IBM i
バージョン 7.2

プログラミング IBM Rational Development Studio for i ILE COBOL 解説書

IBM i
バージョン 7.2

プログラミング
IBM Rational Development Studio for i
ILE COBOL 解説書

お願い

本書および本書で紹介する製品をご使用になる前に、 A-1 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。

本製品およびオプションに付属の電源コードは、他の電気機器で使用しないでください。

本書は、IBM Rational Development Studio for i (製品番号 5770-WDS) に適用されます。また、新しい版で明記されていない限り、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適用されます。このバージョンは、すべての RISC モデルで稼働するとは限りません。また CISC モデルでは稼働しません。

本書にはライセンス内部コードについての参照が含まれている場合があります。ライセンス内部コードは機械コードであり、IBM 機械コードのご使用条件に基づいて使用権を許諾するものです。

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： IBM i
Version 7.2
Programming
IBM Rational Development Studio for i
ILE COBOL Language Reference

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 2014.4

目次

ILE COBOL 解説書

ILE COBOL 解説書

本書では、Integrated Language Environment® COBOL (ILE COBOL) プログラム言語について説明します。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。

本書では、他の IBM® 資料に言及しています。これらの資料の、完全な資料名、資料番号を 10-1 ページの『参考文献』に記載しています。本文で参照するときは、資料名の略称が使われます。

- ステートメント、文節、特殊レジスター
- プログラム構造
- 概念
- 新しい機能

ILE COBOL 解説書について

このセクションでは、この解説書に関する情報を示します。

このセクションでは、この解説書に関する情報を示します。

本書の対象読者

本書は、IBM i (以前の OS/400) システムの ILE COBOL プログラム言語について解説しています。 本書は、データ処理および COBOL プログラム言語の基礎を理解している方々を対象としています。

本書を読む際には、IBM i システムに関する下記の事柄について理解している必要があります。

- 使用するディスプレイ装置 (ワークステーションとも言う) およびその制御についての知識が必要です。そのディスプレイ装置で稼働しているソフトウェアや接続されているハードウェアに関係なく、標準となっている表示画面のエレメントやキーボード上のキーもあります。このようなキーの例を次に示します。
  - カーソル移動キー
  - コマンド・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
- ディスプレイ装置が IBM i システムと接続され、IBM i ソフトウェアを実行している場合の操作方法について知っておく必要があります。これは、次のようなことを行うために、IBM i オペレーティング・システムと制御言語 (CL) について理解するということになります。
  - 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
  - ユーティリティーの呼び出し
  - メッセージへの応答

  このオペレーティング・システムおよびその制御言語について詳しくは、IBM i Information Centerのカテゴリー『*プログラミング*』のセクション『*CL および API*』を参照してください。資料「*CL プログラミング*」も参照することができます。
- Information Center の中の『*データ管理機能*』のトピックについての知識を必要とします。このトピックでは、ユーザー・プログラムの外部にあるデータベース・ファイルや特定の装置ファイルを記述するために必要な、記入項目とキーワードについての詳細説明を提供しています。
- Information Center マニュアルの中『*DDS 解説書*』のトピックについての知識を必要とします。このトピックでは、データ管理機能サポートを使用し、ファイルがアプリケーションで作動できるようにするための情報を提供しています。

  この資料には、以下の情報が含まれています。
  - システムのデータ管理機能サポートの基本的な構造と概念

- スプーリングだけでなく、ディスプレイ装置、プリンター、テープ、およびディスケットのデータ管理機能サポート
- 指定変更および宛先変更 (アプリケーションが実行しているときにファイルに一時的な変更を行うこと)
- データをある場所から別の場所にコピーするための、システム・コマンドを使用したファイルのコピー
- 2 バイト文字データを使用するシステムの調整

- IBM i システム上で使用可能な特定のユーティリティーの呼び出し方法と使用方法について、知っている必要があります。
  - 画面設計機能 (SDA) は、画面の設計およびコーディングに使用します。この情報は、「*適用業務開発ツールセット AS/400 用 画面設計機能 (SDA)*」に含まれています。
  - 原始ステートメント入力ユーティリティー (SEU)。これはソース・メンバーとプロシージャー・メンバーの入力および更新に使用できるフルスクリーン・エディターです。この情報は、「*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*」に含まれています。
  - プログラム開発管理 (PDM) ユーティリティーはリスト処理ツールであり、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するために使用できます。この情報は、「*ADTS/400: Programming Development Manager*」に含まれています。
- IBM i オペレーティング・システムに付随するアプリケーション・プログラム・インターフェース (API) の使用方法について知っている必要があります。この情報は「*System API Programming*」に記載されています。
- 表示メッセージや印刷メッセージの解釈の方法を知っている必要があります。この情報は、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」に含まれています。
- Integrated Language Environment の概念および用語についての知識が必要です。この情報は、資料「*ILE 概念*」に含まれています。

前提条件および関連情報

IBM i 技術情報を検索する場合は、IBM i Information Center を開始点として使用してください。以下の 2 つの方法で、Information Center にアクセスできます。

- 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。

IBM i 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、アドバイスのための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 コンパイラー機能に最も関係のあるものであり 10-1 ページの『参考文献』にその一覧が示されています。

コメントの送付方法

IBM にお客様のご意見をお寄せください。本書や他の IBM i の資料に関するご意見をお持ちの際は、次のいずれかの方法をお選びください。

- 郵便の場合は下記あてにご送付ください。

  IBM Canada Ltd. Laboratory
  Information Development
  8200 Warden Avenue
  Markham, Ontario, Canada L6G 1C7

  「ご意見記入用紙」を米国以外の国からご郵送いただく場合は、ご当地の IBM 事業所または IBM 担当員にお渡しいただいて構いませんが、所定の郵便料金はご負担いただきます。
- FAX の場合は、次の番号にお願いします。1-845-491-7727
- 電子メールの場合は、以下の E メール・アドレスです。
  - 資料に関するご意見

    RCHCLERK@us.ibm.com
  - IBM i Information Center に関するご意見

    RCHINFOC@us.ibm.com

以下の項目の記入をお願いします。

- 資料名。
- 資料番号。
- ご意見の内容に該当するページ番号またはトピック。

新しい機能

ILE COBOL のリリースには、いくつかのリリースがあります。以下は、V3R1 から現行のリリースに至るまでの、各リリースに対して行われた機能強化の一覧です。

- 1-4 ページの『このリリースでの変更点』
- 1-5 ページの『7.1 での変更点』
- 1-7 ページの『V6R1 での変更点』
- 1-7 ページの『V5R4 での変更点』
- 1-8 ページの『V5R3 での変更点』
- 1-9 ページの『V5R2 での変更点』
- 1-10 ページの『V5R1 での変更点』
- 1-12 ページの『V4R4 での変更点』
- 1-12 ページの『V4R2 での変更点』
- 1-14 ページの『V3R7 での変更点』
- 1-15 ページの『V3R6/V3R2 での変更点』
- 1-16 ページの『V3R1 での変更点』

このセクションを使用して、新しい ILE COBOL の機能にリンクして学習することができます。

注: この製品に対するこの情報は、ILE COBOL の 7.2 のリリースにおける最新のものです。コンパイラーの前のリリースを使用している場合は、ユーザーのシステムでどの機能がサポートされるかを判断する必要があります。例えば、6.1 システムを使用している場合には、7.2 のリリースにおける新しい機能はサポートされません。

このリリースでの変更点

以下に、7.2 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。

- 秒の小数部が 0 から 12 桁の TIMESTAMP のサポート

  タイム・スタンプ項目には、0 から 12 桁の秒の小数部を含めることができるようになりました。

  - 組み込み関数 ADD-DURATION、EXTRACT-DATE-TIME、FIND-DURATION、および SUBTRACT-DURATION をタイム・スタンプ項目で指定する際に、期間として PICOSECONDS が許可されるようになりました。
  - FORMAT TIMESTAMP で SIZE キーワードが許可されるようになりました。サイズは、19 (秒の小数部がないことを示す)、または 21 から 32 の値 (1 から 12 桁の秒の小数部を示す) にすることができます。
- XML PARSE で、16MB を超えるサイズの XML ファイルを解析できるようになりました (処理プロシージャーに渡される個別文書が 16MB を超えない場合)。以下の新規 XML-CODE 値がこの変更に関連しています。
  - XML-CODE 62。XML 文書が 16,000,000 バイトを超えていることを示します。
  - XML-CODE 170。XML イベントが 16,000,000 バイトを超えていることを示します。
- PCML 生成
  - PCML 生成で、配列の最大サイズに設定される新規「init」キーワードが追加され、OCCURS DEPENDING ON の配列処理が改善されています。
  - PCML 生成は、データ構造におけるフィラー・データ項目および名前なしデータ項目のために、生成された PCML で自動データ項目命名を提供し、Web サービスが生成された PCML を最初に変更することなく使用できるようにします。これらのデータ項目の名前は、*_filler_1*、*_filler_2* といったようになります。
- 国別 (Unicode) 機能強化
  - 数字国別データ・タイプがサポートされます
  - 数字国別データ項目に対して VALUE 文節で数字リテラルを指定できます
  - 国別データ項目で使用された表意定数 ZERO/ZEROS/ZEROES が 1 つ以上の国別のゼロの桁を表します
  - 新規 PROCESS オプション NATIONALPICNLIT の指定時に、国別 'N' リテラルがサポートされます
- コンパイラー・オプション ARITHMETIC(*EXTEND31) または PROCESS オプション EXTEND31 を指定すると、数字組み込み関数 NUMVAL および NUMVAL-C の精度が 31 桁に上がります。
- CRTBNDCBL / CRTCBLMOD の ARITHMETIC パラメーター:

  新規 *EXTEND31FULL オプション値が以下の機能を提供します。

  - 数字組み込み関数 ANNUITY、MEAN、MEDIAN、MIDRANGE、NUMVAL、NUMVAL-C、PRESENT-VALUE、および VARIANCE の精度が、浮動小数点の精度 (最大 15 桁) から 10 進浮動小数点の精度 (最大 34 桁) に上がります。

| – 固定小数点演算式の中間結果を 34 桁までにすることができ、数値リテラルの最大長を 34 桁にすることができます。

| • 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。

| – NOCHGFLTRND / ALWCHGFLTRND

| MI 命令 SETCA で指定されている浮動小数点丸めモード計算属性を COBOL が使用するかどうかを指定します。SETCA により、浮動小数点の計算結果の丸めモードを丸めまたは切り捨てに設定できます。

| – NATIONALPICNLIT

| 国別リテラルの開始区切り文字として N" および N' を有効にし、PICTURE 記号 N を使用して定義された基本データ項目が暗黙の USAGE NATIONAL 文節を持つことができるようにします。

| – EXTEND31FULL

| **注:** このガイドには iSeries® への廃止された参照を含む画面取りがある可能性があります。

| 7.1 以降に本書に加えられた変更

| この 7.2 のガイド「*IBM Rational® Development Studio for i: ILE COBOL 解説書*」(SD88-5044-08) には、
| 7.1 のガイド (SD88-5044-07) とは異なる場所がいくつかあります。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。本書を使用する上で役立つように、
| 7.2 で行われた技術変更および機能強化による変更個所は縦線 (|) で示されています。

7.1 での変更点

以下に、7.1 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

注: このガイドには iSeries への廃止された参照を含む画面取りがある可能性があります。

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。

- 国別 UCS-2 CCSID サポート

  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、国別データ項目で使用できるように UCS-2 CCSID を指定できるようになりました。このパラメーターによって、国別項目で使用できるように CCSID 1200 のようなデフォルトの 13488 以外の CCSID を指定できます。

- モジュール内の PCML サポート
  - CRTCBLMOD コマンドおよび CRTBNDCBL コマンドの PGMINFO パラメーターは、生成された PCML を置く位置を指定できるように拡張されました。ユーザーが *PCML を PGMINFO キーワードの最初のパラメーターとして指定するときに、*STMF、*MODULE、または *ALL の位置を指定する 2 番目のパラメーターも指定できます。*STMF によって PCML は INFOSTMF パラメーターで指定されたストリーム・ファイルに置かれ、*MODULE によって PCML は生成されたモジュールに置かれ、および *ALL によって PCML はこれらの位置のすべてに置かれます。
  - PROCESS ステートメントのオプション PGMINFO

    このオプションにより、ユーザーは PCML がモジュールに追加され、PGMINFO(PCML MODULE) として指定できるように要求できるようになります。ユーザーが作成コマンドで PCML をストリーム・ファイルに追加するように要求した場合、PCML はモジュールとストリーム・ファイルの両方に追加されます。

- 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・デバッガーは複合 OCCURS DEPENDING ON 配列をデバッグできるようになりました。

- 大規模プログラム・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 サポートが拡張されました。新規ステートメント、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。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することができます。非数値リテラルは、非数値リテラルを指定できるすべてのところで使用できますが、ヌル終了リテラルは、「すべてのリテラル」表意定数でサポートされるわけではありません。

- 新規の CRTBNDCBL / CRTCBLMOD オプション

*NOCOMPRESSDBG/*COMPRESSDBG は、 DBGVIEW オプション *LIST または *ALL が指定されている場合、リスト・ビューの圧縮が、コンパイラーによって実行され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説明します。

- 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、リテラルに対して CONSTANT 文節を含むレベル-01 記入項目を指定することで定義されます。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、 XML 文書を構文解析し、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。

  次の XML 特殊レジスターは、XML パーサーとユーザー作成の処理プロシージャー間で情報を通信するために使用されます。

  - XML-CODE
  - XML-EVENT
  - XML-NTEXT
  - XML-TEXT

- 代替レコード・キーのサポート

  ALTERNATE RECORD KEY 文節を使用すれば、索引付きファイルに関連する代替レコード・キーを定義できます。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 桁に拡張されました。
  - CRTCBLMOD および CRTBNDCBL コマンド、並びに PROCESS ステートメントにおける 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、永続の代替レコード・キーを検出できない場合、一時的な代替レコード・キー索引を作成するかどうかを指定します。
  - *STDINZHEX00 は、値文節のないデータ項目が、16 進数のゼロで初期設定されることを指定します。
  - ARITHMETIC パラメーターに対する *EXTEND63 オプションは、固定小数点演算の中間結果の精度を 63 桁にまで増加させます。
- 新しい PROCESS ステートメント・オプション
  - PROCESS ステートメント・オプション NOCOMPRESSDBG/COMPRESSDBG は、DBGVIEW オプション *LIST または *ALL が指定されている場合、リスト・ビューの圧縮が、コンパイラーによって実行されるべきかどうかを指定します。
  - 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は、COBOL プログラムがエラーを受信したときのエラー情報を含む定義済み構造です。PROGRAM STATUS 文節は、受信されたエラー情報を指定するために使用されます。

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。ローカル・ストレージ・セクションは、再帰的プログラムおよび非再帰的プログラムの両方で指定できます。
- 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、Java プログラムが、より少ない Java コードでこの 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。以下の機能です。

– 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、例えば、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。
  – 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。

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

  Domino® あるいは Java のように、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 (例えば、「\」) だけでなく国際通貨記号 (例えば、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・プロファイル作成サポート

  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

IBM i 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
  - IBM i 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 IBM i 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、IBM i 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 IBM i 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、IBM i 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	LOG10

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
LOWER-CASE
MEAN
NUMVAL
NUMVAL-C
ORD
REVERSE
SIN
SQRT
TAN
UPPER-CASE
WHEN-COMPILED
YEAR-TO-YYYY

- バインディング・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ディング・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、ライブラリー名を使用して IBM i のプログラム・オブジェクトとデータ域を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 IBM i 用と ILE RPG IBM i 用の両方が 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改善されています。

- 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® (S/390®) からのアプリケーションのマイグレーションに便利です。
  - *CHGPOSSGN/*NOCHGPOSSGN オプション

    このオプションは、IBM i 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、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

ILE COBOL 構文表記法

ILE COBOL では、基本的な形式が以下に説明する一定のシステム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
```

  これは表記どおり正確につづらなければなりません。キーワードをいずれか 1 つでも書き忘れると、コンパイラーは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。

  ID–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。これらの演算子は、*特殊文字* の予約語です。ILE COBOL の予約語の完全なリストについては 9-17 ページの『付録 E. ILE COBOL 予約語リスト』を参照してください。
- 構文図に示されているすべての句読点およびその他の特殊文字は、表示するときにその形式の構文上すべて必要なものです。それらを省略すると、プログラムでエラーが起こります。
- 必要な文節でもオプショナルの文節 (ある場合) でも、関連する規則の中で特に指示されていないかぎり、構文図に示されている順序どおりに書かなければなりません。

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パスに従って読みます。

  ▸▸── この記号は、ステートメントの始めを表します。

  ──▸ この記号は、ステートメントの構文が次の行に続くことを表します。

  ▸── この記号は、ステートメントが前の行から続いていることを表します。

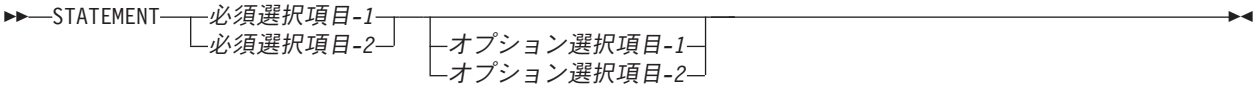
  ──▸◂ この記号は、ステートメントの終わりを表します。

  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、▸▸── 記号で始まり、──▸◂ で終わります。

  注: 段落全体の図内のステートメントは、その始まりか終わりが段落の始まりか終わりと一致しない限り ▸▸── で始まり、──▸◂ で終わることはありません。
- 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。

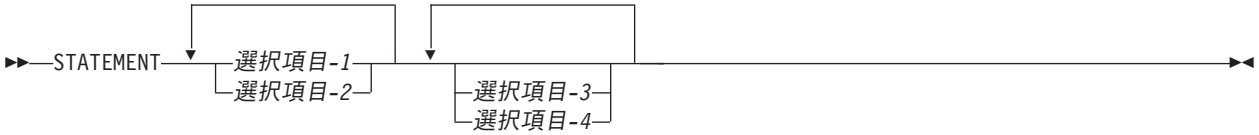
- 複数の項目の中から選択できる場合には、縦に重ねて示されます。

  項目の中から必ず 1 つを選択しなければならない場合には、選択項目の 1 つが主要パスに書かれます。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。

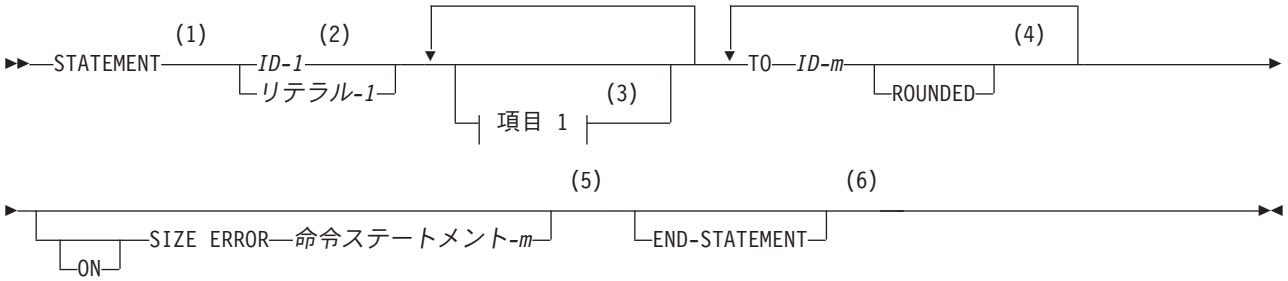
- 必要項目またはオプション項目の上の繰り返しの矢印は、積み重なった項目の中から複数選択することができるか、または 1 つの選択項目を繰り返すことができることを示します。



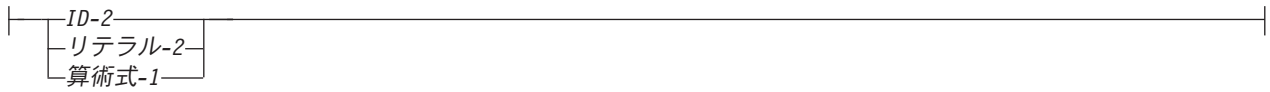
- 2 つの垂直線の間にある 1 つの項目は、構文の一部であり、図の中の別の場所に詳細が定義されています。

以下の例は、構文図の規則の使用方法を示しています。

フォーマット



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 の部分はオプショナルです。アプリケーションに応じて、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には繰り返して指定することができ、各項目の間は、1 つまたは複数の COBOL 分離文字で区切ります。このオペランドに選択可能な項目は、図の最下部に説明してあります。

4 オペランドの ID-m および関連した TO キーワードは必要で、繰り返す場合には各記入項目を COBOL 分離文字で区切ります。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に関連した ON SIZE ERROR 句はオプショナルです。 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、必要な分離文字ではありません。

IBM 拡張

IBM 拡張は、一般的にその直前の規則または制約事項を変更するものです。プログラマーによっては IBM 拡張を採用しないで ILE COBOL 言語を使用することもあるため、最初に標準のものを説明します。その後に、IBM 拡張を**使用する**プログラマーのために、拡張について説明します。

図や表の中の IBM 拡張は、拡張として明示的に識別されていない場合は、枠に囲んで示されます。

構文図中に示されている文節とステートメントが ANSI X3.23b-1985 COBOL に対する ILE COBOL 言語の拡張である場合は、その旨が示されます。

IBM Extension

ANSI X3.23b-1985 COBOL に対する ILE COBOL 言語の拡張が説明文の一部にある場合は、この段落のように、その部分を「IBM 拡張」という見出しが付いた線で囲んであります。

End of IBM Extension

文書化の構文

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、構文図の中で脚注を付けて示されます。

使用されなくなる言語エレメント

使用されなくなる言語エレメントとは、X3.23b-1993 COBOL 規格の ILE COBOL 言語エレメントのうち、この規格の次回の改訂から削除されるものです。使用されなくなる言語エレメントに対しては、ILE COBOL コンパイラーによる構文検査だけが行われます。

DBCS 表記法

リテラル内の DBCS 文字ストリング、すなわちコメントは、シフトアウト (< で示される) 文字とシフトイン (> で示される) 文字で区切られます。これらの文字の EBCDIC コードは、それぞれ X'0E' および X'0F' です。2 バイト文字は D1D2D3 というように表され、DBCS リテラルは G"<D1D2D3>" というように表されます。APOST オプションを指定した場合は、代わりに単一引用符を使用できます。

工業規格

ILE COBOL は、標準 COBOL をサポートするように設計されています。標準 COBOL とは、American National Standard for Information Systems - Programming Language - COBOL, ANSI X3.23-1985, ISO 1989:1985 に定義されている COBOL プログラム言語のことで、下記の文書の内容にしたがって、リストされている順序に更新されたものです。

- 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
- ISO/IEC 1989 DAM2 Programming Languages - COBOL, AMENDMENT 2: Correction and clarification amendment for COBOL.
- *FIPS Publication 21-4, Federal Information Processing Standard 21-4, COBOL*

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。

本書の原典の一部は、「*X3.23b-1993, American National Standard for Information Systems - Programming Language - COBOL*」および「*ISO 1989:1985/Amd 2:1994, Programming languages - COBOL*」からのコピーであり、これらの資料 (米国規格協会が 1985 年の著作権を持つ) から許可を得て複製したものです。これらの資料は、American National Standard Institute (1430 Broadway, New York, New York, 10018) より購入することができます。

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

ILE COBOL コンパイラーでサポートされる業界標準について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の付録 A 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。

> どんな組織であっても、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 の保守に関する手続き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。

概念

ILE COBOL 言語の概念に関するヘルプを表示するには、以下のトピックからいずれかを選択してください。

- 文字、文字ストリング、語、リテラル、コメント、および分離文字
- セクションおよび段落
- COBOL ソース行の参照形式 (シーケンス番号域、標識域、区域 A、区域 B、およびコメント域)
- データ参照および手続き参照の方式
- COBOL 名のタイプおよびそれらの有効範囲
- ステートメント間におけるプログラムの流れの制御権移動
- ユーザー定義データ・タイプ
- サポート情報

サポート情報

このセクションでは、以下のサポート情報について説明します。

- ILE COBOL コンパイラー限界値
- 中間結果と算術精度
- EBCDIC、および ASCII 照合順序
- ILE COBOL 関数名およびコンテキストに依存した語のリスト
- ILE COBOL 予約語リスト
- 属性データ形式
- ACCEPT/DISPLAY および COBOL/2 に関する考慮事項
- 参考文献

COBOL 言語の構成

文字

COBOL で、分割不可能なデータの単位は**文字**です。 COBOL 文字セットを形成する英数字、および特殊文字は表 2-1 に示されています。

Integrated Language Environment* (ILE*) COBOL * 用言語では、文字は、定義された文字セットに制限されます。 データ項目に入る非数字リテラル、コメント行、コメント記入項目および値には、(1 次ソース・ファイルの code character set identifier (CCSID) によって) 現在システムに指定されている文字セットの文字を含めることができます。

IBM Extension

2 バイト文字セット (DBCS) の文字は、特定の COBOL 文字ストリングで有効な文字です。 2 バイト文字は、1 文字を表示するために 2 つの隣接するバイトを占有します。 (詳細は 2-2 ページの『文字ストリング』の DBCS の情報を参照してください。)

End of IBM Extension

個々の文字を組み合わせて、文字ストリング、分離文字、およびテキスト語を構成します。

文字ストリングは、 COBOL 語、 リテラル、PICTURE 文字ストリング、またはコメントを形成する 1 文字または連続する文字です。 文字ストリングは、分離文字によって区切られます。

分離文字は、文字ストリングを区切るために使用される 1 つまたは 2 つの連続する句読文字のストリングです。分離文字については 2-18 ページの『分離文字』に詳しく説明してあります。

テキスト語は、COBOL のライブラリー、ソース・プログラム、または疑似テキストの中のマージン A (桁 7 ～ 桁 8) からマージン R (桁 72 ～ 桁 73) までの間に入っている 1 つの文字、または連続する文字の順序列であり、以下のもののうちのいずれかです。

- スペース、疑似テキスト分離文字、および非数字リテラル分離文字以外の分離文字
- 必要な分離文字を含む、リテラル
- 分離文字によって結合された連続する COBOL 文字のあらゆる順序列のうち、コメント行と語 COPY 以外のもの

表 2-1. COBOL 文字 — その意味と用法

文字	意味	使用法
A–Z	アルファベット (大文字)	英字
a–z	アルファベット (小文字)	英字
0–9	アラビア数字 (数字)	数字
	スペース	句読文字
.	小数点またはピリオド	編集用文字 句読文字
<	より小さい	比較文字

表 2-1. COBOL 文字 — その意味と用法 (続き)

文字	意味	使用法
(	左括弧	句読文字
+	プラス記号	算術演算子 編集用文字
$	円記号	編集用文字
*	アスタリスク	算術演算子 編集用文字 コメント文字
)	右括弧	句読文字
;	セミコロン	句読文字
:	コロン	句読文字
-	マイナス記号またはハイフン	算術演算子 編集用文字 編集用文字 継続文字 COBOL 語のエレメント
_	下線	ユーザー定義語のエレメント
/	ストロークまたは斜線	算術演算子 編集用文字 編集用文字 継続文字
,	コンマ	編集用文字 句読文字
>	より大きい	比較文字
=	等号	句読文字 比較文字
"	引用符	句読文字
'	アポストロフィ	句読文字

注:

1. アポストロフィ (') および下線 (_) 文字は IBM 拡張です。
2. システム・オブジェクトの名前を定義するために使用する非数値リテラル、または日時形式では、他の特定の文字が必要な場合があります。

   文字 # および @ は、IBM i システム名では有効なエレメントです。

   文字 @ および % は、日時形式の定義時に使用できる変換指定子です。

文字ストリング

以下のものを形成するには、1 バイトの文字ストリングを使用できます。

- DBCS 文字による COBOL 語
- COBOL 語
- リテラル
- PICTURE 文字ストリング
- コメント記入項目テキスト

IBM Extension

以下のものを形成するには、ユーザー定義の DBCS ワードを含む文字ストリングを使用できます。

- リテラル
- コメント記入項目テキスト

DBCS 文字ストリングは、2 バイト文字セットの文字を使用して組み立てられます。混合リテラルを含め、DBCS 文字ストリングを非数字ストリングに組み込むことができます。組み込み DBCS 文字ストリングは、シフトアウト制御文字で開始し、シフトイン制御文字で終了することによって範囲を限定する必要があります。 DBCS リテラルの両方のバイトには、X'00' から X'FE' の範囲の文字が含まれます。 DBCS リテラルに X'0F7F' (APOST オプションが選択されている場合は X'0F7D') を含めることはできません。

End of IBM Extension

DBCS 文字による COBOL 語

以下は、DBCS 文字からユーザー定義語を形成するための規則です。

シフトアウト、シフトイン文字の使用

DBCS ユーザー定義語は、シフトアウト文字で始まり、シフトイン文字で終わります。

値の範囲

DBCS ユーザー定義語は、両方のバイトについて 16 進数 41 から 16 進数 FE までの値の範囲をとる文字を含むことができます。

含まれる文字

DBCS ユーザー定義語は、DBCS 文字のみを含むことができます。 DBCS ユーザー定義語は、単一バイトの EBCDIC 文字に相当する文字と単一バイトの EBCDIC 文字に相当しない文字とを含むことができます。単一バイトの EBCDIC 文字に相当する DBCS 文字は、COBOL ユーザー定義語についての通常の規則に従います。つまり、文字 A から Z、 a から z、0 から 9、およびハイフン (-) は許可されます。ハイフンは、先頭文字または最後の文字として使用することはできません。 単一バイトの EBCDIC 文字に相当するものを持たない DBCS 文字は、DBCS ユーザー定義語で使用できます。

連結規則

DBCS ワードは、2 行にわたってはなりません。

最大長 14 DBCS 文字。

ユーザー定義語

次のユーザー定義語のタイプは、ILE COBOL でサポートされています。右側の欄は、DBCS 文字がそのタイプで許可されているかを示しています。

ユーザー定義語のタイプ	DBCS 文字を許可
英字名	Yes
(データの) クラス名	Yes
条件名	Yes
データ名	Yes
ファイル名	Yes
指標名	Yes
ライブラリー名	No
簡略名	Yes

段落名	Yes
プログラム名	No
レコード名	Yes
セクション名	Yes
テキスト名	No

注: DBCS 文字でユーザー定義の COBOL 語を使用するためには、PROCESS オプション GRAPHIC が有効でなければなりません。その他の場合は、DBCS ワードは無効文字として処理されます。

COBOL 語

COBOL 語は、文字、数字、ハイフン、および下線のセットから成る文字ストリングでなければなりません。(ただし、ハイフンおよび下線を先頭文字または最後の文字として使用することはできません。) ILE COBOL 言語では、通常、それぞれの小文字は対応する大文字と等価です。

COBOL 語には、次の 5 つのタイプがあります。

- ユーザー定義語
- システム名
- 関数名

IBM Extension

- コンテキストに依存した語

End of IBM Extension

- 予約語

ソース・プログラム内に含まれる特殊文字語ではないすべての COBOL 語には次の規則が適用されます。

- COBOL 語の最大長は 30 文字です。
- LENGTH、RANDOM、SUM、および WHEN-COMPILED を除く予約語は、ユーザー定義語、システム名、コンテキストに依存した語、または関数名として使用できません。
- ただし同じ COBOL 語が、次のタイプのうちの複数の ILE COBOL 語として使用されることがあります。
  - ユーザー定義語
  - システム名
  - 関数名

  IBM Extension

  - コンテキストに依存した語

  End of IBM Extension

  そのような COBOL 語の特定のオカレンスの種別は、それが出現する句のコンテキストによって判断します。

ユーザー定義語: 以下に、ユーザー定義語と、その構成時に従うべき規則をリストします。

ユーザー定義語のタイプ	一般規則
英字名 クラス名 条件名 定数名 データ名 ファイル名 指標名 ロケール名 簡略名 プログラム名 (含まれているプログラム および最外部プログラムは PROCEDURE リンケージ規則 を使って呼び出される) レコード名 ルーチン名 タイプ名	各語には、最低 1 つの英字が含まれていなければならない。
ライブラリー名 プログラム名 (最外部プログラムは PROGRAM リンケージ規則 を使って呼び出される) テキスト名	各語には、最低 1 つの英字が含まれていなければならない。最初の 10 文字は固有でなければならない。
段落名セクション名	各語には、英字が含まれている必要は**ない**。
レベル番号: 01 ～ 49、66、77、88 セグメント番号: 00 ～ 99	各語は 1 桁または 2 桁の整数でなければならないが、固有である必要はない。 (セグメント情報は構文検査だけ実行される。)

各ユーザー定義語の機能は、それが現れる文節またはステートメントで説明してあります。

ユーザー定義語を参照する際の制約: 通常、ユーザー定義語は、上記のテーブルにリストしたタイプのいずれか 1 つに属していなければなりません。さらに、ユーザー定義語は、属するタイプの中で固有でなければなりません。

一般規則には、以下の 2 つの例外があります。

- レベル番号またはセグメント番号は、固有である必要はありません。レベル番号またはセグメント番号に使用するユーザー定義語は、別のレベル番号またはセグメント番号のユーザー定義語と同じでもかまいません。
- 参照の固有性が保持できれば、以下のいずれか *1* つ で、1 つのユーザー定義語を重複して使用することができます。
  - 条件名、データ名、およびレコード名からなるグループ
  - 段落名
  - テキスト名

  このような名前の参照の固有性の確保について詳しくは 2-27 ページの『データ参照の方法』を参照してください。

ユーザー定義語が宣言されているプログラムのステートメントおよび記入項目は、次のユーザー定義語のタイプを参照できます。

- 段落名
- セクション名

次のユーザー定義語のタイプは、どの COBOL プログラムでも参照できます。

- ライブラリー名
- プログラム名
- テキスト名

次のタイプの名前は、構成セクションで宣言されていれば、構成セクションを含んだプログラム、またはそのプログラム内に含まれるすべてのプログラムのいずれかにあるステートメントおよび記入項目から、参照できます。

- 英字名
- クラス名
- 条件名
- 簡略名

システム名: **システム名**は、 システムにとって特定の意味を持つように IBM によって定義された文字ストリングです。システム名には次の 4 つのタイプがあります。

- コンピューター名
- 言語名
- 実装者名 (環境名および割り当て名を含む)
- ロケール名

コンピューター名は、DBCS 文字で書くことができますが、その他のシステム名は書くことができません。

関数名: **関数名**とは、COBOL ソース・プログラムで使用される、一連の指定された語の 1 つです。

関数名は、ILE COBOL によって提供される、組み込み関数の値を判別するためのメカニズムを指定します。

語 LENGTH、RANDOM、SUM、および WHEN-COMPILED の場合を除き、別のコンテキストでは関数名として使用されている語をプログラムの中でユーザー定義語、システム名、またはコンテキストに依存した語として使用できます。

コンテキストに依存した語:

IBM Extension

コンテキストに依存した語は、予約語の規則に従って形成された COBOL 語で、一般形式で指定されたように使用できます。同じ語が、関数名、ユーザー定義語、またはシステム名として使用されることがあります。

ILE COBOL のコンテキストに依存した語は 9-15 ページの『付録 D. ILE COBOL 関数名およびコンテキストに依存した語のリスト』にリストされています。

End of IBM Extension

予約語: **予約語**は、COBOL ソース・プログラムで事前定義した意味を持つ文字ストリングのことで、言語定義形式の指定のとおりにのみ使用できます。

ILE COBOL の予約語は 9-17 ページの『付録 E. ILE COBOL 予約語リスト』にリストしてあります。

予約語には、次の 5 つのタイプがあります。

- キーワード
- オプショナル語
- 特殊文字語
- 表意定数
- 特殊レジスター

キーワード: **キーワード**とは、特定の文節、 記入項目、または ステートメント内で必要な予約語のことです。

オプショナル語: **オプショナル語**とは、読みやすくするために文節、記入項目、またはステートメントの形式の中に含められる予約語のことです。それらはプログラムの意味や実行に影響を与えることはありません。 オプショナル語は形式内では大文字で表されますが、主パスの下部に書かれます。

特殊文字語: **特殊文字**語には、次の 2 つのタイプがあります。

- **算術演算子**: + - / * **

  7-9 ページの『算術演算子』を参照してください。

- **比較演算子**: < > = <= >=

  7-14 ページの『比較条件』を参照してください。

表意定数: **表意定数**は、特定の定数値を指定および参照する予約済みの語です。表意定数のための予約語とその意味を次に示します。

ZERO/ZEROS/ZEROES

コンテキストに従って、次の 1 つを表します。

- 数字のゼロ (0)
- 非数字のゼロ (0) の 1 つ以上のオカレンス

IBM Extension

- ブール値 B"0"

End of IBM Extension

SPACE/SPACES

1 つまたは複数のブランクやスペースを表しますが、これは非数字リテラルとして扱われます。 SPACES は、DBCS データ項目に対して使用した場合、1 つまたは複数の 2 バイト・スペースを表します。 SPACES は、国別データ項目に対して使用した場合、1 つまたは複数の単一バイト UCS-2 スペースを表します。

HIGH-VALUE/HIGH-VALUES

使用する照合順序の中で最高の順位を持つ文字が 1 つまたは複数あることを表します。 NATIVE および EBCDIC の照合順序では、その文字は X'FF' になり、STANDARD-1 および STANDARD-2 の照

合順序では、その文字は X'07' になります。それ以外の照合順序の場合、実際に使用される文字は、照合順序に依存します。 HIGH-VALUE は非数字リテラルとして扱われます。

LOW-VALUE/LOW-VALUES
使用する照合順序の中で最低の順序を持つ文字が 1 つまたは複数あることを表します。
NATIVE、EBCDIC、STANDARD-1、および STANDARD-2 の照合順序では、その文字は X'00' となり、それ以外の照合順序の場合、実際に使用される文字は、照合順序に依存します。 LOW-VALUE は非数字リテラルとして扱われます。

QUOTE/QUOTES
1 つまたは複数の引用符文字があることを表し、非数字でなければなりません。非数字リテラルを囲むために、QUOTE または QUOTES を引用符やアポストロフィの代わりに用いることはできません。

IBM Extension

APOST がコンパイラー・オプションとして指定されるとき、表意定数 QUOTE にはアポストロフィの EBCDIC 値が含まれます。

End of IBM Extension

ALL リテラル
リテラルを構成する文字ストリングが 1 回または複数あることを表します。リテラルは、ALL リテラル以外の非数字リテラルまたは表意定数でなければなりません。

ALL リテラル以外の表意定数を使用する場合には、ALL という語は余分であり、読みやすくするだけの目的で使用します。 INSPECT、STOP、または STRING ステートメントの中で表意定数の ALL リテラルを使用してはなりません。

注: 表意定数 ALL リテラルが、数字または数字編集項目に関連した場合や、リテラルの長さが 1 より大きい場合、それは古くなったエレメントであり、ANSI 規格の次の改訂から削除されます。

IBM Extension

ALL リテラルに使用できるリテラルは、ブール・リテラル、DBCS リテラル、または国別リテラルです。

End of IBM Extension

IBM Extension

NULL/NULLS
USAGE IS POINTER 文節、USAGE IS PROCEDURE-POINTER 文節、ADDRESS OF 句、または ADDRESS OF 特殊レジスターにより定義されたデータ項目には、有効なアドレスが含まれていないことを示すために使用される値を表します。 NULL は、構文形式で明示的に許可されている場合にだけ使用できます。

ILE COBOL 言語では、NULL の値は定義されていません。

End of IBM Extension

ZERO、SPACE、HIGH-VALUE、LOW-VALUE、QUOTE、および NULL の単数形と複数形は等価なので、いずれを使用してもかまいません。例えば、DATA-NAME-1 が 5 文字のデータ項目の場合、次のステートメントはどれも、DATA-NAME-1 に 5 桁のスペースを埋めます。

