CLASS 文節
- CURRENCY SIGN 文節
- LINKAGE 文節
- CALL ステートメントのプログラム ID
- CANCEL ステートメント
- END PROGRAM ステートメント
- PADDING CHARACTER 文節
- PROGRAM-ID 段落
- ACQUIRE ステートメント
- DROP ステートメント
- COPY ステートメントのテキスト名として
- COPY ステートメントのライブラリー名として

注: COBOL 語や名前に DBCS 文字を使用できます。 COBOL システム名、予約語、およびデータ名やファイル名などのユーザー定義語を様式設定する規則については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

コメントの書き方

プログラム行の 7 桁目にアスタリスク (*) または斜線 (/) を入れることにより、COBOL プログラムで DBCS 文字を含むコメントを書くことができます。どちらの記号の場合も、コンパイラーは 7 桁目以降の情報を文書として処理します。さらに斜線の場合、ページ排出も行われます。 COBOL コンパイラーはコメント行の内容を検査しないため、コメント内の DBCS 文字は検出されません。無効な DBCS 文字が使用されていると、コンパイラー・リストの印刷が正常に行われないことがあります。

見出し部 (IDENTIFICATION DIVISION)

DBCS 文字を含むコメント項目は、見出し部のどの部分にでも入れることができます (PROGRAM-ID 段落以外)。 PROGRAM-ID 段落に指定されるプログラム名は、英数字でなければなりません。

環境部 (ENVIRONMENT DIVISION)

構成セクション (CONFIGURATION SECTION)

コメント項目に DBCS 文字を使用できるのは、構成セクション段落だけです。関数名、簡略名、条件名、および英字名は、すべて英数字で指定する必要があります。SOURCE-COMPUTER および OBJECT-COMPUTER 項目には、次のように英数字のコンピューター名を使用します。

```
IBM-ISERIES
```

構成セクションでは混合リテラルは使用できません。その代わり、SPECIAL-NAMES 段落の CURRENCY SIGN 文節の中で、英字名とリテラルを定義するのに英数字リテラルを使用してください。 DBCS 英字またはクラスは存在しません。代わりに EBCDIC 文字セットを使用してください。

入出力セクション (INPUT-OUTPUT SECTION)

すべてのデータ名、ファイル名、および割り当て名は英数字で指定します。コメントには DBCS 文字を使用できます。

索引付きファイルの場合、RECORD KEY 文節内のデータ名は、レコード内の DBCS データ項目を参照できます。

DBCS 混合データを相対ファイルの RELATIVE KEY として使用することはできません。

ファイル制御 (FILE CONTROL) 段落

ASSIGN 文節

ASSIGN 文節でプリンターまたはデータベースなどの外部メディアを指定するのに、DBCS 文字を含むリテラルを使用することはできません。

データ部 (DATA DIVISION)

ファイル・セクション (FILE SECTION)

FD (ファイル記述) 項目では、VALUE OF 文節に DBCS データ項目またはリテラルを使用できます。 DATA RECORDS 文節が参照できるのはデータ項目だけです。 ILE COBOL コンパイラーは、FILE SECTION の VALUE OF 文節および DATA RECORDS 文節の両方を文書として処理するため、どちらの文節もプログラムの実行時には影響しません。しかし、COBOL コンパイラーは VALUE OF 文節のすべてのリテラルが有効かどうかの検査を行います。

磁気テープの場合、システムが DBCS 文字の読み取り、または DBCS 文字の書き込みを行えるのは EBCDIC 形式のテープだけです。システムは、ASCII 形式のテープに関してはテープ機能を実行できません。 CODE-SET 文節の英字名は、NATIVE または EBCDIC として定義してください。

作業用記憶域セクション (WORKING-STORAGE SECTION)

REDEFINES 文節

データを再定義するための既存の規則は、DBCS 文字を含むデータにも適用されます。再定義するまたは再定義したデータの長さを判別する場合、各 DBCS 文字は英数字の 2 倍の長さであることに注意してください。

さらに、再定義されたデータ項目では、必要な場合には必要な場所にシフト制御文字が含まれているようにしてください。

OCCURS 文節

この文節は、DBCS データを保管するテーブルを定義するのに使用します。ASCENDING/DESCENDING KEY 句を指定した場合、COBOL はテーブルの内容が EBCDIC プログラム照合順序であると見なします。混合データのシフト制御文字は、照合順序に含まれています。

DBCS 文字を含むテーブルの処理の詳細については 689 ページの『テーブル処理 - SEARCH ステートメント』を参照してください。

JUSTIFIED RIGHT 文節

JUSTIFIED RIGHT 文節は、DBCS データを基本受け取り側フィールドの右端にそろえます。受け取り側フィールドが送り側フィールドより短い場合、COBOL は右端の文字を切り捨てます。受け取り側フィールドが送り側フィールドより長い場合、COBOL は受け取り側フィールドの左側の使用されない部分にブランクを埋め込みます。

JUSTIFIED 文節は、VALUE 文節の初期設定に影響を与えません。

VALUE 文節

混合リテラルを使用して、数字以外のデータ項目の初期値を指定したり、レベル 88 条件名項目の値を定義したりすることができます。 DBCS または DBCS 編集データ項目の初期値を指定するには、DBCS リテラルを使用しなければなりません。

リテラル内のシフト制御文字は、次の行への継続に使用されている場合を除いて、そのリテラルのピクチャー・ストリングの一部と見なされます。混合リテラルを継続する場合、コンパイラーは 71 桁目または 72 桁目のシフトイン文字、または継続行の最初にある引用符 (") およびシフトアウト文字を、混合リテラルの一部には含めません。ただし、その混合リテラルが、PICTURE 文節で指定されたデータ項目のサイズを超えないようにしてください。超えた場合は切り捨てが行われます。

DBCS リテラルを使用して、DBCS データ項目を初期設定することができます。

レベル 88 条件名項目の VALUE 文節に DBCS 文字を含むリテラルを使用すると、COBOL はその DBCS 文字を英数字として処理します。したがって、THROUGH オプションの使用も含め、英数字データを指定する際の規則に従ってください。このオプションでは通常の EBCDIC 照合順序を使用しますが、DBCS データにあるシフト制御文字が照合順序に影響を与えることに注意してください。

PICTURE 文節

混合データ項目、および DBCS データ項目の G または N のどちらかを定義するには、PICTURE 記号 X を使用します。 *n* 個の DBCS 文字を含む DBCS データ項目は、次のようにして定義します。

```
PICTURE G(n) または PICTURE N(n)
```

m 個の SBCS 文字と、*n* 文字の DBCS 文字ストリング 1 個を含む混合データ項目は、次のようにして定義されます。

```
PICTURE X(m+2n+2)
```

混合データ項目には、編集した英数字 PICTURE 記号をすべて使用できます。編集記号がこれらの項目内の DBCS データに与える影響は、英数字データ項目に与える影響と同じです。期待した結果が得られたかどうかを確認してください。純粋な DBCS データ項目では、B 編集記号しか使用できません。

RENAMES 文節

この文節は、基本データ項目の代替グループ化を指定するのに使用します。英数字データ項目の名前を変更する際の既存の規則は、DBCS データ項目にも適用されます。

手続き部 (PROCEDURE DIVISION)

組み込み関数

いくつかの組み込み関数では、その引き数として、DBCS データ項目、DBCS リテラル、および混合リテラルを使用することができます。

組み込み関数の引き数のいずれかが DBCS データ項目または DBCS リテラルであるなら、組み込み関数で DBCS データ項目を戻すことも可能です。

DBCS 項目をサポートする組み込み関数については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

条件式

条件名 (レベル 88 項目) では DBCS 文字を含むデータ項目を参照できるため、条件名条件を使用してそのデータをテストできます。 (683 ページの『VALUE 文節』を参照。) 条件変数および条件名を使用する場合は、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」に示されている規則に従ってください。

DBCS データ項目または混合リテラルを、比較条件のオペランドに使用できます。COBOL は混合データを英数字として処理するので、すべての比較は英数字オペランドの規則に従って実行されます。 DBCS データ項目は、別の DBCS データ項目としか比較できません。次のことに注意してください。

- システムは混合の内容を認識できません。
- システムは混合データの比較にシフト・コードを使用します。
- システムは、EBCDIC 照合順序、またはユーザー定義順序のどちらかを使用してデータを比較します。
- DBCS 項目をサイズの異なる類似項目と比較する場合、小さい方の項目の右側にはスペースが埋め込まれます。

詳細については、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」の『SPECIAL-NAMES 段落』を参照してください。

「*IBM Rational Development Studio for i: ILE COBOL 解説書*」で説明されているクラス条件および状況交換条件を使用できます。

入出力ステートメント

ACCEPT ステートメント

形式 1 の ACCEPT ステートメントを使用して装置から受け取る入力データには、DBCS が含まれることがあります。すべての DBCS データは、適切な構文で記述されていなければなりません。シフト制御文字を除いて、DBCS データ項目の既存の内容は、入力データによって置き換えられます。シフト制御文字は、混合データ項目の内容に含まれています。 COBOL はデータの編集やエラー検査を行いません。

形式 3 の ACCEPT ステートメントを使用してファイルの OPEN-FEEDBACK 情報を得た場合、その情報には、ファイルに DBCS データと混合データのどちらが存在するかを示すフィールドが含まれています。

形式 4 の ACCEPT ステートメントを使用してローカル・データ域から受け取った情報には、DBCS または混合文字ストリングが含まれていることがあります。受け取った情報によって、既存の内容が置き換えられます。 COBOL はデータの編集やエラー検査を行いません。このことは、形式 5 の ACCEPT ステートメントによって PIP データ域から受け取る情報、および形式 9 の ACCEPT ステートメントによってユーザー定義データ域から受け取る情報にも当てはまります。

形式 6 の ACCEPT ステートメントを使用すると、ワークステーションのディスプレイ装置およびキーボードの属性を入手できます。 DBCS 文字を表示可能なディスプレイ装置については、システムが ATTRIBUTE-DATA データ項目に適切な値を設定します。装置の名前には DBCS 文字を使用できません。

フィールド・レベルのワークステーション入力に拡張 (形式 7) ACCEPT ステートメントを使用する場合、DBCS データが改行によって分割されないようにしてください。 COBOL はデータのエラー検査や編集を行いません。ただし、シフトイン文字およびシフトアウト文字の除去は必要に応じて行われます。

DISPLAY ステートメント

DISPLAY ステートメントには、DBCS、混合のデータ項目、またはリテラルを指定できます。データのタイプを混合して使用できます。データ項目またはリテラルの DBCS データまたは混合データは、DISPLAY ステートメントに指定されているターゲットであるプログラム装置、またはローカル・データ域かユーザー定義データ域に、そのまま送られます。

COBOL はデータが表示される装置の特性を判別しないので、DBCS および混合データが正確であることを確認しておく必要があります。

注: ALL は、混合リテラルに使用できる有効なオプションです。

形式 3 の DISPLAY ステートメントまたは形式 4 の DISPLAY ステートメントをフィールド・レベルのワークステーション出力に対して使用する場合、DBCS データが改行によって分割されないようにしなければなりません。

READ ステートメント

DBCS データ項目を索引ファイルの RECORD KEY として使用することができます。詳細については 681 ページの『入出力セクション (INPUT-OUTPUT SECTION)』を参照してください。

INTO 句: INTO 句を使用することにより、レコードを DBCS データ項目に読み込むことができます。この句によって、(CORRESPONDING オプションのない) MOVE ステートメントが実行されます。コンパイラーが DBCS データを移動する方法は、英数字データを移動する方法と同じです。コンパイラーはデータが有効であるかどうかを確認しません。

REWRITE ステートメント

DBCS データを DBCS データ項目から既存のレコードに転送するには、このステートメントの FROM 句を使用します。 FROM 句によって、どちらのタイプのデータも READ ステートメントの INTO 句と同様の方法で移動されます。 (『READ ステートメント』を参照。)

START ステートメント

DBCS 文字を索引ファイルのキー内で使用する場合、START ステートメントの KEY 句に対応するデータ項目を指定します。

次のうちのいずれか 1 つが満たされることが必要です。

- データ項目が、FILE-CONTROL 段落の RECORD KEY 文節で指定されたデータ項目と同じである。
- データ項目とレコード・キーの先頭文字が同じであり、データ項目がレコード・キーより長くない。

KEY 句には、有効な演算子 (EQUAL、GREATER THAN、NOT LESS THAN など) を指定できます。システムは、EBCDIC またはユーザー定義のどちらかの照合順序に従うことができます。

WRITE ステートメント

このステートメントの FROM 句を使用して、DBCS データをレコードに書き込むことができます。この句は、REWRITE ステートメントと同様の方法でデータを移動します。 (685 ページの『REWRITE ステートメント』を参照。)

データを装置ファイルに書き込む場合は、シフト制御文字を含めてください。

データ操作ステートメント

算術ステートメント

COBOL が DBCS 文字を処理する方法は SBCS 文字を処理する方法と同じなので、DBCS 文字を数値演算で使用したり、算術ステートメントを使用して操作したりはしないでください。

INSPECT ステートメント

任意の DBCS データ項目を INSPECT ステートメントのオペランドとして使用できます。システムはこれらのオペランドの DBCS 文字を、1 文字の半分ずつ、シフト制御文字を含めて検査および置換します。結果として、データの突き合わせが正確に行われない場合があります。

DBCS 文字オペランドは、別の DBCS 文字リテラルまたはデータ項目との組み合わせでしか使用できません。 混合オペランドは英数字として扱われます。
REPLACING 句を使用すると、検査される混合データ項目の一部を英数字データに置き換えたり、あるいは検査される英数字データ項目の一部を混合データに置き換えたりすることができます。

文字ストリングを、異なる長さのストリングで置き換えることはできません。混合データ項目の SBCS 文字を DBCS 文字で置き換えたり、混合データ項目の DBCS 文字を SBCS 文字で置き換えたりする場合には、このことを考慮してください。

INSPECT ステートメントでの DBCS 文字を含む混合項目の使用を制御したい場合は、シフト制御文字を含むデータ項目を定義してください。シフトアウト文字とシフトイン文字を INSPECT ステートメントの BEFORE/AFTER オペランドとして使用します。

次の例は、INSPECT ステートメントを使用して、混合データ項目内の 1 つの DBCS 文字を別の DBCS 文字に置き換える方法を示しています。

```
01  SUBJECT-ITEM                PICTURE X(50).
01  DBCS-CHARACTERS             VALUE "0EK1K20F".
    05  SHIFT-OUT      PICTURE X.
    05  DBCS-CHARACTER-1         PICTURE XX.
    05  DBCS-CHARACTER-2         PICTURE XX.
    05  SHIFT-IN       PICTURE X.
```

INSPECT ステートメントのコーディングは、次のようなものになります。

```
INSPECT SUBJECT-ITEM
    REPLACING ALL DBCS-CHARACTER-1
              BY  DBCS-CHARACTER-2
    AFTER INITIAL SHIFT-OUT.
```

注: AFTER INITIAL SHIFT-OUT 句を使用すると、EBCDIC 値が DBCS-CHARACTER-1 と同じである 2 つの連続した英数字を誤って置換する危険を避けるのに役立ちます (SUBJECT-ITEM に混合データが含まれる場合)。

さらに、INSPECT ステートメントを使用することにより、適切な処理が実行されるようにデータ中に DBCS 文字が含まれているかどうかを判別することができます。以下に例を示します。

```
01  SUBJECT-FIELD          PICTURE X(50).
01  TALLY-FIELD            PICTURE 9(3) COMP.
01  SHIFTS             VALUE "0E0F".
    05  SHIFT-OUT      PICTURE X.
    05  SHIFT-IN       PICTURE X.
```

手続き部 (PROCEDURE DIVISION) では、次のように入力できます。

```
MOVE ZERO TO TALLY-FIELD.
INSPECT SUBJECT-FIELD TALLYING TALLY-FIELD
                                FOR ALL  SHIFT-OUT.
IF TALLY-FIELD IS GREATER THAN ZERO THEN
    PERFORM DBCS-PROCESSING
ELSE
    PERFORM A-N-K-PROCESSING.
```

MOVE ステートメント

すべての DBCS 文字は英数字ストリングとして移動されます。システムはデータの変換や検査を行いません。

混合リテラルをグループ項目および英数字項目に移動できます。 DBCS データ項目または DBCS リテラルは、DBCS データ項目にしか移動できません。

受け取りフィールドの長さが送り側フィールドと異なる場合、COBOL は次のどちらかを行います。

- 送り出し項目が受け取り項目より長い場合は、送り出し項目の文字を切り捨てます。この操作は、データの保全性を損なうことがあります。
- 送り出し項目が受け取り項目より短い場合は、ブランクを埋め込みます。