```
MOVE SPACE TO DATA-NAME-1
MOVE SPACES TO DATA-NAME-1
MOVE ALL SPACES TO DATA-NAME-1
```

「リテラル」が形式の中のどこにあっても、明示的に禁止されている場合以外は、表意定数を使用できます。形式の中に数字リテラルがある場合には、使用できる表意定数は ZERO だけです。表意定数は、それがある関数への引数となっている算術式の中以外においては、関数の引数として使用できません。

IBM Extension

表意定数 ZERO はブール・リテラルとして使用できます。

End of IBM Extension

表意定数の長さは、プログラムのコンテキストによって決まります。次の規則が適用されます。

- 表意定数がデータ項目と関連している場合 (例えば、表意定数を他の項目へ移動するか、他の項目と比較する場合)、表意定数の文字ストリングの長さは 1 になるか、または関連したデータ項目内の文字位置の数 (いずれか大きい方) に等しくなります。
- ALL リテラル以外の表意定数が別のデータ項目と関連していない場合 (例えば、STOP、STRING、または UNSTRING ステートメントで使用する場合)、文字ストリングの長さは 1 文字です。

特殊レジスター: **特殊レジスター**とは、コンパイラーにより生成される記憶域に名前を付ける予約語のことです。その主な用途は、特定の COBOL 機能が作成した情報を保管することです。このような記憶域はおのおのが固定の名前をもっているので、プログラムの中で改めて定義する必要はありません。

この指定の一般的な形式では、他に制限がなければ、データ名または ID が指定されているところではどこでも、特殊レジスターを使用できます。ただし、特殊レジスターはデータ名または ID と同一のカテゴリーであることが条件です。修飾が許可されている場合は、特殊レジスターに固有性を与えるために修飾できます。

プログラムの制御が、CALL ステートメントによって、実行単位内であるプログラムから別のプログラムに最初に移されると、コンパイラーは、特殊レジスター・フィールドを初期値に設定します。
RETURN-CODE および SORT-RETURN 特殊レジスターは、以下の場合、初期値にリセットされます。

- 参照されたサブプログラムを初期化するために、CANCEL ステートメントが呼び出された場合
- INITIAL 属性を処理するプログラムの場合
- RECURSIVE 属性を処理するプログラムの場合

その他のすべての場合、特殊レジスターが初期値にリセットされることはありません。特殊レジスターは、前回に、プログラム制御が CALL ステートメントによって移されたときに保存された値のままです。

英数字の引数が許可されている場合は、特に禁止されていない限り、英数字レジスターを関数で指定できます。

英数字の引数が許可されている場合は、特に禁止されていない限り、英数字特殊レジスターを関数で指定できます。

それぞれの特殊レジスターについては、次に示されたページから始まるセクションに説明されています。

特殊レジスター
ページ

DEBUG-ITEM
このレジスターは構文検査だけ実行されます。

LINAGE-COUNTER
6-28 ページの『LINAGE-COUNTER 特殊レジスター』

IBM Extension

ADDRESS OF
6-5 ページの『ADDRESS OF 特殊レジスター』

DB-FORMAT-NAME
7-40 ページの『DB-FORMAT-NAME 特殊レジスター』

LENGTH OF
7-78 ページの『LENGTH OF 特殊レジスター』

LOCALE OF
6-41 ページの『LOCALE OF 特殊レジスター』

FORMAT OF
6-42 ページの『FORMAT OF 特殊レジスター』

RETURN-CODE
7-227 ページの『RETURN-CODE 特殊レジスター』

SORT-RETURN
7-138 ページの『SORT-RETURN 特殊レジスター』

WHEN-COMPILED
7-146 ページの『WHEN-COMPILED 特殊レジスター』

XML-CODE
7-270 ページの『XML-CODE 特殊レジスター』

XML-EVENT
7-271 ページの『XML-EVENT 特殊レジスター』

XML-NTEXT
7-272 ページの『XML-NTEXT 特殊レジスター』

XML-TEXT
7-273 ページの『XML-TEXT 特殊レジスター』

End of IBM Extension

リテラル

リテラルは文字ストリングであり、構成されている文字または表意定数の使用のいずれかによって、その値が指定されます (2-7 ページの『表意定数』を参照してください)。リテラルには、次の 5 つのタイプがあります。

IBM Extension

- ブール

- DBCS
- 国別

End of IBM Extension

- 非数字
- 数字

ブール・リテラル:

IBM Extension

ブール・リテラルとは、左側を分離文字 **B"** で、右側を引用符の分離文字で区切る文字ストリングのことです。この文字ストリングは文字 0 または 1 だけで構成されます。ブール・リテラルの値は、囲んでいる分離文字を除いた文字そのものです。

End of IBM Extension

DBCS リテラル:

IBM Extension

DBCS リテラルには、以下の形式があります。

フォーマット

フォーマット

G" または N"
DBCS リテラルの開始分離文字

" DBCS リテラルの終了分離文字

NATIONALPICNLIT PROCESS オプションが有効な場合、開始区切り文字 N" または N' が国別リテラルを識別し、 2-13 ページの『基本国別リテラル』に示されている規則が適用されます。

一般に、非数字リテラルに関係する規則は、DBCS リテラルにもあてはまります。ただし、DBCS リテラルの最大長は 2 バイト文字で 28 文字であり、これらは 2 行に渡って続けることはできません。

DBCS リテラルは、データ部で次のように指定できます。

- DBCS データ記述記入項目の VALUE 文節内で指定する。データ項目の VALUE 文節内に DBCS を指定した場合、リテラルの長さが、データ項目の PICTURE 文節で示されているサイズを超えてはなりません。 DBCS データ項目を USAGE DISPLAY-1 として明示的に定義すると、データ項目を 2 バイトにつき 1 文字ずつ文字形式で保管するよう指定されます。
- JUSTIFIED 文節を使用して指定する。

DBCS リテラルは、手続き部で次のように指定できます。

- DBCS またはグループ項目が受け入れ項目である場合、送り出し項目として指定する。
- 被比較数が DBCS またはグループ項目である場合、関係条件において指定する。
- DBCS リテラルの前の表意定数 SPACE/SPACES、ALL SPACE/SPACES、または ALL として指定する。上記の表意定数だけが DBCS リテラルになることができます。
- DBCS をサポートする組み込み関数への引数として指定する。

DBCS リテラルは、非数字リテラルを使用できる個所ならどこでも指定できます。ただし、次の個所でリテラルとして指定する場合は**例外です**。

- 見出し部
  - PROGRAM-ID 段落
- 環境部 (ENVIRONMENT DIVISION)
  - ALPHABET 文節
  - ASSIGN 文節
  - CLASS 文節
  - CURRENCY SIGN 文節
  - LINKAGE 文節
  - PADDING CHARACTER 文節
  - RERUN 文節
- 手続き部 (PROCEDURE DIVISION)
  - CALL ステートメント (プログラム名)
  - CANCEL ステートメント
  - END PROGRAM ヘッダー
  - STOP ステートメント
  - DROP ステートメント
  - ACQUIRE ステートメント
- COPY
  - COPY ステートメント (テキスト名)
  - COPY ステートメント (ライブラリー名)

End of IBM Extension

国別リテラル:

IBM Extension

国別リテラルは、以下の場所で指定できます。

- 国別データ記述記入項目の VALUE 文節内で
- 手続き部内の送り出し項目として。詳細については 7-139 ページの『MOVE ステートメント』を参照してください。
- 関係条件のオペランドとして
- 国別データをサポートする組み込み関数への引数として

ILE COBOL には、以下の 2 つの国別リテラル形式が用意されています。

- 基本国別リテラル
- 国別 16 進リテラル

表意定数 SPACE/SPACES、ALL SPACE/SPACES は、国別リテラルにすることができます。 SPACE は単一バイトの UCS-2 スペース (NX"0020") です。

基本国別リテラル:

IBM Extension

NATIONALPICNLIT PROCESS オプションが有効な場合、開始区切り文字 N" または N' は国別リテラルを識別します。国別リテラルは、国別のクラスおよびカテゴリーに属します。

NATIONALPICNLIT PROCESS オプションが有効ではない場合、開始区切り文字 N" または N' は DBCS リテラルを識別し、 2-11 ページの『DBCS リテラル』に示されている規則が適用されます。

基本国別リテラルには、以下の形式があります。

フォーマット

N" 国別リテラルの開始分離文字

文字データ
国別リテラルの内容のソース・テキスト表現です。文字データ は、EBCDIC 文字セットからの許容される任意の文字を含むことができます。

" 国別リテラルの終了分離文字

囲みの引用符 (またはアポストロフィ) は、プログラムをコンパイルするときにリテラルから除外されます。引用符を非数字リテラルに組み込むには、2 個の引用符を連続して ("") 指定しなければなりません。

*APOST コンパイラー・オプションが有効な場合には、国別リテラルをアポストロフィ (') で囲まなければなりません。

開始区切り文字で使用されている引用符またはアポストロフィをリテラルの内容に含める場合は、引用符またはアポストロフィをそれぞれ 2 つ続けて指定します。以下に例を示します。

```
N'This literal''s content includes an apostrophe'
N'This literal includes ", which is not used in the opening delimiter'
N"This literal includes "", which is used in the opening delimiter"
```

基本国別リテラルの最大長は 256 文字 (開始と終了の区切り文字を除く) です。リテラルには、1 つ以上の文字が含まれていなければなりません。

文字データのソース・テキスト表現は、実行時に使用するために有効な国別 CCSID に自動的に変換されます (例えば、リテラルが国別カテゴリーのデータ項目に移動されるときや、そのような項目と比較されるとき)。

基本国別リテラルの後に続く ALL は、基本国別リテラルです。

End of IBM Extension

国別 16 進リテラル:

IBM Extension

国別 16 進リテラルには、以下の形式があります。

フォーマット

NX"
国別 16 進リテラルの開始分離文字

" 国別 16 進リテラルの終了分離文字

16 進文字列は、汎用文字セットのバージョン 2 (UCS-2) つまりユニコード文字にマップされる、4 つの 16 進数字のグループから構成されます。

国別 16 進リテラルの最大長は 512 国別文字です。

国別 16 進リテラルが後に続く ALL は、国別 16 進リテラルです。

End of IBM Extension

End of IBM Extension

非数字リテラル: 非数字リテラルとは、引用符 (") で囲まれた文字ストリングであり、EBCDIC 文字セットで許容される任意の文字を含むことができます。非数字リテラルの最大長は 256 文字です。

非数字リテラルは、引用符 (") で囲まなければなりません。

*APOST コンパイラー・オプションが有効な場合には、非数字リテラルをアポストロフィ (') で囲まなければなりません。

囲みの引用符 (またはアポストロフィ) は、プログラムをコンパイルするときにリテラルから除外されます。引用符を非数字リテラルに組み込むには、2 個の引用符を連続して ("") 指定しなければなりません。

例えば、

```
"THIS ISN""T WRONG"
```

IBM Extension

アポストロフィ・リテラルでは、2 重アポストロフィ (") が分離文字である場合は、単一のアポストロフィに変えられます。

例えば、

```
'THIS ISN''T WRONG'
```

は、次のものを表します。

```
THIS ISN'T WRONG
```

End of IBM Extension

非数字リテラル内で使用されるすべての句読文字は、リテラルの値の一部です。

すべての非数字リテラルは英数字データのカテゴリーに入ります。 (データのカテゴリーについては 6-9 ページの『データのクラスおよびカテゴリー』で説明します。)

16 進数リテラル:

IBM Extension

16 進数の表記を使用して **16 進非数字リテラル**を構成することができます。

フォーマット

X" 非数字リテラルの 16 進表記用左分離文字。 (コンパイラー・オプション *APOST または PROCESS ステートメントのオプション APOST が指定される場合、左分離文字は X' です。)

" 非数字リテラルの 16 進表記用右分離文字。 (コンパイラー・オプション *APOST または PROCESS ステートメントのオプション APOST が指定される場合、右分離文字は ' です。)

16 進数値は、0 ～ 9、a ～ f、および A ～ F のいずれかの範囲の文字から成ります。 16 進数は 2 桁で 1 文字を表しますから、16 進数は偶数ごとに指定しなければなりません。

16 進非数字リテラルの最大長は 512 桁です。

継続規則は、他の非数字リテラルに関するものと同じです。

コンパイラーは、16 進リテラルを通常の非数字リテラルに変換します。 16 進非数字リテラルは、非数字リテラルを使用できるところであればどこででも使用できます。

End of IBM Extension

混合リテラル:

IBM Extension

混合リテラルは、1 バイト文字と 2 バイト文字を組み合わせた非数字リテラルです。 2 バイト文字の各ストリングは、シフトアウト制御文字 (16 進数の 0E) で開始し、シフトイン制御文字 (16 進数の 0F) で終了させることによって範囲が限定され、単一バイト・データと区別されます。制御文字は混合リテラルの長さに含まれます。 2 バイト文字ストリングは、2 つの制御文字だけから構成される場合もあります。

COBOL ステートメントは、マシン表示に依存することなく、混合リテラルを処理します。バイト単位で動作するステートメント (例えば、STRING および UNSTRING) は、結果的に 1 バイト文字と 2 バイト文字の有効な混合文字ストリングとならない場合があります。ステートメントが正しく使用されていることを確認するのは、ユーザーの責任です。

混合リテラルは、プログラムが PROCESS ステートメントの GRAPHIC オプションを使用してコンパイルされるような場合にのみ認識されます。それ以外の場合には、単に非数字リテラルとして処理されます。

End of IBM Extension

ヌル終了非数値リテラル:

IBM Extension

非数値リテラルを以下の形式で、ヌル終了することができます。

フォーマット

Z" 非数値リテラルのヌル終了表記用の開始区切り文字。 (コンパイラー・オプション *APOST または PROCESS ステートメントのオプション APOST が指定される場合、左分離文字は Z' です。)ヌル終了リテラル (Z" または Z') の開始区切り文字の両方の文字は、同じソース行になければなりません。

" 非数値リテラルのヌル終了表記用の終了区切り文字。 (コンパイラー・オプション *APOST または PROCESS ステートメントのオプション APOST が指定される場合、右分離文字は ' です。)

開始区切り文字で引用符が使用されている場合、それを終了区切り文字として使用する必要があります。同様に、開始区切り文字でアポストロフィが使用されている場合、それを終了区切り文字として使用する必要があります。

リテラルのコンテントは、1 バイトまたは 2 バイト文字、またはその両方を含むことができますが、値 X'00' を持つ 1 バイト文字を指定することはできません。 X'00' は、リテラルの最後に自動的に追加されるヌル文字です。その他の点では、リテラルのコンテントは混合リテラルと同じ規則と制約事項に従います。

リテラル・コンテントの 1 バイトまたは 2 バイト文字のストリングは、0 から 255 バイトの長さにすることができます。リテラルの実際の長さは、終了ヌル文字を含むため、最大長は 256 バイトになります。

ヌル終了非数値リテラルには、データ・クラスおよびカテゴリー英数字があります。ヌル終了非数値リテラルは、非数値リテラルを指定できるすべてのところで使用できますが、ヌル終了リテラルは、すべてのリテラル表意定数でサポートされるわけではありません。

ヌル終了リテラルを使用して、COBOL プログラムで外部または内部オブジェクトの名前 (プログラム名、ロケール名、ライブラリー名、プロシージャー名など) を指定しないでください。指定した場合、コンパイラーは終了ヌル文字を文字 "0" で置換し、重大度 20 のエラー・メッセージが発行され、ユーザーにこの置換が通知されます。

LENGTH 組み込み関数は、ヌル終了リテラルに適用された場合、終了ヌルを含まないそれより前のリテラル内のバイト数を戻します。(LENGTH 特殊レジスターは、リテラル・オペランドをサポートしません。)

End of IBM Extension

数字リテラル: 数字リテラルとは、数字 0 ～ 9、符号文字 (+ または -)、および小数点の中から選択された文字で構成される文字ストリングのことです。このリテラルに小数点のないものが、整数です。(本書では、形式の中で使用される**整数**という語は、小数点を含まない数字リテラルを表します。コンテキストによっては、このリテラルは、負の値を持つことを許されないか、またはゼロであることを許されません。このような制約事項、および適用される可能性があるその他の制約事項については、形式の説明部分を参照してください)。次の規則が適用されます。

- (デフォルトの) コンパイラー・オプション *NOEXTEND または PROCESS ステートメント・オプション NOEXTEND が指定されているときには、1 ～ 18 桁が可能です。

IBM Extension

- 算術計算モード・コンパイラー・オプション *EXTEND31 または PROCESS ステートメント・オプション EXTEND31 が指定されているときには、1 から 31 桁が可能です。

End of IBM Extension

IBM Extension

- 算術計算モード・コンパイラー・オプション *EXTEND31FULL または PROCESS ステートメント・オプション EXTEND31FULL が指定されているときは、1 から 34 桁が可能です。

End of IBM Extension

IBM Extension

- 算術計算モード・コンパイラー・オプション *EXTEND63 または PROCESS ステートメント・オプション EXTEND63 が指定されているときは、1 から 63 桁が可能です。

End of IBM Extension

- 符号文字は 1 文字だけ使用可能です。符号文字を組み込むのは、リテラルの左端でなければなりません。符号文字のないリテラルは、値が正であると見なされます。
- 小数点は 1 文字だけ使用可能です。小数点が含まれる場合、その語は**仮想小数点**として扱われます (すなわち、リテラル内の文字位置を占めません)。小数点は、リテラルの右端を除いて任意の位置に使用できます。
- 引用符で数字リテラルを囲むと、コンパイラーはこのリテラルを非数字リテラルとして扱います。

数字リテラルの値は、リテラル内の文字によって表される代数的な量です。標準のデータ・フォーマットの文字の数字リテラルの大きさは、ユーザーが指定する桁数と同じです。

すべての数字リテラルは、数字データのカテゴリーに入ります。 (データのカテゴリーについては 6-9 ページの『データのクラスおよびカテゴリー』で説明します。)

浮動小数点リテラル:

IBM Extension

数字リテラルは、固定小数点数または浮動小数点数のいずれかにすることができます。**浮動小数点リテラル値**の規則は次のとおりです。

- 浮動小数点リテラルは、以下の形式で書かれます。

フォーマット

- 仮数と指数の前の符号はオプショナルです。符号を省略すると、コンパイラーは正数を想定します。
- 仮数には、1 から 16 桁までの数字が入ります。仮数には小数点が含まれる必要があります。
- 指数は E によって表示され、その後ろにオプショナルの符号と、1、2、または 3 桁の数字が続きます。
- 浮動小数点リテラル値の絶対値は、2.225073858507201E-308 から 1.797693134862315E+308 までである必要があります。この範囲外の値については、E レベルの診断が行われ、その値はそれぞれ 0 または 1.797693134862315E+308 に置き換えられます。

  注: MVS™ COBOL の範囲は 0.54E-78 から 0.72E+76 で、OS/2® および AIX® の範囲は 2.225E-308 から 1.798E+308 です。

浮動小数点リテラルは、クラスが数字で、カテゴリーが内部浮動小数点になります。一般的に浮動小数点リテラルは、10 進数値リテラルが使用できるところはどこでも使用できます。

End of IBM Extension

PICTURE 文字ストリング

PICTURE 文字ストリングは、通貨記号で組み立てられる特定の記号および COBOL 文字セットの中の文字の組み合わせで構成されています。

PICTURE 文字ストリングの指定の一部として書かれた句読文字は、どれも句読文字とは見なされず、PICTURE 文字ストリングの指定の中で使われている記号と見なされます。(PICTURE 文節の記号は 6-58 ページの表 6-5 に示されています)

コメント記入項目テキスト

コメントとは、EBCDIC 文字セットの任意の文字の組み合わせを含めることができる文字ストリングのことです。これは、プログラムの実行には影響を与えません。コメントには、次の 2 つの形があります。

コメント記入項目
この形式については 4-4 ページの『オプショナルの段落』で説明します。

コメント行
この形式については 2-25 ページの『コメント行』で説明します。

分離文字

分離文字は、1 字の句読文字または句読文字のストリングです。

次に、COBOL 分離文字とその意味についてリストします。

b (スペース)
スペース

, コンマ

. ピリオド

; セミコロン

(左括弧

) 右括弧

" 引用符

== 疑似テキスト分離文字

: コロン

IBM Extension

' アポストロフィ

B" ブール・リテラルの開始分離文字

X" 16 進非数字リテラルの開始分離文字

G" DBCS リテラルの開始分離文字

N" 以下の開始分離文字

- 国別リテラル (NATIONALPICNLIT PROCESS オプションが有効な場合)
- DBCS リテラル (その他の場合)

NX" 国別 16 進リテラルの開始分離文字

Z" ヌル終了非数値リテラル用の開始区切り文字

End of IBM Extension

分離文字の規則

以下の説明では、各分離文字は大括弧で囲まれています。分離文字または分離文字の一部としてスペースを使用する場合は、どこでも複数のスペースを使用できます。

スペース []

スペースは、次の場合を除いて分離文字の直前または直後に使用できます。

- 疑似テキストの開始分離文字 (先行スペースが必要な場合)。
- 引用符内 (または、APOST オプションが有効な場合、アポストロフィ内)。引用符の間のスペースは、分離文字でなく非数字リテラルの一部と見なされます。

ピリオド [.]、コンマ [,]、セミコロン [;]

分離文字ピリオド、分離文字コンマ、または分離文字セミコロンは、ピリオド、コンマ、またはセミコロンとそれに続くスペースから構成されます。分離文字ピリオドが使用できるのは、文の終わりを示すため、または形式で指定された方法による場合だけです。分離文字コンマおよび分離文字セミコロンは、区切りスペースが使用できる所にはどこでも使用できます。

- **見出し部**において、分離文字コンマと分離文字セミコロンは、コメント記入項目の中で使用できます。 各段落の終わりは、分離文字ピリオドでなければなりません。
- **環境部**においては、分離文字コンマまたは分離文字セミコロンによって文節および文節の中のオペランドを区切ることができます。SOURCE-COMPUTER、OBJECT-COMPUTER、SPECIAL-NAMES、および I-O-CONTROL の各段落の終わりは、分離文字ピリオドでなければなりません。
  FILE-CONTROL 段落において、各ファイル制御記入項目は、分離文字ピリオドで終わらなければなりません。
- **データ部**においては、分離文字コンマまたは分離文字セミコロンによって文節および文節の中のオペランドを区切ることができます。ファイル (FD) 記入項目、ソート・マージ・ファイル (SD) 記入項目、およびデータ記述記入項目は、それぞれ分離文字ピリオドで終わらなければなりません。
- **手続き部**においては、分離文字コンマまたは分離文字セミコロンによって文の中のステートメント、およびステートメントの中のオペランドを区切ることができます。各文および各プロシージャーは、分離文字ピリオドで終わらなければなりません。

分離文字

括弧[( ] . . . [ )]
疑似テキスト以外では、括弧は左右の対で使用しなければなりません。括弧は、添え字、一連の関数の引数、参照変更、算術式、および条件を区切ります。

引用符 [b"] . . . ["b]
始めの引用符は、スペースまたは左括弧の直後に書かなければなりません。終わりの引用符は、その直後に分離文字 (スペース、コンマ、セミコロン、ピリオド、または右括弧) を書かなければなりません。引用符は、対で使用しなければなりません。これにより、リテラルが次の行に続く場合を除いて、非数字リテラルをくくります (2-24 ページの『継続行』を参照してください)。

IBM Extension

*APOST コンパイラー・オプション、または APOST PROCESS オプションのもとでは、アポストロフィを引用符の代わりに使用できます。

End of IBM Extension

疑似テキスト分離文字 [b==]... リテラル-2 [==b]
始めの疑似テキストの分離文字は、直前にスペースがなければなりません。終わりの疑似テキストの分離文字は、直後に分離文字 (スペース、コンマ、セミコロン、またはピリオド) がなければなりません。疑似テキストの分離文字は、対で使用しなければなりません。これにより疑似テキストをくくります (8-2 ページの『COPY ステートメント』および 8-5 ページの『REPLACING 句』を参照してください。)

コロン [:]
コロンは分離文字で、一般形式で示される場合に必要となります。

IBM Extension

B" はブール・リテラルを記述するために使用する場合は分離文字です。 B は引用符の直前に書かなければなりません。

X" は 16 進数非数字リテラルを記述するために使用される場合は分離文字です。X は引用符の直前に書かなければなりません。

G" は DBCS リテラルを記述するために使用する場合は分離文字です。 G は引用符の直前に書かなければなりません。

| **N"** は、DBCS リテラルまたは国別リテラル (NATIONALPICNLIT PROCESS オプションが有効になっ
| ている場合) の記述に使用される場合の分離文字です。N は引用符の直前に書かなければなりません。

NX" は国別 16 進リテラルを記述するために使用される場合は分離文字です。NX は引用符の直前に書かなければなりません。

Z" はヌル終了非数値リテラルを記述するために使用する場合は分離文字です。Z は引用符の直前に書かなければなりません。

End of IBM Extension

注: PICTURE 文字ストリング、コメント文字ストリング、または非数字リテラルに含まれる句読文字は、どれも句読文字とは見なされず、むしろ文字ストリングまたは文字リテラルの部分と見なされます。

セクションおよび段落

セクションおよび段落はプログラムを定義します。これらは、文節とステートメントに分けられます。関連する規則に例外が明示されていない限り、必要な文節またはステートメントを、それぞれ形式に記された順序に従って書く必要があります。オプショナルの文節またはステートメントを使用する場合には、それらを形式に示された順序で書く必要があります。これらの規則は、コメントとして扱われる文節およびステートメントについても、適用されます。

文法上の階層は、次のようになります。

- 見出し部

  段落
  項目
  文節

- 環境部 (ENVIRONMENT DIVISION)

  セクション
  段落
  項目
  文節
  句

- データ部 (DATA DIVISION)

  セクション
  項目
  文節
  句

- 手続き部 (PROCEDURE DIVISION)

  セクション
  段落
  文
  ステートメント
  句

記入項目

記入項目とは、一連の文節で、分離文字ピリオドで終わるものです。

文節

文節とは、記入項目の属性を指定する、連続した COBOL 文字ストリングの順序付きセットです。

文

文とは、1 つまたは複数のステートメントが連続し、分離文字ピリオドで終わるものです。

ステートメント

ステートメントとは、COBOL verb とオペランドの有効な組み合わせのことです。ステートメントは、オブジェクト・プログラムが取るべき処置を示します。種々のステートメントの説明は、次に示す節にあります。

- 7-27 ページの『命令ステートメント』
- 7-29 ページの『条件ステートメント』
- 7-29 ページの『範囲区切りステートメント』
- 8-1 ページの『コンパイラー指示ステートメント』.

句

プログラムの文節またはステートメントは、それぞれ、**句**と呼ばれる小さい単位に細分できます。

参照形式

COBOL プログラムは、**必ず** COBOL 参照形式で書く必要があります。図 2-1 に、COBOL の 80 文字ソース行の参照形式を示します。

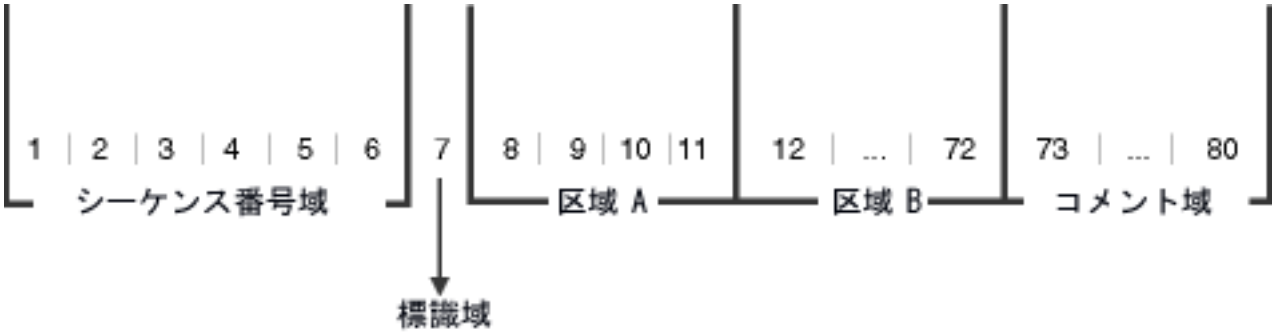


図 2-1. COBOL ソース行の参照形式

シーケンス番号域 (1 ～ 6 桁目)

シーケンス番号は、COBOL コンパイラーがコンパイルする各ステートメントを識別します。シーケンス番号の使用はオプショナルであり、コンピューターの文字セットの中の任意の文字で構成できます。シーケンス番号は、どんな順番でもよく、固有である必要はありません。

IBM Extension

SEQUENCE を指定することにより、コンパイル時に順序検査を行うことができます。

NUMBER オプションを指定した場合には、1 ～ 6 桁目のシーケンス番号が使用されますが、そうでない場合には、ソース・ファイルで与えられるソース・シーケンス番号が使用されます。

End of IBM Extension

標識域 (7 桁目)

標識域は、次のことを指定するために使用します。

- 直前の行から現在行までの行の継続 (2-24 ページの『継続行』の継続行に関する情報を参照)
- 文書またはコメントとしてのテキストの扱い (2-25 ページの『コメント行』のコメントに関する情報を参照)
- デバッグ行 (2-25 ページの『デバッグ行』のデバッグ行に関する情報を参照)

区域 A (8 ～ 11 桁目)

区域 A から始めなければならない項目を次に示します。

- 部のヘッダー
- セクション・ヘッダー
- 段落ヘッダーまたは段落名
- レベル標識 (FD および SD) またはレベル番号 (01 および 77)
- DECLARATIVES および END DECLARATIVES
- END PROGRAM ヘッダー

部のヘッダー

部のヘッダーとは、語の組み合わせの直後に分離文字ピリオドを書き、部の始まりを表すものです。

- IDENTIFICATION DIVISION.
- ENVIRONMENT DIVISION.
- DATA DIVISION.
- PROCEDURE DIVISION.

部のヘッダー (手続き部ヘッダーとともに USING 句が指定された場合を除く) の直後には分離文字ピリオドを書かなければなりません。 USING 句を除き、同一行上にテキストがあってはなりません。

セクション・ヘッダー

セクション・ヘッダーとは、一連の段落の始まり (環境部および手続き部内)、または項目の始まり (データ部内) のいずれかのことです。例えば、前者の場合は FILE-CONTROL があり、後者の場合は、FILE SECTION があります。

セクション・ヘッダーの直後には、手続き部のセグメント番号を指定する場合を除いて、ピリオドを書かなければなりません。(セグメント情報は構文検査だけ実行される。)

段落ヘッダーまたは段落名

段落ヘッダーまたは**段落名**は段落の始まりを意味します。環境部では、段落ヘッダーとその後に続く 1 つまたは複数の項目によって 1 つの段落が構成されます。手続き部では、段落名とそれに続く 1 つまたは複数の文によって 1 つの段落が構成されます。

レベル標識 (FD および SD) またはレベル番号 (01 および 77)

可能な**レベル標識**は FD または SD です。レベル標識は区域 A の中から書き始め、その後にスペースがなければなりません。(6-18 ページの『ファイル・セクション』を参照。) **レベル番号**は、区域 A から始めなければならず、01 または 77 の値を持つ、1 桁または 2 桁の整数です。詳細については 6-36 ページの『レベル番号』を参照してください。

DECLARATIVES および END DECLARATIVES

DECLARATIVES および END DECLARATIVES は、ソース・プログラムの宣言を開始および終了させるキーワードです。手続き部では、これらの語をそれぞれ区域 A から書き始め、その直後に分離文字ピリオドを続ける必要があります。同一の行にそれ以外のテキストを書くことはできません。END DECLARATIVES の後では、次のセクション・ヘッダーの前にテキストを書くことはできません。(7-6 ページの『宣言』を参照。)

END PROGRAM ヘッダー

END PROGRAM ヘッダーは、その後にプログラム名および分離文字ピリオドが続き、COBOL プログラムの終わりを表します。プログラム名は、対応する PROGRAM-ID 段落のプログラム名と同一でなければなりません。すべての COBOL プログラムはこのヘッダーで終わらなければなりません (ネストされたプログラムを含まず、一連の COBOL プログラム順序列の別の COBOL プログラムが後に続かない最外部プログラムを除く)。

区域 B (12 ～ 72 桁目)

次に示す項目は、区域 B から始めなければなりません。

- 記入項目
- 文
- ステートメント
- 文節
- 継続行

項目、文、ステートメント、文節

最初の記入項目、文、ステートメント、あるいは文節は、それらに従うヘッダーまたは段落名と同じ行、または次の非ブランク行でコメント行でない行の区域 B から始まります。後続の文または記入項目は、先行する文または記入項目の同一行の区域 B から始まるか、あるいは次の非ブランク行でコメント行でない行の区域 B から始まります。

記入項目や文の中で、区域 B にある後続行は同じ形式であっても、あるいはプログラム論理を明瞭にするために字下げしてもかまいません。出力リストは、入力ステートメントが字下げされている場合にだけ、字下げされます。字下げを行ってもプログラムの意味には影響ありませんが、どれだけ字下げするかは区域 B の幅の制約を受けます。 2-21 ページの『セクションおよび段落』も参照してください。

継続行

複数行必要な文、記入項目、文節、または句は、次に続くコメント行やブランク行でない行の区域 B に続けることができます。後に続く行を持つ行は**継続される行**と呼ばれ、後に続く方の行は**継続行**と呼ばれます。継続行の標識域にはハイフンが含まれている必要がありますが、区域 A はブランクでなければなりません。ハイフンがない場合は、前行の最終文字の後にスペースが 1 つあるものと見なされます。

ある行の標識域にハイフンがある場合には、その継続行の最初の非ブランク文字と継続される行の最後の非ブランク文字の間にはスペースがなく、すぐ後ろに続いているものと見なされます。

継続される行に終わりの引用符のない非数字リテラルがある場合には、継続される行の終わりのすべてのスペース (72 桁目まで) はそのリテラルの一部と見なされます。この継続行の標識域にはハイフンが必要であり、最初の非ブランク文字は引用符でなければなりません。リテラルの継続は、引用符のすぐ後の文字か

ら始まります。継続される行の最後の文字が 72 桁目にある単一引用符である場合、継続行にある最初の 2 つの非ブランク文字は、単一引用符を非数字リテラルの一部として示すために 2 つの連続した引用符でなければなりません。

疑似テキスト分離文字 (==) に関しては、分離文字を構成する 2 つの文字は同一行上になければなりません。

区域 A または区域 B

次の項目は、区域 A または区域 B のいずれからでも開始できます。

- コメント行
- デバッグ行
- ブランク行
- 疑似テキスト
- USE ステートメント以外のコンパイラー指示ステートメント

コメント行

コメント行とは、その行の標識域 (7 桁目) にアスタリスク (*) または斜線 (/) のある任意の行のことです。コメントは、行の区域 A および区域 B のどこにでも書くことができ、EBCDIC 文字セットの任意の文字の組み合わせから構成できます。コメント行は、見出し部ヘッダーの後なら、プログラムの中のどこにあってもかまいません。

複数のコメント行を書くことができます。それぞれの削除行の、標識域にアスタリスク (*) または斜線 (/) を書く必要があります。

アスタリスク (*) のコメント行は、出力リストの中で先行する最後の行にすぐ続いて印刷されます。斜線 (/) コメント行は次ページの最初の行に印刷され、出力リストの現行のページは排出されます。

コンパイラーはコメント行を文書として扱い、構文チェックは行いません。

IBM Extension

DBCS 文字はコメント行に組み込めますが、次行に続けることはできません。

End of IBM Extension

デバッグ行

デバッグ行とはその行の標識域に 'D' のある任意の行のことです。デバッグ行を書くことができるのは、環境部 (OBJECT-COMPUTER 段落の後)、データ部、および手続き部の中です。デバッグ行の区域 A と区域 B にスペースしかない場合は、ブランク行と見なされます。 5-2 ページの『WITH DEBUGGING MODE 文節』を参照してください。

ブランク行

ブランク行とは、7 ～ 72 桁目にスペースしかない行のことです。ブランク行は、プログラムの中のどこにあってもかまいません。

疑似テキスト

疑似テキストを構成する文字ストリングおよび分離文字は、区域 A または区域 B のいずれかで始めます。しかし、疑似テキスト開始分離文字に続く行の標識域 (7 桁目) にハイフンがある場合、その行の区域 A はブランクでなければならず、継続行の規則はテキスト語の形成に適用されます。

コンパイラー指示ステートメント

次の**コンパイラー指示**ステートメントは区域 A または区域 B のいずれからでも開始できます。

IBM Extension

- *CONTROL(*CBL)

End of IBM Extension

- COPY

IBM Extension

- EJECT

End of IBM Extension

- PROCESS
- REPLACE

IBM Extension

- SKIP1/2/3
- TITLE

End of IBM Extension

コメント域 (73 ～ 80 桁目)

コメント域は、ユーザーが自由に使用できます。例えば、自分のプログラムを識別するために使用します。

データ参照と名前の有効範囲

データ参照と名前の有効範囲の一般的な概念は、COBOL の構文を効率的にまた正しく使用するために重要です。名前の有効範囲は、COBOL プログラムでネストを使用する場合に特に大切です。

このセクションの最初の部分では、データ参照の次の 5 つの方法に注目します。

- 修飾
- 添え字付け
- 参照変更
- 関数 ID
- ユーザー定義データ・タイプ

このセクションの後半では、名前の有効範囲について主に説明します。

データ参照の方法

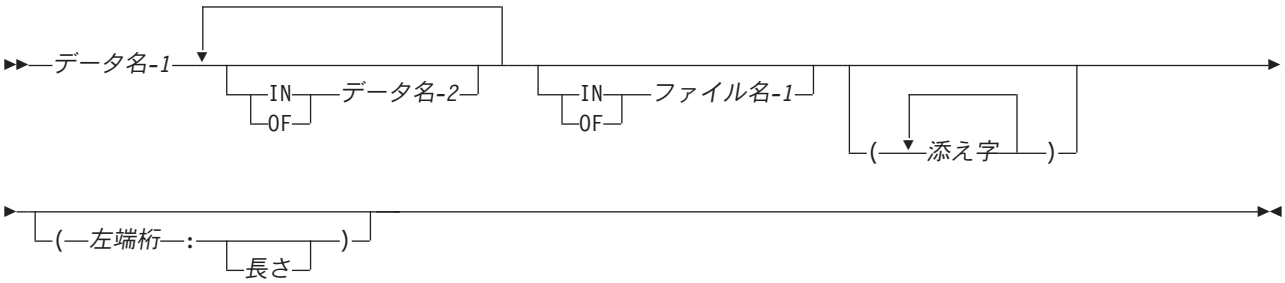
データおよびプロシージャーは、明示的または暗黙的のいずれかで参照できます。

COBOL プログラムのユーザー定義名はすべて、データ処理の問題を解決する目的でリソースに名前を付けるためのものです。リソースを使用するためには、COBOL プログラムのステートメントに、そのリソースを固有に識別する参照を書かなければなりません。参照の固有性を確実にするために、ユーザー定義名の修飾や添え字付けまたは参照変更を行うことができます。しかし、この点を考慮する前に、**ID** という用語を理解しておく必要があります。

ID

構文図で **ID** という語は、プログラムの中で固有でなければ参照の固有性のために必要な場合には、修飾子、添え字、または参照修飾子を構文上正しく組み合わせて、後に続けたユーザー定義名のことを指しています。

形式 1 - ID



データ名-1、データ名-2
レコード名とすることができます。

ファイル名-1
データ部の FD または SD 項目によって識別されなければなりません。

ファイル名-1 は固有に識別できなければなりません。

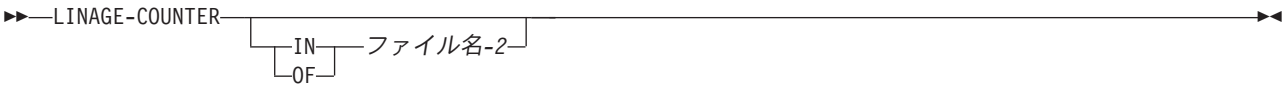
次の規則が適用されます。

- 修飾を行ってもデータ名が固有にならないところでは、データ名を重複して指定できません。
- 同じプログラムにおいて、レベル番号が 01 の 2 つの項目に対するデータ記述記入項目が同じデータ名を持つ場合、そのどちらの記入項目にも EXTERNAL 文節を適用できません。
- 同じデータ部の場合、同じデータ名を指定された 2 つのデータ項目のデータ記述記入項目に GLOBAL 文節を入れてはなりません。

ID には、**LINAGE-COUNTER** と**条件名**という 2 つの特殊なケースがあります。

LINAGE-COUNTER:

形式 2 - LINAGE-COUNTER



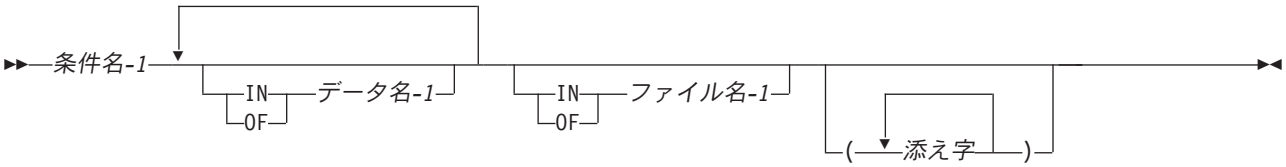
LINAGE-COUNTER
ソース・プログラム内に LINAGE 文節を含むファイル記述記入項目が複数指定されている場合は、参照のたびに修飾しなければなりません。

ファイル名-2
データ部の FD 記入項目によって識別されなければなりません。

ファイル名-2 は当該プログラムの中で固有のものでなければなりません。

条件名:

形式 3 - 条件名



条件名-1
環境部の構成セクションの中の、データ部または SPECIAL-NAMES 段落で定義できます。条件名が構成セクションで定義されている場合、構成セクションを含むプログラムまたはネストしたプログラムの中で参照できます。条件名がデータ部で定義されている場合、グローバル名またはローカル名の有効範囲規則に従って参照できます (2-42 ページの『グローバル名とローカル名』を参照してください)。

明示的に参照される場合、条件名は固有でなければならず、また名前の有効範囲規則そのものが参照の固有性を保証している場合を除き、修飾または添え字付け (あるいはその両方) によって固有になるようにする必要があります。

条件名を固有にするために修飾を使用する場合、最初の修飾子として関連する条件変数を使用できます。 修飾を使用する場合、条件名を固有にするために、条件変数に関連した名前の階層を使用する必要があります。

条件変数を参照するために添え字付けが必要な場合、その条件名のいずれかを参照するには、同様の組み合わせの添え字付けも必要です。

条件名に修飾と添え字付けを組み合わせて使用する場合の形式と制限は、データ名-1 を条件名-1 に置き換えるということを除き、「ID」の場合とまったく同じです。

以下に続く章にある一般的な形式の中で、「条件名」は、必要に応じて修飾されたり、添え字付けされた条件名を指しています。

データ名-1
レコード名とすることができます。

ファイル名-1
データ部の FD または SD 項目によって識別されなければなりません。

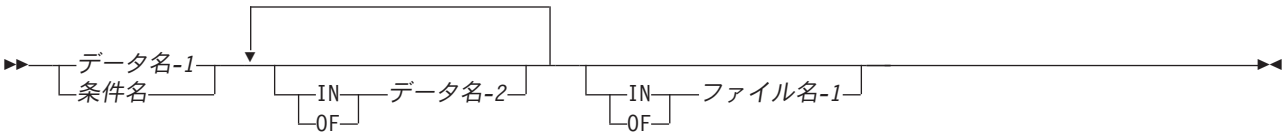
ファイル名-1 は当該プログラムの中で固有のものでなければなりません。

修飾

名前の階層の中に名前が存在し、その階層の中にある 1 レベルまたはそれ以上高いレベルの名前を指定することによってその名前が識別できる場合には、名前を固有にすることができます。この場合の高いレベルの名前を**修飾子**と呼び、このような手順で名前を固有にすることを**修飾**と呼びます。

ユーザー指定の名前の後に 1 つまたは複数の句を書き、それぞれの句は語 IN または OF の後に修飾子を付けて構成することによって、修飾が指定されます。 (IN と OF は論理的に同じです。)

データ部の名前への参照 - 形式 1



データ部の名前への参照 - 形式 2

どの階層においても、最高レベルに関連するデータ名は固有でなければならず、その名前を修飾することはできません。

名前を固有にするためには、十分の修飾を指定する必要がありますが、階層のすべてのレベルを指定する必要はありません。例えば、そのレコードに EMPLOYEE-NO フィールドの入ったファイルが 1 つまたは複数あるが、そのうち 1 つだけが MASTER-RECORD と名付けられたレコードをもっている場合、次のようになります。

- EMPLOYEE-NO OF MASTER-RECORD は、EMPLOYEE-NO の修飾には十分です。
- EMPLOYEE-NO OF MASTER-RECORD OF MASTER-FILE は有効ですが、このような修飾を行う必要はありません。

データ部の名前への参照: プログラム内で明示的に参照されるデータ部の名前は、固有に定義されたものであるか、または修飾によって固有にされたもののいずれかでなければなりません。参照されないデータ名は固有に定義する必要はありません。

レベル番号 01、あるいはファイル・セクション内の FD または SD レベル標識と関連しているデータ名は、データ階層内では最高レベルに位置します。このようなデータ名を参照する場合は、そのデータ名は固有に定義されていなければなりません。なぜなら、そのようなデータ名を修飾することはできないからです。なぜなら、そのようなデータ名を修飾することはできないからです。レベル番号が 02 ～ 49 のデータ項目は、データ階層内のレベルは順次下がっていきます。そのような項目を参照するときは、固有に定義されたものであるか、または修飾によって固有にできるもののいずれかでなければなりません。レベル 77 のデータ名を参照するときは、そのデータ名を修飾することはできないので、必ず固有に定義しておかなければなりません。

手続き部の名前への参照: 明示的に参照される場合、セクション内では段落名を重複させることはできません。段落名をセクション名で修飾する場合は、SECTION という語を書いてはなりません。段落名をそれ

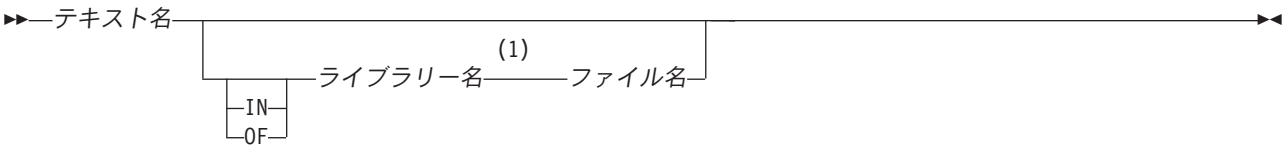
が書かれているセクション内で参照する場合は、修飾の必要はありません。プログラムに書かれる段落名またはセクション名は、他のプログラムからは参照できません。 7-7 ページの『セクション』で説明しているセクション名は、段落名に対して使用可能な最高位の (そして唯一の) 修飾子であり、固有でなければなりません。

手続き部の名前への参照 - 形式 1

手続き部の名前への参照 - 形式 2

COPY ライブラリーへの参照: コンパイル中に複数の COBOL ライブラリーを使用できる場合は、テキスト名は参照されるたびに修飾できます。

COPY ライブラリーへの参照 - 形式 3



注:

1 ファイル名を修飾する場合は、ライブラリー名とファイル名の間にハイフンが必要です。

COPY ライブラリーの参照の規則については 8-2 ページの『COPY ステートメント』を参照してください。

修飾の規則: 名前の修飾に関する規則を次に示します。

- 名前は、その必要がない場合でも、修飾できます。
- 各修飾子は、修飾する名前より高いレベルでなければならず、同じ階層内になければなりません。

  例えば、次のとおりです。