受け取りデータ項目の PICTURE 文節での記号の編集効果についてさらに理解するには、「*IBM Rational Development Studio for i: ILE COBOL 解説書*」を参照してください。

SET ステートメント (条件名形式)

このステートメントで条件名を TRUE に設定すると、COBOL は VALUE 文節のリテラルをそれに対応するデータ項目に移します。 DBCS 文字を含むリテラルを移動させることができます。

STRING ステートメント

STRING ステートメントを使用して、DBCS サブフィールドを含むデータ項目を構成できます。シフト制御文字を含め、ソース・データ項目またはリテラル内のすべてのデータは、一度に 1 文字の DBCS 文字の半分ずつが受け取りデータ項目に移されます。

UNSTRING ステートメント

UNSTRING ステートメントは、DBCS データおよび混合データを英数字データと同様の方法で処理します。 UNSTRING 操作は、一度に 1 文字の DBCS 文字の半分ずつについて実行されます。

データ項目には、同じフィールド内に英数字と DBCS の両方を含めることができます。

データ・フィールドの中で 2 バイトおよび英数字サブフィールドを検索するには、DELIMITED BY 句を使用します。シフト制御文字を含むデータ項目を指定して、それらのデータ項目を DELIMITED BY 句の ID として使用します。これを行う方法については、次の例を参照してください。送り出しフィールドのサブフィールドの走査を継続するには、POINTER 変数を使用します。

システムが UNSTRING 操作を実行した後、DELIMITER IN 句で保管した区切り文字がシフト制御文字値かどうかを検査することによって、DBCS を含むのはどのサブフィールドか、またはどれに英数字が含まれるかを調べることができます。

次の例は、混合データを含む文字ストリングに対する UNSTRING 操作に備えてフィールドを設定する方法を示すものです。

```
01  SUBJECT-FIELD      PICTURE X(40)
01  FILLER.
    05  UNSTRING-TABLE OCCURS 4 TIMES.
        10  RECEIVER   PICTURE X(40).
        10  DELIMTR    PICTURE X.
        10  COUNTS     PICTURE 99 COMP.
01  SHIFTS             VALUE "0E0F".
    05  SHIFT-OUT      PICTURE X.
    05  SHIFT-IN       PICTURE X.
```

UNSTRING ステートメントを次のようにコーディングします。

```
UNSTRING SUBJECT-FIELD  DELIMITED BY SHIFT-OUT
                                  OR SHIFT-IN
INTO RECEIVER (1) DELIMITER IN DELIMTR (1)
                  COUNT     IN COUNTS (1)
INTO RECEIVER (2) DELIMITER IN DELIMTR (2)
                  COUNT     IN COUNTS (2)
INTO RECEIVER (3) DELIMITER IN DELIMTR (3)
                  COUNT     IN COUNTS (3)
INTO RECEIVER (4) DELIMITER IN DELIMTR (4)
                  COUNT     IN COUNTS (4)
ON OVERFLOW PERFORM UNSTRING-OVERFLOW-MESSAGE.
```

この UNSTRING ステートメントは、文字ストリングを英数字部分と DBCS 部分に分割します。文字ストリング内のデータが有効であるとすれば、区切り文字の値がシフトアウトなら対応する受け取りフィールドの内容は英数字データであることを示しており、シフトインなら対応する受け取りフィールドに DBCS データが含まれることを示しています。各受け取りフィールドが文字を受け取ったかどうかは、COUNT データ項目を調べることによって判別できます。次の図は、上記の UNSTRING 操作の結果を示す例です。

```
SUBJECT-FIELD = ABC0EK1K2K30FD0EK4K5K60F
RECEIVER (1) = ABC       DELIMTR (1) = 0E   COUNTS (1) = 3
RECEIVER (2) = K1K2K3    DELIMTR (2) = 0F   COUNTS (2) = 6
RECEIVER (3) = D         DELIMTR (3) = 0E   COUNTS (3) = 1
RECEIVER (4) = K4K5K6    DELIMTR (4) = 0F   COUNTS (4) = 6
SUBJECT-FIELD = 0EK1K2K30FABC0EK40F
RECEIVER (1) = (blanks)  DELIMTR (1) = 0E   COUNTS (1) = 0
RECEIVER (2) = K1K2K3    DELIMTR (2) = 0F   COUNTS (2) = 6
RECEIVER (3) = ABC       DELIMTR (3) = 0E   COUNTS (3) = 3
RECEIVER (4) = K4        DELIMTR (4) = 0F   COUNTS (4) = 2
```

プロシージャー分岐ステートメント

混合リテラルを、STOP ステートメントのオペランドとして使用できます。その場合、対話式ジョブであれば、ワークステーションでリテラルを入力するごとに、システムはそれを表示します。バッチ・ジョブでは、システム・オペレーターのメッセージ待ち行列上で、通常リテラルが表示される位置に、システムによって下線が表示されます。システムはリテラルの内容の編集や検査を行いません。

テーブル処理 - SEARCH ステートメント

形式 1 の SEARCH ステートメント (テーブルの順次検索) を、DBCS データを含むテーブルに対して一度に DBCS 文字の半分ずつ実行できます。

さらに、形式 2 の SEARCH ステートメント (SEARCH ALL) は DBCS テーブルに対して同様に実行できます。テーブルの整列処理は、選択された照合順序に従ってなされます。

注: DBCS データ内のシフト制御文字も、比較の対象になります。

SORT/MERGE

DBCS データ項目を SORT または MERGE ステートメントのキーとして使用できます。ソート操作は、SORT、MERGE、または SPECIAL NAMES 段落で指定された照合順序でデータを並べ替えます。システムは、DBCS および混合キーにシフト制御文字が含まれる場合、それも並べ替えの対象とします。

DBCS 文字を含むレコードを入出力域からソート操作の初期フェーズに転送するには、RELEASE ステートメントを使用します。システムは RELEASE ステートメントの指定された FROM 句を、WRITE ステートメントの指定された FROM 句と同様の方法で実行します。 (686 ページの『WRITE ステートメント』を参照。)

さらに、RETURN ステートメントを使用することによって、DBCS 文字を含むレコードをソートまたはマージの操作の初期フェーズから入出力域に転送できます。シ

ステムは RETURN ステートメントの指定された INTO 句を、READ ステートメントの指定された INTO 句と同様の方法で実行します。 (685 ページの『READ ステートメント』を参照。)

コンパイラー指示ステートメント

COPY ステートメント

COPY ステートメントを使用して、DBCS 文字を含むソース・テキストを COBOL プログラムにコピーすることができます。その場合、コピーブックの名前は英数字データで指定し、それらの名前は「*IBM Rational Development Studio for i: ILE COBOL 解説書*」に記載されている規則に従って指定するようにしてください。

データ記述仕様 (DDS) で定義されたフィールドをコピーするには、形式 2 の COPY ステートメントを使用します。 DBCS (DDS 形式の 35 桁目の値が G)、および混合データ項目 (DDS 形式の 35 桁目の値が O) は、PICTURE X(*n*) 形式で COBOL プログラムにコピーされます。 *PICGGRAPHIC を選択した場合、DBCS データ項目 (形式 G) は、PICTURE G(*n*) 形式でコピーされます。キー・フィールドである場合を除いて、コンパイラー・リストには、これらのフィールドに DBCS 文字が含まれることは示されません。キー・フィールドである場合、システムはキーのコメント・テーブルに O を出力します。

DBCS グラフィック・データ項目は、PICTURE X(*n*) 形式で COBOL プログラムにコピーされます。コンパイラー・リストには、これらのフィールドにグラフィック・データが含まれることが示されます。 DBCS グラフィック・データ・タイプについては 504 ページの『DBCS グラフィック・フィールド』を参照してください。

DDS からコピーしたテキスト・コメントにおいては、関連した DDS フィールドにコメントがある場合、DBCS 文字を使用できます。

COPY ステートメントの REPLACING 句を指定した場合、次の考慮事項に注意してください。

- 疑似テキストには、DBCS および英数字の任意の組み合わせを含めることができます。
- DBCS で構成されるリテラルを使用できます。
- ID は DBCS 文字を含むデータ項目を参照できます。

REPLACE ステートメント

REPLACE ステートメントは COPY ステートメントの REPLACING 句に似ていますが、COPY ライブラリー内のテキストだけでなくソース・プログラム全体を対象とする点が異なります。

REPLACE ステートメントを指定する場合、次の考慮事項に注意してください。

- 疑似テキストには、DBCS および英数字の任意の組み合わせを含めることができます。
- DBCS で構成されるリテラルを使用できます。
- ID は DBCS 文字を含むデータ項目を参照できます。

TITLE ステートメント

DBCS リテラルを TITLE ステートメント内のリテラルとして使用できます。

プログラム間の通信

DBCS または混合文字を含む英数字データ項目を、データ部 (DATA DIVISION) のリンケージ・セクション (LINKAGE SECTION) で指定できます。 DBCS データ項目または DBCS リテラルがプログラムに渡される場合は、受け取り側のリンケージ・セクション項目を DBCS データ項目として定義することができます。

DBCS 文字を 1 つのプログラムから別のプログラムに渡すためには、それらのデータ項目を USING 句で指定します。 混合および DBCS リテラルを渡すには、USING BY CONTENT および USING BY VALUE を使用します。

DBCS 文字を、呼び出し先プログラムのプログラム名として CALL ステートメントで使用することはできません。 CANCEL ステートメントではプログラム名を指定するので、DBCS 文字は使用できません。

FIPS 標識機能

DBCS 文字を使用可能にする COBOL 言語の拡張機能は、IBM 拡張としてコンパイラーに提供されている FIPS (連邦情報処理標準) 標識機能でフラグ付け (識別) されています。

COBOL プログラム・リスト

DBCS 文字は、DBCS 使用可能ソース・ファイルに基づいて、DBCS 可能システムにより作成されたリストに表示できます。

プログラム・リストに表示される DBCS 文字は、ソース・ファイル、COPY ステートメントで作成されたソース・テキスト、または COBOL コンパイラー・メッセージに由来するものです。

DBCS 文字を含むリストは、DBCS データの処理が可能なプリンター・ファイルに出力する必要があります。次の条件の 1 つが満たされれば、DBCS 文字を含むリストは適切に処理されます。

- ソース・ファイルは、CRTSRCPF コマンドの IGCDTA パラメーターを使用して、DBCS データを含められるように定義されていること。この場合、プログラムは、出力印刷装置ファイルの属性の既存値を指定変更します。
- プログラムのコンパイル前に、OVRPRTF コマンドの IGCDTA パラメーターを使用して、出力プリンターに必要な属性がユーザーによって定義されていること。

注: IGCDTA パラメーターが使用可能なのは DBCS システムだけであり、非 DBCS システムではそれを定義または表示することができません。しかし、DBCS システムからコピーすることにより、非 DBCS システム上で DBCS 属性のあるオブジェクトを作成することができます。この場合、非互換になる可能性がないかどうかを検査する必要があります。

コンパイラーは、コンパイラーおよび構文検査メッセージの置換パラメーターとして、ソース・プログラムの文字を使用します。システムは、置換パラメーターの検査または編集を行いません。 DBCS 文字を間違って指定した場合、システムはメッセージの中のいくつかの部分を不正確に出力または表示する可能性があります。

照合順序の関係する組み込み関数

組み込み関数 CHAR および ORD は、文字の順序に依存しています。それらの組み込み関数は、DBCS データ・タイプに関してはサポートされていません (たとえば、1 バイト文字の英数字についてはサポートされています)。それらの関数の結果は、すべて有効な照合順序に基づくものです。 現行の CCSID は、それらの組み込み関数の結果には影響しません。

付録 E. COBOL 定様式ダンプの例

695 ページの図 156 は、COBOL 定様式ダンプの例です。普通、ダンプは、プログラムを実行しようとして問題が発生した場合に使用することができます。

データ部 (DATA DIVISION) のデータ項目をユーザー定義のデータ・タイプとして定義しても、ダンプ内でのデータの表現方法には変わりはありません。 TYPE 文節を使用して定義されたデータ項目の行動は、TYPE 文節を使用せずに定義された場合とまったく同じです。

要求できるダンプには、データ・ダンプと拡張ダンプの 2 種類があります。 695 ページの図 156 にある例は、拡張ダンプです。

データ・ダンプには、次の情報が含まれます。白抜きになったラベルは、定様式ダンプの中でのその情報の位置を示すものです。

A 各変数の名前

B データ・タイプ

C 値

D 16 進値

注: ダンプに表示されるのは、最初の 250 文字だけです。

さらに、**拡張ダンプ**には、次の追加情報が含まれます。白抜きになったラベルは、定様式ダンプの中でのその情報の位置を示すものです。

E 各ファイルの名前

F 各ファイルのシステム名

G 外部 / 内部フラグ

H 最後に試行された入出力操作

I 最新のファイル状況

J 最新の拡張状況

K オープン / クローズ状況

L ブロック化情報

M ブロック化因数

N 入出力フィードバック域情報

O オープン・フィードバック域情報

| **P** 配列エレメントのオフセット (バイト数)

ユーザーが定様式ダンプ内のプログラムの変数の値を見ることができないようにするには、以下のいずれか 1 つを実行します。

- プログラム識別情報を除去して、プログラム内にデバッグ・データが存在しないようにする。

- プログラムを実行するのに十分な権限をユーザーに付与するが、定様式ダンプの実行権限は付与しない。それには、*OBJOPR および *EXECUTE 権限を与えます。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL                CBLGUIDE/SAMPDUMP        ISERIES1   06/02/15 15:10:06        ページ    2
                                ソース
 STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名    変更日付
    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     SAMPDUMP.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600 SOURCE-COMPUTER. IBM-ISERIES.
    6     000700 OBJECT-COMPUTER. IBM-ISERIES.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
    9     001000     SELECT FILE-1 ASSIGN TO DISK-DBSRC.
   11     001100 DATA DIVISION.
   12     001200 FILE SECTION.
   13     001300 FD  FILE-1.
   14     001400 01  RECORD-1.
   15     001500   05 R-TYPE             PIC X(1).
   16     001600   05 R-AREA-CODE        PIC 9(2).
   17     001700     88 R-NORTH-EAST VALUES 15 THROUGH 30.
   18     001800   05 R-SALES-CAT-1      PIC S9(5)V9(2) COMP-3.
   19     001900   05 R-SALES-CAT-2      PIC S9(5)V9(2) COMP-3.
   20     002000   05 FILLER             PIC X(1).
          002100
   21     002200 WORKING-STORAGE SECTION.
   22     002300 01  W-SALES-VALUES.
   23     002400     05 W-CAT-1          PIC S9(8)V9(2).
   24     002500     05 W-CAT-2          PIC S9(8)V9(2).
   25     002600     05 W-TOTAL          PIC S9(8)V9(2).
   24     002500 01 ALPHACODE.
   25     002600   05 STORECODE PIC XX OCCURS 20 TIMES indexed by PMIND.
          002700
   26     002800 01  W-EDIT-VALUES.
   27     002900     05 FILLER           PIC X(8) VALUE "TOTALS: ".
   28     003000     05 W-EDIT-1         PIC Z(7)9.9(2)-.
   29     003100     05 FILLER           PIC X(3) VALUE SPACES.
   30     003200     05 W-EDIT-2         PIC Z(7)9.9(2)-.
   31     003300     05 FILLER           PIC X(3) VALUE SPACES.
   32     003400     05 W-EDIT-TOTAL     PIC Z(7)9.9(2)-.
          003500
   33     003600 01  END-FLAG           PIC X(1) VALUE SPACE.
   34     003700     88 END-OF-INPUT VALUE "Y".
          003800
   35     003900 PROCEDURE DIVISION.
          004000 MAIN-PROGRAM SECTION.
          004100 MAINLINE.
          004200*****************************************************************
          004300* OPEN THE INPUT FILE AND CLEAR TOTALS.                         *
          004400*****************************************************************
   36     004500     OPEN INPUT FILE-1.
   37     004600     MOVE ZEROS TO W-SALES-VALUES.
          004700
          004800*****************************************************************
          004900* READ THE INPUT FILE PROCESSING ONLY THOSE RECORDS FOR THE    *
          005000* NORTH EAST AREA. WHEN END-OF-INPUT REACHED, SET THE FLAG     *
          005100*****************************************************************
   38     005200     PERFORM UNTIL END-OF-INPUT
   39     005300         READ FILE-1
   40     005400             AT END SET END-OF-INPUT TO TRUE
          005500         END-READ
   41     005600         IF R-NORTH-EAST AND NOT END-OF-INPUT THEN
   42     005700             ADD R-SALES-CAT-1 TO W-CAT-1, W-TOTAL
   43     005800             ADD R-SALES-CAT-2 TO W-CAT-2, W-TOTAL
          005900         END-IF
          006000     END-PERFORM.
   44     006100     SET PMIND to 5.
   45     006200     MOVE 'Z1' TO STORECODE(PMIND).
          006100
          006200*****************************************************************
          006300* DISPLAY THE RESULTS AND END THE PROGRAM.                      *
          006400*****************************************************************
   44     006500     MOVE W-CAT-1 TO W-EDIT-1.
   45     006600     MOVE W-CAT-2 TO W-EDIT-2.
   46     006700     MOVE W-TOTAL TO W-EDIT-TOTAL.
   47     006800     DISPLAY W-EDIT-VALUES.
   48     006900     STOP RUN.
                         * * * * *  ソ ー ス 仕 様 の 終 わ り  * * * * *
```

図 156. COBOL 定様式ダンプの生成に使用された COBOL プログラム

```
ライブラリー 'TESTLIB   ' のプログラム 'SAMPDUMP  ' モジュール 'SAMPDUMP  ' ステートメント番号 42 に LNR7200 例外。

ライブラリー 'TESTLIB   ' のプログラム 'SAMPDUMP  ' モジュール 'SAMPDUMP  ' の定様式データ・ダンプ。
名前     属性                          値
DB-FORMAT-NAME  A
         CHAR(10)  B                   "DBSRC     "  C
                                       "C4C2E2D9C34040404040"X  D
END-FLAG
         CHAR(1)                       " "
                                       "40"X
PMIND
         IX(4)                         5
                                       "00000008"X  P
R-AREA-CODE   OF RECORD-1 OF FILE-1
         ZONED(2 0)                     0.
                                       "0000"X
R-SALES-CAT-1   OF RECORD-1 OF FILE-1
         PACKED(7 2)                   00000.00
                                       "00000000"X
R-SALES-CAT-2   OF RECORD-1 OF FILE-1
         PACKED(7 2)                   0000Û.7²
                                       "0000B7A0"X
RETURN-CODE
         BIN(2)                        0000.
                                       "0000"X
STORECODE   OF ALPHACODE
         DIM(1)  (1 20)
STORECODE   OF ALPHACODE
         CHAR(2)
         (1)                           "  "
                                       "4040"X
         ...                           ...
         (5)                           "Z1"
                                       "E9F1"X
         (6)                           "  "
                                       "4040"X
         ...                           ...
W-CAT-1   OF W-SALES-VALUES
         ZONED(10 2)                   00311111.08
                                       "F0F0F3F1F1F1F1F1F0F8"X
W-CAT-2   OF W-SALES-VALUES
         ZONED(10 2)                   00622222.16
                                       "F0F0F6F2F2F2F2F2F1F6"X
W-EDIT-TOTAL   OF W-EDIT-VALUES
         CHAR(12)                      "            "
                                       "40404040404040404040404040"X
W-EDIT-1   OF W-EDIT-VALUES
         CHAR(12)                      "            "
                                       "40404040404040404040404040"X
W-EDIT-2   OF W-EDIT-VALUES
         CHAR(12)                      "            "
                                       "40404040404040404040404040"X
W-TOTAL   OF W-SALES-VALUES
         ZONED(10 2)                   00933333.24
                                       "F0F0F9F3F3F3F3F2F4"X
 E          F
現行の活動ファイル: FILE-1 (DISK-DBSRC)。
ファイル FILE-1 (DISK-DBSRC) と関連した情報。
  ファイルは内部ファイルである。 G
  ファイルに対して試みられた最終入出力操作 : READ。 H
  最終ファイル状況 : '00'. I
  最終拡張ファイル状況 : '    '。 J
  ファイルがオープンされている。 K
  ブロック化が有効となっている。 L
    ブロック化因数 : 17。 M
  入出力フィードバック域。 N
    Number of successful PUT operations: 0.
```

図 157. COBOL 定様式ダンプの例 (1/2)