```
01 FIELD-A
   02 FIELD-B
      05 SUB1
         07 SUB2
   02 FIELD-C
         07 SUB1
```

  階層には、同じかまたはより高いレベル番号が次に現れるまでのすべての従属記入項目が含まれます。したがって、上記の例ではすべての記入項目が FIELD-A の階層に含まれます。 FIELD-B から FIELD-C の前までのすべての記入項目、ただし組み込まれているものではないものが FIELD-B の階層に含まれます。

  FIELD-A の階層では、SUB1 を 2 回使用できます。 1 回は FIELD-B に従属するものとして、さらにもう 1 回は FIELD-C に従属するものとして使用できます。 SUB-1 を参照する場合には、SUB-1 OF

FIELD-B または SUB-1 OF FIELD-C のように修飾しなければなりません。 FIELD-B または FIELD-C では、SUB1 は自分自身に従属することはできません。

- 1 つのデータ名の全修飾子のリストは、別のデータ名の修飾子の部分的リストと同じであってはなりません。
- 1 つのデータ名または条件名が複数のデータ項目に割り当てられている場合は、そのデータ項目を参照するたびに修飾しなければなりません。(1 つ例外がありますので 6-75 ページの『REDEFINES 文節』を参照してください。)
- 修飾により参照を固有にできる場合は、指定されたプログラムまたはコンパイル単位の複数の場所でデータ名を定義できます。
- 固有性を確実にする修飾子の組み合わせが 2 組以上存在する場合には、どの組み合わせを使用してもかまいません。
- プログラム内でセクション名を参照する場合は、セクション名は固有でなければなりません。
- プログラム内で段落名を参照する場合、段落名はセクションの中で固有でなければなりません。段落名をセクション名で修飾する場合は、SECTION という語を書いてはなりません。段落名をそれが書かれているセクション内で参照する場合は、修飾の必要はありません。
- ソース・プログラム内に LINAGE 文節を含むファイル記述記入項目が複数指定されている場合は、LINAGE-COUNTER は参照するたびに修飾しなければなりません。
- ライブラリー名はシステム内で固有でなければなりません。したがって、ライブラリー名の上位 10 文字は固有でなければなりません。
- テキスト名 (メンバー名) は、それが属するライブラリー名およびファイル名によって修飾しなければなりません。 (ライブラリー名とファイル名の間には、スペースを入れずにハイフンを置くことが必要です。) ライブラリーが指定されない場合は、ライブラリー・リストが検索されます。ファイル名が指定されない場合は、QCBLLESRC が使用されます。

IBM Extension

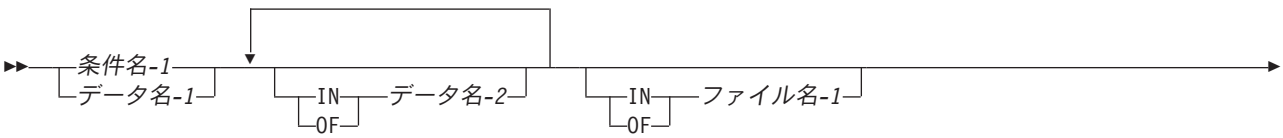
形式 1 の COPY ステートメントでは、ファイル名の指定はオプショナルです。ファイル名が指定されない場合のデフォルトは QCBLLESRC です。

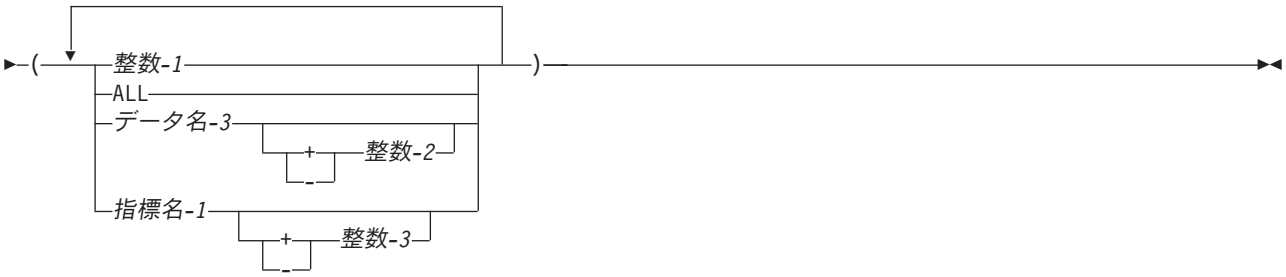
End of IBM Extension

添え字付け

添え字付けとは、正の整数を使用してテーブル参照を行う手法のことです。**添え字**とは、テーブル・エレメントのオカレンス番号を指定する正の整数値 (または整数データ項目) です。

添え字付け - 形式





条件名-1
OCCURS 文節を含むデータ記述記入項目に従属している必要があります。

データ名-1
OCCURS 文節を含んでいるか、または OCCURS 文節を含むデータ記述記入項目に従属している必要があります。

整数-1
符号付きでもかまいません。符号付きの場合には、正でなければなりません。

ALL
さまざまな数の引数を認める関数の、関数引数として使用されます。添え字付き ID が関数の引数として使用されるときのみ使用でき、条件名が指定されているときには使用できません。

データ名-3
整数を表す数字基本項目でなければなりません。

データ名-3 は修飾できます。

指標名-1
指標名を指定している INDEXED BY 句を含む、参照されているテーブルの階層のデータ記述記入項目に対応します。

整数-2、整数-3
符号のない整数でなければなりません。

括弧で囲まれた添え字は、テーブル・エレメントの名前の修飾を直後に続けて書きます。このような参照の添え字の数は、そのエレメントが参照されているテーブルの次元の数と等しくなければなりません。すなわち、データ名そのものも含めて、データ名の入っている階層の各 OCCURS 文節に 1 つの添え字がなければなりません。

複数の添え字が必要とされるときには、それらはデータ編成を含めた小さい方の次元から連続した順序で書かれます。複数の次元のテーブルが連続するネストされたテーブルであると見なされ、ネストの一番外側のテーブルが大テーブルで、一番内側が小テーブルであると見なされる場合、添え字は大、中、小の順番で左から右へ書かれます。

例えば、TABLE-THREE が次のように定義されている場合、

```
01  TABLE-THREE.
    05  ELEMENT-ONE OCCURS 3 TIMES.
        10  ELEMENT-TWO OCCURS 3 TIMES.
            15  ELEMENT-THREE OCCURS 2 TIMES    PIC X(8).
```

TABLE-THREE への有効な添え字付き参照は次のとおりです。

```
ELEMENT-THREE (2 2 1)
```

項目への参照は、その項目がテーブル・エレメント**または**テーブル・エレメントに関連した項目または条件名でなければ、添え字付きにしてはなりません。

各テーブル・エレメント参照は、次のような参照が行われる場合を除き、添え字を付ける必要があります。

- USE FOR DEBUGGING ステートメントの中
- SEARCH ステートメントのサブジェクトとして
- REDEFINES 文節の中
- OCCURS 文節の KEY IS 句の中
- LIKE 文節の中

添え字によって表される最低の許容オカレンス数は 1 です。どのような特定の場合でも最高の許容オカレンス数は、OCCURS 文節で指定される項目のオカレンスの最大数です。

整数またはデータ名を使った添え字付け: 整数またはデータ名が添え字を表すために使用されるとき、それらは異なったテーブル内の項目を参照するために使用できます。これらのテーブルは、同じサイズのエレメントを持つ必要はありません。同じ整数またはデータ名を、1 つの項目を持つ唯一の添え字および別の項目を持つ複数の添え字の中の 1 つとして表すことができます。各データ名は、修飾できますが、添え字付きまたは指標付きとすることはできません。例えば、TABLE-THREE への有効な添え字付き参照には次のものがあります (SUB1、SUB2、および SUB3 はすべて、SUBSCRIPT-ITEM に従属する項目であると想定します)。

```
ELEMENT-THREE (SUB1 SUB2 SUB3)
ELEMENT-THREE IN TABLE-THREE (SUB1 OF SUBSCRIPT-ITEM, SUB2 OF
SUBSCRIPT-ITEM, SUB3 OF SUBSCRIPT-ITEM)
```

指標名を使用した添え字付け (指標付け): 指標付けにより、テーブル検索や特定項目の操作などの操作を行うことができます。指標付けを使用するには、データ記述記入項目に OCCURS 文節が入っているデータ記述記入項目に 1 つまたは複数の指標名を関連付けます。指標名と関連する指標は添え字として機能し、その値は指標名と関連する項目のオカレンス番号に対応します。

テーブルと関連する指標名を識別する INDEXED BY 句は、OCCURS 文節のオプショナルの部分です。指標名の定義は完全にシステム依存なので、指標名を記述する別の項目はありません。指標名は、このプログラムだけが使用するためにコンパイラーが生成したレジスターと考えることができます。指標名はデータまたはデータ階層の一部ではなく、COBOL プログラムの中で固有でなければなりません。

各指標名は、ユーザー定義語の形成規則に従っていなければなりません。

各指標名は、コンパイラーが生成したレジスターまたは記憶域を指します。

オブジェクト時の指標の初期値が定義されないときは、指標は添え字として使用される前に初期設定されなければなりません。指標の初期値は以下のいずれかにより割り当てられます。

- VARYING 句を伴う PERFORM ステートメント
- ALL 句を伴う SEARCH ステートメント
- SET ステートメント

CALL ステートメントの USING 句のパラメーターとしてまたは関係条件比較の中で指標名を参照できるのは、PERFORM、SET、または SEARCH ステートメントだけです。

テーブル・エレメントまたはテーブル・エレメント内の項目を参照する添え字としての整数またはデータ名を使っても、そのテーブルに関連付けられている指標に変化はありません。

指標名は、INDEXED BY 句によって関連付けられるテーブルだけへの参照として使用できます。

分離文字

テーブルの形式内に配列されるデータは、しばしば検索されます。 SEARCH ステートメントには、逐次検索および非逐次 (例えば、2 進) 検索の機能があります。これは、特定の条件を満たすテーブル・エレメントの検索およびそのテーブル・エレメントを示す関連した指標の値の調整に使用されます。

指標値が実行時に有効であるためには、指標値は、オカレンス番号が 1 以上であり、かつ許容できる最高値以下のテーブル・エレメントのオカレンスに対応している必要があります。

指標名についての詳細は、OCCURS 文節の INDEXED BY 句に記述されています。 6-52 ページの『INDEXED BY 句』を参照してください。

相対添え字付け: 相対添え字付けの場合、テーブル・エレメントの名前の後に、データ名または指標名の次に + または - の演算子と符号なしの整数リテラルという形式の添え字が続きます。演算子の + および - の前後には、スペースを置かなければなりません。添え字にデータ名が入っている場合、添え字の値は、データ名が整数の値によって設定された後の値と同じ値になります。添え字に指標名が入っている場合、整数はオカレンス番号と見なされ、指標名への加算や減算の前に指標値に変換されます。相対指標付けの使用によって、オブジェクト・プログラムが指標の値を変化させることはありません。

指標の値は、指標データ項目の値を保管することによってオブジェクト・プログラムへアクセス可能にすることができます。指標データ項目は、USAGE IS INDEX 文節を含んでいるデータ記述記入項目によってプログラムに記述されます。指標値は、SET ステートメントを実行することによって指標データ項目に移されます。

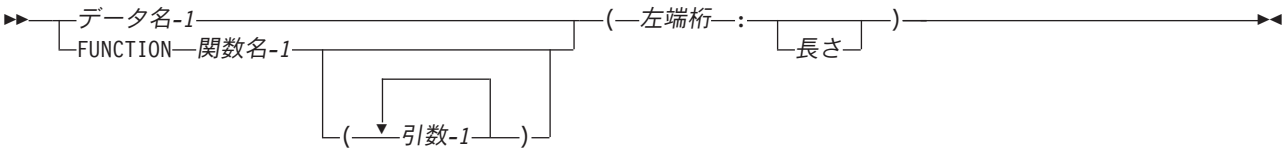
有効な指標値は、オカレンス番号が 1 以上で許容できる最高の値以下のテーブル・エレメントのオカレンスに対応していなければなりません。

指標名についての詳細は、OCCURS 文節の INDEXED BY 句に記述されています。 6-52 ページの『INDEXED BY 句』を参照してください。

参照変更

参照変更は、他のコンピューター言語におけるサブストリング化と類似しています。参照変更は、データ項目の開始桁と長さを指定することによって、データ項目を定義します。

フォーマット



データ名-1

使用法が DISPLAY、DISPLAY-1、または NATIONAL として暗黙に、または明示的に定義されているデータ項目を参照する必要があります。データ名-1 には修飾や添え字付けができます。

IBM Extension

ブール・データ項目を参照することはできません。

End of IBM Extension

IBM Extension

TYPE 文節を使用して定義した項目を参照してはなりません。

End of IBM Extension

関数名-1
引数を含む、英数字関数、DBCS 関数、または国別関数でなければなりません。

詳細については 7-274 ページの『組み込み関数』を参照してください。

引数-1
引数-1 は、ID、リテラル (表意定数以外)、または算術式でなければなりません。

左端桁
算術式でなければなりません。左端桁の位置の評価結果は、データ名-1 または関数名-1 によって参照されるデータ項目内の文字数以下である正の非ゼロ整数にならなければなりません。

長さ
算術式でなければなりません。

左端桁と長さの和から 1 を引いた値は、データ名-1 または関数名-1 によって参照されるデータ項目内の文字数以下でなければなりません。長さを省略すると、使用される長さは、データ名-1 または関数名-1 によって参照されるデータ項目内の文字数に 1 を加え、さらにそこから左端桁を引いた値と同じになります。長さの評価の結果は、正の非ゼロの整数でなくてはなりません。

注: 算術式の結果が固定小数点の非整数の場合は、切り捨てが行われて整数になります。算術式の結果が浮動小数点の非整数の場合は、丸めが行われて整数になります。

IBM Extension

DBCS または国別データ項目の場合、桁および長さは 2 バイト文字の数を参照します。

End of IBM Extension

参照変更は、ID による英数字、DBCS、または国別データ項目の参照が認められるところでは、許可されるのが普通です。

IBM Extension

日時クラスのデータ項目を参照変更することはできません。

End of IBM Extension

データ名-1 または関数名-1 が参照するデータ項目の各文字には、左端桁から右端桁へ 1 ずつ増分する序数が割り当てられます。左端桁には序数 1 が割り当てられます。データ名-1 のデータ記述項目に SIGN IS SEPARATE 文節が含まれている場合、符号の桁にはそのデータ項目内の序数が割り当てられます。

参照変更は、データ名-1 または関数名-1 およびその引数によって参照されるデータ項目のサブセットとその引数である固有のデータ項目を作成します。この固有データ項目は、JUSTIFIED 文節なしの基本項目と見なされます。

データ名-1 が参照変更される場合、固有データ項目は、データ名-1 によって参照されるデータ項目に対して定義されたものと同じクラスとカテゴリーを持っています。しかし、データ名-1 のカテゴリーが数字、数字編集、英数字編集、または外部浮動小数点である場合、固有データ項目のクラスとカテゴリーは英数字です。

関数が参照変更される場合、固有データ項目は、関数の引数によって英数字、DBCS、または国別のクラスおよびカテゴリーを持ちます。

長さが指定されない場合、作成された固有のデータ項目は、左端桁によって識別された文字から、データ名-1 または関数名-1 によって参照されたデータ項目の右端桁までの範囲となります。

オペランドの計算: 1 つのオペランドの参照変更は次のように計算されます。

- オペランドに対して添え字付けが指定されている場合、参照変更の計算は添え字の計算直後に行われます。
- オペランドに対して添え字付けが指定されていない場合、参照変更の計算は、もし添え字が指定されていたとしたら添え字付けが計算されるであろうときに行われます。オペランドに対して ALL 添え字付けが指定されている場合、テーブルの暗黙的に指定された各エレメントに対して、参照変更が適用されます。
- 組み込み関数に参照変更が指定されている場合には、参照変更の評価は、関数の評価の直後に行われます。

参照変更の例: この例では、変数 whole-name 内の最初の 25 字を、変数 last-name に転送しています。

```
MOVE whole-name(1:25) TO last-name
```

範囲エラー: 範囲外の参照変更コンポーネント、例えば左端の桁がゼロの場合などには、システム・メッセージが生成されます。 これは、添え字範囲や文字ストリング境界内でのエラーを知らせるメッセージと同じものです。 (このメッセージは、CRTCBLMOD または CRTBNDCBL コマンドで RANGE オプションが指定された場合にだけ生成されます。)

参照変更に関する制約事項:

IBM Extension

INDICATORS 句は参照変更をサポートしません。

End of IBM Extension

次の制約事項は以下にリストするステートメントに適用されます。

ステートメント
制約事項

STRING
ID-3 の参照変更は不可。

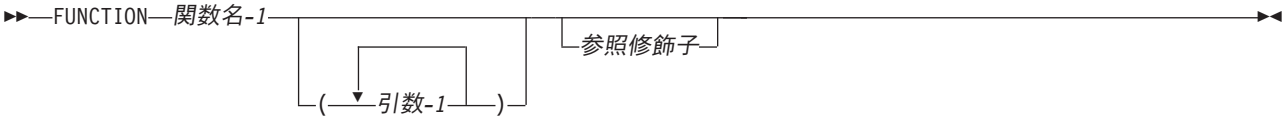
UNSTRING
ID-1 の参照変更は不可。

START
データ名-1 の最終オカレンスのに参照変更が可。

関数 ID

関数 ID は、関数の評価で得られるデータ項目を固有に参照する、文字ストリングと分離文字の構文的に正しい文字ストリングです。

フォーマット



引数-1

ID、リテラル (表意定数以外)、または算術式でなければなりません。

関数名-1

関数名-1 は、組み込み関数名の 1 つでなければなりません。

詳細については 7-274 ページの『組み込み関数』を参照してください。

参照修飾子

英数字関数、DBCS 関数、または国別関数にだけ指定できます。

英数字項目への参照: 英数字関数を参照する関数 ID は、ID が認められていて、関数への参照が特に禁じられていない場合は、どの場所においても指定できますが、次の場合を除きます。

- 任意のステートメントの受け入れオペランドとして指定する場合。
- データ項目が特定の特性 (クラスおよびカテゴリー、使用法、サイズ、および暗黙的値など) を持つ必要があり、定義に従った関数の評価および指定されている特定の引数がこれらの特性を持たない可能性がある場合。

整数への参照: 整数または数字関数への参照を行う関数 ID は、算術式が使用できるところであればどこででも使用できます。

DBCS 項目への参照:

IBM Extension

DBCS ID が認められていて、一般形式に関する規則で特に関数への参照が禁止されていない場合は、一般形式のどの場所においても DBCS 関数を指定できますが、次の場合を除きます。

- 任意のステートメントの受け入れオペランドとして指定する場合。
- 一般形式に関する規則で、参照されているデータ項目が特定の特性 (クラスおよびカテゴリー、使用法、サイズ、および暗黙的値など) を持つ必要があり、定義に従った関数の評価および指定された特定の引数が、これらの特性を持たない可能性がある場合。

DBCS 関数の参照変更が認められています。関数に参照変更が指定されている場合には、参照変更の評価は、関数の評価の直後に行われます。

DBCS 関数は、DBCS 引数を認めている関数の引数として参照できます。

End of IBM Extension

国別項目への参照:

IBM Extension

国別 ID が許可されており、一般形式に関連する規則で特に関数への参照が禁止されていない場合は、一般形式のどの場所においても国別 ID を指定できますが、次の場合を除きます。

- 任意のステートメントの受け入れオペランドとして指定する場合。
- 一般形式に関する規則で、参照されているデータ項目が特定の特性 (クラスおよびカテゴリー、使用法、サイズ、および暗黙的値など) を持つ必要があり、定義に従った関数の評価および指定された特定の引数が、これらの特性を持たない可能性がある場合。

国別関数の参照変更が認められています。関数に参照変更が指定されている場合には、参照変更の評価は、関数の評価の直後に行われます。

国別関数は、国別引数を認めている関数の引数として参照できます。

End of IBM Extension

日時項目への参照:

IBM Extension

日時 ID が許可されており、一般形式に関連する規則で特に関数への参照が禁止されていない場合は、一般形式のどの場所においても日時関数を指定できますが、次の場合を除きます。

- 任意のステートメントの受け入れオペランドとして指定する場合。
- 一般形式に関する規則で、参照されているデータ項目が特定の特性 (クラスおよびカテゴリー、使用法、サイズ、および暗黙的値など) を持つ必要があり、定義に従った関数の評価および指定された特定の引数が、これらの特性を持たない可能性がある場合。

日時関数は、日時を引数として取ることができる関数の引数として参照できます。

End of IBM Extension

ブール項目への参照:

IBM Extension

ブール ID が許可されており、一般形式に関連する規則で特に関数への参照が禁止されていない場合は、一般形式のどの場所においてもブール関数を指定できますが、次の場合を除きます。

- 任意のステートメントの受け入れオペランドとして指定する場合。
- 一般形式に関する規則で、参照されているデータ項目が特定の特性 (クラスおよびカテゴリー、使用法、サイズ、および暗黙的値など) を持つ必要があり、定義に従った関数の評価および指定された特定の引数が、これらの特性を持たない可能性がある場合。

ブール関数は、ブール値を引数として取ることができる関数の引数として参照できます。

End of IBM Extension

ユーザー定義データ・タイプ

IBM Extension

ユーザー定義データ・タイプ (またはタイプ名) は TYPEDEF 文節を含む 01 レベルの基本項目またはグループ項目です。このような項目にストレージが割り振られることはありません。これは、データ名とそれに従属する項目を記述するテンプレートと見なすことができます。したがって、タイプ名を TYPE 文節内で指定することにより、データ名 (または別のタイプ名) を定義することが可能です。このようにして定義したデータ名は、TYPE 文節内で指定したタイプ名の特性を持つことになります。このタイプ名がグループ項目の場合は、定義されたデータ名にはそのタイプ名に従属する項目と同じ名前、階層、および特性を持つ項目が従属します。

ユーザー定義データ・タイプを使用して TYPE 文節でデータ名 (またはタイプ名) を定義する際に、その TYPE 文節と他の文節との論理積をとってこのデータ名の記述を完全なものにすることができます。ただし、そのために使用できるのは以下の文節だけです。

- EXTERNAL 文節
- GLOBAL 文節
- OCCURS 文節
- TYPEDEF 文節
- VALUE 文節

タイプ名の有効範囲に関する規則は、データ名の場合の規則と同じです。

TYPE 文節と TYPEDEF 文節の詳細については 6-87 ページの『TYPE 文節』および 6-88 ページの『TYPEDEF 文節』を参照してください。

TYPEDEF 文節: TYPEDEF 文節では、ユーザー定義データ・タイプ (タイプ名) となる基本データ項目またはグループ・データ項目を宣言します。いったん定義したタイプ名は、その後も他のデータ項目を定義するために (TYPE 文節内で) 使用できます。

TYPE 文節: TYPE 文節では、ユーザー定義データ・タイプ (タイプ名) を使用してデータ項目を定義できます。これは (TYPEDEF 文節を使用して宣言した) タイプ名を TYPE 文節内で指定することによって行います。そのタイプ名がグループ項目の場合は、定義するデータ項目も同様にグループ項目となります。この場合、このデータ項目に従属する記入項目の名前、階層および特性は、当該のタイプ名に従属する記入項目のそれらと対応します。

タイプ名-1
対象とするデータ名を定義するために使用するタイプ名

End of IBM Extension

名前の有効範囲

このセクションでは、COBOL 名のタイプについて簡潔に説明し、次に名前有効範囲の規則を説明します。

名前のタイプ

英字名
英字名は、特定の文字セット、または環境部の SPECIAL-NAMES 段落の照合順序 (あるいはその両方) に名前を割り当てます。

クラス名
クラス名は、真理値を定義できる環境部の SPECIAL-NAMES 段落の前置詞に名前を割り当てます。

条件名
条件名は、値と条件変数を関連付けます。

定数名
定数名は定数を示すもので、データ部で定義されます。

データ名
データ名はデータ項目の名前です。

ファイル名
ファイル名はファイル・コネクターの名前です。

指標名
指標名は特定のテーブルに関連した指標の名前です。

ライブラリー名
ライブラリー名は指定されたソース・プログラムのコンパイル時にコンパイラーが使用する COBOL ライブラリーの名前です。

簡略名
簡略名は実装者名にユーザー定義語を割り当てます。

段落名
段落名は手続き部の段落の名前です。

プログラム名
プログラム名は、外部または内部の (ネストされた) プログラムの名前です。

2-45 ページの『プログラム名に関する規則』を参照してください。

レコード名
レコード名はレコードの名前です。

セクション名
セクション名は手続き部にあるセクションの名前です。

シンボリック文字
シンボリック文字はユーザー定義の表意定数を指定します。

テキスト名
テキスト名は、COPY 指示ステートメントによって使用されるソース・メンバーを含むライブラリーを示す名前です。

IBM Extension

タイプ名

タイプ名は、TYPE 文節内でデータ項目の定義に使用することができるユーザー定義データ・タイプを命名します。

End of IBM Extension

ネストされたプログラム

COBOL プログラムには、他の COBOL プログラムを**含める**ことができます。**含まれた** (つまりネストされた) プログラムにさらに別のプログラムが含まれている場合もあります。そのような場合には、あるプログラムに**直接**含まれるプログラムと**間接的に**含まれるプログラムが存在することになります。

図 2-2 は、直接的および間接的に含まれたプログラムのプログラム構造を説明しています。

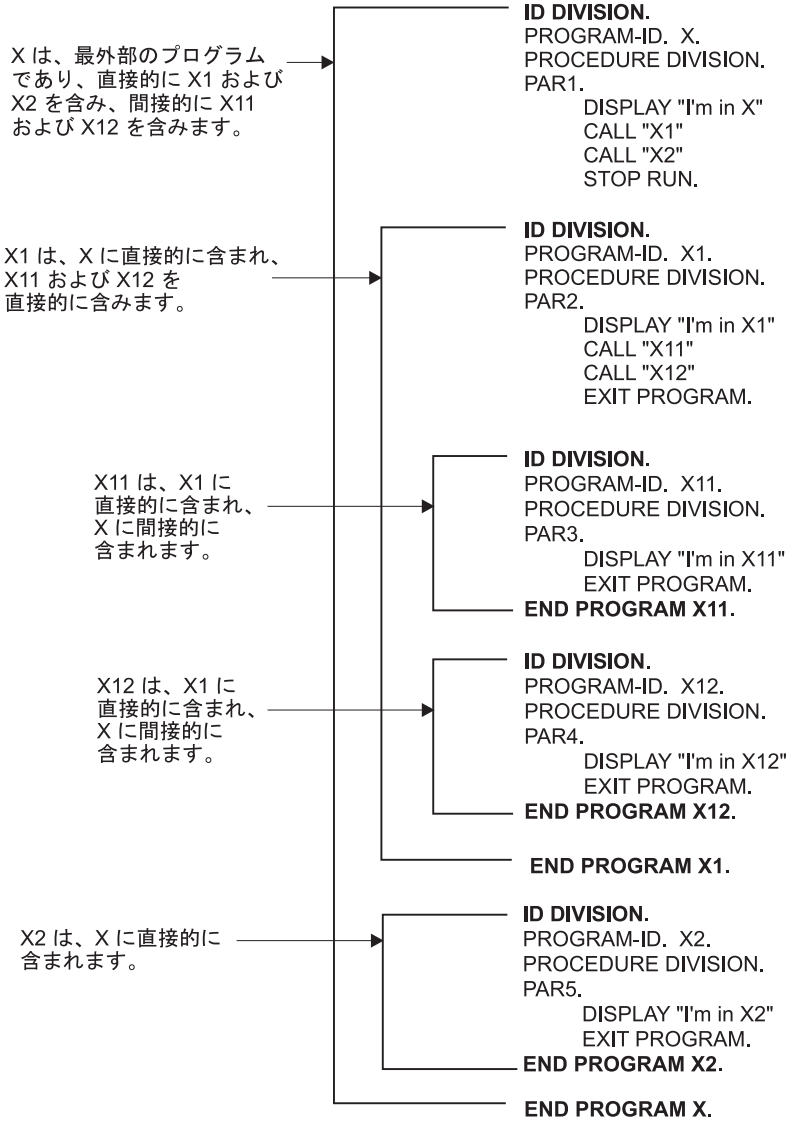


図 2-2. 直接的または間接的に含まれたプログラムのネストされたプログラム構造

異なるプログラムで異なるオブジェクトを定義するために、同じユーザー定義語を使用できます。特定のプログラムの場合、オブジェクトへの参照は常に**そのプログラムの中で**定義されたオブジェクトを参照します。

グローバル名とローカル名

名前にはグローバル属性またはローカル属性を含めることができます。名前の中には常にグローバルであるものもあれば、常にローカルであるものもあります。また、ローカルになるかグローバルになるかは名前が宣言されたプログラムで行う指定によって決まるという名前もあります。

プログラムは、それに含まれているプログラム内で宣言されたどの条件名、データ名、ファイル名、指標名、段落名、レコード名、セクション名、またはタイプ名も、参照することはできません。

グローバル名は、その名前が宣言されたプログラム内か、グローバル名を宣言したプログラムを含む他のプログラム内のいずれかで関連したオブジェクトを参照するために使用されます。

しかし**ローカル名**は、そのローカル名を宣言したプログラム内だけから関連したオブジェクトを参照します。

データ名、レコード名、条件名、タイプ名、またはファイル名がグローバルとして宣言されていない場合は、その名前はローカルです。

注: 特定の規則では、あるデータ記述記入項目、ファイル記述記入項目、またはレコード記述記入項目に対して GLOBAL 文節の指定を禁止する場合があります。

定数名
GLOBAL 文節が指定されている場合、定数名はグローバルです。

データ名
GLOBAL 文節が、データ名を宣言したデータ記述記入項目またはデータ記述記入項目が従属である別の記入項目のいずれかで指定された場合、そのデータ名はグローバルです。

ファイル名
ファイル名は、そのファイル名のファイル記述記入項目に GLOBAL 文節が指定されていると、グローバル名になります。

1 つの実行単位内の 2 つのプログラムは、次の状況の下では共通ファイル・コネクターを参照できます。

1. 外部ファイル・コネクターは、そのファイル・コネクターを記述したどのプログラムからでも参照できます。
2. プログラムが別のプログラムに含まれている場合、両方のプログラムは共通ファイル・コネクターを参照できます。それは、ファイル・コネクターのあるプログラムか、ファイル・コネクターのあるプログラムを直接的または間接的に含んだプログラムのいずれかで宣言された関連グローバル・ファイル名を参照することによって行われます。

レコード名
レコード名がグローバルになるのは、そのレコード名を宣言したレコード記述記入項目に GLOBAL 文節が指定されているときか、またはファイル・セクションのレコード記述記入項目の場合は、そのレコード記述記入項目と関連したファイル名のファイル記述記入項目に GLOBAL 文節が指定されているときです。

条件名
データ記述記入項目が GLOBAL 文節を指定する別の記入項目に従属する場合、条件名はグローバルで、データ記述記入項目に宣言されます。

条件名は、構成セクション内で宣言されるとき、常にグローバルです。

プログラム名
プログラム名はローカルでもグローバルでもありません。 2-45 ページの『プログラム名に関する規則』を参照してください。

セクション名および段落名
これらの名前は常にローカルです。

ライブラリー名およびテキスト名
これらの名前はプログラムに対して外部であり、どの COBOL プログラムからでも参照できます。またコンパイラー・システムは関連ライブラリーをサポートしており、参照されたエンティティーがそのシステムに分かるようになっています。

英字名
英字名は常にグローバルです。

クラス名
クラス名は常にグローバルです。

簡略名
簡略名は常にグローバルです。

指標名
グローバル属性を持つデータ項目が指標によりアクセスするテーブルを含む場合、その指標もグローバル属性を持ちます。そのため、指標名の有効範囲はテーブル名を付けるデータ名の有効範囲と同一になります。そのテーブルの指標への命名はその指標名によって行われ、データ名の名前の規則の有効範囲が適用されます。指標名を修飾することはできません。

IBM Extension

タイプ名
あるタイプ名が宣言されているデータ記述記入項目内に GLOBAL 文節が指定されている場合は、そのタイプ名はグローバルです。ただし、タイプ名の GLOBAL 属性はそのタイプ名のみに制限されるため、そのタイプ名を使用して TYPE 文節で定義したデータ項目がその属性を獲得することはありません。

End of IBM Extension

外部オブジェクトと内部オブジェクト

アクセス可能なデータ項目には通常、保管されたデータの一定の表示が必要です。ファイル・コネクターには通常、保管されたファイルについての情報が必要です。データ項目またはファイル・コネクターに関連したストレージは、オブジェクトを宣言したプログラムの**外部**または**内部**に取ることができます。

オブジェクトに関連したストレージが、実行単位内の特定プログラムではなく実行単位に関連している場合、データ項目またはファイル・コネクターは外部オブジェクトです。外部オブジェクトは、オブジェクトを記述する実行単位内のプログラムによって参照されることがあります。同一の名前を持つオブジェクトの個々の記述を使って、異なるプログラムから外部オブジェクトを参照するとき、それらのオブジェクトは常に同一になります。実行単位には、ただ 1 つの外部オブジェクトの表示だけがあります。

オブジェクトに関連したストレージがオブジェクトを記述するプログラムにだけ関連する場合は、内部オブジェクトです。

外部および内部オブジェクトはグローバル名かローカル名のいずれかを持つことができます。

EXTERNAL 文節がそのデータ記述記入項目に存在することによって、作業用ストレージ・セクションに記述されたデータ・レコードには外部属性が与えられます。さらにデータ記述記入項目 (外部レコードを記述している記入項目に従属する) で記述したデータ項目には、外部属性が与えられます。レコードまたはデータ項目に外部属性がない場合、それは記述したプログラムの内部データの一部です。

関連ファイル記述記入項目に EXTERNAL 文節がある場合は、ファイル・コネクターに外部属性が与えられます。ファイル・コネクターに外部属性がない場合、ファイル・コネクターは関連ファイル名を記述したプログラムに対して内部です。ソート・マージ・ファイルに対して EXTERNAL 文節を指定することはできません。

EXTERNAL 文節を含んでいないファイル記述記入項目に従属するように記述されたデータ・レコードや、ソート・マージ・ファイル記述記入項目に従属するように記述されたデータ・レコードは、そのようなデータ・レコードに対するデータ記述記入項目に従属するように記述されたあらゆるデータ項目同様に、ファイル名を記述するプログラムに対して常に内部です。 EXTERNAL 文節がファイル記述記入項目にある場合、データ・レコードおよびデータ項目には、外部属性が与えられます。

プログラムのリンケージ・セクションに記述されたデータ・レコードや従属データ項目は、そのデータを記述するプログラムに対して常に内部と見なされます。特別な考慮事項がリンケージ・セクションに記述したデータに適用され、記述されたデータ・レコードと別のプログラムにアクセス可能な他のデータとを関連させます。

データ属性の指定

明示的なデータ属性は、実際の COBOL のコーディング中にプログラマーが指定します。

暗黙のデータ属性は、デフォルト値です。データの属性を明示的に指定しない場合には、コンパイラーはデフォルト値を使用します。

例えば、データ項目の USAGE は指定する必要はありません。これを省略した場合のデフォルトは USAGE DISPLAY であり、この場合は、暗黙のデータ属性です。しかし COBOL のコーディングの中に USAGE DISPLAY を指定すると、これは明示的なデータ属性になります。

名前の解決

プログラム B というプログラムが直接的または間接的に別のプログラム A というプログラムの中に含まれる場合、両方のプログラムは同一ユーザー定義語を使用してオブジェクトを定義できます。 (オブジェクトには、例えば、条件名、データ名、ファイル名、レコード名、関数名、またはタイプ名などが含まれます。) そのような重複した名前をプログラム B で参照するときには、次の規則を使用して参照オブジェクトを判別します。

1. 参照オブジェクトは、プログラム B で定義されたすべての名前セットから、また、プログラム A に直接含まれる定義されたすべてのグローバル名、さらに直接的または間接的にプログラム A を含むプログラムで定義されたすべてのグローバル名から識別されます。この名前のセットを使用するときには、修飾に関する通常の規則や参照の固有に関する他のすべての規則が適用されます。これは 1 つまたは複数のオブジェクトが識別されるまで続きます。
2. 1 つのオブジェクトだけが識別される場合は、それが参照オブジェクトです。
3. 複数のオブジェクトが識別される場合、適切な修飾によってオブジェクトの参照をそれぞれ固有なものにしないかぎり、ただ 1 つのオブジェクトだけがプログラム B に対してローカルな名前を持つことができます。プログラム B に対してローカルな名前を持つオブジェクトが 1 つあるか、または何もないとき、次の規則が適用されます。

- 名前がプログラム B で宣言される場合、プログラム B は参照オブジェクトです。
- また、プログラム A が別のプログラムの中に含まれる場合、参照オブジェクトは次のとおりです。
  - 名前がプログラム A で宣言される場合、プログラム A のオブジェクトが参照オブジェクトです。
  - 名前がプログラム A で宣言されるのではなく、プログラム A を含んだプログラムで宣言される場合、その収容プログラムのオブジェクトが参照オブジェクトです。この規則は、単一の有効なオブジェクトが見つかるまでさらに大きな範囲の収容プログラムに適用されます。
- 参照されたオブジェクトが関数である場合は、その関数定義によって、時にはプログラマーが、その特定の参照のために関数の値を判別する 1 つまたは複数の引数の値を設定または指定することを要求されます。**関数 ID** という用語は、COBOL ソース・プログラムの手続き部内の組み込み関数を指すのに使用されます。関数によって示されるデータ項目は、関数名とその引数 (ある場合) によって固有に識別されます。

プログラム名に関する規則

プログラムのプログラム名は、そのプログラムの見出し部の PROGRAM-ID 段落で指定されます。プログラム名を参照できるのは次のものに限られます。

- CALL ステートメント
- CANCEL ステートメント
- END PROGRAM ヘッダー

IBM Extension

- SET ステートメント

End of IBM Extension

実行単位を構成しているプログラム名は固有である必要はありませんが、実行単位内の 2 つのプログラムが同一の名前であるとき、2 つのプログラムのうち少なくとも 1 つは直接的または間接的に別のプログラム (別個にコンパイルされた) になければなりません。そのコンパイル済みのプログラムには、2 つのうちの残った方のプログラムを含めることはできません。

プログラム名の有効範囲は次の規則によって規制されています。

- プログラム名が、COMMON 属性をもたないプログラムの名前で、別のプログラムに直接含まれている場合、収容プログラムにあるステートメントだけがそのプログラム名を参照できます。
- プログラム名が、COMMON 属性を持つプログラムの名前で、別のプログラム内に直接含まれている場合、収容プログラムにあるステートメント、およびその収容プログラム内に直接または間接的に含まれているプログラムのステートメントだけがそのプログラム名を参照できます。例外として COMMON 属性を持つプログラムそれ自体およびそのプログラム内に含まれているプログラムはどれも参照できません。
- プログラム名が、別個にコンパイルしたモジュール・オブジェクト中の最外部の COBOL プログラムの名前である場合、直接的または間接的に含まれるプログラムを除けば、実行単位内の他のどんなプログラムにあるステートメントでもプログラム名を参照できます。

プログラム名の有効範囲を規制する規則: 次の規則は別のプログラム内に含まれたプログラムのプログラム名を参照する際に適用されます。ここでは、プログラム-B およびプログラム-C を直接含むプログラム-A、プログラム-D およびプログラム-F を直接含むプログラム-C、さらにプログラム-E を直接含むプロ

グラム-D について取り上げます。

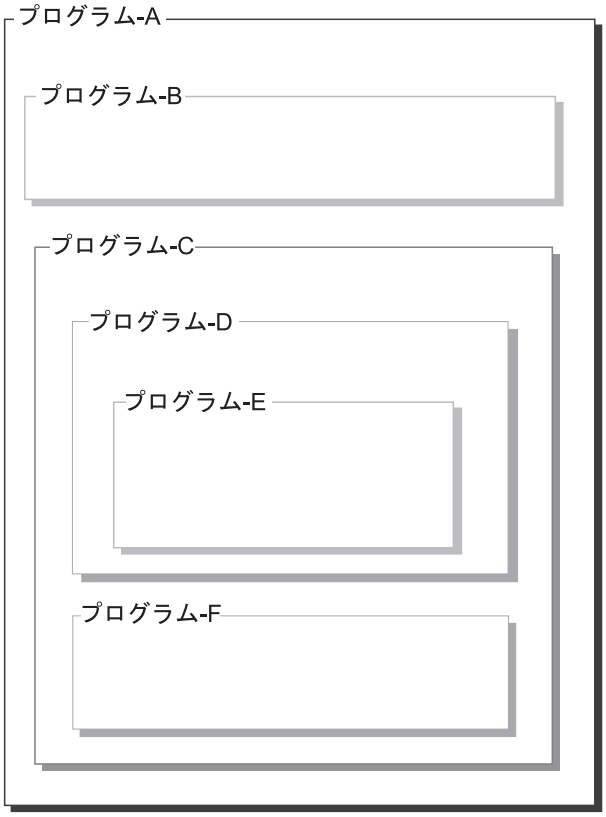


図 2-3. プログラム名の有効範囲を規制する規則

プログラム-D が COMMON 属性を持っていない場合には、プログラム-D は、プログラム-D を直接含んでいるプログラム (つまりプログラム-C) によってのみ参照されます。

プログラム-D が COMMON 属性を持っている場合、プログラム-C はプログラム-D を含んでいるのでそれを参照できます。さらにプログラム-D およびプログラム-D に含まれるプログラム以外は、プログラム-C に含まれるどのプログラムでもプログラム-D を参照できます。言い換えると、プログラム-D が COMMON 属性を持っている場合、プログラム-D はプログラム-C およびプログラム-F で参照できますが、プログラム-E、プログラム-A、プログラム-B、またはプログラム-D のステートメントでは参照できません。

制御の転送

手続き部では、**明示的な**制御の転送が存在せず、 次の実行可能ステートメントが存在するのであれば、プログラムの流れは、ステートメントが書かれた順番に従って次々と制御を転送していきます。 (以下の注を参照してください。) この通常のプログラムの流れが、**暗黙の**制御の転送です。

連続するステートメントの間での暗黙の制御の転送の他、プロシージャーの分岐ステートメントが実行されずに正規の流れが変更される場合にも暗黙の制御の転送が行われます。以下の例は、ステートメントからステートメントへの制御の転送を無効にする、**暗黙の**制御の転送を示しています。

- 別の COBOL ステートメントの制御下で実行される COBOL プロシージャーの最終ステートメントが実行されると、暗黙のうちに制御が転送されます。 (COBOL プロシージャーの実行を制御する COBOL ステートメントには、MERGE、PERFORM、SORT、および USE などがあります。)

- SORT または MERGE ステートメントの実行中に、INPUT または OUTPUT のプロシージャーに暗黙のうちに制御が転送されます。
- 宣言プロシージャーを実行させるような COBOL ステートメントの実行中、制御はそのプロシージャーに暗黙のうちに転送されます。
- 宣言プロシージャーの実行が終了すると、制御はその実行の原因となったステートメントに関連した制御メカニズムに暗黙のうちに返されます。
- プログラムに手続き部がないか、またはプログラムで非宣言のセクションのいずれかが呼び出されたとき、その呼び出し側プログラムは暗黙のうちに EXIT PROGRAM を出します。

COBOL はまた、プロシージャーの分岐または条件ステートメントの実行によって、**明示的な**制御の転送を行います。

次の実行可能ステートメント

注: 「次の実行可能ステートメント」という用語は、上記の規則に従って、制御が転送される次の COBOL ステートメントを指します。以下の場合には、**次の実行可能ステートメント**は存在しません。

- プログラムに手続き部がない場合。
- 宣言セクションの最後のステートメントが書かれている段落が別の COBOL ステートメントの制御下で実行されていない場合の、その最後のステートメントの後。
- あるプログラムの最後のステートメントが書かれている段落がそのプログラム内の別の COBOL ステートメントの制御下で実行されていない場合の、その最後のステートメントの後。
- 宣言セクションの最後のステートメントが異なるセクションで実行される活動 PERFORM ステートメントの範囲内にあり、また活動 PERFORM ステートメントの出口であるプロシージャーの最後のステートメントでない場合の、その宣言セクションの最後のステートメントの後。
- COBOL プログラムの外に制御を転送させる STOP RUN、EXIT PROGRAM、または GOBACK ステートメントの後。
- END PROGRAM ヘッダーの後。

次の実行可能ステートメントがなく、制御が COBOL プログラムの外に転送されない場合には、プログラム実行が CALL ステートメントの制御下でプログラムの非宣言手続き部分にない限り、プログラムの制御の流れは不確定になります。この場合、暗黙の EXIT PROGRAM ステートメントが実行されます。

COBOL プログラムの構造

一般構造

COBOL ソース・プログラムは、正しい構文の一連の COBOL ステートメントです。

COBOL ソース・プログラムは、他の COBOL ソース・プログラムを含むことがあります。含まれる側のプログラムは、含む側のプログラムのリソースの一部を参照することがあります。

このようなプログラムを含むという概念は、ネストと呼ばれ、含まれるプログラムは**ネストされたプログラム**と呼ばれます。ネストされたプログラムは、直接または間接に別のプログラムに含まれます。例えば、プログラム B がプログラム A に含まれている場合、プログラム A に含まれプログラム B を含むような介在プログラムがなければ、プログラム B は「直接含まれる」ということになります。プログラム A に含まれプログラム B を含むような介在プログラムがあると、プログラム B は「間接的に」プログラム A に含まれます。

含まれるプログラムおよび含むプログラムについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」のネストされたプログラムに関するセクションを参照してください。

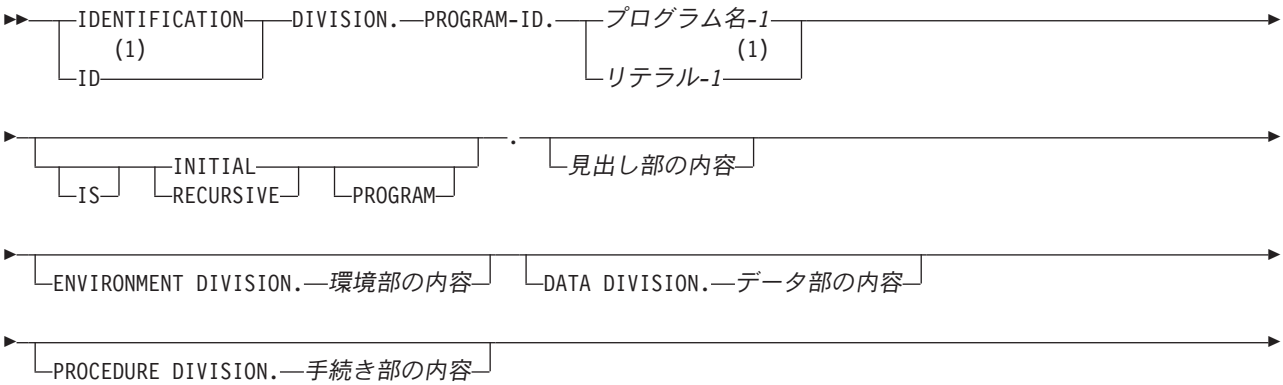
コンパイラー指示ステートメントと END PROGRAM ヘッダーを除き、COBOL ソース・プログラムのステートメント、項目、段落、およびセクションは、次の 4 つの部に分けられます。

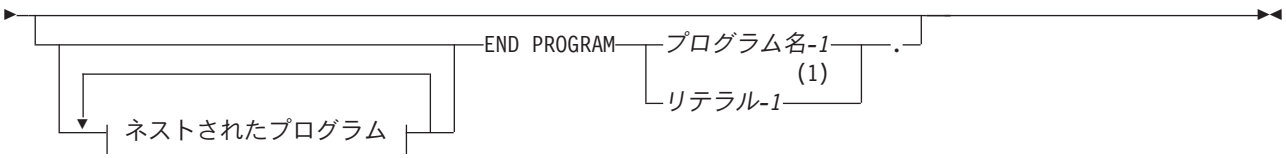
- 見出し部
- 環境部
- データ部
- 手続き部

COBOL ソース・プログラムの終わりは、END PROGRAM ヘッダー (指定された場合) で示されるか、またはそれ以上ソース・プログラムの行がないことによって示されます。

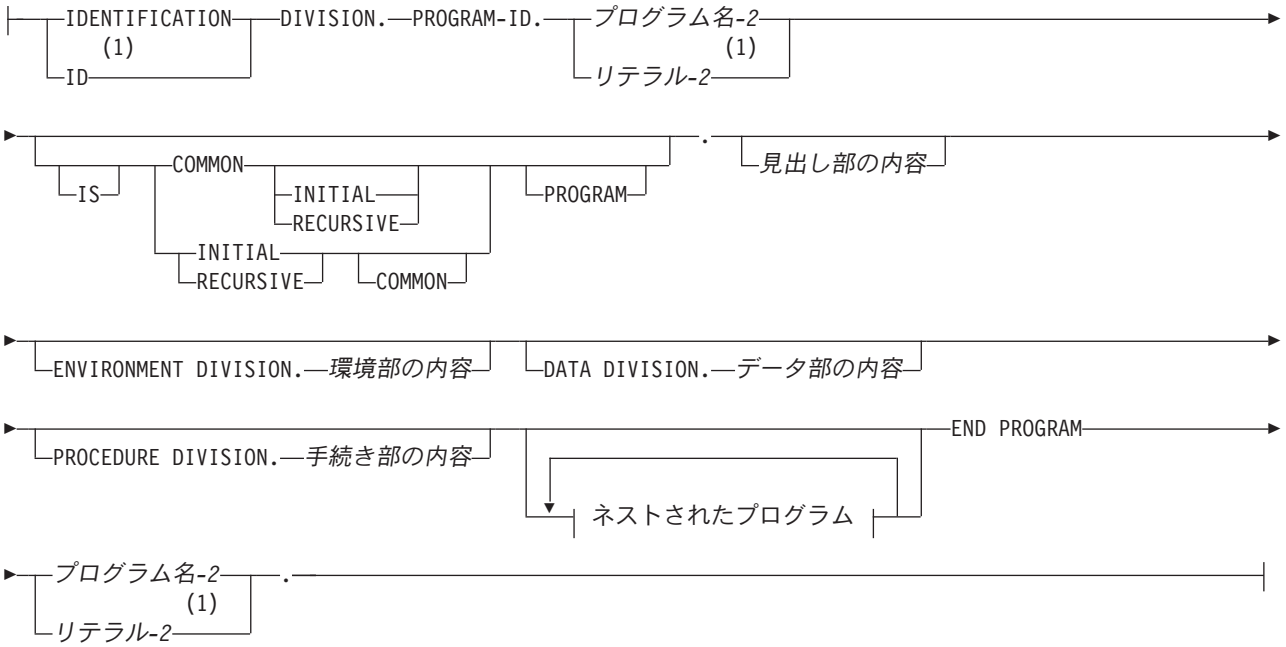
次の図に、COBOL プログラムの一般構造を簡単に図示してあります。

COBOL ソース・プログラム - 形式





ネストされたプログラム:



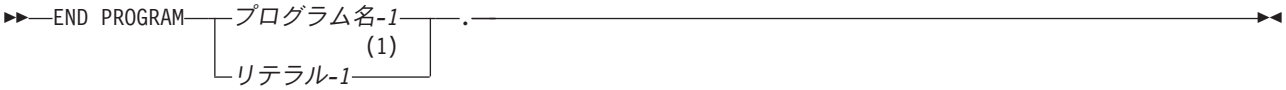
注:

1 IBM 拡張

END PROGRAM ヘッダー

END PROGRAM ヘッダーは、名前付き COBOL ソース・プログラムの終わりを示します。また、このヘッダーは、一連のソース・プログラムの中の個々のプログラムを分離します。一連のソース・プログラムの最後のプログラムがネストされたプログラムを含んでいなければ、その最後のプログラムの END PROGRAM ヘッダーはオプショナルです。

END PROGRAM ヘッダー - 形式



注:

1 IBM 拡張

プログラム名-1

直前の PROGRAM-ID 段落で宣言されたプログラム名と同一でなければならないユーザー定義語。プログラム名の作成法の規則については 4-2 ページの『PROGRAM-ID 段落』のプログラム名 を参照してください。

IBM Extension

リテラル-1

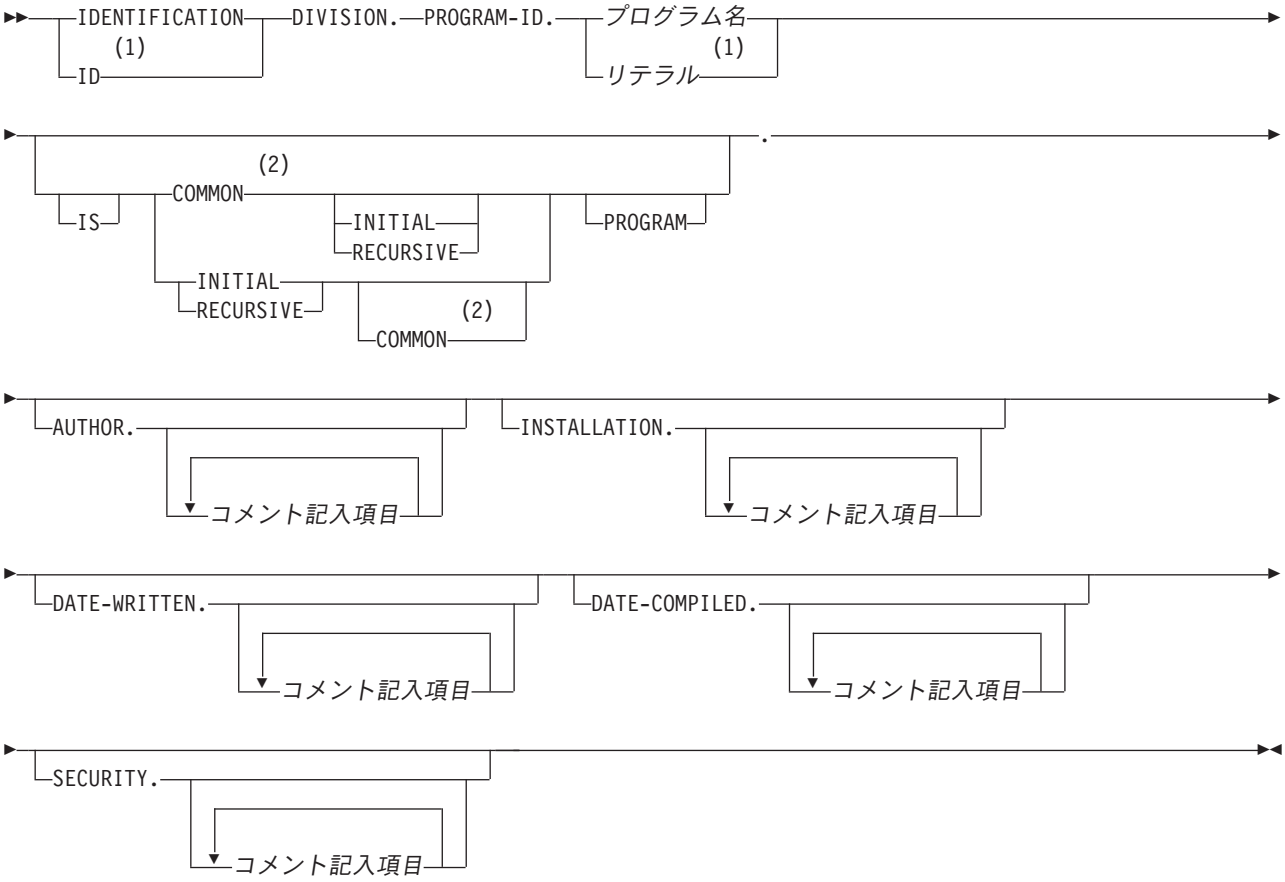
非数字リテラルでなければなりません。リテラルの作成法については 4-2 ページの『PROGRAM-ID 段落』のリテラル を参照してください。

End of IBM Extension

見出し部

見出し部は、あらゆる COBOL ソース・プログラムの最初の部分として必要です。この部分にはプログラムの名前、プログラム作成日、コンパイルの日付、その他プログラムについての記述を入れることができます。

見出し部 - 形式



注:

1 IBM 拡張

2 ネストされた COBOL プログラムでのみ使用できます。

見出し部の最初の段落は、PROGRAM-ID 段落でなければなりません。他の段落はオプショナルですが、書く場合には、形式に示されている順序でなければなりません。

IBM Extension

標準的な部のヘッダーの代わりに省略形 ID DIVISION を使用でき、オプショナルの段落は任意の順序とすることができます。

注: SEU 構文検査プログラムでは、次に示す段落ヘッダーの最初の文が段落ヘッダーと同じ行から始まる必要があります。

- PROGRAM-ID
- AUTHOR
- INSTALLATION
- DATE-WRITTEN
- DATE-COMPILED
- SECURITY

図 4-1 に、見出し部のコーディングの例を示します。

End of IBM Extension