```
   Number of successful GET operations: 1.
   Number of successful PUTGET operations: 0.
   Number of other successful operations: 0.
   Current data management operation: 1.
   Record format: 'DBSRC     '.
   Device class and type: '  '.
   Program device name: '          '.
   Length of last record: 228.
   Number of records for blocked PUT or GET: 17.
   Length of all data returned: 0.
   Number of blocks successfully read or written: 0.
   Offset: '090'. Value: '00000000000000000001000004800004'.
   Offset: '0A0'. Value: '00000000000000000001000000110000'.
   Offset: '0B0'. Value: '0000'.
オープン・フィードバック域。 0
   Actual file name: 'DBSRC     '.
   Actual library name: 'TESTLIB   '.
   Member name: 'SALES     '.
   File type: 21.
   Open file count: 1.
   Max record length: 0.
   CCSID: 65535.
   Offset: '000'. Value: 'C4C2C4C2E2D9C34040404040D9D4C9E2'.
   Offset: '010'. Value: 'E3D9E840404000000000000000000000'.
   Offset: '020'. Value: '0000000000000000000000000E40000'.
   Offset: '030'. Value: 'E2C1D3C5E24040404040FFFFFFE00000'.
   Offset: '040'. Value: '00000015000000000000000000011C1'.
   Offset: '050'. Value: 'D900D5A5000000000000500000000000'.
   Offset: '060'. Value: '00000000000000000000011000000EF00'.
   Offset: '070'. Value: '0003E000000000000000000000000001'.
   Offset: '080'. Value: '0000000102007300000FFFF0000000000'.
   Offset: '090'. Value: '00010001C4C1E3C1C2C1E2C540400000'.
   Offset: '0A0'. Value: '00000000000000000302000E00450045'.
   Offset: '0B0'. Value: '0045004500450045006F004500450045'.
   Offset: '0C0'. Value: '00450BFD068E0045000D001100000001'.
   Offset: '0D0'. Value: '00000000000000000000000000000000'.
   Offset: '0E0'. Value: '00000000000000000000000000000000'.
   Offset: '0F0'. Value: '00000000000000000000000000000000'.
   Offset: '100'. Value: '00000000000000000000000000000000'.
   オフセット : '110'。値 : '000000000000'。
```

図 157. COBOL 定様式ダンプの例 (2/2)

付録 F. XML 参照資料

この付録は、XML パーサーが特殊レジスター XML-CODE で戻す XML 例外コードを説明します。 また、パーサーが XML 仕様の内のどの整形式制約を検査するかを記述します。 XML パーサーは、IBM の Enterprise COBOL から移植されており、エラー・コードのうちのいくつかは i5/OS サーバー上では適用されない可能性がありますが、表を完全にするために含まれていることに注意してください。

関連リファレンス
『継続可能な XML 例外』
703 ページの『継続不可能な XML 例外』
707 ページの『XML 順応』
「*XML specification*」(www.w3c.org/XML/)

継続可能な XML 例外

以下の表は、 XML パーサーが XML データの処理を継続できる場合にパーサーが特殊レジスター XML-CODE で戻す、 XML イベント EXCEPTION に関連した例外コードを示しています。それは、コードが以下の範囲のいずれかのうちにある場合です。

- 1 ～ 99
- 100,001 ～ 165,535
- 200,001 ～ 265,535

表は、例外を説明し、また例外発生後にパーサーに継続するように要求したときにパーサーが取るアクションを説明しています。これらの説明では、「XML テキスト」という表現は、構文解析している XML 文書が英数字データ項目であるか、国別データ項目であるかによって、それぞれ XML-TEXT か XML-NTEXT を意味します。

表 35. **継続可能な XML 例外**

コード	説明	継続時のパーサーのアクション
1	パーサーは、エレメント・コンテントの外側の空白を走査中に無効文字を検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
2	パーサーは、エレメント・コンテント外で処理命令、エレメント、コメント、または文書タイプ宣言の無効な開始部分を検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
3	パーサーは、重複する属性名を検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。

表 35. **継続可能な XML 例外** *(続き)*

コード	説明	継続時のパーサーのアクション
4	パーサーは、マークアップ文字の「<」を属性値内で検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
5	エレメントの開始タグと終了タグの名前が一致しなかった。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
6	パーサーは、エレメント・コンテント中に無効文字を検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
7	パーサーは、エレメント・コンテント中にエレメント、コメント、処理命令、または CDATA セクションの無効な開始部分を検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
8	パーサーは、エレメント・コンテント中で、一致する CDATA 開始文字シーケンス「<![CDATA[」を持たない CDATA 終了文字シーケンス「]]>」を検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
9	パーサーは、コメント中に無効文字を検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
10	パーサーは、文字シーケンス「--」(ハイフン 2 つ) で、その次に「>」が続かないものをコメント中に検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
11	パーサーは、処理命令のデータ・セグメント中に無効文字を検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
12	処理命令のターゲット名は「xml」(小文字、大文字、または大文字小文字混合) であった。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。

表 35. **継続可能な XML 例外** *(続き)*

コード	説明	継続時のパーサーのアクション
13	パーサーは、16 進の文字参照 (&#xdddd; の形式) 中に無効な桁を検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
14	パーサーは、10 進の文字参照 (&#dddd; の形式) 中に無効な桁を検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
15	XML 宣言のエンコード宣言の値が、小文字または大文字の A から Z で始まらなかった。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
16	文字参照が、有効な XML 文字を参照していなかった。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
17	パーサーは、エンティティー参照名中に無効文字を検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
18	パーサーは、属性値中に無効文字を検出した。	パーサーは、文書の終端に達するか、継続可能でないエラーが発生するまで、エラーの検出を続けます。パーサーは、END-OF-DOCUMENT イベントを除き、それ以降の通常のイベントを通知しません。
50	文書は EBCDIC でエンコードされており、COBOL ソース・メンバーの CCSID はサポートされている EBCDIC の CCSID であるが、文書のエンコード宣言は、認識できるエンコード方式を指定しなかった。	パーサーは、COBOL ソース・メンバーの CCSID に指定されたエンコード方式を使用します。
51	文書は EBCDIC でエンコードされており、文書のエンコード宣言はサポートされている EBCDIC エンコード方式を指定したが、パーサーは COBOL ソース・メンバーの CCSID をサポートしていない。	パーサーは、文書のエンコード宣言に指定されたエンコード方式を使用します。

表 35. **継続可能な XML 例外** *(続き)*

コード	説明	継続時のパーサーのアクション
52	文書は EBCDIC でエンコードされており、COBOL ソース・メンバーの CCSID はサポートされている EBCDIC の CCSID であるが、文書のエンコード宣言は、ASCII エンコード方式を指定した。	パーサーは、COBOL ソース・メンバーの CCSID に指定されたエンコード方式を使用します。
53	文書は EBCDIC でエンコードされており、COBOL ソース・メンバーの CCSID はサポートされている EBCDIC の CCSID であるが、文書のエンコード宣言は、サポートされている Unicode エンコード方式を指定した。	パーサーは、COBOL ソース・メンバーの CCSID に指定されたエンコード方式を使用します。
54	文書は EBCDIC でエンコードされており、COBOL ソース・メンバーの CCSID はサポートされている EBCDIC の CCSID であるが、文書のエンコード宣言は、パーサーのサポートしない Unicode エンコード方式を指定した。	パーサーは、COBOL ソース・メンバーの CCSID に指定されたエンコード方式を使用します。
55	文書は EBCDIC でエンコードされており、COBOL ソース・メンバーの CCSID はサポートされている EBCDIC の CCSID であるが、文書のエンコード宣言は、パーサーのサポートしないエンコード方式を指定した。	パーサーは、COBOL ソース・メンバーの CCSID に指定されたエンコード方式を使用します。
56	文書は ASCII でエンコードされており、COBOL ソース・メンバーの CCSID はサポートされている ASCII の CCSID であるが、文書のエンコード宣言は、認識できるエンコード方式を指定しなかった。	パーサーは、COBOL ソース・メンバーの CCSID に指定されたエンコード方式を使用します。
57	文書は ASCII でエンコードされており、文書のエンコード宣言はサポートされている ASCII エンコード方式を指定したが、パーサーは COBOL ソース・メンバーの CCSID の指定する CCSID をサポートしていない。	パーサーは、文書のエンコード宣言に指定されたエンコード方式を使用します。
58	文書は ASCII でエンコードされており、COBOL ソース・メンバーの CCSID はサポートされている ASCII の CCSID であるが、文書のエンコード宣言は、サポートされている EBCDIC エンコード方式を指定した。	パーサーは、COBOL ソース・メンバーの CCSID に指定されたエンコード方式を使用します。

表 35. **継続可能な XML 例外** *(続き)*

コード	説明	継続時のパーサーのアクション
59	文書は ASCII でエンコードされており、COBOL ソース・メンバーの CCSID はサポートされている ASCII の CCSID であるが、文書のエンコード宣言は、サポートされている Unicode エンコード方式を指定した。	パーサーは、COBOL ソース・メンバーの CCSID に指定されたエンコード方式を使用します。
60	文書は ASCII でエンコードされており、COBOL ソース・メンバーの CCSID はサポートされている ASCII の CCSID であるが、文書のエンコード宣言は、パーサーのサポートしない Unicode エンコード方式を指定した。	パーサーは、COBOL ソース・メンバーの CCSID に指定されたエンコード方式を使用します。
61	文書は ASCII でエンコードされており、COBOL ソース・メンバーの CCSID はサポートされている ASCII の CCSID であるが、文書のエンコード宣言は、パーサーのサポートしないエンコード方式を指定した。	パーサーは、COBOL ソース・メンバーの CCSID に指定されたエンコード方式を使用します。
100,001 ～ 165,535	文書は EBCDIC でエンコードされており、COBOL ソース・メンバーの CCSID および文書のエンコード宣言の CCSID が指定するエンコード方式は、両方ともサポートされている EBCDIC の CCSID であるが、同じではない。 XML-CODE は、エンコード宣言の CCSID に 100,000 を足したものを含みます。	EXCEPTION イベントから戻る前に XML-CODE を 0 に設定すると、パーサーは、COBOL ソース・メンバーの CCSID に指定されたエンコード方式を使用します。 XML-CODE を、文書のエンコード宣言の CCSID に設定すると (100,000 を減算することにより)、パーサーはそのエンコード方式を使用します。
200,001 ～ 265,535	文書は ASCII でエンコードされており、COBOL ソース・メンバーおよび文書のエンコード宣言の CCSID が指定するエンコードは、両方ともサポートされている ASCII の CCSID であるが、同じではない。 XML-CODE は、エンコード宣言の CCSID に 200,000 を足したものを含みます。	EXCEPTION イベントから戻る前に XML-CODE を 0 に設定すると、パーサーは、COBOL ソース・メンバーの CCSID に指定されたエンコード方式を使用します。 XML-CODE を、文書のエンコード宣言の CCSID に設定すると (200,000 を減算することにより)、パーサーはそのエンコード方式を使用します。

関連作業

330 ページの『XML 文書中のエラーの処理』

継続不可能な XML 例外

これらの XML 例外では、XML-CODE を 0 に設定し、例外の処理後にパーサーに制御を戻したとしても、パーサーからそれ以降のイベントが戻されることはありません。 NOT ON EXCEPTION 句で指定されたステートメントに、もしくは NOT ON EXCEPTION 句をコーディングしていない場合は Parse ステートメントの終わりに、制御が移ります。

表 36. **継続不可能な XML 例外**

コード	説明
100	パーサーは、XML 宣言の開始部を走査中に、文書の終端に達した。
101	パーサーは、XML 宣言の終端を探している途中に、文書の終端に達した。
102	パーサーは、ルート・エレメントを探している途中に、文書の終端に達した。
103	パーサーは、XML 宣言のバージョン情報を探している途中に、文書の終端に達した。
104	パーサーは、XML 宣言のバージョン情報の値を探している途中に、文書の終端に達した。
106	パーサーは、XML 宣言のエンコード宣言の値を探している途中に、文書の終端に達した。
108	パーサーは、XML 宣言のスタンドアロン宣言の値を探している途中に、文書の終端に達した。
109	パーサーは、属性名を走査中に、文書の終端に達した。
110	パーサーは、属性の値を走査中に、文書の終端に達した。
111	パーサーは、属性値の中の文字参照またはエンティティー参照を走査中に、文書の終端に達した。
112	パーサーは、空のエレメント・タグを走査中に、文書の終端に達した。
113	パーサーは、ルート・エレメント名を走査中に、文書の終端に達した。
114	パーサーは、エレメント名を走査中に、文書の終端に達した。
115	パーサーは、エレメント・コンテント中の文字データを走査中に、文書の終端に達した。
116	パーサーは、エレメント・コンテント中の処理命令を走査中に、文書の終端に達した。
117	パーサーは、エレメント・コンテント中のコメントもしくは CDATA セクションを走査中に、文書の終端に達した。
118	パーサーは、エレメント・コンテント中のコメントを走査中に、文書の終端に達した。
119	パーサーは、エレメント・コンテント中の CDATA セクションを走査中に、文書の終端に達した。
120	パーサーは、エレメント・コンテント中の文字参照またはエンティティー参照を走査中に、文書の終端に達した。
121	パーサーは、ルート・エレメントの閉じ部の後を走査中に、文書の終端に達した。
122	パーサーは、無効な文書タイプ宣言の開始部である可能性のある個所を検出した。
123	パーサーは、2 番目の文書タイプ宣言を検出した。
124	ルート・エレメント名の先頭文字は、文字、「_」、もしくは「:」ではなかった。
125	エレメントの最初の属性名の先頭文字は、文字、「_」、もしくは「:」ではなかった。
126	パーサーは、エレメント名中またはエレメント名後に無効文字を検出した。
127	パーサーは、属性名の次に、「=」以外の文字を検出した。
128	パーサーは、無効な属性値区切り文字を検出した。
130	属性名の先頭文字は、文字、「_」、もしくは「:」ではなかった。

表 36. **継続不可能な XML 例外** *(続き)*

コード	説明
131	パーサーは、属性名中または属性名後に無効文字を検出した。
132	空のエレメント・タグが、「/」に続く「>」で終了しなかった。
133	エレメント終了タグ名の先頭文字は、文字、「_」、もしくは「:」ではなかった。
134	エレメント終了タグ名が、「>」で終了していなかった。
135	エレメント名の先頭文字は、文字、「_」、もしくは「:」ではなかった。
136	パーサーは、エレメント・コンテント中にコメントまたは CDATA セクションの無効な開始部分を検出した。
137	パーサーは、コメントの無効な開始部分を検出した。
138	処理命令のターゲット名の先頭文字は、文字、「_」、もしくは「:」ではなかった。
139	パーサーは、処理命令ターゲット名の中または次に無効文字を検出した。
140	処理命令が、終了文字シーケンスの「?>」で終了しなかった。
141	パーサーは、文字参照またはエンティティー参照の「&」に続く無効文字を検出した。
142	XML 宣言にバージョン情報が存在しなかった。
143	XML 宣言の「version」の次に、「=」が続かなかった。
144	XML 宣言のバージョン宣言の値が欠落しているか、正しく区切り文字で区切られていない。
145	XML 宣言のバージョン情報の値が正しくない文字を指定しているか、開始と終了の区切り文字が一致しなかった。
146	パーサーは、XML 宣言のバージョン情報の値の終了区切り文字の次に、無効文字を検出した。
147	パーサーは、XML 宣言で、オプションのエンコード宣言の代わりに無効な属性を検出した。
148	XML 宣言の「encoding」の次に「=」が続かなかった。
149	XML 宣言のエンコード宣言の値が欠落しているか、正しく区切り文字で区切られていない。
150	XML 宣言のエンコード宣言の値が正しくない文字を指定しているか、開始と終了の区切り文字が一致しなかった。
151	パーサーは、XML 宣言のエンコード宣言の値の終了区切り文字の次に、無効文字を検出した。
152	パーサーは、XML 宣言で、オプションのスタンドアロン宣言の代わりに無効な属性を検出した。
153	XML 宣言の「standalone」の次に「=」が続かなかった。
154	XML 宣言のスタンドアロン宣言の値が欠落しているか、正しく区切り文字で区切られていない。
155	スタンドアロン宣言の値は、「yes」または「no」のどちらか 1 つだけではなかった。
156	XML 宣言のスタンドアロン宣言の値が正しくない文字を指定しているか、開始と終了の区切り文字が一致しなかった。
157	パーサーは、XML 宣言のスタンドアロン宣言の値の終了区切り文字の次に、無効文字を検出した。

表 36. **継続不可能な XML 例外** *(続き)*

コード	説明
158	XML 宣言が、正しい終了文字シーケンスの「?>」で終了しなかったか、無効な属性を含んでいた。
159	パーサーは、ルート・エレメントの終端の後に文書タイプ宣言の開始部を検出した。
160	パーサーは、ルート・エレメントの終端の後にエレメントの開始部を検出した。
300	文書は EBCDIC でエンコードされているが、 COBOL ソース・メンバーの CCSID はサポートされている ASCII CCSID である。
301	文書は EBCDIC でエンコードされているが、 COBOL ソース・メンバーの CCSID は Unicode である。
302	文書は EBCDIC でエンコードされているが、 COBOL ソース・メンバーの CCSID はサポートされていない CCSID である。
303	文書は EBCDIC でエンコードされており、COBOL ソース・メンバーの CCSID は サポートされておらず、文書のエンコード宣言は空であるか、エンコード方式のサポートされていない英字の別名を含んでいた。
304	文書は EBCDIC でエンコードされているが、 COBOL ソース・メンバーの CCSID はサポートされておらず、文書はエンコード宣言を含まなかった。
305	文書は EBCDIC でエンコードされているが、 COBOL ソース・メンバーの CCSID はサポートされておらず、文書のエンコード宣言はサポートされている EBCDIC エンコード方式を指定しなかった。
306	文書は ASCII でエンコードされているが、 COBOL ソース・メンバーの CCSID は、サポートされている EBCDIC CCSID である。
307	文書は ASCII でエンコードされているが、 COBOL ソース・メンバーの CCSID は Unicode である。
308	文書は ASCII でエンコードされているが、 COBOL ソース・メンバーの CCSID はサポートされておらず、文書はエンコード宣言を含まなかった。
309	COBOL ソース・メンバーの CCSID はサポートされている ASCII の CCSID であるが、文書は Unicode でエンコードされていた。
310	COBOL ソース・メンバーの CCSID はサポートされている EBCDIC の CCSID を指定したが、文書は Unicode でエンコードされていた。
311	COBOL ソース・メンバーの CCSID はサポートされていない CCSID を指定し、文書は Unicode でエンコードされていた。
312	文書は ASCII でエンコードされているが、COBOL ソース・メンバーの CCSID は サポートされておらず、文書のエンコード宣言は空であるか、エンコード方式のサポートされていない英字の別名を含んでいた。
313	文書は ASCII でエンコードされているが、 COBOL ソース・メンバーの CCSID はサポートされておらず、文書はエンコード宣言を含まなかった。
314	文書は ASCII でエンコードされているが、 COBOL ソース・メンバーの CCSID はサポートされておらず、文書のエンコード宣言はサポートされている ASCII エンコード方式を指定しなかった。
315	文書は、UTF-16 リトル・エンディアンでエンコードされているが、これはこのプラットフォームではパーサーでサポートしていない。
316	文書は、UCS-4 でエンコードされているが、これはパーサーでサポートしていない。

表 36. **継続不可能な XML 例外** *(続き)*

コード	説明
317	パーサーは、文書のエンコード方式を判別できない。文書は、損傷されている可能性があります。
318	文書は、UTF-8 でエンコードされているが、これはパーサーでサポートしていない。
319	文書は、UTF-16 ビッグ・エンディアンでエンコードされているが、これはこのプラットフォームではパーサーでサポートしていない。
500 ～ 999	内部エラー。サービス技術員に報告してください。

関連作業

330 ページの『XML 文書中のエラーの処理』

XML 順応

ILE COBOL に含まれた XML パーサーは、 XML 仕様の定義に従った順応 XML プロセッサーではありません。パーサーは、構文解析する XML 文書を妥当性検査しません。パーサーは、多くの整形式エラーは検査しますが、 XML 非妥当性検査プロセッサーに必要なすべてのアクションを行うわけではありません。

特に、内部の文書タイプ定義 (DTD 内部サブセット) を処理しません。そのため、デフォルトの属性値を提供せず、属性値を正規化せず、また事前定義エンティティー以外の内部エンティティーの置換テキストを含みません。その代わりに、文書タイプ宣言全体を DOCUMENT-TYPE-DESCRIPTOR XML イベントの XML-TEXT または XML-NTEXT の中身として渡し、それにより、必要ならばアプリケーションがそれらのアクションを行うことができます。

パーサーはオプションとして、いくつかのエラー発生後も XML 文書の処理を継続することを許可します。その目的は、XML 文書および処理プログラムのデバッグを容易にするためです。

XML 仕様の定義を要約すると、テキスト・オブジェクトは、以下の条件を満たしているなら整形式 XML 文書です。

- 全体が、XML 文書の文法に準拠している。
- XML 仕様で挙げられているすべての明示的な整形式制約を守っている。
- 文書内で直接または間接的に参照されている、各構文解析対象エンティティー (テキストの一部) は、整形式である。

COBOL XML パーサーは、文書タイプ宣言を除いて、文書が XML の文法に準拠していることを検査します。文書タイプ宣言は、未検査のまま、その全体がアプリケーションに渡されます。

以下の内容は、XML 仕様からの注釈です。 W3C は、元の URL (www.w3.org/TR/REC-xml) にないコンテンツに対しては責任を持ちません。注釈はすべて、標準を定めるものではなく、*イタリック* で示されています。

XML 仕様は、12 の明示的な整形式制約も含みます。 COBOL XML パーサーが部分的にまたは完全に検査する制約は、**太字**で示されています。

1. 内部サブセットのパラメーター・エンティティー (PE)
   「内部 DTD サブセットでは、パラメーター・エンティティー参照は、マークアップ宣言を配置できる個所のみに配置でき、マークアップ宣言の中には配置できない。 (これは、外部パラメーター・エンティティー内に位置する参照や、外部サブセットへの参照にはあてはまりません。) ”

   パーサーは、内部 DTD サブセットを処理しないので、この制約を強制しません。
2. 外部サブセット
   「外部サブセットが存在する場合は、生成規則 extSubset に適合しなければならない。」

   パーサーは、外部サブセットを処理しないので、この制約を強制しません。
3. 宣言間のパラメーター・エンティティー
   「DeclSep 中のパラメーター・エンティティー参照の置換テキストは、プロダクション extSubsetDecl に一致しなければならない。」

   パーサーは、内部 DTD サブセットを処理しないので、この制約を強制しません。
4. **エレメント・タイプ一致**
   「エレメントの終了タグ中の Name は、エレメントの開始タグ中のエレメント・タイプに合致しなければならない。」

   パーサーは、この制約を強制します。
5. **固有な属性指定**
   「いかなる属性名も、同じ開始タグまたは空エレメント・タグ中に 2 回以上現れてはならない。」

   パーサーは、各エレメントにつき、10 個の属性名まで固有であることを検査することで、この制約を部分的にサポートします。この限度を越えた属性名は、アプリケーションで検査することができます。
6. 外部エンティティー参照なし
   「属性値は、直接または間接の、外部エンティティーへのエンティティー参照を含むことができない。」

   パーサーは、この制約を強制しません。

7. 属性値中に「<」なし
   「属性値の中で直接的または間接的に参照されるどのエンティティーの置換テキストも、「<」を含んではならない。」

   パーサーは、この制約を強制しません。

8. **有効文字**
   「文字参照を使用して参照される文字は、生成規則 Char に適合しなければならない。」

   パーサーは、この制約を強制します。

9. エンティティーが宣言されている
   「DTD がない文書、パラメーター・エンティティー参照を含まない内部 DTD サブセットをのみがある文書、または standalone='yes' がある文書では、外部サブセット中またはパラメーター・エンティティー中でないエンティティー参照において、エンティティー参照で与えられた Name は外部サブセットまたはパラメーター・エンティティー中でないエンティティー宣言の Name と一致しなければならない。ただし、整形式文書では、エンティティー amp、lt、gt、apos、および quot はいずれも宣言する必要はありません。一般エンティティーの宣言は、属性リスト宣言中のデフォルト値の中にあるそのエンティティーに対する参照より前に位置する必要があります。

   エンティティーが外部サブセットまたは外部パラメーター・エンティティー内で宣言されている場合は、非妥当性検査プロセッサーはそれらの宣言を読み取り、処理する必要はないということに注意してください。そのような文書では、エンティティーが宣言されていなくてはならないという規則は、standalone='yes' である場合のみの整形式制約です。」

   パーサーは、この制約を強制しません。

10. 構文解析対象エンティティー
    「エンティティー参照は、構文解析対象外エンティティーの名前を含んではならない。構文解析対象外エンティティーは、タイプが ENTITY または ENTITIES として宣言された属性値の中でのみ参照することができます。」

    パーサーは、この制約を強制しません。

11. 再帰なし
    「構文解析対象エンティティーは、直接的にも間接的にも、自身に対する再帰的参照を含んではならない。」

    パーサーは、この制約を強制しません。

12. DTD 中である
    「パラメーター・エンティティー参照は、DTD 中にしか現れてはならない。」

    このエラーが発生することが不可能なので、パーサーはこの制約を強制しません。

上記の資料は、XML 仕様に含まれる注釈です。 W3C は、元の URL (www.w3.org/TR/REC-xml) にないコンテンツに対しては責任を持ちません。これらの注釈はすべて、標準を定めるものではありません。この資料は、W3C メンバーおよび関係者によって検討され、ディレクターによって W3C 推奨として承認されました。これは信頼できる資料であり、参照資料としての利用および、標準を定めるリファレンスとして他の資料からの引用が可能です。仕様の標準版は、W3C サイトにある英語版であり、翻訳文書は翻訳による誤りを含む可能性があります。

関連概念

311 ページの『COBOL の XML パーサー』

関連リファレンス

「*XML specification*」(www.w3c.org/XML/)

『2.8 Prolog and document type declaration』(www.w3.org/TR/REC-xml#sec-prolog-dtd の「*XML specification*」)

XML 生成例外

以下の表に、XML の生成中に発生する可能性のある例外コードを示します。例外コードは、特殊レジスター XML-CODE で戻されます。これらの例外のいずれかが発生すると、制御は ON EXCEPTION 句内のステートメントか、(ON EXCEPTION 句をコーディングしていない場合は) XML GENERATE ステートメントに渡されます。

表 37.

コード	説明
400	受信側が小さすぎて、生成された XML 文書を含めることができませんでした。 COUNT IN データ項目が指定されている場合は、実際に生成された文字位置のカウントがこのデータ項目に含まれます。
401	Unicode への変換時に XML エレメント名としては無効な文字がデータ名に含まれていました。
411	PROCESS ステートメントの CCSID オプション d で指定された CCSID が、サポートされている 1 バイト CCSID ではありません。
450	XML ファイルは既に存在しています。
451	間違った CCSID が既存の XML ファイルに含まれています。
600 から 699	内部エラー。サービス技術員に報告してください。
650	内部エラー。ストリーム・ファイルの OPEN、WRITE、CLOSE、または REPLACE が失敗しました。サービス技術員に報告してください。
3000 から 3600	ストリーム・ファイルで内部エラーが発生しました。サービス技術員に報告してください。

関連作業

346 ページの『XML 出力の生成中のエラーの処理』

付録 G. OPM COBOL/400 および ILE COBOL の間のマイグレーションおよび互換性に関する考慮事項

この付録では、ILE COBOL と OPM COBOL/400 の相違点について説明します。

既存の OPM COBOL/400 プログラムおよびアプリケーションを ILE COBOL に移す場合、次に示す OPM COBOL/400 コンパイラーと ILE COBOL との相違点について理解しておくことが必要です。場合によっては、プログラムの変更が必要になります。

マイグレーションの方法

既存の OPM COBOL/400 プログラムおよびアプリケーション・プログラムを ILE COBOL にマイグレーションするには、次のマイグレーション方法が推奨されます。

- プログラムを 1 つずつマイグレーションする代わりに、アプリケーション全体 (または COBOL 実行単位) を、初期設定済み ILE 環境に同時にマイグレーションします。
- COBOL 実行単位を ILE 活動化グループにマッピングします。たとえば、複数の COBOL プログラムを含む COBOL 実行単位の場合、COBOL 実行単位のセマンティクスを保つために次のいずれか 1 つを行うことができます。
  - COBOL プログラムのすべてを CRTBNDCBL コマンドを使用して作成する。この場合、プログラムのすべては QILE 活動化グループで実行されます。
  - COBOL プログラムのすべてを CRTCBLMOD コマンド、およびそれに続いて ACTGRP(anyname) を指定した CRTPGM を使用して作成する。この場合、プログラムのすべては「anyname」で指定した名前の活動化グループで実行されます。
  - 最初の COBOL プログラムを、CRTPGM コマンドによって ACTGRP(*NEW) を指定して作成し、アプリケーション・プログラム内の残りのプログラムを、ACTGRP(*CALLER) を指定して作成する。この場合、プログラムのすべては、最初の COBOL プログラムの *NEW 活動化グループで実行します。
- CRTPGM コマンドに ACTGRP(*CALLER) オプションを指定して作成したプログラムの呼び出し側が OPM プログラムではないようにしてください。

  注: OPM COBOL/400 プログラムと ILE COBOL プログラムを同じ実行単位に混在させることは、お勧めできません。
- 複数の異なる有効範囲オプションが可能なシステム機能には、特に注意してください。たとえば、次のシステム機能では、ILE 活動化グループで使用される場合に、デフォルトの有効範囲が *ACTGRPDFN (活動化グループ・レベル) に変更されます。 OPM プログラムで使用される場合には *CALLLVL (呼び出しレベル) など他のデフォルトがあります。
  - OPNDBF および OPNQRYF の場合、アプリケーション・プログラムに応じて OPNSCOPE を変更する必要が生じることがあります。たとえば、アプリケー

ション・プログラムが異なる複数の活動化グループで実行されていて、ファイルを共有する必要がある場合、有効範囲を *JOB に変更する必要があります。
  - 指定変更。
  - コミットメント制御。
- RCLRSRC は ILE 活動化グループに影響を与えません。 ILE 活動化グループをクリーンアップするには、代わりに RCLACTGRP を使用します。

互換性に関する考慮事項

ここでは、ILE COBOL と OPM COBOL/400 の互換性に関する考慮事項について説明します。

一般的な考慮事項

領域検査

ILE COBOL において、領域検査が活動状態になるのは行の最初のトークンに対してだけです。それ以降のトークンに対しては、適切な領域に存在するかどうかを判別する検査は行われません。

OPM COBOL/400 コンパイラーは、すべてのトークンを検査します。

コンパイラー・リストのデータ部マップ・セクションの属性フィールド

ILE COBOL では、構文が検査されるだけの属性 (たとえば、SAME SORT AREA、SAME SORT-MERGE AREA、SAME AREA、LABEL 情報) は、コンパイラー・リストのデータ部マップ・セクションには報告されません。

ILE COBOL では、条件名はコンパイラー・リストのデータ部マップ・セクションに示されません。

OPM COBOL/400 では条件名が示されますが、属性情報は指定しません。

MIXED、COMMUNICATIONS、および BSC ファイル

MIXED、COMMUNICATIONS、および BSC ファイルは、ILE COBOL ではサポートされていません。 System/38 環境で有効なこれらのファイル・タイプは、ILE COBOL コンパイラーではコンパイル時 (COPY DDS の際) または実行時にサポートされていません。

予約語

ILE COBOL は、OPM COBOL/400 で現在サポートされていないいくつかの予約語をサポートしています。たとえば、SORT-RETURN および RETURN-CODE は特殊レジスターです。 SORT-RETURN または RETURN-CODE が OPM COBOL/400 プログラム内に現れると、これらが COBOL の他の機能で使用される予約語であることを示す重大度 10 のメッセージが生成されます。

ILE COBOL はこれらの語を予約語として認識します。そして類似の状況下で、ILE COBOL はユーザー定義語が必要であるところに予約語が存在していることを示す重大度 30 のメッセージを出します。

SAA CPI データ構造のソース・ファイル

ILE COBOL では、SAA CPI データ構造のソース・ファイルは、ライブラリー QSYSINC のファイル QCBLLESRC の中に存在します。

OPM COBOL/400 では、SAA CPI データ構造のソース・ファイルは、ライブラリー QLBL および QLBLP のファイル QILBINC の中に存在します。

CL コマンド

CRTCBLPGM コマンドに代わる CRTCBLMOD および CRTBNDCBL コマンド

OPM COBOL/400 コンパイラーは CRTCBLPGM CL コマンドによって呼び出されます。 CRTCBLPGM CL コマンドは *PGM オブジェクトを作成します。

ILE COBOL コンパイラーは、CRTCBLMOD または CRTBNDCBL の CL コマンドによって呼び出されます。 CRTCBLMOD CL コマンドは *MODULE オブジェクトを作成し、CRTBNDCBL CL コマンドは *PGM オブジェクトを作成します。

次の CRTCBLPGM のパラメーターおよびオプション (およびそれらに関連した PROCESS ステートメントのオプション) は、CRTCBLMOD および CRTBNDCBL にはありません。

- GENOPT パラメーター (残りの GENOPT 明細は、すべて OPTION 明細に移されました)
- PRTFILE パラメーター
- SAAFLAG パラメーター
- DUMP パラメーター
- ITDUMP パラメーター
- OPTION パラメーターの NOSRCDBG/SRCDBG オプション
- OPTION パラメーターの NOLSTDBG/LSTDBG オプション
- OPTION パラメーターの PRINT/NOPRINT オプション
- GENOPT パラメーターの LIST/NOLIST オプション
- GENOPT パラメーターの NOPATCH/PATCH オプション
- GENOPT パラメーターの NODUMP/DUMP オプション
- GENOPT パラメーターの NOATR/ATR オプション
- GENOPT パラメーターの NOOPTIMIZE/OPTIMIZE オプション
- GENOPT パラメーターの STDERR/NOSTDERR オプション
- GENOPT パラメーターの NOEXTACCDSP/EXTACCDSP オプション
- GENOPT パラメーターの FS21DUPKY/NOFS21DUPKY オプション

次のパラメーターおよびオプションが変更されました。

- SRCFILE パラメーターでは、デフォルトのソース・ファイル名は QCBLLESRC です。
- CVTOPT パラメーターでは、CRTCBLPGM の GRAPHIC/NOGRAPHIC キーワードは、CRTCBLMOD および CRTBNDCBL の PICXGRAPHIC/NOPICXGRAPHIC に変更されました。
- MSGLMT パラメーターのデフォルトの最大重大度レベルは 30 です。
- GENLVL パラメーターのデフォルトの重大度レベルは 30 です。

- FLAGSTD パラメーターでは、CRTCBLPGM にあった NOSEG/SEG1/SEG2 および NODEB/DEB1/DEB2 オプションは、CRTCBLMOD または CRTBNDCBL にはありません。
- OPTION パラメーターでは、NOUNREF/UNREF オプションのデフォルトは NOUNREF に変更されました。
- OPTION パラメーターでは、NOSECLVL/SECLVL オプションのデフォルトは NOSECLVL に変更されました。

次のパラメーターおよびオプションは、CRTCBLMOD および CRTBNDCBL コマンドで新しく使用されるようになったものです。

- MODULE パラメーター (CRTCBLMOD パラメーターのみ)
- PGM パラメーター (CRTBNDCBL パラメーターのみ)
- OUTPUT パラメーター
- DBGVIEW パラメーター
- OPTIMIZE パラメーター
- LINKLIT パラメーター
- SIMPLEPGM パラメーター (CRTBNDCBL のみ)
- OPTION パラメーターの MONOPRC/NOMONOPRC オプション
- OPTION パラメーターの NOSTDTRUNC/STDTRUNC オプション
- OPTION パラメーターの NOIMBEDERR/IMBEDERR オプション
- OPTION パラメーターの NOCHGPOSSGN/CHGPOSSGN オプション
- OPTION パラメーターの NOEVENTF/EVENTF オプション
- OPTION パラメーターの MONOPIC/NOMONOPIC オプション
- CVTOPT パラメーターの NOPICGGRAPHIC/PICGGRAPHIC オプション
- CVTOPT パラメーターの NOPICNGRAPHIC/PICNGRAPHIC オプション
- CVTOPT パラメーターの NOFLOAT/FLOAT オプション
- CVTOPT パラメーターの NODATE/DATE オプション
- CVTOPT パラメーターの NOTIME/TIME オプション
- CVTOPT パラメーターの NOTIMESTAMP/TIMESTAMP オプション
- CVTOPT パラメーターの NOCVTTODATE/CVTTODATE オプション
- ENBPFRCOL パラメーター
- PRFDTA パラメーター
- CCSID パラメーター
- ARITHMETIC パラメーター
- NTLPADCHAR パラメーター
- LICOPT パラメーター
- STGMDL パラメーター
- DBGENCKEY パラメーター
- BNDDIR パラメーター (CRTBNDCBL のみ)
- ACTGRP パラメーター (CRTBNDCBL のみ)

パラメーターおよびオプションに対するすべての削除、変更、および追加は、PROCESS ステートメントのオプションに対する関連した変更にも反映されます。

NOGRAPHIC PROCESS ステートメントのオプションは、PROCESS ステートメントの GRAPHIC オプションのデフォルト値として、ILE COBOL に追加されています。

以下の OPM COBOL/400 の PROCESS ステートメントのオプションは、ILE COBOL には存在しません。

- FS9MTO0M/NOFS9MTO0M
- FS9ATO0A/NOFS9ATO0A

コード化文字セット ID (CCSID)

ILE COBOL では、コンパイルでのソース・メンバーの CCSID の標準化が、1 次ソース・ファイルの CCSID に対して行われます。 OPM COBOL/400 の場合は、コンパイル時ジョブの CCSID に対して行われます。

デフォルトのソース・メンバー・タイプ

ILE COBOL では、デフォルトのソース・メンバー・タイプは CBLLE です。 OPM COBOL/400 では、デフォルトのソース・メンバー・タイプは CBL です。

エラー・メッセージ

ILE COBOL では、コンパイル時のエラー・メッセージに接頭部 LNC が付きます。また、メッセージ・メンバーによっては、必ずしも OPM COBOL/400 の場合と同じではありません。

GENLVL パラメーター

ILE COBOL は、GENLVL に指定された重大度より大きいか等しい 重大度レベルのエラーが生じた場合、コードを生成しません。

OPM COBOL/400 は、GENLVL に指定された重大度より大きい 重大度レベルのエラーが生じた場合、コードを生成しません。

SAA 標識機能

SAA 標識機能は ILE COBOL ではサポートされていません。

STRCBLDBG および ENDCBLDBG CL コマンド

STRCBLDBG コマンドと ENDCBLDBG コマンドは、ILE COBOL ではサポートされていません。

コンパイラー指示ステートメント

COPY ステートメント

可変長フィールドの後のコメント: OPM COBOL/400 では、データ・タイプ G および VARLEN の DDS ソースでは、次のものが生成されます。