```
IDENTIFICATION DIVISION.
PROGRAM-ID. IDSAMPLE.
AUTHOR. PROGRAMMER NAME.
INSTALLATION. COBOL DEVELOPMENT CENTER.
DATE-WRITTEN. 12/02/94.
DATE-COMPILED. 12/09/94 12:57:53.
SECURITY. NON-CONFIDENTIAL.
```

図 4-1. 文が段落と同じ行で始まることを示すための見出し部のコーディング例

PROGRAM-ID 段落

PROGRAM-ID 段落は、COBOL プログラムの名前を指定します。また、最も外側のプログラムでは、この段落にプログラム・オブジェクト (*PGM) またはモジュール・オブジェクト (*MODULE)、あるいはその両方の名前を指定できます。これは必要であり、しかも見出し部の最初の段落でなければなりません。

システムが認識するプログラム・オブジェクトの名前は、CRTBNDCBL コマンドの PGM パラメーターによって指定変更できます。モジュール・オブジェクトの認識用の名前は、CRTCBLMOD コマンドの MODULE パラメーターによって指定変更できます。 PGM または MODULE パラメーターについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

プログラム名

プログラムまたはモジュール・オブジェクトをシステムに対して認識させるユーザー定義語。プログラムおよびモジュール・オブジェクトの場合、プログラム名の最初の 10 文字だけがそのオブジェクトの識別名として使用されます。 ILE プロシージャー名の場合、プログラム名の最初の 250 文字が使用されます。CRTBNDCBL コマンドまたは CRTCBLMOD コマンドに *MONOPRC オプションを指定すると、プログラム名の先頭文字は英字にしなければなりません。数字にすると、次のように変換されます。

```
0       ->  J
1 ～ 9   ->  A ～ I
```

ハイフンが 2 ～ 10 桁目にある場合は、ゼロ (0) に変換されます。

*PGMID が、プログラムまたはモジュール名に対して CRTBNDCBL または CRTCBLMOD コマンドで指定されていて、*NOMONOPRC オプションが指定されている場合、PROGRAM-ID 段落で指定されているプログラム名が、小文字を含む非数値リテラルを決して含まないようにしてください。使用できないオブジェクトが作成される可能性があります。 プログラムまたはモジュール名に対する *PGMID がない場合で

も、複数のソース・プログラムには注意する必要があります。複数のソース・プログラムは、SIMPLEPGM = *YES の場合の PROGRAM-ID 段落に対して小文字を持つ非数値リテラルを含む、2 番目以降の ILE COBOL プロシージャーに対して、使用できないオブジェクトが作成される可能性があります。

リテラル

非数字リテラルでなければなりません。

囲み用の分離文字のない非数字リテラルは、プログラム名になります。上記のプログラム名の項に定義してあるとおり、モジュール名、プログラム名、およびプロシージャー名の作成でも、同じ規則が適用されます。ただし、*MONOPRC オプションを指定しないと、リテラル内の小文字は相当する大文字に変換されます。

RECURSIVE 文節

IBM Extension

RECURSIVE 文節は、COBOL プログラムの再帰的再入を可能にする、オプションの文節です。この文節は、プログラムおよびその中に含まれるすべてのプログラムが再帰可能であることを示します。ILE COBOL では、ネストされたプログラムでの RECURSIVE 文節の指定が可能です。さらに、再帰的プログラムは、ネストされたサブプログラムを含むことができます。RECURSIVE 文節が指定されていると、以前の呼び出しがまだアクティブなときに、プログラム名-1 に再帰的に入ることができます。 RECURSIVE 文節を指定しないと、アクティブ・プログラムに再帰的に再入することはできません。

再帰的プログラムの作業用ストレージ・セクションは、プログラムへ最初に入ったときに静的に割り振られ、初期化されるストレージを定義し、最後に使われた状態で、すべての再帰呼び出しに使用することができます。再帰的プログラム (および非再帰的プログラム) のローカル・ストレージ・セクションは、呼び出しごとに、割り振り、初期化、割り振り解除が自動的に行われるストレージを定義します。

再帰的プログラムのファイル・セクションで、FD に対応する内部ファイル・コネクターは静的に割り振られます。内部ファイル・コネクターの状況は、複数の呼び出し間で持続するプログラムの最後に使われた状態の一部です。

再帰的プログラムでは、以下の言語エレメントはサポートされていません。

- ALTER。 7-67 ページの『ALTER ステートメント』を参照
- 指定されたプロシージャー名のない GO TO。 7-119 ページの『GO TO ステートメント』を参照
- RERUN。 5-50 ページの『RERUN 文節』を参照
- SEGMENTATION。
- USE FOR DEBUGGING。 8-34 ページの『USE FOR DEBUGGING』を参照

このプログラムを直接または間接的に含むプログラムが最初のプログラムの場合、 RECURSIVE 文節は指定しないでください。

End of IBM Extension

COMMON 文節

COMMON 文節を使用すると、プログラム名で命名したプログラムを、その兄弟、およびその兄弟の中に含まれているプログラムによって呼び出すことができます。 COMMON 文節は、ネストされたプログラムでしか使用できません。

INITIAL 文節

プログラム名が呼び出されると、そのプログラム名と、その中に含まれるすべてのプログラムはそれぞれの初期状態に設定されることを指定します。 (作業用ストレージ項目はすべてそれぞれの初期状態にリセットされ、すべての INTERNAL ファイルはクローズされます。)

プログラムは次のような場合に初期状態に設定されます。

- ある実行単位内で最初にそのプログラムが呼び出されたとき
- INITIAL 属性をもったプログラムが呼び出されるたび
- そのプログラムを参照する CANCEL ステートメントか、またはそのプログラムを直接または間接的に含むプログラムを参照する CANCEL ステートメントの実行後、最初にそのプログラムが呼び出されたとき
- INITIAL 属性をもち、しかも直接または間接的にそのプログラムを含むプログラムを参照する CALL ステートメントの実行後、最初にそのプログラムが呼び出されたとき

  例えば、プログラム A がプログラム B を呼び出す場合に、プログラム B が INITIAL 属性をもち、さらにプログラム C を含んでいるとすると、A が B を呼び出した後で最初にプログラム C が呼び出されたときにプログラム C はその初期状態に設定されます。

プログラムが初期状態に設定されると、次のようになります。

- 作業用ストレージ・セクションおよびローカル・ストレージ・セクションに含まれるそのプログラムの内部データが初期設定される。データ項目の記述に VALUE 文節が使用されていると、そのデータ項目は定義値に初期設定される。 VALUE 文節がデータ項目に関連付けられていない場合は、そのデータ項目の初期値はコマンド CRTCBLMOD または CRTBNDCBL にオプション *STDINZ、*STDINZHEX00 または *NOSTDINZ が指定されているかいないかによって変わります。
- そのプログラムに関連付けられた内部ファイル・コネクターを持つファイルは、オープン・モードにならない。
- そのプログラムに含まれるすべての PERFORM ステートメントの制御メカニズムは、それぞれの初期状態に設定される。
- そのプログラムに含まれる変更済み GO TO ステートメントは、それぞれの初期状態に設定される。

このプログラムを直接または間接的に含むプログラムが再帰的プログラムの場合、 INITIAL 文節は指定しないでください。

オプショナルの段落

見出し部のオプショナルの段落は、省略可能です。

AUTHOR
プログラム作成者の名前。これは構文検査だけを受けます。

INSTALLATION
会社名または場所名。これは構文検査だけを受けます。

DATE-WRITTEN
プログラムが作成された日付。これは構文検査だけを受けます。

DATE-COMPILED
プログラムがコンパイルされた日付。

---

1. 兄弟プログラムとは、同じプログラムに直接含まれるプログラムのことです。

SECURITY
プログラムの機密性のレベル。

コメント記入項目

すべてのオプショナルの段落の**コメント記入項目**として、コンピューターの文字セットにある任意の文字を組み合わせて使用できます。コメント記入項目とコメント行を混同しないでください。 (後者は、標識域にスラッシュまたはアスタリスクで示されます。) コメント記入項目は、区域 B に 1 行または複数行を書きます。しかし SEU 構文検査プログラムでは、最初の文は段落ヘッダーと同じ行から始まっている必要があります。

コメント記入項目は説明として機能するだけです。この記入項目はプログラムの内容には影響を与えません。コメント記入項目では標識域 (7 桁目) のハイフンを使用することはできません。

段落名 DATE-COMPILED およびそれに関連したコメント記入項目は、コンパイル中に、次の形式の段落に置き換えられます。

```
DATE-COMPILED. current date.
```

SECURITY 段落のコメント記入項目の最初の 8 行には、作成するモジュール・オブジェクトの著作権情報が入ります。

IBM Extension

コメント記入項目では、その行の任意の場所に *CBL、*CONTROL、EJECT、 SKIP1、SKIP2、SKIP3、または TITLE ステートメントを入れられます。これらのステートメントは、コメント記入項目の行に単独で入れられている場合に作用するもので、コメント記入項目を終了することはありません。

コメントには、2 バイト文字と 1 バイト文字を組み合わせて使用できます。2 バイト・ストリングを含むコメント記入項目には複数の行を指定できますが、シフトアウト文字とシフトイン文字は、対にして同一行で使用しなければなりません。

注: 混合ストリングについては 2-2 ページの『文字ストリング』で説明されています。

End of IBM Extension

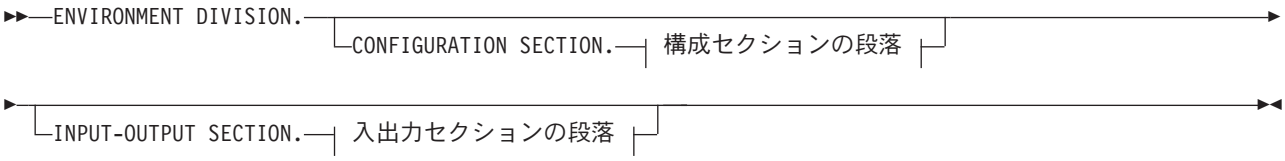
環境部

環境部は次の 2 つのセクションに分かれます。

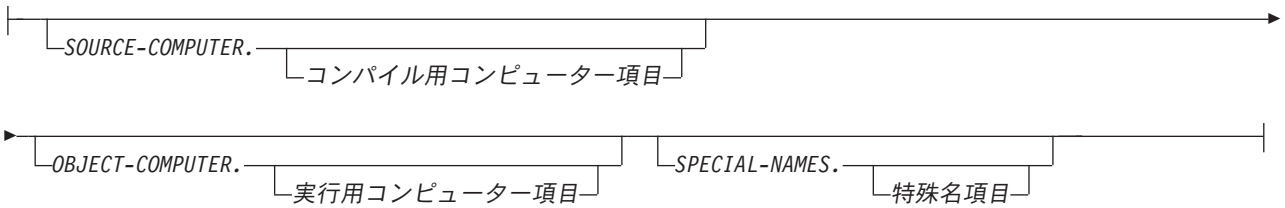
- 構成セクション
- 入出力セクション (5-23 ページの『入出力セクション』を参照してください。)

COBOL ソース・プログラムにおいては、環境部はオプショナルです。

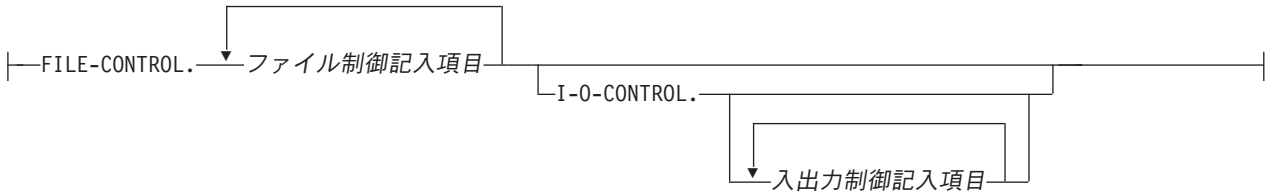
環境部 - 形式



構成セクションの段落:



入出力セクションの段落:



構成セクション

構成セクションはオプショナルです。これを指定すると、ソース・プログラムがコンパイルされるコンピューター、およびオブジェクト・プログラムが実行されるコンピューターを記述することができます。しかし構成セクションは、ネストされたプログラムには指定してはなりません。プログラムの構成セクションに指定する項目は、そのプログラムに含まれるすべてのプログラムに対して適用されます。

さらに構成セクションは、次のことを行うことができます。

- IBM 定義の環境名とユーザーが定義した簡略名とを関連させる。
- 照合順序を指定する。
- 通貨符号のための、単一または複数の通貨文字ストリングおよび置換文字を指定する。

- PICTURE 文節および数字リテラル内のコンマとピリオドの機能を交換する。
- 英字名を文字セットまたは照合順序に関連付ける。
- クラス名と文字セットを関連付ける。
- CALL、CANCEL、または SET...ENTRY ステートメントで行うリンケージのタイプを指定する。
- 日付または時刻のデータ・タイプのデフォルト形式を指定する。

各段落には、段落の最後の項目の直後に分離文字ピリオドが 1 つだけなければなりません。

注: SEU 構文検査機能を使用する場合、以下の段落の最初の文節は段落名と同じ行に記入しなければなりません。

- SOURCE-COMPUTER
- OBJECT-COMPUTER
- SPECIAL-NAMES

環境部の構成セクションには、次の 3 つの段落があります。

- SOURCE-COMPUTER 段落
- OBJECT-COMPUTER 段落
- SPECIAL-NAMES 段落

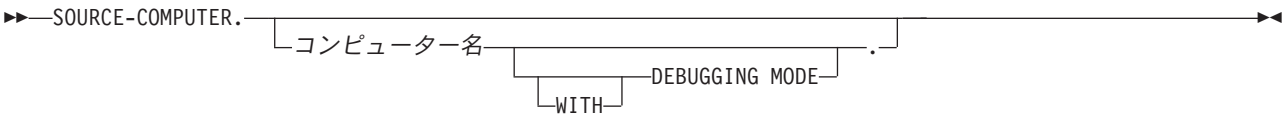
コーディング例

```
# ENVIRONMENT DIVISION.
# CONFIGURATION SECTION.
# SOURCE-COMPUTER. IBM-I.
# OBJECT-COMPUTER. IBM-I.
# SPECIAL-NAMES. C01 IS TOP-OF-PAGE.
```

SOURCE-COMPUTER 段落

SOURCE-COMPUTER 段落は、ソース・プログラムをコンパイルするコンピューターを記述します。

SOURCE-COMPUTER 段落 - 形式



コンピューター名

システム名。コンピューター名は、例えば次のようになります。

```
# IBM-I
```

WITH DEBUGGING MODE 文節を除き、SOURCE-COMPUTER 段落は、構文検査されますが、プログラムの実行には影響しません。

WITH DEBUGGING MODE 文節

ソース・プログラムに書かれたデバッグ行のコンパイル時スイッチを活動化します。

デバッグ行とは、コンパイル時スイッチが 活動化されたときしかコンパイルされないステートメントのことです。例えば、デバッグ行によってプロシージャーのある点でのデータ名の値を検査できます。

WITH DEBUGGING MODE 文節があると、USE FOR DEBUGGING プロシージャーはすべてコンパイルされます。この文節がなければ、これらのプロシージャーはコメントとして扱われ、無視されます。

プログラムにデバッグ行を指定するには、7 桁目 (標識域) に 'D' または 'd' をコーディングします。デバッグ行を連続して入れても構いませんが、各行の 7 桁目に 'D' または 'd' を入れなければならず、文字ストリングを 2 行にまたがって分けることはできません。

コンパイルされるか、それともコメントとして扱われるかにかかわらず、デバッグ行はすべてプログラムが構文的に正しくなるように書かなければなりません。

DEBUGGING MODE 文節があるかどうかは、すべての COPY ステートメントが処理された後に判別されます。詳細は 8-2 ページの『COPY ステートメント』を参照してください。

デバッグ行は、環境部 (OBJECT-COMPUTER 段落の後)、データ、あるいは手続き部にコーディングできます。

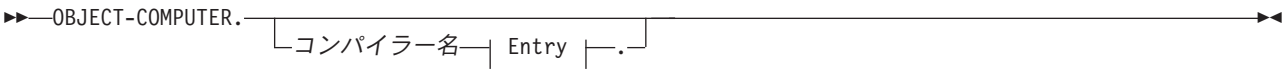
デバッグ行の区域 A および区域 B にスペースしかない場合は、ブランク行と同様に扱われます。

WITH DEBUGGING MODE 文節を省略すると、デバッグ行はコメント行として扱われます。

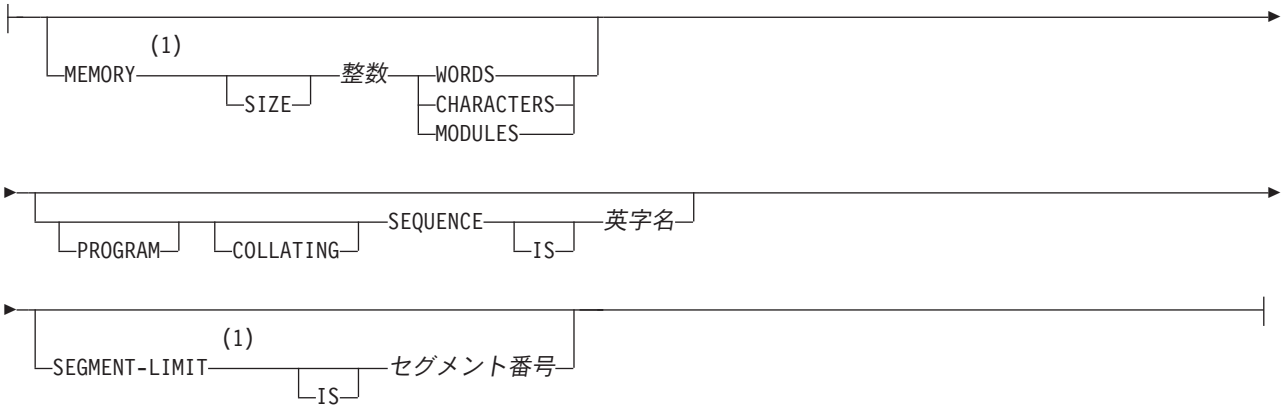
OBJECT-COMPUTER 段落

OBJECT-COMPUTER 段落は、オブジェクト・プログラムが実行されるシステムを指定します。

OBJECT-COMPUTER 段落 - 形式



記入項目:



注:

1 構文検査だけ行われます。

コンピューター名

システム名は構文検査されますが、プログラムの実行には影響しません。コンピューター名は、例えば次のようになります。

```
#      IBM-I
```

MEMORY SIZE

オブジェクト・プログラムを実行するために必要な主ストレージ量。MEMORY SIZE 文節には、構文検査だけが行われます。

整数

語、文字、またはモジュールで表されます。

PROGRAM COLLATING SEQUENCE IS

このプログラム (およびそれに含まれるすべてのプログラム) で使用される照合順序は、指定した英字名と関連付けられた照合順序です。

英字名

照合順序。

PROGRAM COLLATING SEQUENCE によって、次の非数字比較の真理値が決まります。

- 比較条件に明示的に指定されたもの
- 条件名条件に明示的に指定されたもの

MERGE または SORT ステートメントに COLLATING SEQUENCE 句が指定されていない限り、PROGRAM COLLATING SEQUENCE 文節はすべての非数字のマージまたはソート・キーにも適用されます。PROGRAM COLLATING SEQUENCE 文節が省略されると、EBCDIC 照合順序が使用されます。

これらの照合順序については 9-10 ページの『付録 C. EBCDIC および ASCII 照合順序』を参照してください。

SEGMENT-LIMIT IS

オブジェクト・プログラムの永久セグメントと見なされるセグメントを決定します。この文節には、構文検査だけが行われます。

セグメント番号

1 ～ 49 の可変値の整数でなければなりません。

SPECIAL-NAMES 段落

SPECIAL-NAMES 段落では、以下を行います。

- IBM 指定の環境名とユーザーが定義した簡略名とを関連付ける。
- 英字名を文字セットまたは照合順序に関連付ける。
- クラス名を文字セットに関連付ける。
- 通貨符号のための、単一または複数の通貨文字ストリングおよび置換文字を指定する。
- PICTURE 文節および数字リテラル内のコンマと小数点の機能が交換されることを指定する。

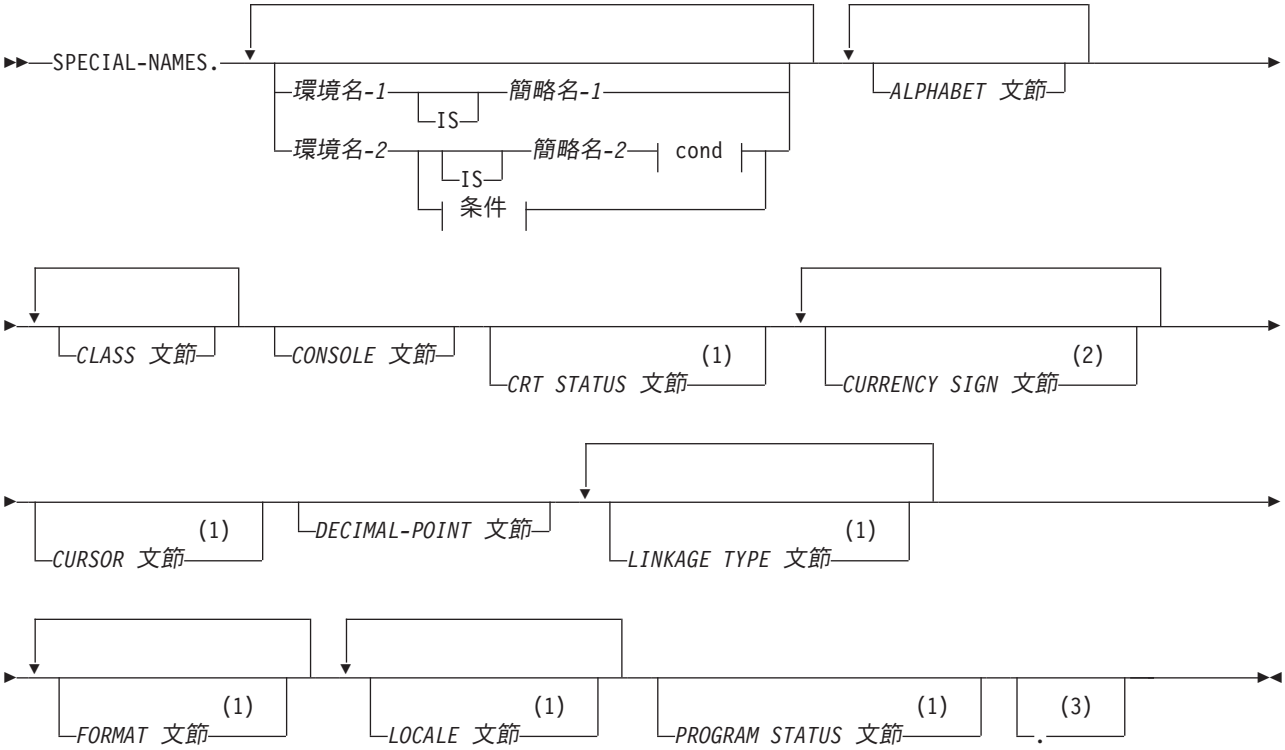
IBM Extension

- ロケール・オブジェクト名およびそれらと関連付けられているライブラリーをユーザー定義の簡略名に関連付ける。
- 日付または時刻のデータ・タイプのデフォルト形式を指定する。
- ACCEPT または DISPLAY ステートメントが、拡張 ACCEPT または DISPLAY ステートメントとして取り扱われるか、または動的画面管理プログラムのセッション・サービス API に対する要求として取り扱われるように指定する。
- ACCEPT ステートメントと関連する追加機能を指定する。

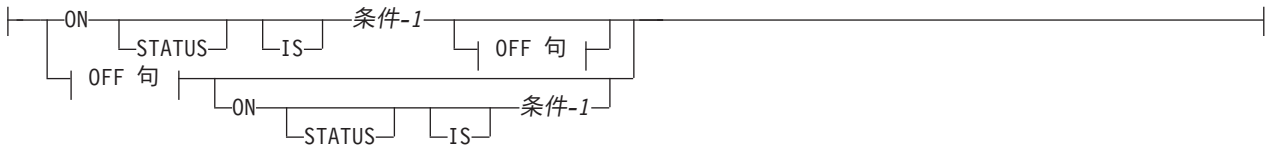
- プログラムの CALL または CANCEL や、SET ステートメントでのプロシージャー・ポインターの設定に使用するリンケージのタイプを指定する。

End of IBM Extension

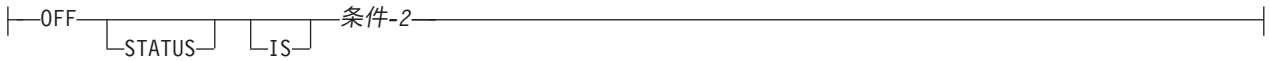
SPECIAL-NAMES 段落 - 形式



条件:



OFF 句:



注:

1 IBM 拡張

2 以降の反復は IBM 拡張です。

3 分離文字ピリオドは、オプショナルの文節が選択されたときに使用しなければなりません。文節は、任意の順序で入れられます。

環境名-1

コンパイラーが取るシステム装置または標準システムの処置。

表 5-1 は、環境名-1 の簡略名に関連付けられた処置を示しています。

表 5-1. 環境名-1 の種類およびその機能

環境名-1	環境名に関係付けられた簡略名が使用されるステートメント	使用法
CSP	WRITE	印刷を行う際、スペース送りを抑止します。装置が PRINTER のときだけ使用します。
C01	WRITE	次のページにスキップします。装置が PRINTER のときだけ使用します。
ATTRIBUTE-DATA	ACCEPT	トランザクション・ファイルにより獲得されたプログラム装置についての属性データを検索します。ただしファイルがオープンされているときに限られます。
I-O-FEEDBACK	ACCEPT	ファイル上で行われた最後の入出力操作についての情報を与えます。ただしファイルがオープンされているときに限られます。
DATA-AREA	ACCEPT、DISPLAY	システム・データ域を検索または更新します。
OPEN-FEEDBACK	ACCEPT	ファイルについての情報を与えますが、ファイルがオープンされているときに限られます。
CONSOLE、SYSTEM-CONSOLE	ACCEPT、DISPLAY	システム・オペレーターのメッセージ待ち行列 (QSYSOPR) と通信します。
LOCAL-DATA	ACCEPT、DISPLAY	ジョブごとにシステムが作成したローカル・データ域へデータを転送、あるいはローカル・データ域からデータを検索します。
PIP-DATA	ACCEPT	開始前のジョブの一部として実行されるプログラム用に、プログラム初期設定パラメーター (PIP) データ域からデータを取り出します。
REQUESTOR	ACCEPT、DISPLAY	ユーザー・ワークステーション (対話型ジョブ)、バッチ入力ストリーム、またはジョブ・ログ (バッチ・ジョブ) と通信します。
SYSIN	ACCEPT	REQUESTOR と同じ (ACCEPT ステートメントの場合のみ)。
SYSOUT	DISPLAY	REQUESTOR と同じ (DISPLAY ステートメントの場合のみ)。

環境名-2

環境名-2 は、UPSI-0 ～ UPSI-7 と定義しても、または SYSTEM-SHUTDOWN と定義してもかまいません。 UPSI とは、1 バイトのユーザー・プログラマブル状況標識スイッチのことです。

UPSI-7 ～ UPSI-0 は、COBOL プログラムの外部で定義されたプログラム・スイッチをオブジェクト時に識別する COBOL 名です。これらの内容は英数字と見なされます。ゼロの値はオフになり、1 の値がオンになります。

各スイッチは、制御言語 CHGJOB、SBMJOB、JOB、および JOBD コマンドの 8 文字の SWS パラメーターからの 1 バイトを次のように表します。

UPSI-0　1 番目のバイト (左端)
UPSI-1　2 番目のバイト
UPSI-2　3 番目のバイト
.
.
.
UPSI-7　8 番目のバイト (右端)

SYSTEM-SHUTDOWN は、システム・オペレーターがシステムをシャットダウン保留状態にした場合またはジョブがある制御手段によってキャンセルされた場合に ON にセットされる内部スイッチです。関連する ON または OFF 条件名は、条件名が有効なところではどこでも参照できます。これらの状況をプログラムによって変更することはできません。

簡略名-1、簡略名-2

簡略名-1 および簡略名-2 はユーザー定義名の作成の規則に従います。簡略名-1 は、ACCEPT、DISPLAY、および WRITE ステートメントで使うことができます。簡略名-2 は、SET ステートメントにおいてのみ参照することができます。簡略名-2 は条件-1 または条件-2 の名前を修飾できます。

簡略名と環境名は、固有名でなくてもかまいません。環境名にも使用される簡略名を選択すると、その名前に対する任意の参照では、簡略名としてのその定義の方が、環境名としての定義よりも優先されます。

ON STATUS IS、OFF STATUS IS

UPSI スイッチは、年初や年末の処理のようなプログラム内の特殊条件を処理します。例えば、手続き部の最初に UPSI スイッチをテストできます。それが ON であれば、特殊な分岐がとられます。
(7-22 ページの『スイッチ状況条件』を参照。)

条件-1、条件-2

環境名-2 が外部スイッチを参照すると、そのスイッチのオン/オフ状況が、条件-1、条件-2 といった条件名と関連付けられることがあります。スイッチの状況は、条件名を介して得ることができます。条件名は、ユーザー定義名の規則に従います。少なくとも 1 文字は英字でなければなりません。条件名に関連付けられた値は、英数字と見なされます。

手続き部では、UPSI スイッチ状況は、関連した条件名によりテストされます。すべての条件名はレベル 88 項目に等しくなります。それに関連した簡略名は、指定された場合には、条件変数と見なされ、修飾に使用できます。

パラメーターの SPECIAL-NAMES 段落に宣言された名前はすべて、包含されるどのプログラムからでも参照できます。

コーディング例

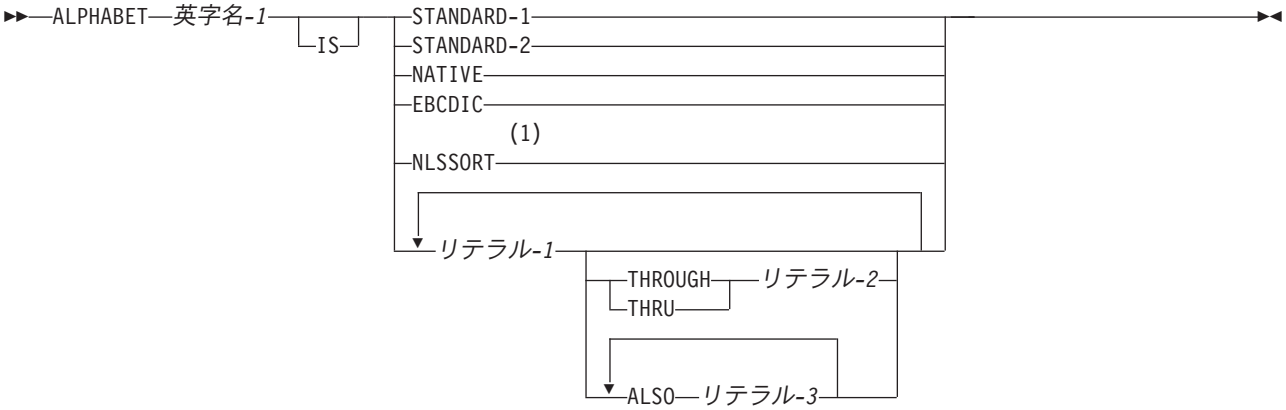
このコーディング例では、SPECIAL-NAMES 段落で一般に使用される環境名に簡略名を割り当てています

```
SPECIAL-NAMES.  SYSTEM-CONSOLE IS SYSTM
   REQUESTOR IS WORK-STATION
   C01 IS NEXT-PAGE
   LOCAL-DATA IS LOCAL-DATA-AREA
   ATTRIBUTE-DATA IS ATTRB-DATA
   SYSTEM-SHUTDOWN IS SHUTDOWN-SWITCH
       ON STATUS IS SHUTDOWN-PENDING
   UPSI-0 IS UPSI-SWITCH-0
       ON STATUS IS U0-ON
       OFF STATUS IS U0-OFF
   UPSI-1 IS UPSI-SWITCH-1
       ON STATUS IS U1-ON
       OFF STATUS IS U1-OFF
   IBM-ASCII IS STANDARD-1
   CURRENCY SIGN IS "Y".
```

ALPHABET 文節

ALPHABET 文節を使用すると、指定した文字コード・セットまたは照合順序に英字名を関連付けることができます。

ALPHABET 文節 - 形式



注:

1 IBM 拡張

次のいずれかで使用される際に、**照合順序**を指定します。

- OBJECT-COMPUTER 段落の PROGRAM COLLATING SEQUENCE 文節
- SORT または MERGE ステートメントの COLLATING SEQUENCE 句 。

次のもので指定するときには、**文字コード・セット**を指定します。

- CODE-SET 文節の FD 記入項目

注: EBCDIC 照合順序は、英字名文節が省略されると使用されます。

- ALPHABET 文節のコーディング例

英字名-1
英字名-1 は、ユーザー定義名の規則にしたがいます。少なくとも 1 文字は英字でなければなりません。英字名-1 は、特定の文字コード・セットまたは照合順序を識別します。

STANDARD-1
ASCII 文字セットを指定します。

STANDARD-2
情報処理交換のための国際規格 646 (7 ビット・コード化文字セットに定義されている) ISO 7 ビット・コードの国際基準バージョンを指定します。

NATIVE
EBCDIC 文字セットを指定します。

EBCDIC
EBCDIC 文字セットを指定します。

NLSSORT
英字名の別の照合順序のさまざまな面でコンパイラー・オプション (または暗黙のデフォルト) に

SRTSEQ および LANGID 仕様を使用します。 NLSSORT と関連した英字名に対する参照は、SORT および MERGE ステートメントの PROGRAM COLLATING SEQUENCE 文節か、または COLLATING SEQUENCE 句でしか行えません。

リテラル-1、リテラル-2、リテラル-3

次のような規則に従って、プログラムが照合順序を決定するということを指定します。

- リテラルが現れる順序は、この照合順序での文字の順序番号を昇順で指定します。
- 指定される各数字リテラルは、符号なしの整数で 1〜256 (EBCDIC 文字セットの最大文字数) の値をもたなければなりません。各リテラルの値は EBCDIC 文字セット内の文字の相対位置を指定します。例えば、次のとおりです。
  - リテラル 112 は EBCDIC 文字 ? を表します。
  - リテラル 234 は EBCDIC 文字 Z を表します。
  - リテラル 241 は EBCDIC 数字 0 を表します。

  EBCDIC および ASCII 照合順序での各文字の順序番号は 9-10 ページの『付録 C. EBCDIC および ASCII 照合順序』にリストされています。
- 非数字リテラルの中の各文字は、EBCDIC 文字セットの実際の文字を表します。 (非数字リテラルに複数の文字が含まれる場合には、このリテラルの左端から、各文字が照合順序の昇順で順次その位置を割り当てられます。)
- 明示的に指定されなかった EBCDIC 文字は、明示的に指定されたどの文字よりも上位の照合順序を持つものと見なされます。指定されなかった文字の相対順序は、EBCDIC 照合順序から変更されません。
- 1 つの英字名文節の中で、同一の文字を重複して指定してはなりません。
- THROUGH 句または ALSO 句に関連する各非数字リテラルは、長さが 1 文字でなければなりません (それ以上長い場合は、先頭文字だけが保管され、警告が発行されます)。
- THROUGH 句を指定すると、リテラル-1 で指定された文字で始まり、リテラル-2 で指定された文字で終わる連続した EBCDIC 文字に、この照合順序における連続した昇順の位置が割り当てられます。この順序は、元の EBCDIC 順序内の、昇順または降順のどちらでもかまいません。例えば、文字 Z 〜 S が指定されると、この照合順序での昇順値は ZYXWVUTS になります。
- ALSO 句を指定すると、リテラル-1、リテラル-3、... などと指定された EBCDIC 文字には、この照合順序内の同じ位置が割り当てられます。例えば、次のように指定します。

  ```
  "D" ALSO "N" ALSO "%"
  ```

  上記では、文字 D、N、および % はすべて照合順序の同じ位置にあると見なされます。
- SPECIAL-NAMES 段落のリテラルとして指定された表意定数 HIGH-VALUE および LOW-VALUE は、それぞれ 16 進値 FF および 16 進値 00 と関連付けられます。
- SPECIAL-NAMES 段落のすべての文節の処理が終わると、この照合順序で**最上位の**順序位置の文字は、表意定数 HIGH-VALUE と関連付けられます。ALSO 句の指定があるために複数の文字が最上位の位置をもっていると、最後に指定された文字 (または、特定の文字が固有照合順序内に明示的に指定されていないときのデフォルトの文字) は、DISPLAY のようなプロシージャー・ステートメントの HIGH-VALUE 文字と見なされるか、または MOVE ステートメントの送り出しフィールドと見なされます。 (固有照合順序内のすべての文字が明示的に指定され、かつ、上の例の ALSO 句が照合順序の高位文字として指定された場合には、HIGH-VALUE 文字は % になります。)
- SPECIAL-NAMES 段落のすべての文節の処理が終わると、この照合順序で**最下位の**順序位置の文字は、表意定数 LOW-VALUE と関連付けられます。ALSO 句の指定があるために、2 文字以上が最低

位置にある場合は、最初に指定された文字が LOW-VALUE 文字になります。 (上の例の ALSO 句が照合順序の低位文字として指定された場合、LOW-VALUE 文字は D になります。)

リテラル-1、リテラル-2、 またはリテラル-3 を指定した場合は、英字名を CODE-SET 文節で参照しては**なりません** (6-28 ページの『CODE-SET 文節』を参照)。

IBM Extension

DBCS リテラルおよび浮動小数点リテラルは、ユーザー指定の照合順序においては使用しないでください。

End of IBM Extension

コーディング例

次の例は、ALPHABET 文節のいくつかの使用法を示したものです。

PROGRAM COLLATING SEQUENCE IS USER-SEQUENCE の場合に、USER-SEQUENCE IS “D”、“E”、“F” と英字名文節を指定し、2 つのデータ部を次のように定義するとします。

```
77 ITEM-1 PIC X(3) VALUE "ABC".
77 ITEM-2 PIC X(3) VALUE "DEF".
```

上記の場合、次の比較は真になります。

```
IF ITEM-1 > ITEM-2
```

文字 D、E、および F はこの照合順序の順序位置の 1、2、および 3 です。文字 A、B、および C はこの照合順序の順序位置の 197、198、および 199 です。

英字名文節が USER-SEQUENCE IS 1 THRU 247、251 THRU 256、“7”、ALSO “8”、ALSO “9” の場合に、256 個の EBCDIC 文字がすべて指定されていて、そして 2 つのデータ部を次のように定義するとします。

```
77 ITEM-1 PIC X(3) VALUE HIGH-VALUE.
77 ITEM-2 PIC X(3) VALUE "789".
```

上記の場合、次の比較は真になります。

```
IF ITEM-1 = ITEM-2 . . .
IF ITEM-2 = HIGH-VALUE . . .
```

これらは、値 “7”、“8”、および “9” がすべてこの USER-SEQUENCE 照合順序で同じ位置 (HIGH-VALUE) を占めるので、真として比較します。

英字名文節を USER-SEQUENCE IS “E”, “D”, “F” と指定し、データ部のテーブルを次のように定義するとします。

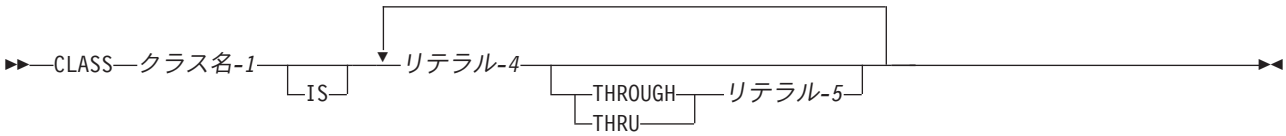
```
05 TABLE A OCCURS 6 ASCENDING KEY IS
   KEY-A INDEXED BY INX-A.
   10 FIELD-A ...
   10 KEY-A ...
```

上記の場合に、KEY-A の各オカレンスの昇順の内容が A、B、C、D、E、G であると、このテーブルの SEARCH ALL ステートメントの実行結果は、KEY-A の内容が昇順でないので無効になります。正しい昇順は、E、D、A、B、C、G です。

CLASS 文節

CLASS 文節は、その文節にリストされた指定文字セットに名前を関連付けます。

CLASS 文節 - 形式



クラス名-1

クラス名-1 はユーザー定義語であって、少なくとも 1 つの英字が入っていなければなりません。CLASS 文節のクラス名は、DBCS ユーザー定義語とすることができます。クラス名-1 は、クラス条件においてのみ参照できます。詳しくは、 7-10 ページの『クラス条件』を参照してください。この文節のリテラルの値により指定される文字は、このクラス名を構成する限られた文字セットを定義します。

リテラル-4、リテラル-5

数字の場合は、符号なしの整数で 1 ～ 256 の値になります (EBCDIC 文字セットの最大文字数)。

各リテラルの値は、EBCDIC 文字セット内での文字の相対位置、または順序番号を指定します。EBCDIC 照合順序の各文字の順序番号は 9-10 ページの『付録 C. EBCDIC および ASCII 照合順序』にリストされています。

IBM Extension

浮動小数点リテラル、DBCS リテラル、または国別リテラルとして指定できません。

End of IBM Extension

非数字の場合は、リテラルは EBCDIC 文字セット内の実際の文字です。非数字リテラルの値に複数の文字が入っている場合には、リテラル中の各文字はクラス名によって識別される文字セットに含まれます。

非数字リテラルが THROUGH 句に関連付けられている場合は、1 文字の長さでなければなりません。

THROUGH、THRU

THROUGH を指定すると、クラス名にはリテラル-4 の値で始まり、リテラル-5 の値で終わる文字が入ります。さらに、THROUGH 句によって指定される文字は、昇順または降順のいずれかで文字を指定することができます。

CONSOLE 文節

IBM Extension

CONSOLE IS CRT を指定すると、特定の形式固有の句をもたないすべての ACCEPT または DISPLAY ステートメント (LOCAL-DATA または PIP-DATA) は、拡張された ACCEPT または DISPLAY ステートメントとして扱われます。

同様に、CONSOLE IS DISPLAY を指定すると、特定の形式固有の句をもたないすべての ACCEPT または DISPLAY ステートメントは、動的画面管理プログラム・セッション・サービス API に対する要求とし

て取り扱われます。これらの API については、**IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL および API*』を参照してください。

CONSOLE 文節 - 形式

注:

1 IBM 拡張

CONSOLE IS 文節を指定しないと、特定の形式固有の句をもたないすべての ACCEPT または DISPLAY ステートメントは、標準の ANSI COBOL の ACCEPT または DISPLAY ステートメントとして取り扱われます。

ACCEPT または DISPLAY ステートメントが拡張ステートメントであるか標準ステートメントであるかを判別する条件の説明は 7-60 ページの『拡張 ACCEPT および拡張 DISPLAY の考慮事項』および 7-100 ページの『形式 3 - 拡張 DISPLAY ステートメント』を参照してください。

End of IBM Extension

CRT STATUS 文節

IBM Extension

CRT STATUS 文節は、拡張 ACCEPT ステートメント実行後の状況値の転送先のデータ項目を指定します。

CRT STATUS 文節 - 形式

注:

1 IBM 拡張

データ名-2
このデータ名は、WORKING-STORAGE または LOCAL-STORAGE SECTIONS 内で記述されていなければならず、また、6 バイト英数字フィールドまたは 6 バイトの符号なしのゾーン整数でなければなりません。データ名-2 をネストされたプログラムから参照する場合には、これを最も外側のプログラムでグローバルとして定義しなければなりません。

CRT STATUS 文節についての考慮事項

CRT STATUS 文節が SPECIAL-NAMES 段落で指定されている場合は、拡張 ACCEPT ステートメントが実行されるたびに、その ACCEPT 操作の結果を示す値がデータ名-2 に入れられます。データ名-2 は、操作の完了によって生じる指定可能な条件を示すように設定されたいくつかの状況キーにより構成されます。

CRT 状況キー 1: データ名-2 の最初の 2 バイトが CRT 状況キー 1 となり、これは PIC 99 と記述しなければなりません。このキーは、ACCEPT 操作を終了させる原因となった条件を示します。指定可能な値は次のとおりです。

0 終了キー (例えば、改行キー)、あるいは最終フィールドからの自動スキップを示します。

1 ファンクション・キーであることを示します。

9 エラーがあることを示します。

ACCEPT ステートメントに ON EXCEPTION 句が含まれている場合は、 CRT 状況キー 1 内にどのような値 (0 を除く) が入っていても、ON EXCEPTION 句内の命令ステートメントが実行されることになります。

CRT 状況キー 2: データ名-2 の次の 2 バイトが CRT 状況キー 2 を形成し、ACCEPT 操作を終了させる原因となった条件の詳細を示すコードが入ります。このキーの形式および可能な値は、次の表に示すように、CRT 状況キー 1 内の値によって異なります。

表 5-2. CRT STATUS キー 1 および 2 の有効な組み合わせ

キー 1	キー 2		意味
	フォーマット	値	
0	PIC 99	0	オペレーターが終了キーを押した
0	PIC 99	1	最後のフィールドからの自動スキップ[1]
1	PIC 99	1 ～ 24	ファンクション・キーの番号
9	PIC 99	0	エラー条件 (画面内に入る項目が存在しない)
注: [1] 最後のフィールドからの自動スキップが生じた場合、CRT STATUS KEY 2 の値 1 がサポートされている制御装置に返され、値 0 がサポートされていない制御装置に返されます。この関係は表 5-3 に記載されています。			

表 5-3. 制御装置タイプによって戻される自動スキップ値

制御装置のタイプ	戻された自動スキップ値 1
IBM i 制御装置: ローカル・ワークステーション制御装置 リモート 5251-12 型 リモート 5294 リモート 5394 リモート 3174 リモート 3274	 はい 該当せず いいえ 新ワークステーション制御装置コードを 使ったインストールのときは、あり *NOUNDSPCHR オプションを使ったとき は、なし *NOUNDSPCHR オプションを使ったとき は、なし
PC 接続機構 DOS およびオペレーティング・ システム /2 (OS/2) の操作 環境	 No
システム間パススルー IBM i システムから IBM i システム System/36™ から IBM i システム System/38™ から IBM i システム	 はい いいえ No

CRT 状況キー 3: データ名-2 の最後の 2 バイトが CRT 状況キー 3 です。CRT 状況キー 1 が 0 のときには、CRT 状況キー 3 には、ACCEPT 操作を終了させたキーボード・キーのコードが入ります。 CRT STATUS キー 1 が 9 のときは、エラーであることがオペレーティング・システムによって通知され、CRT STATUS キー 3 が 99 に設定されることになります。

これらのキーのコードには、次のものがあります。

- 00 実行キー
- 90 次ページ・キー
- 91 前ページ・キー
- 93 ヘルプ・キー
- 94 クリア・キー

ヘルプ・キーとクリア・キーは、ローカル IBM i ワークステーションでのみデータを受け入れます。

End of IBM Extension

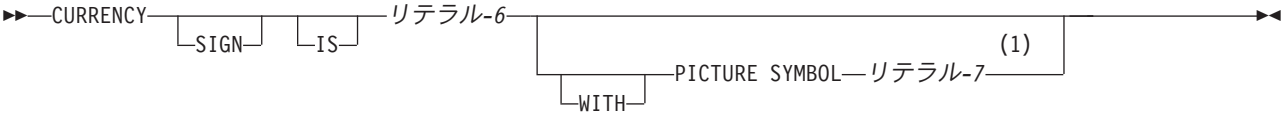
CURRENCY SIGN 文節

CURRENCY SIGN 文節は、次のような**通貨記号**を定義するために使用されます。

- 受け入れ項目として使用されるときの数字編集データ項目に挿入される。
- 項目の未編集数値を決定するときに、(編集解除した) 数字データ項目から除去される。

さらに、この文節は、PICTURE 文字ストリング内で通貨ストリングを表すために使用する記号を指定する際に使用される場合もあります。この記号を**通貨記号**と呼びます。

CURRENCY SIGN 文節 - 形式



注:

1 IBM 拡張

IBM Extension

注: CURRENCY SIGN 文節を繰り返すことによって、COBOL プログラム内で複数の通貨ストリングを使用することができます。ただし、通貨記号の値は**重複させないで**ください。

End of IBM Extension

CURRENCY SIGN 文節を省略する場合には、円記号 (¥) を通貨ストリングの値および通貨記号の両方に使用する必要があります。

PICTURE SYMBOL 句のないリテラル-6

通貨記号として使用される文字と、さらに**通貨ストリング**の値を指定します。単一文字で、非数字リテラルでなければならず、また次の文字であっては**なりません**。

- 0 ～ 9 の数字
- 大文字の英字 A B C D P R S V X Z またはこれらの文字の小文字

- スペース
- 特殊文字 * + - /, . ; ( ) = "
- 表意定数

IBM Extension

- プログラムが外部浮動小数点項目を定義している場合は、大文字の英字 E。
- プログラムが DBCS または国別項目を定義している場合は、大文字の英字 G および N。
- 小文字の英字 e、g、および n。

End of IBM Extension

通貨記号には大文字小文字の区別があり、プログラムの中では必ず CURRENCY SIGN 文節で使用した文字とまったく同じ文字を指定する必要があります。ただし、OPTION パラメーター値 *NOMONOPIC、または PROCESS ステートメント・オプション NOMONOPIC が指定されていなければ、PICTURE 文字ストリングで使用される英字通貨記号は、実際の表記に関係なく大文字と見なされます。したがって、英字通貨記号は、NOMONOPIC オプションが指定されていないのであれば、常に英大文字で入力する必要があります。

PICTURE SYMBOL 句付きリテラル-6

PICTURE SYMBOL 句が指定されている場合、リテラル-6 は**通貨ストリング**の値を指定し、リテラル-7 は通貨記号を表します。リテラル-6 は、任意の長さ (複数文字) であり、コンピューターの文字セットから以下を**除いた**任意の文字から構成できます。

- 0 ～ 9 の数字
- 特殊文字 * + - / . ,
- 他の文字がない 1 つまたは複数のスペース

リテラル-7

PICTURE SYMBOL 句が指定されている場合、リテラル-7 は**通貨記号**として使用される文字を指定します。これは、単一文字で、非数字リテラルでなければならず、またプログラムで定義された他の通貨記号と同じ値であっては**なりません**。この文字の値は、PICTURE SYMBOL 句が省略される場合に通貨記号 (リテラル-6) に適用されるものと同じ制約事項に従います。

CURSOR 文節

IBM Extension

CURSOR 文節は、拡張 ACCEPT ステートメントで使用されるカーソル・アドレスが入るデータ項目を指定します。

CURSOR 文節 - 形式

注:

1 IBM 拡張

データ名-1

4 バイトまたは 6 バイトの英数字フィールド、あるいは 4 バイトまたは 6 バイトの符号なしのゾーン整数フィールドでなければなりません。データ名-1 の長さが 4 文字のときは、最初の 2 文字は行番号として解釈され、残りの 2 文字は桁番号として解釈されます。データ名-1 の長さが 6 文字のときは、最初の 3 文字が行番号として解釈され、残りの 3 文字は桁番号として解釈されます。

データ名-1 に無効な位置値 (例えば、ゼロ、非数字の値、または画面の範囲を超える値など) が入っている場合は、この文節は効力をもちません。

データ名-1 は、WORKING-STORAGE または LOCAL-STORAGE SECTIONS に記述しなければなりません。データ名-1 をネストされたプログラムから参照する場合には、最も外側のプログラムでグローバルと定義しなければなりません。

CURSOR 文節についての考慮事項

拡張 ACCEPT 操作の開始時に、データ名-1 に画面上の有効な位置を示す値が入っている場合は、その位置がカーソルの初期位置として使用されます。有効な位置とは、画面上の座標の 1 つを指します (つまり、行 1、列 1 から行 24、列 80 までの範囲内にあるもの)。 ACCEPT 操作の終了後、データ名-1 に入っている位置が有効なものであった場合には、その時点のカーソル位置を示すようにデータ名-1 が更新されます。

CURSOR IS ID に無効な値が入っている場合 (例えば、スペース、小さすぎる値、大きすぎる値、または画面範囲外の値など) には、カーソルは、画面上で活動状態にある最初の入力フィールドの先頭に置かれます。

CURSOR IS は、画面上のフィールドの位置付けには影響を与えません。

End of IBM Extension

DECIMAL-POINT IS COMMA 文節

DECIMAL-POINT IS COMMA 文節は、PICTURE 文字ストリングおよび数字リテラルのピリオドとコンマの機能を交換させます。

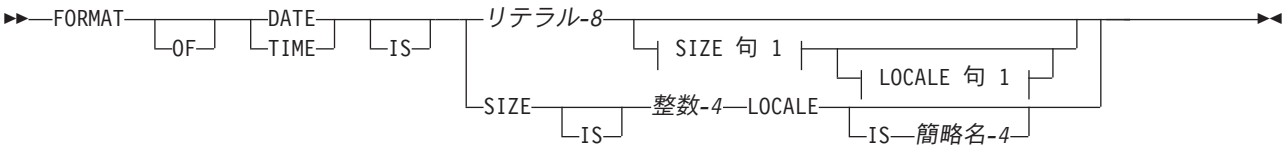
DECIMAL-POINT IS COMMA 文節 - 形式

FORMAT 文節

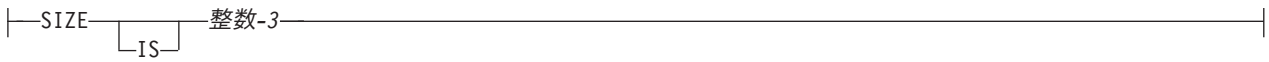
IBM Extension

FORMAT 文節は、DATA DIVISION の日付項目または時刻項目のデフォルトの形式を指定するために使用します。また、FORMAT 文節は、組み込み関数のデフォルトの日付形式または時刻形式を指定することもできます。

FORMAT 文節 - 形式



SIZE 句 1:



LOCALE 句 1:

リテラル-8

日付項目または時刻項目のデフォルトの形式を指定します。リテラル-8 は、その長さが最小でも 2 文字の非数字リテラルでなければなりません。リテラル-8 には 1 つまたは複数の変換指定子とゼロまたはそれ以上の分離文字が含まれている必要があります。リテラル-8 が LOCALE 句に及ぼす影響の詳細は 5-19 ページの『LOCALE 句』を参照してください。リテラル-8 で使用することができる変換指定子のリストは 表 5-4 に記載されていますので、それを参照してください。

次の規則が適用されます。

- リテラル-8 とともに LOCALE 句を指定しない場合は、変換指定は COBOL ロケールに基づく値に置き換えられる。 COBOL ロケールについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。
- 日付項目に関しては、リテラル-8 には変換結果が年間通算日となる変換指定子が含まれていなければならない。リテラル-8 に年と月の変換指定は含まれていても、日についての変換指定が含まれていない場合は、当該月の最初の日が想定されます。 IBM i の日付形式およびそれらと等価のリテラル-8 日付形式のリストは 6-41 ページの表 6-2 に記載されていますので、それを参照してください。
- 日付項目に対して FORMAT 文節を指定しない場合は、ISO 形式がデフォルトの日付項目形式として使用される。
- リテラル-8 を指定しない場合は LOCALE 句を指定しなければならない。
- 時刻項目に関しては、リテラル-8 には時および分の変換指定が含まれていなければならない。秒 (またはミリ秒) を指定しない場合は値 0 が想定されます。 IBM i の時刻形式およびそれらと等価のリテラル-8 時刻形式のリストは 6-41 ページの表 6-3 に記載されていますので、それを参照してください。
- 時刻項目に対して FORMAT 文節を指定しない場合は、ISO 形式がデフォルトの時刻項目形式として使用される。

表 5-4 に、リテラル-8 で使用することができる変換指定子をリストします。

表 5-4. リテラル-8 で使用できる変換指定子

指定子	内容	長さ	用途
@C	世紀を表す 1 つの整数 [0,9] に置き換えられる (0⁴ は 20 世紀)	1 バイト	D

表 5-4. リテラル-8 で使用できる変換指定子 (続き)

指定子	内容	長さ	用途
%d	月の中の日にちを表す 1 つの整数 [01,31] に置き換えられる	2 バイト	D
%D	%m/%d/%y と同じ	8 バイト	D
%H	時を表す 1 つの整数 [00,23] に置き換えられる (24 時間時計)	2 バイト	T
%I	時を表す 1 つの整数 [01,12] に置き換えられる (12 時間時計)	2 バイト	T
%j	年間通算日 (年初から数えた日にち) を表す 1 つの整数 [001,366] に置き換えられる	3 バイト	D
%m	月を表す 1 つの整数 [01,12] に置き換えられる	2 バイト	D
%M	分を表す 1 つの整数 [00,59] に置き換えられる	2 バイト	T
%p	ロケールと等価の a.m. または p.m. のいずれかに置き換えられる	ロケール	T
@p	AM および PM については、大文字小文字を任意に混合させることができる	2 バイト	T
%r	a.m. および p.m. での時刻表記に置き換えられる。POSIX ロケールでは %I:%M:%S %p と等価。	ロケール、最小でも 8 バイト	T
%R	24 時間表記の時刻 [%H:%M] に置き換えられる	5 バイト	T
%S	秒を表す 1 つの整数 [00,61] に置き換えられる	2 バイト	T
@Sh	100 分の 1 秒単位の秒を表す 1 つの整数 [00,99] に置き換えられる	2 バイト	T
@Sm	100 万分の 1 秒単位の秒を表す 1 つの整数 [000000,999999] に置き換えられる	6 バイト	T
@So	1000 分の 1 秒単位の秒を表す 1 つの整数 [000,999] に置き換えられる	3 バイト	T
@St	10 分の 1 秒単位の秒を表す 1 つの整数 [0,9] に置き換えられる	1 バイト	T
%y	年を表す 1 つの整数 [00,99] に置き換えられる (世紀を併記せず)	2 バイト	D
%Y	年を表す 1 つの整数に置き換えられる (世紀を併記)	通常 4 バイト	D
@Y	年を表す 1 つの整数に置き換えられる (世紀を併記)	4 バイト	D
%%	1 つの % に置き換えられる	1 バイト	D、T
@@	1 つの @ に置き換えられる	1 バイト	D、T

表 5 の注:

1. 変換指定子では大文字と小文字は区別されます。
2. 用途欄の記号の意味は次のとおりです。
   - D - DATE 項目
   - T - TIME 項目
3. 長さの欄は EBCDIC 単一バイト・エンコード体系 (CCSID 37) が適用されるデフォルトの COBOL ロケールに基づいています。
4. デフォルトでは、ゼロの値は 20 世紀 (1900 年 ～ 1999 年) を表します。この値は DATTIM PROCESS ステートメント・オプションに指定する基本世紀 (base century) に基づきます。

SIZE 句

SIZE 句では、日付項目または時刻項目の合計サイズを桁数で指定します。この桁数は形式リテラルのサイズ以上でなければなりません。形式リテラルのサイズは、変換指定子をそれらの最大値によって置き換え、必要な場合はさらに実行時 CCSID への変換を行うことによって判別されます。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」に記載されている、CRTCBLMOD の CCSID パラメーターの説明を参照してください。

日付項目または時刻項目の長さがコンパイル時に判別できない場合は、これらの項目について SIZE 句を指定する必要があります。コンパイラーが日付項目または時刻項目のサイズを判別できない場合を以下に示します。

- リテラル-8 と LOCALE 句の両方が指定されている場合。これは、日付項目または時刻項目には指定されたロケールから実行時に判別される部分が含まれるため、実際の長さは実行時に決まるということを意味します。
- リテラル-8 は指定されているが LOCALE 句は指定されておらず、リテラル-8 の中に指定されている変換のうちの 1 つを実行した結果が可変長項目になる可能性がある場合。
- リテラル-8 が指定されていない 場合。これは、日付項目または時刻項目のすべての内容は指定されたロケールから実行時に判別されるため、実際の長さが決まるのは実行時であることを意味します。

整数-3、整数-4

整数-3 および整数-4 では、デフォルトの日付項目または時刻項目のサイズを桁数で指定します。日付項目または時刻項目のサイズがコンパイル時に判別できない場合は、整数-3 または整数-4 を指定する必要があります。日付項目および時刻項目については、整数-3 および整数-4 は 4 以上でなければなりません。日時クラスの項目の最大サイズは、その項目に USAGE DISPLAY が指定されている場合、またはその項目の USAGE PACKED-DECIMAL に 31 が指定されている場合は 256 桁となります。

LOCALE 句

LOCALE 句は、日付項目および時刻項目のフォーマット設定に使用する、特定の文化圏固有のロケールを指定するために使用します。

リテラル-8 を*指定せずに* LOCALE 句を指定する場合は、日付項目または時刻項目の形式および分離文字はすべてロケールに基づきます。リテラル-8 を*指定した上で* LOCALE 句を指定すると、項目の形式はリテラル-8 から判別されますが、正確な表記 (例えば、%p) のロケールに従う変換指定子の置き換えに使用される値は、そのロケールに基づいた値です。

簡略名-3、簡略名-4

簡略名-3 または簡略名-4 が指定されている 場合は、日付項目または時刻項目のために使用されるロケールは SPECIAL-NAMES 段落内の簡略名-3 または簡略名-4 に関連付けられているロケールとなります。簡略名-3 または簡略名-4 が指定されていない 場合は、現行ロケールが使用されます。現行ロケールを判別する場合は、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の説明を参照してください。

簡略名-3 および簡略名-4 はロケール簡略名でなければなりません。ロケール簡略名は SPECIAL-NAMES 段落の LOCALE 文節で指定します。 5-21 ページの『LOCALE 文節』を参照してください。

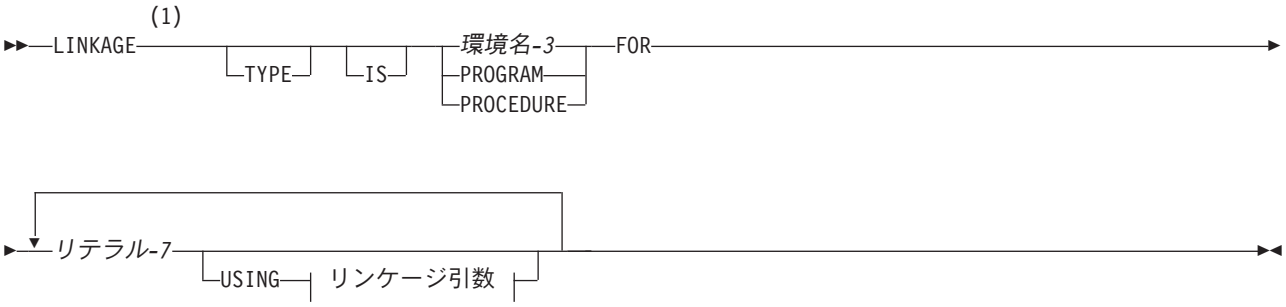
End of IBM Extension

LINKAGE TYPE 文節

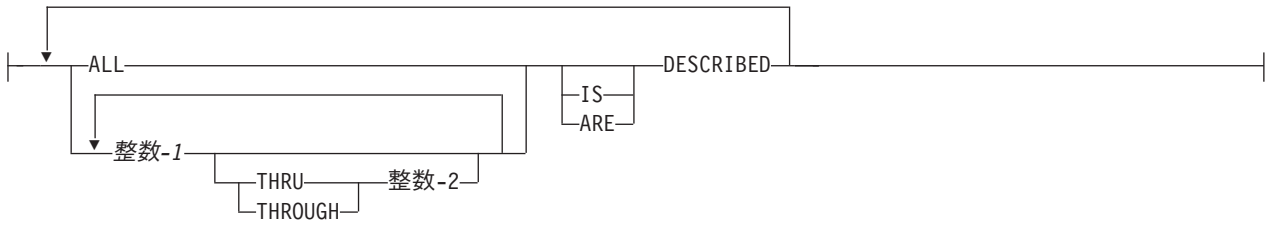
IBM Extension

LINKAGE TYPE 文節は、リテラル-7 で指定されたプログラムに対する CALL またはそのプログラムの CANCEL で行うリンケージのタイプを指定するとともに、また、それを SET ステートメントで行うリンケージのタイプにすることを指定します。

LINKAGE TYPE 文節 - 形式



リンケージ引数:



注:

1 IBM 拡張

環境名-3

環境名-3 は次のように定義できます。

PGM プログラム・オブジェクト (*PGM) へのリンケージが生成されます。

PRC ILE プロシージャーへのリンケージが生成されます。

SYS システム提供のプロシージャーへのリンケージが生成されます。

PROGRAM

プログラム・オブジェクト (*PGM) へのリンケージが生成されます。これは、PGM の環境名-3 と同義です。

PROCEDURE

ILE プロシージャーへのリンケージが生成されます。これは、PRC の環境名-3 と同義です。

リテラル-7

リテラル-7 は、プログラム・オブジェクトまたはプロシージャーの名前です。リテラル-7 は、拡張名を含むことがあります。その名前は、プログラムでは最長 10 文字、プロシージャー名では最長 256 文字にできます。リテラル-7 は、OPTION(*MONOPRC) パラメーターの影響を受けます。

*MONOPROC を指定すると、小文字は大文字に変換され、プログラム名を作成する際の規則が適用されます。詳細は、 4-2 ページの『PROGRAM-ID 段落』にある**プログラム名**のセクションを参照してください。

USING
どのパラメーターの操作記述子を呼び出し先のプロシージャーから使用できるようにするのかを指定します。これらのパラメーターは、USAGE DISPLAY または USAGE DISPLAY-1 の指定付き基本データ項目として定義する必要があります。これらを参照変更することはできません。

USING 文節は、リンケージ・タイプのプロシージャーに使用でき、CALL ステートメントにしか適用されません。

整数-1、整数-2
ゼロ以外の正の整数でなければなりません。操作記述子を使って記述されるすべてのパラメーターの順序位置を指定します。

整数-2 は整数-1 より大きくなければなりません。

DESCRIBED
整数-1 THROUGH 整数-2 で指定したパラメーターは、対応する操作記述子とともに渡されます。ALL を指定すると、適用できるのであれば、そのプロシージャー用に定義されたすべてのパラメーターが対応する操作記述子とともに渡されます。

LINKAGE TYPE 文節についての考慮事項

いくつかの方法で、CALL、CANCEL、または SET 用に生成されるリンケージのタイプに影響を与えることができます。その方法は、優先順位の順番でリストされます。CALL、CANCEL または SET ステートメントの LINKAGE 句が、最優先順位になります。ステートメントに LINKAGE 句の指定がなく、しかもネストされた可視のプログラムがない場合に LINKAGE TYPE 文節が指定されると、これが使用されます。優先順位は次のとおりです。

- ステートメントの LINKAGE 句
- ネストされたプログラムに対する CALL または CANCEL
- SPECIAL-NAMES 段落の LINKAGE TYPE 文節
- CRTCBLMOD または CRTBNDCBL コマンドの LINKLIT パラメーター

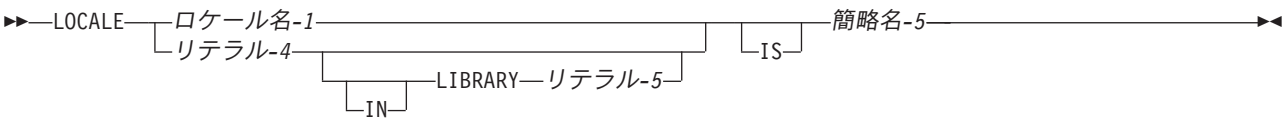
End of IBM Extension

LOCALE 文節

IBM Extension

LOCALE 文節は、ロケール簡略名と IBM i においてそれらと等価のロケール・オブジェクト名およびライブラリーを定義するために使用します。

LOCALE 文節 - 形式



ロケール名-1
ロケール・オブジェクトを参照するシステム特定名を指定します。 ILE COBOL の場合、サポートされているロケール名-1 は POSIX だけです。

リテラル-4
リテラル-4 はロケール・オブジェクト名でなければなりません。これは、最大長が 10 文字までの非数字リテラルでなければなりません。

リテラル-5
リテラル-5 は、ロケール・オブジェクトを検出するオペレーティング・システム・ライブラリーの名前を指定する際に使用されます。これは、最大長が 10 文字までの非数字リテラルでなければなりません。特殊値 *LIBL (ジョブのライブラリー・リストを使用する検索) を指定できます。 LIBRARY 句を省略すると、ジョブのライブラリー・リストを使用して、ロケール・オブジェクトを検索します。

簡略名-5
簡略名-5 は、ロケール名-1 で識別されるロケールへの参照、またはリテラル-4 およびリテラル-5 に指定された値への参照を提供します。これを使用できるのは、FORMAT 文節、PICTURE 文節、SET ステートメントの形式-8、または数種の組み込み関数の引数リストの中だけです。

End of IBM Extension

PROGRAM STATUS 文節

IBM Extension

PROGRAM STATUS 文節は、エラーがプログラム内で発生した後、定義済みのプログラム状況構造の値の転送元の値にデータ項目を指定します。

PROGRAM STATUS 文節 - 形式

データ名–1
WORKING-STORAGE SECTION にある英数字フィールドでなければなりません。データ名-1 をネストされたプログラムから参照する場合には、最も外側のプログラムでグローバルと定義しなければなりません。データ名-1 は、プログラム状況構造サブフィールドの長さの倍数でなければなりません。

整数–1
プログラム状況構造の開始位置を指定します。整数-1 が指定されていない場合は、開始位置は 0 と仮定されます。整数-1 は、プログラム状況構造サブフィールドの開始位置と一致しなければなりません。

PROGRAM STATUS 文節が、SPECIAL-NAMES 段落で指定されている場合、データ名-1 は、定義済みプログラム状況構造の値で更新されます。この構造は、発生したプログラムの例外/エラーに関する情報を与えるサブフィールドを含んでいます。 5-23 ページの表 5-5 は、データ構造およびそれが含む情報のサブフィールドのレイアウトを規定します。

表 5-5. プログラム状況データ構造の内容

開始位置	長さ	フォーマット	内容
0	10	文字	プログラム名。
10	10	文字	プログラム・ライブラリー名。
20	10	文字	モジュール名。
30	10	文字	ステートメント番号。選択不可の場合は、*N。
40	6	文字	最適化レベル。
46	7	文字	例外メッセージ ID。
53	10	文字	ジョブ名。
63	6	文字	ジョブ番号。
69	1	文字	ジョブ・タイプ。
70	10	文字	プログラムを実行するユーザー・プロファイル。
80	14	文字	エラーが発生した時刻用のタイム・スタンプ (YYYYMMDDHHMMSS 形式)

長さおよび開始位置のコーディングによって、データ名-1 へ移動させられるプログラム状況構造のサブフィールド (複数のサブフィールド) を選択します。 コンパイラーは、長さおよび開始位置を使用して、データ名-1 のマップ先であるプログラム状況のサブフィールド (複数のサブフィールド) を決定します。 長さおよび開始位置は、プログラム状況構造の定義済みサブフィールドと 1 つ以上一致しなければなりません。

End of IBM Extension

入出力セクション

入出力セクションには、各ファイルを定義し、その外部ストレージ・メディアを識別し、1 つ以上の入出力装置にファイルを割り当て、そして外部メディアと COBOL プログラムの間でデータを送受信するのに必要な情報を指定します。

ファイル・カテゴリー

IBM i システムには、データベース・ファイル、装置ファイル、DDM ファイル、および保管ファイルの 4 つのカテゴリーのファイルがあります。

本書では、これらのファイルを意味する用語として「ファイル」を使用します。

データベース・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けられます。データベース・ファイルは、物理ファイルと論理ファイルの 2 つのタイプに分かれます。

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、1 つまたは複数の物理ファイルのデータにアクセスするために使用するデータベース・ファイルです。このデータの形式および編成は、物理ファイル内のデータのものとは異なっています。各論理

ファイルは、物理ファイル内のデータに別のアクセス・パス (索引) を定義でき、物理ファイルに定義されているフィールドを除外して、その番号を付け直すことができます。

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 サーバー上に分散させることができます。データベース要求を分割して複数のシステムで処理することにより、大規模な照会の処理効率を向上させることが可能です。分散ファイルは、多くの点で DATABASE ファイルと同様の振る舞いをします。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。

分散ファイルへのアクセスについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

装置ファイル

装置ファイルは、装置またはリモート・システムからの読み取りや、またはそれらへの書き込みを行います。装置ファイルは、物理装置またはリモート・システムとプログラムの間で行われるデータの送受信を制御します。

DDM ファイル

分散データ管理機能 (DDM) を用いると、DDM をサポートするリモート・システムに存在するデータにアクセスできます。他のシステムに存在するファイル内のデータ・レコードを、検索、追加、更新、または削除できます。

リモート・ファイルへのアクセスについて詳しくは、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) 内のカテゴリー『データベースおよびファイル・システム』のセクション『*DB2® for i*』を参照してください。

保管ファイル

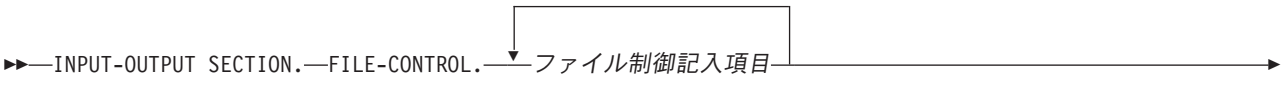
保管ファイルとは、バックアップおよびリカバリー用、または別のシステムへの転送用に適した形式でデータを作成するために使用されるファイルのことです。このファイルには、ライブラリー保管 (SAVLIB) またはオブジェクト保管 (SAVOBJ) CL コマンドから作成された出力が入ります。保管ファイルについては、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) 内のカテゴリー『データベースおよびファイル・システム』のセクション『*DB2 for i*』を参照してください。

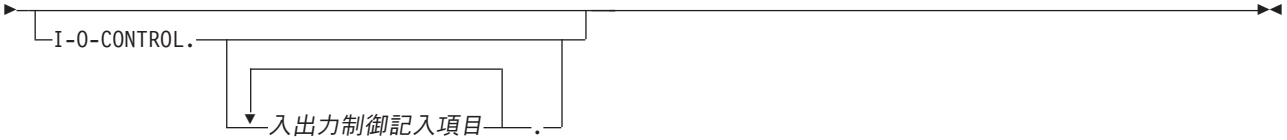
段落

環境部の入出力セクションには 2 つの段落があります。

- FILE-CONTROL 段落
- I-O-CONTROL 段落

入出力セクション - 形式





FILE-CONTROL 段落
ファイルに名前を付け、外部メディアと関連付けます。

キーワード FILE-CONTROL は、FILE-CONTROL 段落の先頭に、1 回だけ現れることができます。区域 A から始め、分離文字ピリオドが続かなければなりません。

ファイル制御記入項目
SELECT 文節の区域 B から始めなければなりません。これは分離文字ピリオドで終わらなければなりません。 『FILE-CONTROL 段落』を参照してください。

I-O-CONTROL 段落
外部メディアと COBOL プログラムの間でデータを伝送するために必要な情報を指定します。

入出力制御記入項目
連続する記入項目は、分離文字ピリオドで終わらなければなりません。 5-47 ページの『I-O-CONTROL 段落』を参照してください。

入出力セクションの正確な内容は、ファイル編成と使用されるアクセス方式によって異なります。 5-33 ページの『ORGANIZATION 文節』および 5-35 ページの『ACCESS MODE 文節』を参照してください。

FILE-CONTROL 段落

FILE-CONTROL 段落は、COBOL プログラム中の各ファイルを外部メディアと関連付けて、ファイル編成、アクセス・モード、およびその他の情報を指定します。

COBOL では 4 つの種類の異なるファイル入出力を行うことができます。

- 順次
- 相対
- 索引付き

IBM Extension

- トランザクション

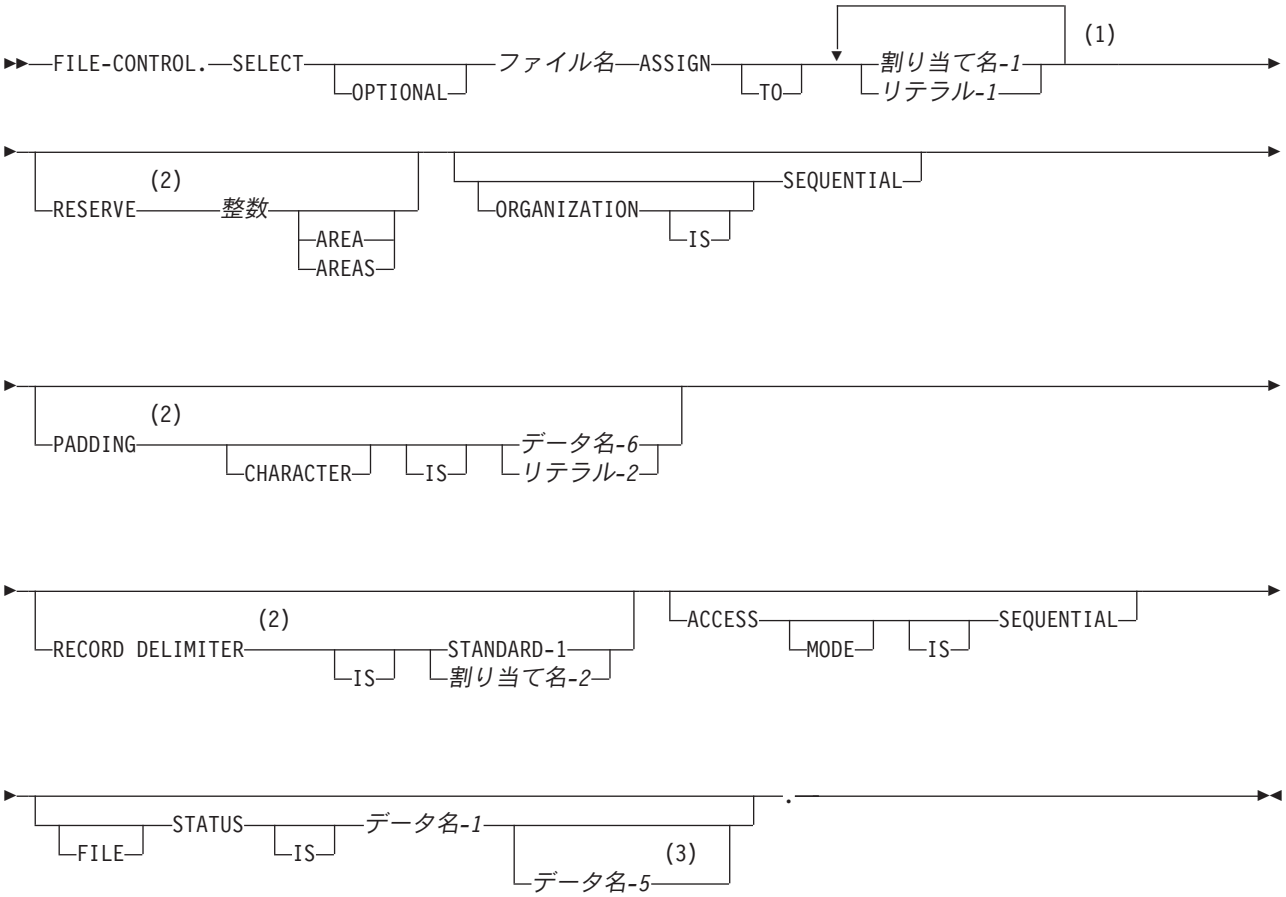
End of IBM Extension

FILE-CONTROL 段落は、"FILE-CONTROL" という語で始まり、その後に分離文字ピリオドが付きます。この段落には、データ部の FD 記入項目または SD 記入項目に記述される各ファイルの記入項目が、1 つだけ含まれていなければなりません。各項目内の先頭には、必ず SELECT 文節がなければなりません。それ以外の文節はどのような順序で記入してもかまいません。

各データ名は、データ部のデータ記述記入項目に現れなければなりません。各データ名は、修飾できますが、添え字または指標を付けることはできません。

FILE-CONTROL 段落 - 形式 1 - 順次ファイル

FILE-CONTROL 段落 - 形式 1 - 順次



注:

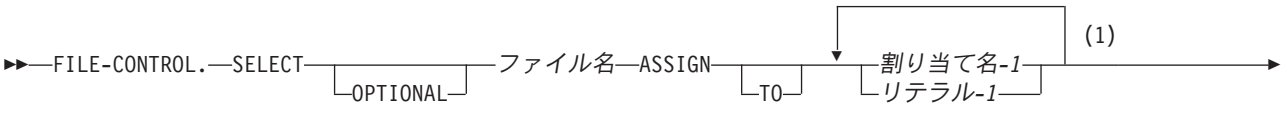
1 以降の反復に対する構文検査だけが行われます。

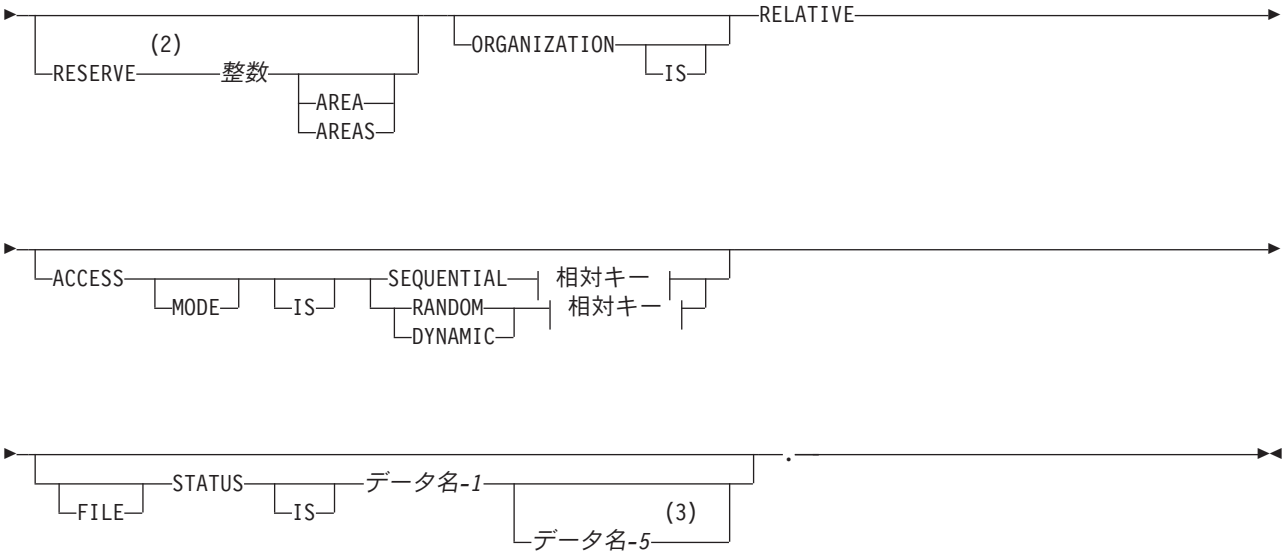
2 構文検査だけ行われます。

3 IBM 拡張

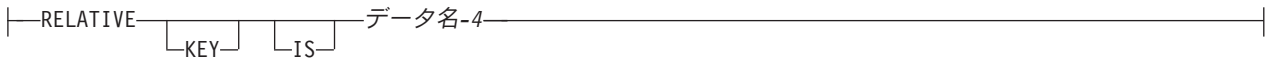
FILE-CONTROL 段落 - 形式 2 - 相対ファイル

FILE-CONTROL 段落 - 形式 2 - 相対





相対キー:



注:

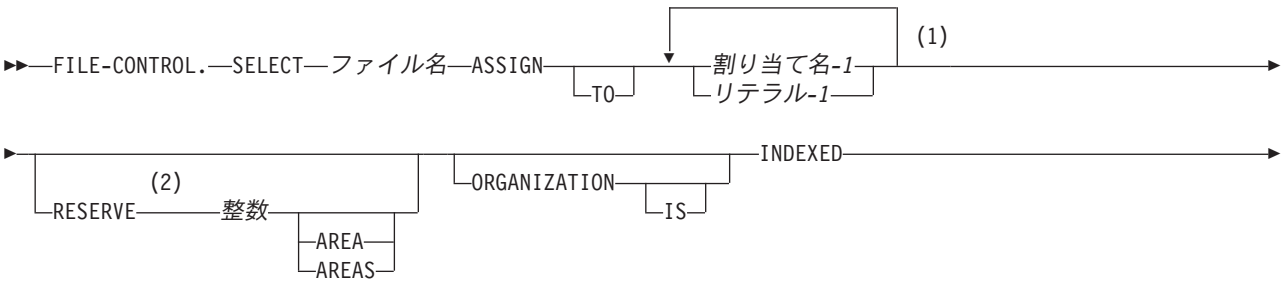
1 以降の反復に対する構文検査だけが行われます。

2 構文検査だけ行われます。

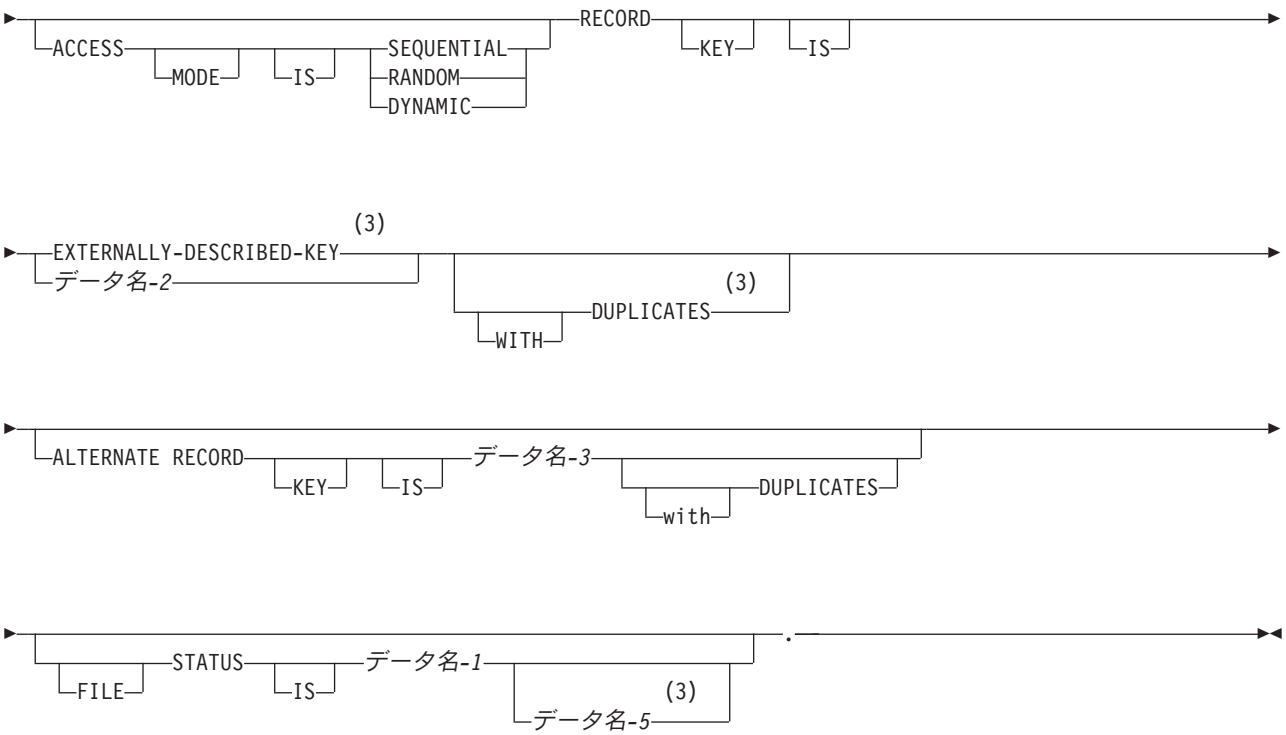
3 IBM 拡張

FILE-CONTROL 段落 - 形式 3 - 索引付きファイル

FILE-CONTROL 段落 - 形式 3 - 索引付き



FILE-CONTROL 段落



注:

1 以降の反復に対する構文検査だけが行われます。

2 構文検査だけ行われます。

3 IBM 拡張

FILE-CONTROL 段落 - 形式 4 - ソート・ファイルまたはマージ・ファイル

FILE-CONTROL 段落 - 形式 4 - ソートまたはマージ

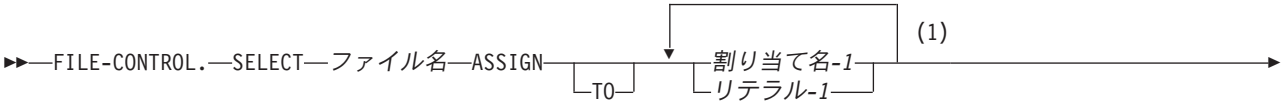
注:

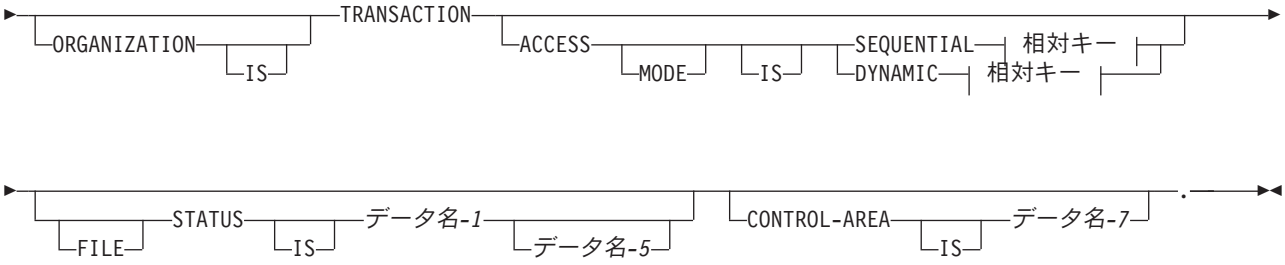
1 構文検査だけ行われます。

FILE-CONTROL 段落 - 形式 5 - トランザクション・ファイル

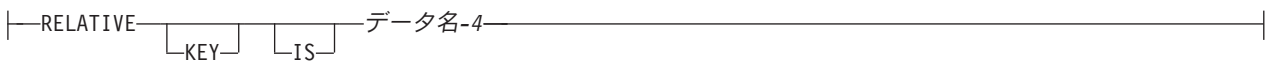
IBM Extension

FILE-CONTROL 段落 - 形式 5 - トランザクション





相対キー:



注:

1 以降の反復に対する構文検査だけが行われます。

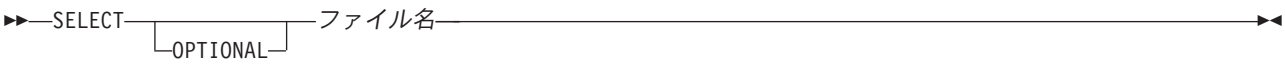
トランザクション・ファイルの処理について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」のトランザクション・ファイルに関する章を参照してください。

End of IBM Extension

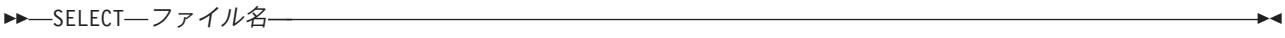
SELECT 文節

SELECT 文節はファイルを選択します。

SELECT 文節 - 形式 - 順次および相対ファイル



SELECT 文節 - 索引付き、ソート・マージ、トランザクション・ファイル



SELECT OPTIONAL (形式 - 順次ファイル、相対ファイル)
入力、入出力、または拡張モードでオープンする順次ファイルと相対ファイルのときだけ指定できます。オブジェクト・プログラムを実行するごとに、必ずしも示されない入力ファイルに対して SELECT OPTIONAL を指定しなければなりません。

ファイル名
データ部の FD または SD 項目によって識別されなければなりません。ファイル名は COBOL ユーザー定義名の規則に従い、少なくとも 1 つの英文字が入り、かつ、このプログラム内で固有でなければなりません。

ASSIGN 文節

ASSIGN 文節は、ファイルと外部メディアを関連付けます。

ASSIGN 文節

ASSIGN 文節 - 形式

注:

1 以降の反復に対する構文検査だけが行われます。

ソート・ファイルまたはマージ・ファイル (SD 記入項目と関連した) では、外部メディアは使用しません。関連付けられた ASSIGN 文節は、構文が検査されるだけです。

割り当て名-1、リテラル-1

割り当て名-1 あるいはリテラル-1 で、ファイルと外部メディアを関連付けます。

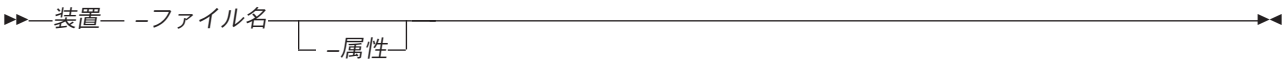
2 番目以降の割り当て名-1 またはリテラル-1 はすべて構文検査されますが、プログラムの実行には影響を与えません。

割り当て名-1 またはリテラル-1 を構成する 3 つの部分は次のとおりです。

- 装置
- ファイル名
- 属性

これは、次の一般構造をもちます。

フォーマット



装置

この部分はファイルが使用する装置のタイプを指定します。これによってコンパイラーは、ファイルが一貫した方法で記述され、使用されているかどうかを検査します。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

注:

1. コンパイラーは、外部ファイルと関係のある装置が割り当て名-1 またはリテラル-1 の装置部分に指定されたタイプであるかどうかの検査はしません。
2. コンパイラーは、入出力 verb が整合性のない方法で使用されていないかぎり、診断をしません。
3. プログラムが実行されると、オペレーティング・システムはエスケープ・メッセージを出すか、または機能が装置に適用できなければそれを無視することがあります。ファイルのオーバーライドについて詳しくは、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) 内のカテゴリー『データベースおよびファイル・システム』のセクション『ファイル・システムと管理』を参照してください。

IBM Extension

ファイルが使用する装置は OVRxxxF CL コマンドによって実行時に変更できます。矛盾のない結果を確実にするためには、ファイルに関連した装置タイプは割り当て名に与えられたものと一致させる必要がありま

す。

End of IBM Extension

装置は次のどちらでもかまいません。

装置　関連付けられるファイル

PRINTER
PRINTER はプログラム記述プリンター・ファイルにのみ指定されます。

FORMATFILE
FORMATFILE は、外部記述プリンター・ファイルにのみ指定されます。外部記述プリンター・ファイルの使用方法について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

TAPEFILE
テープ・ファイル

DISKETTE
ディスケット・ファイル

DISK 任意の物理データベース・ファイルまたは単一様式の論理データベース・ファイル。 DISK が装置である場合、データベースの拡張は使用できませんが、動的ファイル作成はサポートされています。 DISK フィールドについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。動的ファイル作成の詳細は 7-151 ページの『OPEN ステートメントに関する考慮事項』を参照してください。

DATABASE
任意のデータベース・ファイル (または DDM ファイル)。 DATABASE が装置である場合、外部記述データおよびデータベース拡張は使用できますが、動的ファイル作成はサポートされていません。

WORKSTATION
ディスプレイ・ファイルまたは ICF ファイル。

ファイル名

割り当て名のこの部分は、実際の外部ファイル (物理データベースまたは論理データベース、あるいは装置) の、10 文字以内のハイフンなしのシステム名でなければなりません。この外部ファイルが、プログラムの中で COPY ステートメント、DDS (データ記述仕様) または DD 様式で使用されている場合には、このプログラムのコンパイルの前にこの外部ファイルは作成されていなければなりません。

引用符付きのファイル名を、リテラル-1 に指定できます。例えば、 IBM i システム・ファイルに、"sysfile" という引用符付きの名前が付いている場合、リテラル-1 の記入項目は次のようにコーディングします。