```
06    FILLER      PIC X(10)
      (Variable length field)
```

ILE COBOL では、可変長フィールドのコメントの後に、より詳細なコメントが追加されます。

```
06    FILLER      PIC X(10)
      (Variable length field)
      (Graphic field)
```

デフォルトのソース・ファイル名: ILE COBOL では、ソース・ファイル・メンバーをコンパイルする場合は、デフォルトのソース・ファイル名は QCBLLESRC で

す。ストリーム・ファイルをコンパイルする場合は、そのストリーム・ファイルを指定する必要があります。ILE COBOL では、ソース・ファイル・メンバーをコンパイルする場合は、ソース修飾子のない COPY ステートメントは、QCBLLESRC を使用します。デフォルトのファイル名が使用されていて、ソース・メンバーがファイル QCBLLESRC に存在しない場合、さらにファイル QLBLSRC も検索されます。ストリーム・ファイルをコンパイルする場合は、コンパイラーは、コピーブックを解決するために、異なる検索順序に従います。詳細については、「*ILE COBOL 解説書*」を参照してください。

OPM COBOL/400 では、デフォルトのソース・ファイル名は QLBLSRC です。

PROCESS ステートメント

***CBL/*CONTROL ステートメント:** PROCESS ステートメントに *CONTROL が存在する場合、それはディレクティブではなく無効な PROCESS オプションとして処理されます。 *CBL/*CONTROL ディレクティブは、1 つの行に存在する唯一のステートメントであることが必要です。

INTERMEDIATE および MINIMUM オプション (FIPS 標識機能): ILE COBOL では、FIPS 標識機能が CRTCBLMOD または CRTBNDCBL コマンドで要求されておらず、しかも COPY ステートメントが PROCESS ステートメント内に存在する場合、INTERMEDIATE または MINIMUM が COPY ステートメントの後に指定されていると、FIPS 標識機能はコピー・メンバーに対して実行されません。しかし、INTERMEDIATE または MINIMUM が COPY ステートメントの前に指定されているときには、FIPS 標識機能はコピー・メンバーに対して実行されます。

OPM COBOL/400 では、INTERMEDIATE または MINIMUM が COPY STATEMENT の前または後に指定されていてもいなくても、FIPS 標識機能はコピー・メンバーに対して実行されます。

NOSOURCE オプション: OPM COBOL/400 では、NOSOURCE オプションが PROCESS ステートメントに指定されている場合、有効なオプション値がコンパイラー・リストに印刷されます。

ILE COBOL では、NOSOURCE オプションが PROCESS ステートメントに指定されている場合、有効なオプション値はコンパイラー・リストに印刷されません。

USE FOR DEBUGGING

OPM COBOL/400 では、WITH DEBUGGING MODE が指定されている場合、USE FOR DEBUGGING が使用可能です。

ILE COBOL は USE FOR DEBUGGING をサポートしていません。次のセクションの最初または DECLARATIVES の終わりまでのテキストは、コメントとして処理されます。重大度 0 のエラー・メッセージ、および重大度 20 のエラー・メッセージが出されます。

環境部 (ENVIRONMENT DIVISION)

DATA DIVISION と ENVIRONMENT DIVISION の順序

OPM COBOL/400 では、DATA DIVISION と ENVIRONMENT DIVISION の順序が混合していてもあまり問題はありません。 OPM COBOL/400 では、文節、句、セクション、および部が適切な順序になっていないことが検出されると、重大度 10 および重大度 20 のメッセージが出ます。

ILE COBOL では、DATA DIVISION と ENVIRONMENT DIVISION との順序が混合していてはなりません。 ILE COBOL では、文節、句、セクション、および部が適切な順序になっていないことが検出されると、重大度 30 のメッセージが出ます。

FILE-CONTROL および I-O-CONTROL 段落

FILE-CONTROL 項目または I-O-CONTROL 項目に重複した文節が存在していて、使用できる文節が 1 つだけである場合、OPM COBOL/400 は最後に指定された文節を使用します。

同じ状況下で、ILE COBOL は最初に指定された文節を使用します。

SELECT 文節

OPM COBOL/400 コンパイラーでは、指定された属性に一貫性があれば、1 つのファイル名を参照する複数の SELECT 文節を使用できます。エラー・メッセージが出されない場合もあります。それ以外の場合、重大度 10 または重大度 20 のメッセージが出されます。指定された属性に一貫性がない場合、重大度 30 のメッセージが出されます。

ILE COBOL コンパイラーでは、1 つのファイル名を参照する複数の SELECT 文節に対して、必ず重大度 30 のメッセージが出されます。

データ部 (DATA DIVISION)

DATA DIVISION と ENVIRONMENT DIVISION の順序

『環境部 (ENVIRONMENT DIVISION)』の『DATA DIVISION と ENVIRONMENT DIVISION の順序』を参照してください。

FD または SD 項目

FD 項目または SD 項目に重複した文節が存在していて、使用できる文節が 1 つだけである場合、OPM COBOL/400 は最後に指定された文節を使用します。

同じ状況下で、ILE COBOL は最初に指定された文節を使用します。

作業用記憶域セクション (WORKING-STORAGE SECTION)

ILE COBOL では、独立した作業記憶域項目の記憶割り振りはそれらの項目が作業用記憶域セクション (WORKING-STORAGE SECTION) で宣言された順序を反映しませんが、OPM COBOL/400 では反映されていました。

ストレージの割り振り方法がこのように変更されることによる潜在的な影響は、OPM COBOL/400 の 32K という最大テーブル・サイズ制限を緩和する回避スキー

マを使用するプログラムに現れます。複数の独立した作業記憶域項目が一貫して宣言されていて、範囲検査がオフに設定されている場合に、テーブル・サイズを増加させる回避スキーマを使用するプログラムの場合、そのスキーマは機能しなくなります。そのようなスキーマを使用するプログラムを ILE COBOL を使用して実行すると、結果は予測不能になります。

ILE COBOL では、最大テーブル・サイズが 16,711,568 バイトになったので、この回避スキーマが起動されるという問題はなくなりました。しかし、この回避スキーマを使用するプログラムはすべて、コーディングしなおす必要があります。

LIKE 文節

REDEFINES 文節が LIKE 文節の後に存在する場合、OPM COBOL/400 コンパイラーは、REDEFINES 文節が LIKE 文節の後に現れたために無視されたことを示す重大度 20 のメッセージを出します。

同じ状況下で、ILE COBOL コンパイラーは REDEFINES 文節が検出されると重大度 10 のメッセージを出してその REDEFINES 文節を受け入れますが、さらに LIKE 文節に REDEFINES 文節との互換性がないことを示す重大度 30 のメッセージも出します。

LIKE および USAGE、または LIKE および PICTURE など互換性のない他の文節の場合にも、このようなことが生じる場合があります。

LINAGE 文節

OPM COBOL/400 は、符号付き LINAGE 整数にメッセージ LBL1350 のフラグを立てますが、符号付き FOOTING、TOP、および BOTTOM に対してはメッセージを出しません。

ILE COBOL は、4 つのすべての場合にメッセージ LNC1350 を出します。

PICTURE 文節

PICTURE ストリング .¥¥ は、ILE COBOL コンパイラーでは使用できません。同様に、PICTURE ストリング +.¥¥ および -.¥¥ も使用できません。

CR または DB が文字ストリングの文字位置 30 および 31 に存在する場合、ILE COBOL コンパイラーはそれらを有効な文字ストリングとして処理しません。PICTURE ストリングは全体で 30 文字以内でなければなりません。

REDEFINES 文節

OPM COBOL/400 は、再定義された項目を初期設定します。

ILE COBOL は、再定義された項目を初期設定しません。初期値は、元のデータ項目のデフォルト値によって判別されます。

VALUE 文節

ILE COBOL では、VALUE 文節に指定された数字リテラルが、それを定義する PICTURE ストリングよりも長い場合、その数字リテラルは切り捨てられます。OPM COBOL/400 では、値 0 と見なされます。

手続き部 (PROCEDURE DIVISION)

一般的な考慮事項

2 進データ項目: OPM COBOL/400 では、2 進データ項目に、ピクチャー文節で記述された値を超える値が指定された場合、結果は予測不能になります。一般的には、この項目が使用される場合、ピクチャー文節で記述された実際の桁数に切り捨てられる場合と、切り捨てられない場合とがあります。通常それは、値のコピーにパックされた中間結果が使用されたかどうかによって異なります。

ILE COBOL でも結果は予測不能ですが、OPM COBOL/400 から生成される結果とは異なるものとなります。

8 バイト 2 進データの位置合わせ: OPM COBOL/400 では、*SYNC オプションが CRTCBLPGM コマンドの GENOPT パラメーターに指定されている場合、8 バイト 2 進数項目は 4 バイト境界に位置合わせされます。

ILE COBOL では、*SYNC オプションが CRTCBLMOD または CRTBNDCBL コマンドの OPTION パラメーターに指定されている場合、8 バイト 2 進数項目は 8 バイト境界に位置合わせされます。

重複した段落名: 重複した段落名が COBOL プログラム中に存在すると、OPM COBOL/400 コンパイラーは重大度 20 のメッセージを生成します。

同様の状況下で、ILE COBOL コンパイラーは重大度 30 のエラー・メッセージを出します。

添え字の数: 項目に指定された添え字の数が適切でない場合 (過多、過少、添え字が必要な項目に指定されていない、添え字が不必要な項目に指定されている、という場合)、ILE COBOL では重大度 30 のメッセージが生成されます。

同じ状況下で、OPM COBOL/400 は重大度 20 のメッセージを生成します。

セグメント化: セグメント化は ILE COBOL ではサポートされていません。したがって、セグメント数の構文検査は実行されません。

共通句

(NOT) ON EXCEPTION 句: DISPLAY ステートメントには、(NOT) ON EXCEPTION 句が追加されました。それらの句が追加されたことによってコンパイル時エラーが出ないようにするために、有効範囲を区切る END-DISPLAY を追加することが必要になる可能性があります。

以下に例を示します。

```
ACCEPT B AT LINE 3 COLUMN 1
  ON EXCEPTION
      DISPLAY "IN ON EXCEPTION"
  NOT ON EXCEPTION
      MOVE A TO B
END-ACCEPT.
```

ACCEPT ステートメントに対して ON EXCEPTION 句と NOT ON EXCEPTION 句の両方が指定されました。しかし、下に示すように、END-DISPLAY がない場合、NOT ON EXCEPTION は DISPLAY ステートメントの一部であると見なされます。

```
 ACCEPT B AT LINE 3 COLUMN 1
   ON EXCEPTION
      DISPLAY "IN ON EXCEPTION"
      END-DISPLAY
   NOT ON EXCEPTION
      MOVE A TO B
END-ACCEPT.
```

INVALID KEY 句: ILE COBOL では、相対ファイルの順次アクセスに INVALID KEY 句を使用できません。これらの状況下では無効キーの意味が不確定だからです。この状況下では、ILE COBOL コンパイラーは重大度 30 のエラー・メッセージを出します。

OPM COBOL/400 コンパイラーは、この場合にエラー・メッセージを出しません。

ON SIZE ERROR 句: ILE COBOL で算術演算および条件式を使用する際、ON SIZE ERROR が指定されていない場合にサイズ・エラーが生じると、結果は予測できません。その結果は、OPM COBOL/400 の場合とは違う可能性があります。

ILE COBOL で算術演算および条件式を使用する際、ON SIZE ERROR が指定されていない場合にゼロ除算が生じると、結果は予測できません。その結果は、OPM COBOL/400 の場合とは違う可能性があります。

DECLARATIVE プロシージャー

ILE プロシージャーとして組み込まれた宣言: ILE COBOL では、DECLARATIVE プロシージャーはそれぞれ ILE プロシージャーです。したがって、それぞれの DECLARATIVE プロシージャーは、他の宣言から独立したもので、ILE COBOL プログラムの非宣言部分とは独立して、独自の呼び出しで実行されます。その結果、メッセージの送受信、RCLRSC CL コマンド、および指定変更などの、呼び出し依存システム機能の使用方法は、ILE COBOL と OPM COBOL/400 では異なります。

別の宣言から宣言を呼び出す: ILE COBOL では、前の宣言が何らかの理由によって呼び出されていない場合、その宣言は入出力エラーによって別の宣言から呼び出されることがあります。

OPM COBOL/400 では、宣言が入出力エラーによって別の宣言から呼び出されることはありません。

式

クラス条件式: ILE COBOL では、クラス条件式の ID として、1 つまたは複数の符号付き数値基本項目を含むグループ項目は使用できません。

短縮された条件式: ILE COBOL では、短縮された結合関係条件に括弧は使用できません。 OPM COBOL/400 では、この規則は要求されていません。

形象定数と形象定数の比較: OPM COBOL/400 では、形象定数を別の形象定数と比較すると、重大度 20 のエラー・メッセージが出され、そのステートメントは受け入れられます。

ILE COBOL で形象定数を別の形象定数と比較すると、重大度 30 のエラー・メッセージが出され、そのステートメントは拒否されます。

ゾーン項目と非数値項目との比較: ゾーン項目と非数値項目とを比較すると、OPM COBOL/400 では重大度 20 のメッセージが出されます。 ILE COBOL ではそのようなメッセージは出ません。

関係式での NOT: ″A NOT NOT = B″ という式は、OPM COBOL/400 では受け入れられますが、重大度 20 のメッセージが生成されます。

同じ状況下で、ILE COBOL は重大度 30 のメッセージを生成します。

NOT LESS THAN OR EQUAL TO: ILE COBOL では、OPM COBOL/400 で許容されないいくつかの形式の条件式が使用できます。特に、その中には NOT LESS THAN OR EQUAL および NOT GREATER THAN OR EQUAL が含まれます。

特殊レジスター

DEBUG-ITEM 特殊レジスター: ILE COBOL では、DEBUG-ITEM 特殊レジスターのサポートはなくなりました。見つかった場合、それは構文検査されるだけです。

LINAGE-COUNTER 特殊レジスター: OPM COBOL/400 で LINAGE 文節に整数を指定した場合、LINAGE-COUNTER は 2 バイト 5 桁の 2 進数項目として定義されます。

ILE COBOL で LINAGE 文節に整数を指定した場合、LINAGE-COUNTER は 4 バイト 9 桁の 2 進数項目として定義されます。

WHEN-COMPILED 特殊レジスター: OPM COBOL/400 では、WHEN-COMPILED 特殊レジスターを使用できるのは MOVE ステートメントだけです。

ILE COBOL では、WHEN-COMPILED 特殊レジスターは任意のステートメントに使用することができます。

拡張 ACCEPT および DISPLAY ステートメント

コンパイル時の考慮事項: OPM COBOL/400 では、拡張 ACCEPT および DISPLAY ステートメントを使用可能にするために、CRTCBLPGM コマンドの GENOPT パラメーターに EXTACCDSP の値を指定する必要があります。 ILE COBOL では、CRTCBLMOD/CRTBNDCBL コマンドに EXTACCDSP オプションはありません。 ILE COBOL では、拡張 ACCEPT および DISPLAY ステートメントは、常に使用可能です。 ILE COBOL では PROCESS ステートメントの EXTACCDSP オプションはなくなったので、PROCESS ステートメントにこのオプションを指定している OPM COBOL/400 プログラムを ILE COBOL コンパイラーでコンパイルした場合、そのプログラムは異なる動作をすることがあります。 ILE COBOL コンパイラーは、SPECIAL NAMES 段落中の CONSOLE IS CRT を検索することによって、または形式 7 の ACCEPT ステートメントか形式 3 の

DISPLAY ステートメントにある句を検索することによって、ACCEPT または DISPLAY ステートメントが拡張されているかどうかを判別します。

ILE COBOL では、以下の語は常に COBOL に予約されています。
- AUTO
- BEEP
- BELL
- FULL
- BLINK
- COL
- COLUMN
- PROMPT
- UPDATE
- NO-ECHO
- REQUIRED
- AUTO-SKIP
- HIGHLIGHT
- UNDERLINE
- ZERO-FILL
- EMPTY-CHECK
- LEFT-JUSTIFY
- LENGTH-CHECK
- REVERSE-VIDEO
- RIGHT-JUSTIFY
- TRAILING-SIGN

OPM COBOL/400 の場合、DISPLAY ステートメントでは固定テーブルのみサポートされます。 ILE COBOL では、ACCEPT および DISPLAY ステートメントはともに、どのテーブルもサポートします。

ILE COBOL では、拡張 ACCEPT および DISPLAY ステートメントで参照変更データがサポートされます。これは OPM COBOL/400 ではサポートされていません。

OPM COBOL/400 では、基本 DBCS 文字セットに加えて拡張文字セットを使用できるようにするには、*NOUNDSPCHR オプションを使用する必要があります。 ILE COBOL では、*NOUNDSPCHR または *UNDSPCHR のどちらを使用しても、DBCS 文字を適切に管理できます。

OPM COBOL/400 コンパイラーは、画面の容量よりも長いデータ項目を検出すると、重大度 30 のエラー・メッセージを出します。 ILE COBOL コンパイラーは、このようなエラー・メッセージは出しません。

ILE COBOL は、COLUMN 句の ID または整数が 8 桁を超える場合、重大度 30 のエラー・メッセージを出します。 OPM COBOL/400 は、エラー・メッセージを出しません。

拡張 DISPLAY ステートメントの中の、構文が検査されるだけの句については、ILE COBOL コンパイラーは PROMPT、BACKGROUND-COLOR、および FOREGROUND-COLOR 句の完全な構文検査を実行します。これらの句のどれかの

コーディングが誤っている場合、ILE COBOL コンパイラーは重大度 30 のエラー・メッセージを出します。 OPM COBOL/400 コンパイラーでは、PROMPT、BACKGROUND-COLOR、および FOREGROUND-COLOR 句に対する完全な構文検査は行われず、コンパイル時エラー・メッセージは出されません。

実行時の考慮事項: OPM COBOL/400 では、拡張 ACCEPT の操作中には PRINT キーが使用できなくなります。 ILE COBOL では、PRINT KEY は、常に無条件で使用可能です。

OPM COBOL/400 で、SIZE 句がサポートされるのは DISPLAY ステートメントだけです。 ILE COBOL では、SIZE 句は ACCEPT と DISPLAY の両方のステートメントでサポートされます。指定されたサイズが PICTURE 文節のデータ長で暗黙のうちに指定されたサイズを超えている場合、OPM COBOL/400 は英数字データの位置調整を行うのに左側にブランクを埋め込みます。 ILE COBOL は常に右側にブランクを埋め込みます。

OPM COBOL/400 では、データ項目が画面内に入りきらない場合、エラー・メッセージ LBE7208 が出されます。 ILE COBOL では、画面に入りきらない英数字データは切り捨てられ、画面に入りきらない数値データは表示されません。実行時エラーは出されません。

3174 または 3274 遠隔制御装置に接続されたワークステーション上で ACCEPT 操作を完了するために HELP および CLEAR キーが使用された場合、ILE COBOL では実行時エラーが出されます。 OPM COBOL/400 ではこの ACCEPT 操作が正常に終了して、実行時エラーは出されません。

OPM COBOL/400 では、ACCEPT ステートメントで処理されたフィールドのすべてが常に更新されます。 ILE COBOL では、1 つの ACCEPT ステートメントごとに、ユーザーが ENTER キーを押す前に変更したフィールドだけが更新されます。そのため、2 つのコンパイラーは次の 3 つの状況で異なる処理を行います。

- SECURE 句が ACCEPT ステートメントに指定されていて、値が入力されていない場合
- ACCUPDNE オプションが有効にされていて、数字編集が行われていないデータが ACCEPT ステートメントによって処理される場合
- フィールドに英数字データが事前表示されていて、RIGHT-JUSTIFIED 句が ACCEPT ステートメントに指定されている場合

CALL ステートメント

CALL/CANCEL リテラルまたは ID での小文字: OPM COBOL/400 では、CALL/CANCEL リテラルまたは ID に小文字を含めることができます。ただし、引用符付きのシステム名 (拡張名) でないプログラム・オブジェクト名には小文字を使用できません。その結果 CALL/CANCEL 操作は正常に終了しません。

ILE COBOL では、CRTCBLMOD および CRTBNDCBL コマンドの OPTION パラメーターの値として *MONOPRC と *NOMONOPRC の 2 つが新たにサポートされています。デフォルト値である *MONOPRC では、CALL/CANCEL リテラルまたは ID に使用されている小文字がすべて大文字に変換されます。 *NOMONOPRC 値は、CALL/CANCEL リテラルまたは ID に使用されている小文字を大文字に変換しないことを指定します。

USING 句でファイル名を渡す: OPM COBOL/400 および ILE COBOL のどちらにおいても、ファイル名を CALL ステートメントの USING 句に渡すことが可能です。ただし、OPM COBOL/400 は FIB (ファイル情報ブロック) へのポインターを渡しますが、ILE COBOL は NULL ポインターへのポインターを渡します。

再帰呼び出し: ILE COBOL では、再帰的プログラムを再起呼び出しすることができます。ILE COBOL は、非再帰的プログラムで再帰を検出すると実行時エラー・メッセージを生成します。

OPM COBOL/400 は、再帰を防止しません。しかし、OPM COBOL/400 で再帰を実行しようとすると、予測不能の結果になることがあります。

CANCEL ステートメント

ILE COBOL では、CANCEL ステートメントが取り消すのは、同じ活動化グループ内の ILE COBOL プログラムのみです。 ILE COBOL では、活動化グループ (実行単位) レベルで、呼び出されたプログラム・オブジェクトのリストが維持管理されています。取り消そうとするプログラムがこのリストに存在しない場合、その取り消しは無視されます。

OPM COBOL/400 では、取り消そうとするプログラムがライブラリー・リストに存在しない場合、CANCEL ステートメントでエラー・メッセージが出されます。

COMPUTE ステートメント

場合によっては、ILE COBOL で累乗を計算した結果が、OPM COBOL/400 で累乗を計算した結果と多少異なることがあります。

仮数が負の値で指数が負の小数値である累乗式に COMPUTE ステートメントを実行すると、OPM COBOL/400 は未定義の結果になります。同じ状況下で、ILE COBOL は CEE2020 例外を生成します。

式に指数演算が含まれている場合、固定小数点演算で実行された COMPUTE ステー
トメントの結果は、OPM COBOL/400 での結果と多少異なることがあります。 ILE
COBOL では、指数演算は内部的に浮動小数点数演算で実行されます。 COMPUTE
ステートメントに浮動小数点データ項目がない場合、指数の結果が固定小数点フォ
ーマットに変換され、式の残りを計算します。この変換により、OPM COBOL/400
の結果が多少異なることがあります。

DELETE ステートメント

OPM COBOL/400 では、DELETE ステートメント上でファイルに対して無効なレコード様式が使用される場合、ファイル状況が 90 に設定されます。

ILE COBOL 用では、DELETE ステートメント上でファイルに対して無効なレコード様式が使用される場合、ファイル状況が 9K に設定されます。

EVALUATE ステートメント

OPM COBOL/400 では、ZERO THRU *英字 id* に WHEN 句が指定された場合、そのステートメントは受け入れられ、診断メッセージは出されません。

同様の状況下で、ILE COBOL は重大度 30 のエラー・メッセージを出します。

注: ILE COBOL では、*英字 id* THRU *英字 id* の場合はこの規則は緩和されています。英字 ID に含めることができるのは英字だけだからです。

IF ステートメント

OPM COBOL/400 では、IF ステートメントのネストの深さの限度は 30 です。

ILE COBOL では、IF ステートメントのネストの深さに実際的な限度はありません。

OPM COBOL/400 では、END-IF 句と同じ IF ステートメント内で NEXT SENTENCE 句を使用した場合、制御が END-IF 句に続くステートメントに渡されます。

ILE COBOL では、同じ状況下で、制御は次の分離文字ピリオドに続くステートメント、すなわち、次の文の最初のステートメントに渡されます。

注: OPM「*AS/400 V3 言語 : COBOL/400 解説書*」の資料には、予想される動作が、ILE COBOL 用の場合に実際に発生する動作と同じであることが示されています。

INSPECT ステートメント

ILE COBOL では、INSPECT ステートメントでの参照変更がサポートされています。

OPM COBOL/400 にはこのサポートが含まれません。

MOVE ステートメント

英数字リテラルおよび索引名: 英数字リテラルを索引名に移動すると、OPM COBOL/400 は重大度 20 のエラー・メッセージを出します。同様の状況下で、ILE COBOL は重大度 30 のエラー・メッセージを出します。

英数字値および数字編集リテラル: 数字だけを含む英数字値を数字編集リテラルに移動する場合 (たとえば、`MOVE "12.34" TO NUMEDIT`)、OPM COBOL/400 はリテラルをデフォルトの 0 にします。同様の状況下で、ILE COBOL は重大度 30 のエラー・メッセージを出します。

ブール値: OPM COBOL/400 では、ブール値を参照変更された英字 ID に移動できます。 ILE COBOL では、これは実行できず、重大度 30 のエラー・メッセージが出されます。

CORRESPONDING 句: ILE COBOL の MOVE、ADD、および SUBTRACT CORRESPONDING ステートメントは、対応する項目を判別するために OPM COBOL/400 とは異なるアルゴリズムを使用します。 OPM COBOL/400 ではメッセージが出されない状況下でも、ILE COBOL では重大度 30 のエラー・メッセージが生成されることがあります。