```
"device-""sysfile""-SI"
```

データベース・ファイルの場合、メンバー名をプログラム中に指定することはできません。最初のメンバー以外のメンバーを指定すべき場合は、実行時に、データベース・ファイル指定変更 (OVRDBF) CL コマンドを使ってメンバー名を指定しなければなりません。

このファイル名は、プログラム参照表示 (DSPPGMREF) コマンドによって表示される IBM i オブジェクトの名前です。 SD ファイルに外部メディアが使用されないので、DSPPGMREF コマンドは、SD ファイル用に定義されたファイルをリストしません。

ASSIGN 文節

ファイル名は、実行時に OVRxxxF CL コマンドの TOFILE パラメーターを使って変更できます。一貫性のある結果が得られるようにするためには、TOFILE パラメーターに関連付けられる装置タイプを、割り当て名-1 またはリテラル-1 に指定されているものと同じにすることが必要です。

属性

割り当て名-1 またはリテラル-1 のこの部分は *SI* または *ALWNULL* にすることができます。

SI ファイル FORMATFILE または WORKSTATION に関する DDS の中で独立標識域が指定されたことを示します。

ALWNULL
ALWNULL が指定されている場合は、プログラムはデータベース・ファイル内のヌル可能フィールドを操作できます。このキーワードを使用できるのは装置タイプが DATABASE の場合だけです。

SI または ALWNULL 属性の使用および ASSIGN 文節について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

割り当て名-1 またはリテラル-1 の各フィールドの有効な記入項目は、装置によって異なります。フィールドの有効な組み合わせを表 5-6 に示します。

表 5-6. 割り当て名-1 およびリテラル-1 の有効な記入項目

装置	ファイル名	デフォルト・ファイル名	SI	ALWNULL
PRINTER	O	QPRINT	N	N
FORMATFILE	R		O	N
TAPEFILE	O	QTAPE	N	N
DISKETTE	O	QDKT	N	N
DISK	R		N	N
DATABASE	R		N	O
WORKSTATION	R		O	N
キー: R=必須 O=オプション N=許可されない				

RESERVE 文節

RESERVE 文節は入出力域を予約します。構文検査は行われますが、文書化のための記述として処理されます。

RESERVE 文節 - 形式

注:

1 構文検査だけ行われます。

ORGANIZATION 文節

ORGANIZATION 文節は、ファイルの論理構造を指定します。ファイル編成は、ファイル作成時に確立され、あとでそれを変更することはできません。

IBM Extension

データベース・ファイルの場合、ORGANIZATION 文節は、プログラムでのファイルの現行プログラム使用を示します。したがって、同じデータベース・ファイルで、ORGANIZATION 文節に SEQUENTIAL、RELATIVE、または INDEXED を使えます (キー順アクセス・パスが存在すると想定して)。このファイルを使用する他のプログラムに何が指定されていても、同じことがいえます。

キー順アクセス・パスは、物理ファイル作成 (CRTPF) または論理ファイル作成 (CRTPF) CL コマンドへの入力として使われた DDS にキーが指定されると、常に作成されます。

End of IBM Extension

ORGANIZATION IS SEQUENTIAL (形式 1)

ORGANIZATION 文節 - 順次ファイル

ファイル中のレコードの先行・後続関係は、ファイルが作成または拡張されるときにレコードがファイルに置かれる順番によります (到着順アクセス・パス)。

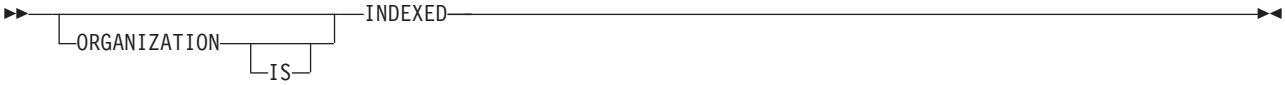
ORGANIZATION IS RELATIVE (形式 2)

ORGANIZATION 文節 - 相対ファイル

ファイル中の各レコードの位置は、到着順アクセス・パス内のその相対レコード番号によって決まります。

ORGANIZATION IS INDEXED (形式 3)

ORGANIZATION 文節 - 索引付きファイル



ファイル内の各論理レコードの位置は、ファイルと共に作成され、システムによって維持されるキー順アクセス・パスにより決まります。アクセス・パスは、ファイルのレコード内の埋め込みキーに基づいています。

ORGANIZATION IS TRANSACTION (形式 4)

IBM Extension

ORGANIZATION 文節 - トランザクション・ファイル

注:

1 IBM 拡張

COBOL プログラムと、ワークステーション・ユーザーまたは別のシステムとの対話を意味します。トランザクション・ファイルについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

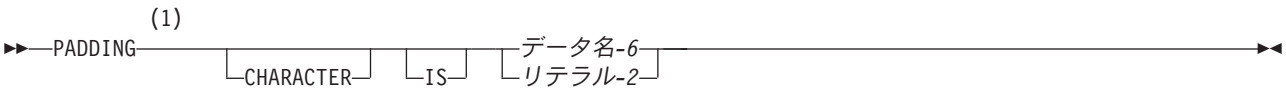
End of IBM Extension

PADDING CHARACTER 文節

PADDING CHARACTER 文節は、順次ファイルのブロック埋め込みに使用される文字を指定します。

PADDING CHARACTER 文節は構文検査はされますが、コンパイル時または実行時の検査は行われません。この文節はプログラムの実行には影響しません。

PADDING CHARACTER 文節 - 形式



注:

1 構文検査だけ行われます。

データ名-6
データ部に 1 文字データ項目として英数字で定義しなければならず、ファイル・セクションに定義してはなりません。データ名-6 は修飾できます。

リテラル-2
1 文字の非数字リテラルでなければなりません。

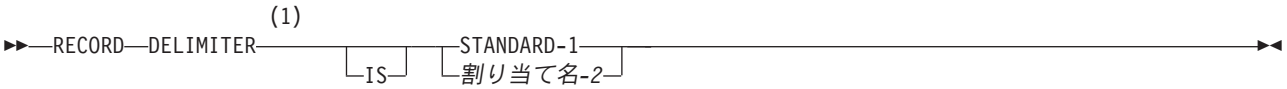
EXTERNAL ファイルの場合、データ名-6 が指定されると、それは EXTERNAL データ項目を参照しなければなりません。

RECORD DELIMITER 文節

RECORD DELIMITER 文節は、外部メディアでの可変長レコードの長さの判別法を示します。この文節は、可変長レコードにしか指定できません。

RECORD DELIMITER 文節の構文は検査されますが、コンパイル時にも実行時にも検査は行われません。この文節は説明として取り扱われます。

RECORD DELIMITER 文節 - 形式



注:

1 構文検査だけ行われます。

STANDARD-1
STANDARD-1 を指定する場合、外部メディアは磁気テープ・ファイルでなければなりません。

割り当て名-2
任意の COBOL 語を使えます。

ACCESS MODE 文節

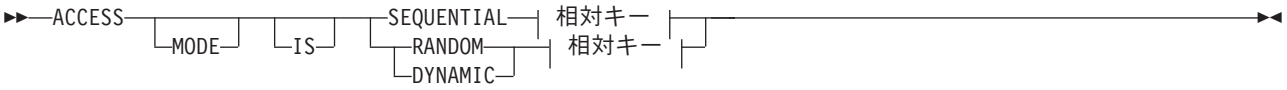
ACCESS MODE 文節は、ファイル内のレコードが処理のために使用可能にされる際の方法を定義します。
ACCESS MODE 文節を指定しないと、SEQUENTIAL アクセスと見なされます。

ACCESS MODE 文節 - 形式 1 - 順次ファイル

ACCESS MODE 文節 - 形式 1 - 順次ファイル

ACCESS MODE 文節 - 形式 2 - 相対ファイル

ACCESS MODE 文節 - 形式 2 - 相対ファイル



相対キー:

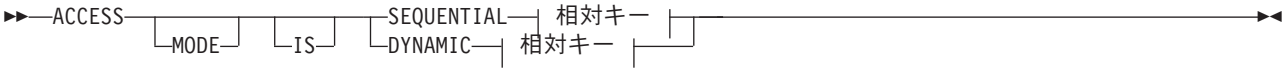
ACCESS MODE 文節 - 形式 3 - 索引付きファイル

ACCESS MODE 文節 - 形式 3 - 索引付きファイル

ACCESS MODE 文節 - 形式 4 - トランザクション・ファイル

IBM Extension

ACCESS MODE 文節 - 形式 4 - トランザクション・ファイル



相対キー:

End of IBM Extension

ACCESS MODE IS SEQUENTIAL

次の 3 種類のファイルのどれにでも指定できます。

順次

ファイル内のレコードは、そのファイルの作成時または拡張時 (到着順) に確立された順序でアクセスされます。

相対

ファイル内のレコードは、ファイル内の既存のレコードの相対レコード番号の昇順にアクセスされます。

索引付き

ファイル内のレコードは、そのファイルの照合順序にしたがって、昇順のレコード・キー値の順番にアクセスされます。

IBM Extension

外部記述されたファイルの使用時に、フィールドをキー・フィールドと指定するのに DDS キーワードの DESCEND を使うと、そのファイル内のレコードは、索引内の降順のレコード・キー値の順番にアクセスされます。 DESCEND キーワードまたは ASCEND キーワード (DESCEND が指定されない場合) のいずれかが、COBOL ソース・プログラム・リストのコメント・テーブルに RETRIEVAL というヘッディングで現れます。

End of IBM Extension

ACCESS MODE IS RANDOM

相対ファイルまたは索引付きファイルにしか指定できません。また、SORT または MERGE ステートメントの USING 句または GIVING 句に指定するファイル名には、ACCESS MODE IS RANDOM を指定してはなりません。

相対

相対キー・データ項目に入っている値が、アクセスされるレコードを指定します。

索引付き

現行参照キーのレコード・キー・データ項目に入っている値で、アクセスするレコードを指定します。

ACCESS MODE IS DYNAMIC

相対ファイルまたは索引付きファイルにしか指定できません。

相対

特定の入出力要求の形式に応じて、ファイル内のレコードに順次またはランダムにアクセスできます。

索引付き

特定の入出力要求の形式に応じて、ファイル内のレコードに順次またはランダムにアクセスできます。

データ編成とアクセス・モード

データ編成とは、ファイルの永続的な論理構造のことです。コンピューターにファイルからレコードを検査する方法を伝えるには、**アクセス・モード**を指定します。 COBOL では、4 種類のデータ編成、および 3 つのアクセス・モードをどれでも指定できます。順次編成されたデータには順次アクセスしかできませんが、索引編成または相対編成になっているデータには、3 つのアクセス方法のどれででもアクセスできます。

データ編成

COBOL プログラムのデータ編成には次のものがあります。

- 順次
- 相対
- 索引付き

IBM Extension

- トランザクション

End of IBM Extension

順次編成: ファイル内にレコードが置かれる物理的順序が、レコードの順序を決定します。ファイルのレコード間の関係は、ファイルが拡張されない限り、変わりません。キーはありません。データベース・ファイルおよび装置ファイルの両方が順次編成にできます。

ファイル内の各レコードには、最初のものを除いて、固有の先行レコードがあり、また最後のレコード以外の各レコードには、固有の後続レコードがあります。

相対編成: このファイルは、おのおのが 1 つのレコードを持つレコード域がいくつか順に並んでいるものと考えることができます。各レコード域は相対的なレコード番号で識別されますが、アクセス方式の場合、その相対レコード番号に基づいてレコードを保管したり取り出したりします。例えば、最初のレコード域は相対レコード番号 1 でアドレスされ、10 番目のレコード域は相対レコード番号 10 でアドレスされます。相対ファイルは DISK または DATABASE に割り当てなければなりません。

表 5-7には、相対出力ファイルに影響する条件が要約されています。

表 5-7. 相対出力ファイルの初期設定

ファイル・アクセスおよび CL 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		書き出されないレコードが初期設定される	全増分

表 5-7. 相対出力ファイルの初期設定 (続き)

ファイル・アクセスおよび CL 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 *NOMAX サイズ		CLOSE は正常に完了ファイル状況は 0Q	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されている		全増分
ランダムまたは動的 *NOMAX サイズ	OPEN は失敗ファイル状況は 9Q		ファイルは空

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・テキストに記述されているとおりに、CHGPF (物理ファイル変更) コマンドを使用してください。

相対レコード番号処理は、物理ファイルまたは、ただ 1 つの物理ファイルに基づいている論理ファイルに使用できます。

ファイル境界の拡張: ファイルの作成後に、ファイル・サイズを拡張できます。ファイルのファイル状況として 0Q を受け取ったら、その処理を行う前にさらにレコードをそのファイルに追加することが必要な場合があります。 INZPFM (物理ファイル・メンバー初期設定) コマンドを使えば、削除済みレコードをそのファイルに追加できます。

例えば、10 000 のレコードのファイルを作成してから、毎回 1 000 のレコードを 3 回増分するとします。

1. (最初の 10 000 件の) レコードを初期設定します。
2. さらに多くのデータを保管する必要が生じます。そこで、RECORDS(*DLT) オプションを指定して INZPFM コマンドを再度実行し、13 000 のレコードをすべて初期設定します。
3. そのうえ、さらにデータを保管する必要が生じますが、13 000 のレコードはすべて使いきっています。もう一度 INZPFM コマンドを実行すると、次のうちのいずれかを行うよう求める対話式エラー・メッセージ (重大度 99 の) を受け取ります。
   a. INZPFM 要求を取り消す
   b. 要求を継続する (つまり、次の 1 000 のレコードを初期設定する)
4. 上記のステップの 2 番目のオプションを選択すると、全部で 14 000 のレコードが初期設定されます。このようにして、それ以前に定義された最大値以上にファイル・サイズを拡大します。

索引編成: ファイル中の各レコードには、索引に関連付けられた埋め込みキー (キー・データ項目と呼ばれます) があります。各索引はそれに関連する埋め込みレコード・キー・データ項目の内容に従って、データ・レコードへの論理パスを提供します。データベース・ファイルと DISK ファイルのみを索引編成にできます。

レコードの挿入、更新、または削除が行われるとき、それらのレコードはその基本キーの値によってのみ識別されます。基本キー・データ項目名は、FILE-CONTROL 段落の RECORD KEY 文節に指定します。

IBM Extension

OUTPUT 用にオープンされている論理ファイルは、そのベースとなっている物理ファイル内のすべてのレ

コードを取り除くわけではありません。そうではなく、ファイルは書き込み操作のみ許されるようにオープンされ、レコードをファイルに追加することができます。

End of IBM Extension

TRANSACTION 編成:

IBM Extension

ワークステーションおよびデータ通信ファイルは、TRANSACTION 編成にできます。「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」のトランザクション・ファイルの章を参照してください。

End of IBM Extension

アクセス・モード

アクセス・モードとは、論理ファイルまたは物理ファイルのデータを処理する方法を定義する COBOL の用語です。アクセス・モードには、順次、ランダム、および動的の 3 つがあります。

順次アクセス・モード: 連続してファイルのレコードの読み取り、および書き込みを行えるようにします。参照する順は、そのファイル内のレコード位置によって決まります。

ランダム・アクセス・モード: プログラムの指定どおりにレコードの読み取りおよび書き込みを行えるようにします。ファイルの連続参照の制御は、ユーザーが提供する特殊な定義のキーで示します。

動的アクセス・モード: 特定の入出力要求によって、アクセス・モードを決定できます。したがって、レコードを順次またはランダム (あるいはその両方) で処理することができます。

データ編成とアクセス・モードの関係

順次ファイル: 順次編成のファイルへは、順次にアクセスします。レコードがアクセスされる順序は、レコードが書かれた順番です。

相対ファイル: 3 つのすべてのアクセス・モードが指定可能です。

順次アクセス・モードでは、レコードがアクセスされる順序は、ファイル内に現在あるすべてのレコードの相対番号の昇順です。

ランダム・アクセス・モードでは、レコードがアクセスする順序をユーザーが制御します。目的のレコードにアクセスするには、RELATIVE KEY データ項目にそのレコードの相対レコード番号を入れます。RELATIVE KEY を、このファイルのレコード記述記入項目に定義してはなりません。

動的アクセス・モードでは、適切な形式の入出力ステートメントを使って順次アクセスからランダム・アクセスへ変更できます。

索引付きファイル: 3 つのすべてのアクセス・モードが指定可能です。

順次アクセス・モードでは、レコードがアクセスする順序は、基本レコード・キーの値で決定されます。参照キーである別のレコード・キー内にも同じ重複値を持つレコードは、WRITE ステートメントか、またはその重複値を作成する REWRITE ステートメントの実行で解放されたときと同じ順序で使用できるようになります。

ACCESS MODE 文節

ランダム・アクセス・モードでは、レコードがアクセスする順序をユーザーが制御します。目的のレコードへアクセスするには、RECORD KEY データ項目にそのレコード・キーを入れます。一連のレコードが代替レコード・キー値を持つときは、最初に書き込まれたレコードだけが使用可能になります。

動的アクセス・モードでは、適切な形式の入出力ステートメントを使って順次アクセスからランダム・アクセスへ変更できます。

トランザクション・ファイル:

IBM Extension

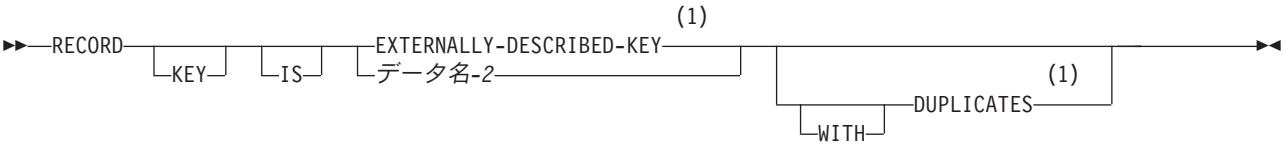
トランザクション・ファイルのアクセス・モードに関する考慮事項については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」のトランザクション・ファイルの章を参照してください。

End of IBM Extension

RECORD KEY 文節

RECORD KEY 文節は、索引付きファイルに指定しなければなりません。 RECORD KEY 文節は、索引付きファイルのレコード・キーであるレコード内のデータ項目を指定します。

RECORD KEY 文節 - 形式



注:

1 IBM 拡張

DUPLICATES 句

IBM Extension

DUPLICATES 句は、重複レコード・キーを割り当てられたファイルにしか指定できません。これによってファイルは同じ値を持つ複数のキーを持つことができます。ファイルに複数の形式がある場合、異なった形式の 2 つのキーは、キーの長さおよびキーの内容が同じであるときにのみ、同じ値を持ちます。

例えば、次の 2 つの形式を持つファイルがあるとします。

- A、B、C のキーを持つ形式 F1
- A、B、D のキーを持つ形式 F2

フィールド C および D が同じ長さで、同じデータ・タイプ、および同じ値を持つ場合、ファイルには 1 つの重複キーを持つ 2 つのレコードが入ります。*重複キー* という用語は、形式の完全なレコード・キーにのみ適用します。形式についてのレコード・キーは、データベースにあるレコードの DDS 形式に定義されるキー・フィールドから構成されます。この用語はファイルの共通キー (上の例のフィールド A および B のみ) には適用しません。

ユーザーは、RECORD KEY 文節に DUPLICATES を指定できます。ファイル状況 95 は、以下の条件で正常にオープンした後に返されます。

- DUPLICATES 句が COBOL プログラムで指定されており、ファイルは、DDS で UNIQUE と指定して作成された。
- DUPLICATES 句は COBOL プログラムで指定されず、ファイルは非固有キーで作成された。

これらのいずれかの条件が存在するときのファイルの処理は、予測できない結果をもたらすことがあります。

重複を認め、ランダムにもしくは動的に処理されるファイル内では、更新または削除される重複レコードは適切なものでなければなりません。そのためには、REWRITE または DELETE 操作に先立って処理された最後の入出力ステートメントは、NO LOCK 句のない READ ステートメントが正常に処理されたものでなければなりません。

DDS ファイル・レベル・キーワード LIFO (後入れ先出し法) を指定した場合には、物理ファイル内の重複レコードは後入れ先出し法の順番で検索されます。

End of IBM Extension

データ名-2

データ名-2 は、RECORD KEY データ項目です。これはファイルと関連するレコード記述項目内の固定長英数字項目として記述しなければなりません。またこれは、変数オカレンス・データ項目の入ったグループ項目を参照してはなりません。データ名-2 は修飾できますが、添え字を付けてはなりません。

レコード・キーの長さには制限があり、バイト数でのキー長は 2 000 を超えてはなりません。詳しくは、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) 内のカテゴリー『データベースおよびファイル・システム』のセクション『*DB2 for i*』を参照してください。

索引付きファイルに可変長レコードが入っていると、データ名-2 はそのレコードの最初の「x」位置に入らなければなりません。ここで、「x」はそのファイル用に指定された最小レコード・サイズです。

EXTERNAL ファイルの場合、EXTERNAL ファイルに関連しそして実行単位内にあるすべてのファイル記述項目には、関連したレコード内の同じ相対ロケーションを使って、同じデータ記述項目をデータ名-2 に指定しなければなりません。そうしないと、予期しない結果を生じます。

IBM Extension

ファイルが DATABASE 装置タイプに割り当てられている場合は、RECORD KEY データ項目 (データ名-2) は日時項目または数字項目であっても構いません。数字項目は、DISPLAY、COMP-1、COMP-2、COMP (COMP-3)、 COMP-4、COMP-5、PACKED-DECIMAL、または BINARY 形式の場合があります。数字項目は、外部浮動小数点データ項目にすることもできます。

End of IBM Extension

IBM Extension

ILE COBOL は、広範な日時データ項目形式をサポートしています。これらの形式のうち、多くのものは DDS のサポート外です。サポート外の場合、下層の DDS フィールドは文字フィールドまたは数字フィールドとして定義する必要があります。 COBOL は日時項目を定義しているが下層 DDS フィールドは日時用ではない場合は、データベースに対するレコード検索またはレコード書き込みは下層

DDS のデータ・タイプによって判別される順序に従って行われます。

End of IBM Extension

キーは、ファイルの作成時に使われた照合順序で順序づけされます。

データ名-2 のデータ記述およびレコード内の相対位置は、ファイルが DDS に定義されたときに使用したものと同じでなければなりません。

データ名-2 を定義するレコード記述は、入出力操作でレコード・キー・フィールドへその後アクセスするごとに使用されます。

EXTERNALLY-DESCRIBED-KEY

IBM Extension

予約語 EXTERNALLY-DESCRIBED-KEY は、このファイルのキーが DDS で外部記述されているものであることを指定できます。キーは、このファイルの FD で、COPY ステートメント、DDS、DD、DDSR、または DDR 形式によりコピーされるレコード形式により決定されます。

キーは、各形式ごとに、バッファー内の異なるオフセットから開始できます。この状況では、ランダム READ または START ステートメントを使って、あるレコード形式から別のレコード形式へ変更するときは注意しなければなりません。キーは、ファイルのランダム・アクセスで使用される形式のレコード形式で正しいオフセットに置かねばなりません。希望のレコードのキーが最後のレコード読み取りの一部であったデータに基づく場合、予期しない結果が起こることがあります。これは、キー・フィールドへのデータの移動が、オーバーラップ・フィールドに影響を与えるためです。

ある形式内のキーは、複数の、不連続 (隣接しない) フィールドから作ることができます。ファイルについて FD 内でコピーされるそれらのレコード形式のみが、FORMAT 句により参照されます。ファイル内で定義されているある形式が参照されても、その形式がプログラムにコピーされていない場合、キーはコピーされた最初のレコード形式に定義されるキー・フィールドを用いて作られます。これにより予期しない結果が起こる可能性があります。

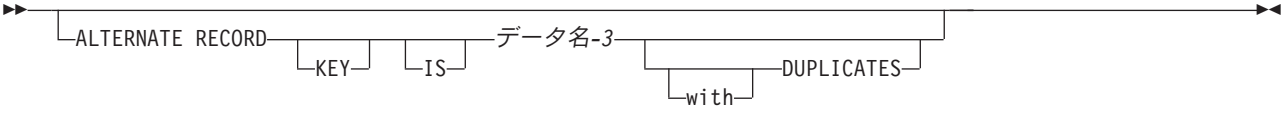
キーの一部が論理ファイルの中で (独立した宣言項目ではなく) 連結項目のエレメントとしてのみ宣言されている場合には、CONCAT 操作の結果を可変長項目にしてはなりません。

予約語 EXTERNALLY-DESCRIBED-KEY を、ALTERNATE RECORD KEY 文節で指定することはできません。

End of IBM Extension

ALTERNATE RECORD KEY

ALTERNATE RECORD KEY 文節は、索引付きファイルの代替レコード・キーであるレコード内のデータ項目を指定します。これらの代替レコード・キーを使用すれば、ILE COBOL プログラムは、ファイル・レコードのさまざまな論理配列を使用して、ファイルにアクセスできます。



データ名-3

データ名-3 は、ALTERNATE RECORD KEY データ項目です。これはファイルと関連するレコード記述記入項目内の固定長英数字項目として記述しなければなりません。またこれは、変数オカレンス・データ項目の入ったグループ項目を参照してはなりません。データ名-3 は修飾できますが、添え字を付けてはなりません。

代替レコード・キーの長さには制限があり、バイト数でのキー長は 2000 を超えてはなりません。索引付きファイルに可変長レコードが入っていると、データ名-3 はそのレコードの最初の「x」位置に入らなければなりません。ここで、「x」はそのファイル用に指定された最小レコード・サイズです。

EXTERNAL ファイルについては、実行単位内にあってしかも EXTERNAL ファイルと関連しているすべてのファイル記述記入項目には、次のものを指定しなければなりません。

- データ名-3 用の同じデータ記述記入項目
- 関連レコード内の同じ相対位置
- 代替レコード・キーの同じ番号
- 同じ DUPLICATES 句

IBM Extension

ファイルが DATABASE 装置タイプに割り当てられている場合は、ALTERNATE RECORD KEY データ項
目 (データ名-3) は日時項目または数字項目であっても構いません。数字項目は、DISPLAY、COMP-
1、COMP-2、COMP (COMP-3)、 COMP-4、COMP-5、PACKED-DECIMAL、または BINARY 形式の場合
があります。数字項目は、外部浮動小数点データ項目にすることもできます。 ILE COBOL は、広範な日
時データ項目形式をサポートしています。これらの形式のうち、多くのものは DDS のサポート外です。サ
ポート外の場合、下層の DDS フィールドは文字フィールドまたは数字フィールドとして定義する必要があ
ります。 ILE COBOL は日時項目を定義しているが下層 DDS フィールドは日時用ではない場合は、レコ
ード検索は下層 DDS のデータ・タイプによって判別される順序になります。

End of IBM Extension

IBM Extension

キーは、ファイルの作成時に使われた照合順序で順序づけされます。データ名-3 のデータ記述、レコード内のその相対位置、およびその長さは、ファイルが DDS に定義されたときに使用したものと同じでなければなりません。データ名-3 の 1 番左端の文字位置は、RECORD KEY またはその他の ALTERNATE RECORD KEY の 1 番左端の文字位置と同じであってはなりません。DUPLICATES 句が指定されていない場合には、ALTERNATE RECORD KEY データ項目に含まれている値は、ファイルにあるレコードの中で固有のものでなければなりません。 代替キー索引が一時的なものである場合、重複レコードの検索順序が、特定の順序である保証はありません。代替キー索引が永続的なものである場合、DDS ファイル・レベル・キーワード LIFO、FIFO、FCFO が、重複レコードの検索順序を指定するために使用されます。 代替キー索引について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

End of IBM Extension

使用に関する考慮事項

- 代替キーの順序付けは、基本キーと同じです。基本キーが、ファイルにある複数の DDS キー・フィールドにわたる場合、代替キー順序は、最初の基本キー・フィールドによって決定されます。永続代替索引が使用される場合、論理ファイルのキー順序は、物理ファイルとも同じでなければなりません。つま

り、DDS キーワード DESCEND が物理ファイル DDS で指定されている場合は、論理ファイル DDS でも指定されなければなりません。 その他の場合、ILE COBOL は永続代替索引を検出することができません。

- 代替キー付きのファイルは、外部で定義された基本レコード・キーを持つことはできません。
- 1 つのファイルに対して許可された代替キーの最大数は、253 です。
- ブロック化は、代替キー付きのファイルに対して、暗黙的に使用不可となっています。
- 指定変更コマンドで指定されたパラメーター値は、ILE COBOL が代替索引を作成した場合は、TOFILE、MBR、LVLCHK、WAITRCD、SEQONLY、および INHWRT 以外については、無視されます。
- 代替レコード・キーを使用するためには、データベース・ファイルは以下の要件を満たさなければなりません。 そうしないと、OPEN 操作が失敗し、ファイル状況が 39 に設定されます。
  1. 代替キーとして使用されているデータベース・ファイルのフィールド (複数のフィールドの場合もあり) は、入力、出力、または入出力フィールドでなければなりません。
  2. データベース・ファイルは、分散データ管理機能 (DDM) ファイルとすることはできません。
  3. データベース・ファイルは、オープン・データ・パスを共有してはなりません。
  4. プログラムの各キーに対して指定された DUPLICATES 文節は、データベース・ファイルの重複属性と一致しなければなりません。これには、基本キーも含まれます。永続代替索引を使用している場合、DDS キーワード UNIQUE が、固有のキーを指定するために使用されます。 このキーワードがないと、ファイルが複写キーを許可するということを暗黙指定します。 一時的な代替索引を使用し、DUPLICATES 文節が指定されていない場合は、必ず、データベース・ファイルにある既存のレコードが、プログラムでキーとして定義されているフィールドに、重複値を持たないようにしなければなりません。
- 以下の場合、OPEN オペレーションが失敗し、ファイル状況が 90 に設定されます。
  1. ILE COBOL は、それぞれの代替キーに対して 1 つの追加ファイルをオープンします。これらのファイルは、「QARK」で始まるオープン ID でオープンします。オープン ID は、プログラムが実行中である活動化グループ内で固有のものでなければなりません。 OPEN 操作は、ILE COBOL が非固有のオープン ID を検出した場合に失敗します。これが起こる可能性があるのは、ILE COBOL プログラムと共に OPNDBF および/または OPNQRYF コマンドを使用し、「QARK」で始まるオープン ID を指定した場合です。
  2. CRTARKIDX オプションが指定されておらず、ILE COBOL が永続索引を検出できない場合、OPEN 操作は失敗します。
  3. 代替キーを形成するために使用できる連続する DDS フィールドの最大数は 156 です。この限界を超えている場合、OPEN 操作は失敗します。

RELATIVE KEY 文節

RELATIVE KEY 文節は、相対ファイル内の特定の論理レコードの相対レコード番号を指定するデータ名を識別します。

RELATIVE KEY 文節 - 形式

データ名-4

PICTURE 記号の P の入っていない記述をもった符号なしの整数データ項目として定義しなければな

りません。データ名-4 は、この相対ファイルと関連したレコード記述項目に定義してはなりません。すなわち、RELATIVE KEY は、レコードの一部では**ありません**。データ名-4 は修飾できます。

順次アクセスで読み取りを行うと、使用可能になったレコードの相対レコード番号を使って RELATIVE KEY データ項目は更新されます。

データ名-4 は、START ステートメントが使用されるときにのみ ACCESS IS SEQUENTIAL に必要です。これは常に ACCESS IS RANDOM および ACCESS IS DYNAMIC には必要です。 START ステートメントが出されると、システムは、RELATIVE KEY データ項目の内容を使用して、順次処理をどのレコードから開始するかを判別します。

ある値がデータ名-4 に置かれ、かつ START ステートメントが出されない場合、その値は無視され、処理はファイルの最初のレコードから開始されます。

IBM Extension

ファイルのオープン時に、OVRDBF CL コマンドの POSITION パラメーターを使ってファイル位置標識をセットすることができます。これによって最初のレコード以外のレコードから処理を開始します。詳しくは、**IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『*プログラミング*』のセクション『*CL および API*』を参照してください。

End of IBM Extension

相対ファイルが START ステートメントで参照される場合、そのファイルに RELATIVE KEY 文節を指定しなければなりません。

ACCESS MODE IS RANDOM 文節は、SORT または MERGE ステートメントの USING あるいは GIVING 句に指定されたファイル名に指定してはなりません。

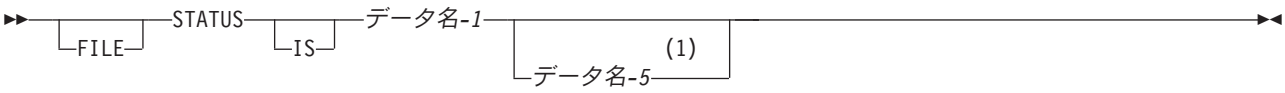
EXTERNAL ファイルの場合、データ名-4 は外部データ項目を参照しなければならず、それに関連した各ファイル制御記入項目の RELATIVE KEY 句もその同じ外部データ項目を参照しなければなりません。 (相対キーはサブファイルと一緒に使用します。)

トランザクション・ファイルの考慮事項については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」のトランザクション・ファイルの章を参照してください。

FILE STATUS 文節

FILE STATUS 文節は、ファイルへの各入出力要求の実行をモニターします。

FILE STATUS 文節 - 形式



注:

1 IBM 拡張

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。その値は、そのステートメントの実行状況を示します。 (7-37 ページの『共通の処理機能』の『状況キー』のセクションを参照。)

FILE STATUS 文節

コンパイラーが、出力レコードをブロック化するため、または入力レコードをブロック解除するためにコードを生成すると、オペレーティング・システムの例外が原因で生じるファイル状況値は、ブロックが処理されるときにのみ設定されます。可能な値については9-23 ページの『付録 F. ファイル構造サポートの要約およびファイル状況キーの値』を参照してください。出力レコードのブロックおよび入力レコードのブロック解除について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

データ名-1

状況キー・データ項目は、データ部で 2 桁の英数字項目として定義されていなければなりません。データ名-1 をファイル・セクションの中で定義してはなりません。データ名-1 は修飾できます。

IBM Extension

データ名-5

オプショナルの状況キー・データ項目を、ファイル処理用に指定できます。

トランザクション・ファイルの場合には、データ項目は 4 桁の英数字項目でなければなりません。

非トランザクション・ファイルの場合には、データ項目は 6 バイトのグループ項目でなければなりません。この項目は、すべての非トランザクション・ファイル (動的に作成されるものを除く) のための文書化として扱われます。 OPEN OUTPUT を指定すると動的に作成されるファイルの場合、拡張ファイル状況は 0900 に設定されます。データ名-5 は修飾できます。

End of IBM Extension

CONTROL-AREA 文節

IBM Extension

この文節は、TRANSACTION ファイルの入出力操作を制御するのに使用する、装置依存情報とシステム依存情報を指定します。

CONTROL-AREA 文節 - 形式

注:

1 IBM 拡張

データ名-7

LINKAGE、LOCAL-STORAGE または WORKING-STORAGE SECTION に定義される、次のような形式のデータ項目 (2、12、または 22 文字の長さ)。