```
01 A.
   05  B.
     10  C PIC X(5).
   05  C PIC X(5).
01 D.
```

```
        05  B.
          10  C PIC X(5).
        05  C PIC X(5).
MOVE CORRESPONDING A TO D.
```

OPM COBOL/400 はメッセージを出しませんが、ILE COBOL はメッセージ LNC1463 を出します。

ソース・ストリングとターゲット・ストリングが重なっている場合: MOVE ステートメントでソース・ストリングとターゲット・ストリングが重なり合っている場合、結果は予測できません。移動の仕方は、同様の状況下で OPM COBOL/400 を使用した場合と異なることがあります。

OPEN ステートメント

動的ファイル作成: 動的ファイル作成に関連して、互換性に関する考慮事項が 2 つあります。

- OPM COBOL/400 コンパイラーでは、索引ファイルの動的作成がサポートされています。

  ILE COBOL コンパイラーは、このようなサポートを*行いません*。
- ファイルは、DISK の COBOL 装置タイプに割り当てられたときのみ、動的に作成されます。

  OPM COBOL/400 では、データベース・ファイル指定変更 (OVRDBF) がある場合、DISK 以外の COBOL 装置タイプに割り当てられるファイル (データベース・ファイル) を作成します。

FORMATFILE のオープン: ILE COBOL では、FORMATFILE をオープンできるのは OUTPUT の場合のみです。 FORMATFILE に出力レコードを書き込むには、WRITE ステートメントを使用できます。

OPM COBOL/400 では、FORMATFILE は INPUT、I-O、および OUTPUT 用としてオープンできます。

OPTIONAL ファイルに対する OPEN OUTPUT または OPEN I-O: ILE COBOL では、ファイルの編成が INDEXED である場合に、OPTIONAL ファイルに対して OPEN OUTPUT または OPEN I-O を実行して、ファイルが存在しない場合、ファイルは作成されません。

OPM COBOL/400 では、ファイルが作成されます。

PERFORM ステートメント

ILE COBOL では、PERFORM ステートメントの VARYING...AFTER 句で、*id-2* は *id-5* の設定前に増分されます。 OPM COBOL/400 では、*id-5* は *id-2* の増分前に設定されます。

AFTER 句の指定された形式 4 の PERFORM ステートメントの結果は、ILE COBOL と OPM COBOL/400 とでは異なります。次の例を考えてみます。

```
    PERFORM PARAGRAPH-NAME-1
            VARYING X FROM 1 BY 1 UNTIL X > 3
                    AFTER Y FROM X BY 1 UNTIL Y > 3.
```

OPM COBOL/400 では、*PARAGRAPH-NAME-1* は (X,Y) 値が (1,1)、(1,2)、(1,3)、(2,1)、(2,2)、(2,3)、(3,2)、(3,3) の場合について実行されます。

ILE COBOL では、*PARAGRAPH-NAME-1* は (X,Y) 値が (1,1)、(1,2)、(1,3)、(2,2)、(2,3)、(3,3) の場合について実行されます。

READ ステートメント

AT END は相対ファイルのランダム読み取りでは使用できない: ILE COBOL では、相対ファイルのランダム読み取りに AT END 句を使用できません。このような状況ではランダム読み取りの意味が不確定だからです。この状況下では、ILE COBOL コンパイラーは重大度 30 のエラー・メッセージを出します。

OPM COBOL/400 コンパイラーは、この場合にエラー・メッセージを出しません。

エラー・メッセージ: ILE COBOL では、READ ステートメントを FORMATFILE に対して実行しようとした場合、FORMAT 句に対してエラー・メッセージ LNC0651 ではなく LNC1408 が出されます。

読み取り用装置が DATABASE 以外のものである場合、エラー・メッセージ LNC1408 が出されます。装置が DATABASE であっても ORGANIZATION が索引編成でない場合、エラー・メッセージ LNC0651 が出されます。

REWRITE ステートメント

OPM COBOL/400 では、REWRITE ステートメント上でファイルに対して無効なレコード様式が使用される場合、ファイル状況が 90 に設定されます。

ILE COBOL では、REWRITE ステートメント上でファイルに対して無効なレコード様式が使用される場合、ファイル状況が 9K に設定されます。

SET ステートメント

条件名を TRUE に設定し、関連した条件変数が編集項目である場合、OPM COBOL/400 は、条件変数に移動される際に条件名の値を編集します。

ILE COBOL は、条件名の値が条件変数に移動される際に編集を行いません。

SORT/MERGE ステートメント

GIVING 句と SAME AREA/SAME RECORD AREA 文節: ILE COBOL では、GIVING 句に関連したファイル名は、同じ SAME AREA または SAME RECORD AREA 文節では指定できません。 ILE COBOL コンパイラーは、この状況が検出されると、重大度 30 のエラー・メッセージを出します。

OPM COBOL/400 コンパイラーは、この場合にメッセージを出しません。

STOP RUN ステートメント

ILE 活動化グループで STOP RUN が出されると、暗黙のうちに COMMIT が実行されます。OPM COBOL/400 では、そのようにはなりません。

注: ジョブ・デフォルト活動化グループ (*DFTACTGRP) で出された STOP RUN では、暗黙の COMMIT は実行されません。

STRING/UNSTRING ステートメント

OPM COBOL/400 では、STRING/UNSTRING 操作での暗黙の関係条件の真値を判別するために、PROGRAM COLLATING SEQUENCE が使用されます。

ILE COBOL では、STRING/UNSTRING 操作での暗黙の関係条件の真値を判別する際、PROGRAM COLLATING SEQUENCE は無視されます。

アプリケーション・プログラミング・インターフェース (API)

ILE COBOL バインド可能 API

ILE COBOL では、OPM 実行時ルーチンではなく、新しいバインド可能 API を使用します。

- QlnRtvCobolErrorHandler ILE バインド可能 API により、QLRRTVCE は置き換えられます。
- QlnSetCobolErrorHandler ILE バインド可能 API により、QLRSETCE は置き換えられます。
- QlnDumpCobol ILE バインド可能 API により、定様式ダンプを作成する QLREXHAN は置き換えられます。
- QLRCHGCM は、ILE COBOL ではサポートされていません。複数の実行単位を得るには、名前付き ILE 活動化グループを使用します。

OPM COBOL/400 API の呼び出し

OPM COBOL/400 API は ILE COBOL から呼び出すことができますが、それらが影響を与えるのは OPM COBOL/400 実行単位だけです。

ILE COBOL 実行単位に影響を与えるには、対応する ILE API または CRTPGM コマンドの ACTGRP パラメーターを使用します。

実行時

OPM 互換実行単位のセマンティクスの保存

次の中では、OPM 互換実行単位のセマンティクスを正確に保存することができます。

- ILE COBOL プログラムだけから構成されるアプリケーション・プログラム、または
- OPM COBOL/400 プログラムと ILE COBOL プログラムが混在するアプリケーション・プログラム

ILE COBOL アプリケーションにおける OPM 互換実行単位のセマンティクスの保存: ILE COBOL アプリケーションにおいて、OPM 互換実行単位のセマンティクスを保存するには、次の条件に適合していなければなりません。

- 実行単位の参加プログラムのすべて (ILE COBOL または他の ILE プログラム / プロシージャー) は、単一の ILE 活動化グループ内で実行する必要があります。

  注: すべての参加プログラム・オブジェクトに対して、名前付きの ILE 活動化グループを使用することにより、実行の前に特定の ILE COBOL プログラムをメイン・プログラムとして指定する必要はなくなります。一方、実行前に特

定の ILE COBOL プログラムがメイン・プログラムとして認識されている場合は、ILE COBOL プログラムを UEP として使用して *PGM オブジェクトを作成するときに、ACTGRP オプションに *NEW 属性を指定することができます。 それ以外のすべての参加プログラムは、 ACTGRP オプションに *CALLER 属性を指定する必要があります。

- ILE 活動化グループの最も古い呼び出しが、その ILE COBOL のものでなければなりません。これは実行単位のメイン・プログラムです。

これらの条件に適合しない場合、ILE 活動化グループ内に暗黙のまたは明示的な STOP RUN が存在しても、活動化グループが終了しないことがあります。活動化グループは活動状態のままで、さまざまな ILE COBOL プログラムが最後に使用された状態になります。

注: 上記の条件のために、*DFTACTGRP で実行される ILE COBOL プログラムは、一般に、OPM と互換性のない実行単位で実行されます。 *DFTACTGRP で実行される ILE COBOL プログラムは、その静的ストレージは、ジョブの終了まで物理的に再利用されません。 CRTPGM コマンドの ACTGRP パラメーターに *CALLER が指定された ILE COBOL プログラムを OPM プログラムから呼び出すと、それは *DFTACTGRP で実行されます。

OPM COBOL/400 と ILE COBOL の混合アプリケーションにおける OPM 互換実行単位のセマンティクスの保存: OPM COBOL/400 プログラムと ILE COBOL とを混合し、かつ OPM 互換実行単位のセマンティクスを可能な限り正確に保存するには、次の条件に適合していなければなりません。

- OPM COBOL/400 プログラムの呼び出し (ILE COBOL の呼び出しではない) が、最初の COBOL 呼び出しであることが必要です。
- STOP RUN は OPM COBOL/400 プログラムによって出されます。
- (OPM COBOL/400) 実行単位内のすべての参加プログラムは、*DFTACTGRP 活動化グループ内で実行されていなければなりません。

上記の条件に適合しない場合、OPM と ILE の混在アプリケーション・プログラムについては、OPM 互換実行単位のセマンティクスは保存されません。たとえば、ILE COBOL プログラムが *DFTACTGRP 内で実行されていて、それが STOP RUN を出すと、OPM COBOL/400 プログラムと ILE COBOL プログラムは両方とも最後に使用した状態で残ります。

ILE COBOL では、CALL、CANCEL、EXIT PROGRAM、STOP RUN、および GOBACK の制御操作の流れのため、OPM 互換実行単位が使用されていないなら、実行単位が異なる機能を実行する結果になります。

エラー・メッセージ

ILE COBOL では、実行時エラー・メッセージに接頭部 LNR が付けられます。また、メッセージ・メンバーによっては、必ずしも OPM COBOL/400 の場合と同じではありません。

ILE COBOL では、実行単位が異常終了した場合、メッセージ CEE9901 が呼び出し側に戻されます。 OPM COBOL/400 では、同様の状況下でメッセージ LBE9001 が呼び出し側に戻されます。

ILE 例外処理と OPM 例外処理の違いのために、ILE COBOL ステートメントでは OPM COBOL/400 ステートメントと比較してより多くの例外を受け取ることがあります。

ファイル状況の 9A から 0A への変更

OPM COBOL/400 では、制御によりジョブが終了された場合、ファイル状況は 9A に設定されます。

ILE COBOL では、制御によりジョブが終了された場合、ファイル状況は 0A に設定されます。

ファイル状況の 9M から 0M への変更

OPM COBOL/400 では、最後のレコードがサブファイルに書き込まれた時点で、ファイル状況は 9M に設定されます。

ILE COBOL では、最後のレコードがサブファイルに書き込まれた時点で、ファイル状況は 0M に設定されます。

付録 H. 略語の用語集

略語	意味	説明
AG	活動化グループ (Activation Group)	ジョブ内のリソースの区分化。 1 つの活動化グループは、1 つ以上のプログラムに割り当てられたシステム・リソース (プログラムまたはプロシージャー値のためのストレージ、コミットメント定義、およびオープン・ファイル) から構成されます。
API	アプリケーション・プログラミング・インターフェース (Application Programming Interface)	オペレーティング・システムまたは別途注文可能なライセンス・プログラムにおいて提供される機能インターフェースで、高水準言語で記述されたアプリケーション・プログラムが、オペレーティング・システムまたはライセンス・プログラムの特定のデータや機能を使用できるようにするためのもの。
ANSI	米国規格協会 (American National Standards Institute)	認可を受けた組織が米国での自主業界基準を作成および保守するためのプロシージャーの確立を目的とした、生産者、消費者、および一般関係者で構成される組織。
ASCII	情報交換用米国標準コード (American National Standard Code for Information Interchange)	データ処理システム、データ通信システム、および関連装置で情報交換を行うための、米国規格協会が開発したコード。 ASCII 文字セットは 8 ビットで構成され、そのうちの 7 ビットは制御文字およびシンボリック文字で、1 ビットはパリティー検査ビットです。
CICS®®	顧客情報管理システム (Customer Information Control System)	リモート・ワークステーションで入力されたトランザクションをユーザー作成のアプリケーション・プログラムで同時に処理することを可能にする、IBM ライセンス・プログラム。このライセンス・プログラムには、データベースを作成、使用、および保守する機能、および他のオペレーティング・システム上にある CICS と通信する機能が含まれます。
CL	制御言語 (Control Language)	ユーザーがシステム機能を要求するためのすべてのコマンドの集まり。
DBCS	2 バイト文字セット (Double-Byte Character Set)	各文字が 2 バイトで表されている文字の集まり。日本語、中国語、および韓国語などのように、256 個のコード点では表現できないほど多くの記号を含む言語では、2 バイト文字セットが必要です。 1 文字に 2 バイトが必要なので、DBCS 文字の入力、表示、および印刷には、DBCS をサポートするハードウェアおよびソフトウェアが必要です。システムでは日本語、韓国語、中国語簡体字、および中国語繁体字の 4 つの 2 バイト文字セットがサポートされています。 1 バイト文字セット (SBCS) と対比される語。
DDM	分散データ管理 (Distributed Data Management)	あるシステム上のアプリケーション・プログラムまたはユーザーがリモート・システム上に保管されたデータ・ファイルを使用できるようにするためのオペレーティング・システムの機能。それらのシステムが通信ネットワークで接続されていること、およびリモート・システムでも DDM が実行されていることが必要です。

略語	意味	説明
DDS	データ記述仕様 (Data Description Specifications)	定形様式でシステムに入力される、ユーザーのデータベースまたは装置ファイルの記述。ファイルの作成においては、その記述が使用されます。
EBCDIC	拡張 2 進化 10 進コード (Extended Binary-Coded Decimal Interchange Code)	256 個の 8 ビット文字で構成されるコード化文字セット。
EPM	拡張プログラム・モデル (Extended Program Model)	i5/OS 上で高水準言語によってソース・コードのコンパイルおよびプログラムの作成を行うための機能の集まりで、プロシージャー呼び出しを定義するもの。
FIPS	連邦情報処理標準 (Federal Information Processing Standard)	ビジネスで使用するコンピューターおよびデータ処理の使用および管理を向上させるための公式標準規格。
HLL	高水準言語 (high-level language)	概念および構造が人間の思考になじみやすいプログラミング言語。例として、C、COBOL、および RPG などがあります。高水準言語は、コンピューターの構造および操作構造からは独立しています。
IBM i	IBM i	i5 のオペレーティング・システム。 以前は OS/400 と呼ばれていました。
ICF	システム間通信機能 (Intersystem Communications Function)	プログラムが他のプログラムまたはシステムと対話式に通信できるようにするためのオペレーティング・システムの機能。
ILE	Integrated Language Environment	すべての ILE 対応の高水準言語に、共通の実行時環境と、実行時バインドが可能なアプリケーション・プログラミング・インターフェース (API) とを提供する、構成およびインターフェースの集まり。
I/O	入出力 (Input/Output)	コンピューターに供給されるデータ、またはコンピューター処理の結果のデータ。
LVLCHK	レベル検査 (Level Checking)	オープンするファイルのレコード様式レベル ID と、コンパイルされたプログラムの一部であるファイル記述とを比較して、ファイルのレコード様式がプログラムのコンパイル後に変更されたかどうかを調べる機能。
ODP	オープン・データ・パス (open data path)	ファイルがオープンされると作成される制御ブロック。ODP には、マージされたファイル属性に関する情報、および入出力操作から戻された情報が含まれます。 ODP が存在するのは、ファイルがオープンされている間だけです。
ODT	オブジェクト定義テーブル (Object Definition Table)	コンパイル時にシステムによって作成されるテーブルで、プログラムで宣言されたオブジェクトを記録するためのもの。このテーブル内のプログラム・オブジェクトには、変数、定数、ラベル、オペランド・リスト、および例外記述が含まれます。このテーブルは、コンパイルされたプログラム・オブジェクト内に存在しています。
OPM	オリジナル・プログラム・モデル (original program model)	Integrated Language Environment (ILE) モデルの導入前に、i5/OS 上でソース・コードのコンパイルおよび高水準言語プログラムの作成を行うための機能の集まり。
PEP	プログラム入り口プロシージャー (program entry procedure)	動的プログラム呼び出しの ILE プログラムの入り口点となる、コンパイラーによって提供されるプロシージャー。ユーザー入り口プロシージャー (UEP) と対比。

略語	意味	説明
SDA	画面設計機能 (Screen Design Aid)	ライセンス・プログラム用の適用業務開発ツール・セット機能の 1 つで、ユーザーが画面やメニューを設計、作成、および保守する際に役立つもの。
SEU	原始ステートメント入力ユーティリティー	ライセンス・プログラム用の適用業務開発ツール・セット機能の 1 つで、ソース・メンバーを作成したり変更したりするために使用するもの。
SQL/400	構造化照会言語 /400 (Structured Query Language/400)	リレーショナル・データベースをサポートする IBM ライセンス・プログラムで、データベースに情報を入力したり、データベースから選択した情報を出力および編成したりするために使用するもの。
UEP	ユーザー入り口プロシージャー (user entry procedure)	動的プログラム呼び出しのターゲットとなる、アプリケーション・プログラマーによって作成された入り口プロシージャー。このプロシージャーは、プログラム入り口プロシージャー (PEP) によって呼び出されます。*プログラム入り口プロシージャー (PEP)* と対比。
UPSI	ユーザー・プログラム状況標識スイッチ (User Program Status Indicator switch)	ハードウェア・スイッチの機能を実行する外部プログラム・スイッチ。 UPSI 0 ～ 7 の 8 つのスイッチが用意されています。

注: オペレーティング・システムのコマンドの省略形については、ここでは取り上げていません。 IBM i コマンドとその使用法については、Web サイト http://www.ibm.com/systems/i/infocenter/ にある **i5/OS Information Center** の「プログラミング」カテゴリーの中の『*CL および API*』セクションを参照してください。

付録 I. ILE COBOL の資料

この付録では、ILE COBOL 関連の入手可能なオンライン情報やハードコピーについて説明します。資料の発注については、IBM 担当員にご連絡ください。

オンライン情報

オンライン情報には、次のいくつかの方法でアクセスすることができます。

- 「バインド COBOL PGM の作成 (CRTBNDCBL)」画面または「COBOL モジュールの作成 (CRTCBLMOD)」画面から F1 キーを押す。
- 「コンパイラー・オプション」ダイアログ・ボックスまたは「プログラム検査装置」ダイアログ・ボックスから F1 キーを押すか、「連携開発環境プログラム/400 の編集プログラム」内から F1 キーを押す。
- i5/OS Information Center (http://www.ibm.com/eserver/iseries/infocenter) にアクセスする。

  地域、言語、そして最新のリリースを選択してください。左側で、「**System i 補足資料**」を選択します。次に、「**全資料リスト (Complete list of manuals)**」をクリックします。表示されたオンライン・ブックのリストから、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」および「*IBM Rational Development Studio for i: ILE COBOL 解説書*」にアクセスすることができます。

ハードコピー情報

ILE COBOL 製品に関して、以下の印刷資料が入手できます。

- *IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*, SD88-5045-07
- *IBM Rational Development Studio for i: ILE COBOL 解説書*, SD88-5044-07
- *ILE 概念*, SC41-5606-09.

注: 上記以外の資料についても有料で注文できます。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒242-8502
神奈川県大和市下鶴間1623番14号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Ltd. Laboratory
Information Development
8200 Warden Avenue
Markham, Ontario, Canada L6G 1C7

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、IBM 機械コードのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。サンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。IBM は、その予見の有無を問わず、お客様の当該サンプル・コードの使用から生ずるいかなる損害に対しても一切の責任を負いません。

プログラミング・インターフェース情報

本書の目的は、Integrated Language Environment® ILE COBOL プログラムをお客さまが作成する手助けをすることです。本書は、ILE COBOL コンパイラーを使用する場合に必要な情報を記述しています。

ILE COBOL コンパイラーのサービスを要求したり、サービスを受けたりするプログラムを作成する際に使用するプログラミング・インターフェースについては説明していません。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

Java およびすべての Java 関連の商標およびロゴは Sun Microsystems, Inc.の米国およびその他の国における商標です。

注記

IBM は、ILE COBOL コンパイラーにおいて下記の研究プロダクトを使用しています。

S/SL ©Copyright 1981 by the University of Toronto

参考文献

i5/OS システムでの ILE COBOL プログラミングに関連するトピックの詳細は、以下の IBM i5/OS 資料を参照してください。

- 「*AS/400 プログラム開発管理機能 (PDM)*, SC88-5197-00」適用業務開発ツール・セットのプログラム開発管理機能 (PDM) を使用して、ライブラリー、オブジェクト、メンバー、およびユーザー定義オプションのリストを処理し、コピー、削除、名前変更などの操作を容易に行うための情報を提供します。 PDM を学習するのに役立つ活動および参照情報が入っています。最もよく使用される操作およびファンクション・キーについては、例を用いて詳しく説明されています。
- 「*AS/400 適用業務開発ツールセット AS/400 用原始ステートメント入力ユーティリティー 使用者の手引きと参照*, SD88-5047-00」適用業務開発ツール・セットの原始ステートメント入力ユーティリティー (SEU) を使用したソース・メンバーの作成および編集について説明します。この資料では、SEU セッションを開始および終了する方法と、このフルスクリーン・テキスト・エディターの多くの機能を使用する方法を説明しています。この解説書には、新規ユーザーと経験の深いユーザーの両方にとって簡単な行編集コマンドから高水準言語やデータ形式用の事前定義プロンプトの使用まで、各種の編集作業の実施に役立つ例が入っています。
- 「*AS/400 Advanced Series Application Display Programming*, SC41-5715-02」次の情報を提供します。
  - DDS を使用してアプリケーション用の表示画面を作成したり保守したりする方法。
  - システムでディスプレイ・ファイルを作成したり処理したりする方法。
  - オンライン・ヘルプ情報の作成。
  - UIM を使用してアプリケーション用のパネルおよびダイアログを定義する方法。
  - パネル・グループ、レコード、および文書の使用
- 「*Recovering your system*, SC41-5304-10」次の機能のセットアップと管理に関する情報を提供します。
  - ジャーナリング、アクセス・パス保護、およびコミットメント制御
  - ユーザー補助記憶域プール (ASP)
  - ディスク保護 (装置パリティー、ミラー保護、およびチェックサム)

  バックアップ・メディアおよび保管/復元操作についてのパフォーマンス情報も提供します。さらに、活動時保管サポートの使用、異なるリリースへの保管と復元、およびプログラミングのヒントと技法など、拡張バックアップおよびリカバリーに関するトピックも記載されています。
- 「*CICS for iSeries Application Programming Guide*, SC41-5454-02」 iSeries 用 CICS のアプリケーション・プログラミングについて説明します。この資料には、CICS アプリケーション・プログラム・インターフェースとシステム・プログラミング・インターフェース・コマンドに関する手引きおよび参照情報、さらに、新しいアプリケーションの開発や他の CICS プラットフォームからの既存のアプリケーションのマイグレーションに関する一般情報が記載されています。
- 「*CL プログラミング*, SD88-5038-06」オブジェクトとライブラリーに関する概説、CL プログラミング、制御の流れとプログラム間通信、CL プログラム内のオブジェクト処理、および CL プログラムの作成などの、iSeries プログラミングに関するトピックを広範囲にわたって説明しています。その他のトピックとしては、事前定義メッセージと即時メッセージおよびメッセージの処理、ユーザー定義のコマンドとメニューの定義と作成、デバッグ・モード、ブレークポイント、追跡、および表示機能を含むアプリケーションのテストなどが入っています。
- 「*Communications Management*, SC41-5406-02」通信環境における作業管理、通信状況、通信問題のトレースと診断、エラー処理とリカバリ

ー、パフォーマンス、および特定の回線速度とサブシステム・ストレージについての情報を提供します。

- 「*Experience RPG IV Tutorial Kit*, GK2T-9882-00」RPG III と RPG IV の違いおよび新しい ILE 環境での作業方法を説明する、対話式のチュートリアルです。付属のワークブックには追加の練習問題が載せられていて、学習終了後には参照資料として役立ちます。 ILE RPG コンパイラーのコーディング例が、自習用プログラムとともに配布され、それらは iSeries 上で直接実行できます。
- 「*GDDM Programming Guide*, SC41-0536-00」IBM i 図形データ表示管理プログラム (GDDM) を使用してグラフィックス・アプリケーション・プログラムを作成する方法について説明します。多くのプログラム例およびこのプロダクトをデータ処理システムに適合させる方法を理解するのに役立つ情報が含まれています。
- 「*GDDM Reference*, SC41-3718-00」 IBM i 図形データ表示管理プログラム (GDDM) を使用してグラフィックス・アプリケーション・プログラムを作成する方法について説明します。この資料では、GDDM で使用可能なすべてのグラフィック・ルーチンが詳細に説明されています。さらに、GDDM に対する高水準言語インターフェースに関する情報を提供します。
- 「*ICF Programming*, SC41-5442-00」 iSeries 通信および IBM i システム間通信機能 (IBM i-ICF) を使用するアプリケーション・プログラムを作成するのに必要な情報を提供します。さらに、データ記述仕様 (DDS) キーワード、システム提供の様式、戻りコード、ファイル転送サポート、およびプログラム例が含まれています。
- 「*IDDU Use*, SC41-5704-00」 iSeries 対話式データ定義ユーティリティー (IDDU) を使用してデータ・ディクショナリー、ファイル、およびレコードをシステムに記述する方法を説明します。次の情報が含まれています。
  - コンピューター・ファイルおよびデータ定義の概念の紹介
  - IDDU を使用して照会および文書において使用するデータを記述する方法の紹介
  - データ・ディクショナリー、ファイル、レコード様式、およびフィールドの作成、保守、および使用に関する代表的な作業
  - IDDU を使用して他のシステムで作成されたファイルを処理するための詳細情報、およびエラー・リカバリーと問題予防に関する情報
- 「*IBM Rational Development Studio for i: ILE C/C++ Programmer's Guide*, SC09-2712-07」ILE C コンパイラー言語を使用してアプリケーションを開発する方法について説明しています。この資料には、プログラムの作成、実行、およびデバッグに関する情報が含まれています。また、言語をまたがるプログラムとプロシージャー呼び出し、ロケール、例外処理、データベース、外部記述ファイル、および装置ファイルのプログラミング上の考慮事項が解説されています。 パフォーマンス上のヒントもいくつか説明されています。 付録には、ソース・コードを EPM C/400 またはシステム C/400 から ILE C コンパイラーにマイグレーションする方法についても説明されています。
- 「*IBM Rational Development Studio for i: ILE C/C++ 言語参照*, SC88-4026-02」では、C および C++ プログラミング言語の構文、セマンティクス、および IBM による実装について説明しています。
- 「*IBM Rational Development Studio for i: ILE COBOL 解説書*, SD88-5044-07」ILE COBOL プログラム言語について説明しています。 ILE COBOL プログラム言語の構造および ILE COBOL ソース・プログラムの構造についての情報を提供します。さらにこの資料には、見出し部の段落、環境部の文節、データ部の文節、手続き部のステートメント、およびコンパイラー指示ステートメントのすべてについても説明されています。
- 「*ILE 概念*, SC41-5606-09」 iSeries ライセンス・プログラムのIntegrated Language Environment (ILE) アーキテクチャーに関する概念および用語について説明します。扱われるトピックには、モジュールの作成、バインディング、プログラムの実行、プログラムのデバッグ、および例外処理が含まれています。
- 「*IBM Rational Development Studio for i: ILE RPG プログラマーの手引き*, SD88-5042-07」iSeries システムにおけるIntegrated Language Environment (ILE) での RPG IV 言語のインプ

リメンテーションである ILE RPG コンパイラー・プログラム言語について説明します。この資料には、プログラムの作成と実行、およびプロシージャー呼び出しと言語間通信の考慮事項についての情報が含まれています。さらに、デバッグおよび例外処理について扱い、RPG プログラム内で iSeries ファイルおよび装置を使用する方法を説明します。付録には、RPG IV へのマイグレーションに関する情報と、コンパイラー・リストのサンプルが含まれています。この資料は、データ処理の概念および RPG プログラミング言語の基礎を理解している方を対象としています。

• 「*IBM Rational Development Studio for i: ILE RPG 解説書*, SD88-5043-07」ILE RPG コンパイラー・プログラム言語について説明しています。この解説書は位置ごとおよびキーワードごとに、すべての RPG IV 仕様書に有効な項目を説明し、またすべての演算コードおよび組み込み関数を詳細に説明しています。またこの解説書には、RPG の論理サイクル、配列とテーブル、編集機能、および標識の説明があります。

• 「*AS/400 アドバンスト・シリーズ 装置構成*, SD88-5003-00」 iSeries サーバーでのローカル装置の構成について説明します。これには、次のものを構成する方法が含まれます。

– ローカル・ワークステーション制御装置 (平衡型制御装置を含む)

– テープ制御装置

– ローカル接続装置 (平衡型装置を含む)

• 「*印刷装置プログラミング*, SD88-5073-03」印刷を理解し、制御するのに役立つ情報を提供します。エレメントの印刷と iSeries サーバーの概念、印刷操作用のプリンター・ファイルと印刷スプーリング・サポート、およびプリンターの接続性に関する具体的な情報を提供します。パーソナル・コンピューターの使用に関する考慮事項、他の印刷機能 (IBM ビジネス・グラフィックス・ユーティリティー (IBGU) など)、高機能印刷 (AFP™)、および iSeries の印刷エレメントを扱う例 (スプール出力ファイルをある出力待ち行列から別の出力待ち行列に移動する方法など) が記載されています。さらに、印刷作業負荷の管理のために使用する制御言語 (CL) に関する付録が含まれています。 iSeries で使用できるフォントも示されています。フォント置換テーブルによって、接続されたプリンターがアプリケーション指定のフォントをサポートしていない場合の、置き換えフォントの相互参照が提供されます。

• 「*Security reference*, SC41-5302-11」システム・セキュリティー・サポートを使用して、システムやデータが適切な権限のないユーザーによって使用されるのを防ぐ方法、データを故意または偶発的な損傷または破壊から保護する方法、セキュリティー情報を最新の状態に保つ方法、およびシステムにセキュリティーをセットアップする方法について説明します。

• 「*IBM i および関連ソフトウェアのインストール、アップグレードおよび削除*, SD88-5002-11」初期インストールと、IBM 提供のライセンス・プログラム、プログラム一時修正 (PTF)、および 2 次言語のインストール手順をステップごとに示します。この資料は、あるリリースを iSeries サーバーにインストール済みのユーザーで、新しいリリースをインストールしたい方も対象としています。

システム・アプリケーション体系 (SAA) 共通プログラミング・インターフェース (CPI) COBOL についての情報は、次の資料を参照してください。

- 「*システム・アプリケーション体系 (SAA) 共通プログラミング・インターフェース COBOL 解説書*, N:SC26-4354」

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

［サ行］

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[ワ行]

[数字]

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

[特殊文字]

プログラム番号: 5770-WDS

Printed in USA

SD88-5045-07

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21