```
01    data-name-7
      05   function-key  PIC X(2)
      05   device-name   PIC X(10)
      05   record-format PIC X(10)
```

ここで、

ファンクション・キー
オペレーターがトランザクションを開始するのに押したファンクション・キーを識別するための、ワークステーション・インターフェースからフィールドに挿入される 2 桁の数字です。

数字	意味
00	実行キー
01 ～ 24	1 ～ 24 のファンクション・キー
90	次ページ / ページ送りキー
91	前ページ / ページ戻しキー
92	印刷キー
93	ヘルプ・キー
94	クリア・キー
95	ホーム・キー
99	未定義

装置名
プログラム装置の名前です。

レコード形式
それ以前の最後の入出力ステートメントの実行で参照された DDS レコード形式の名前です。

End of IBM Extension

I-O-CONTROL 段落

入出力セクションの I-O-CONTROL 段落は、さまざまなファイルが共用する記憶域を指定します。この段落は COBOL プログラムではオプショナルです。

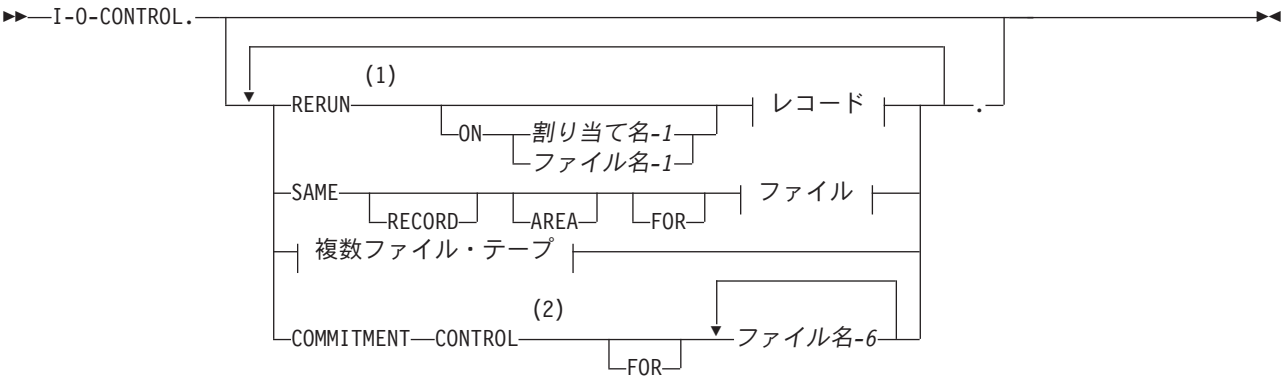
キーワード I-O-CONTROL は、I-O-CONTROL 段落の先頭に、1 回だけ表示できます。 I-O-CONTROL という語は、区域 A から始めて、その後に分離文字ピリオドを付けなければなりません。

段落の中の各文節は、分離文字コンマまたは分離文字セミコロンによって、次の文節と区切ることができます。 I-O-CONTROL 段落文節が書かれる順序は、重要ではありません。 I-O-CONTROL 段落の終わりには、分離文字ピリオドを付けます。

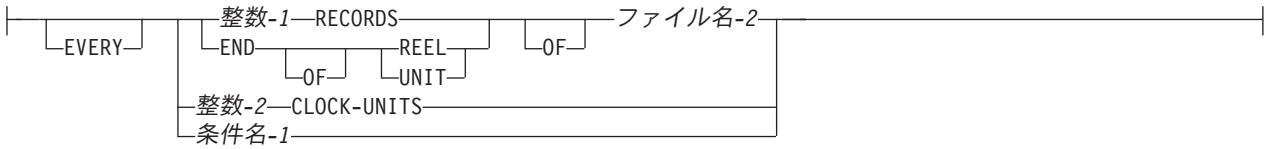
I-O-CONTROL 段落 - 形式 1 - 順次ファイル

I-O-CONTROL 段落 - 形式 1 - 順次

I-O-CONTROL 段落

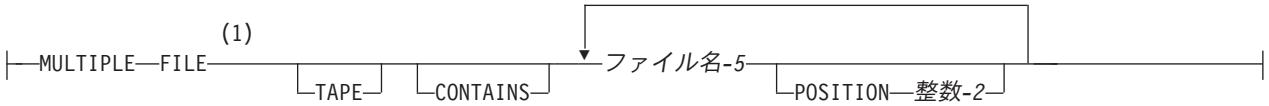


レコード:



ファイル:

複数ファイル・テープ:



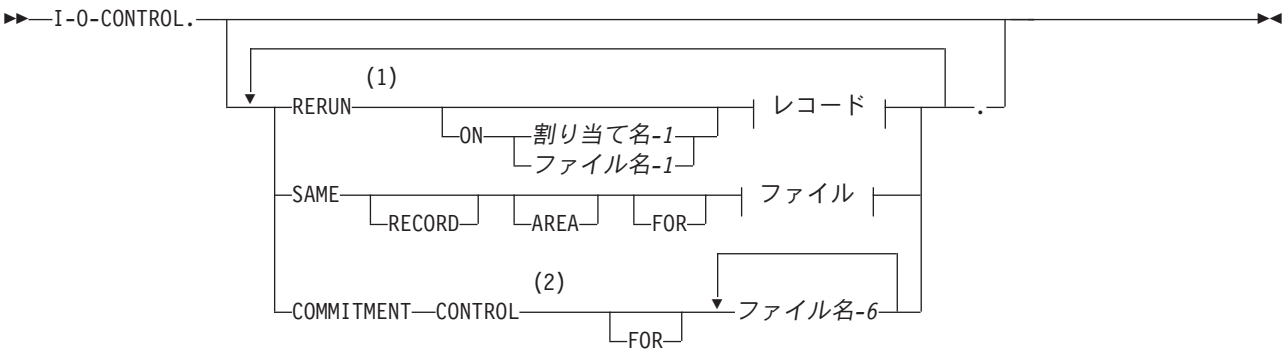
注:

1 構文検査だけ行われます。

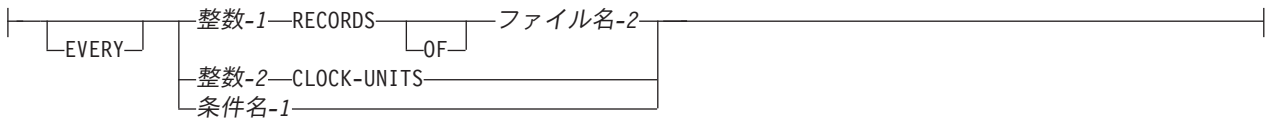
2 IBM 拡張

I-O-CONTROL 段落 - 形式 2 - 相対および索引付きファイル

I-O-CONTROL 段落 - 形式 2 - 相対 / 索引付き



レコード:



ファイル:

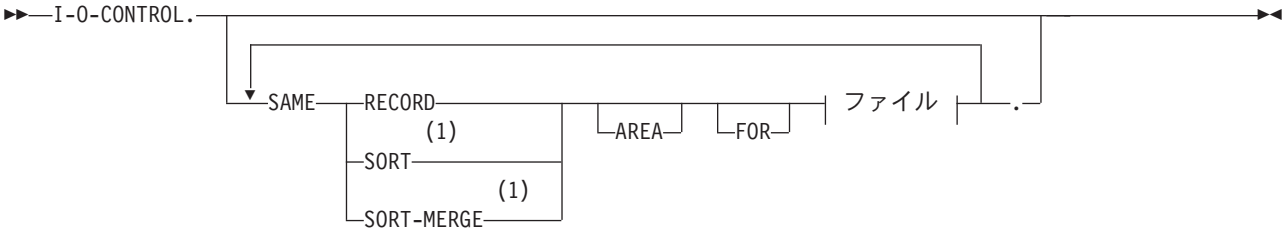
注:

1 構文検査だけ行われます。

2 IBM 拡張

I-O-CONTROL 段落 - 形式 3 - ソート・ファイルまたはマージ・ファイル

I-O-CONTROL 段落 - 形式 3 - ソート・マージ



ファイル:

注:

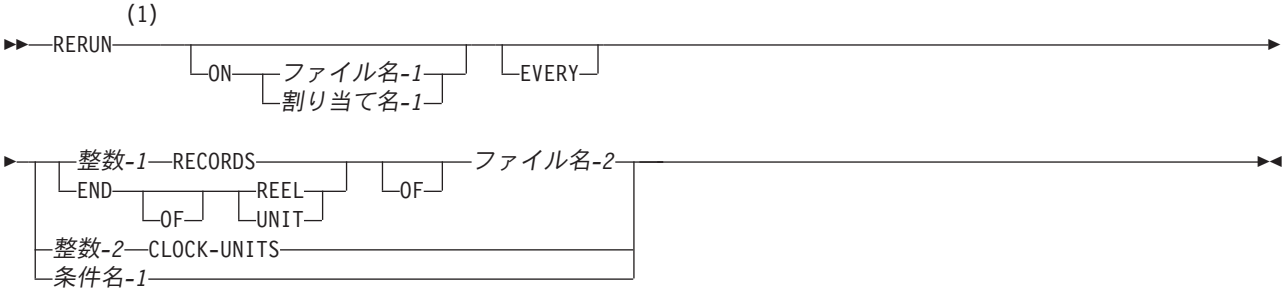
1 構文検査だけ行われます。

RERUN 文節

RERUN 文節は、チェックポイント・レコードを取ることを指定します。

RERUN 文節は構文検査されますが、文書化のための記述として処理されます。

RERUN 文節 - 形式



注:

1 構文検査だけ行われます。

ファイル名-1
順次編成されたファイルの名前です。 RERUN 文節に指定するファイルは、GLOBAL と定義されたファイルであっても、I-O-CONTROL 段落と同じプログラムで定義されたファイルでなければなりません。

割り当て名-1
この名前は、どのユーザー定義語にもすることができます。 RERUN 文節に指定するファイルは、GLOBAL と定義されたファイルであっても、I-O-CONTROL 段落と同じプログラムで定義されたファイルでなければなりません。

EVERY 整数-1 RECORDS
処理されるファイル名-2 内の整数-1 レコードごとに 1 つのチェックポイント・レコードが書き込まれます。

整数-1 の RECORDS 句を複数指定するときは、そのうちのどの 2 つにも同じファイル名-2 を指定してはなりません。

整数-1 は符号なしの整数でなければなりません。この文節は、RERUN 情報が書き込まれる前に処理するレコード数を指定します。

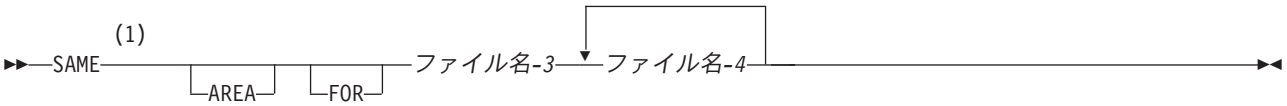
EVERY END OF REEL/UNIT
複数の END OF REEL 句または END OF UNIT 句に同じファイル名-2 を指定できません。 UNIT の定義は、それぞれの割り当て名-1 で決まります。

EVERY 整数-2 CLOCK-UNITS
CLOCK-UNITS 句を含んだ RERUN 文節は 1 つしか指定できません。

SAME AREA 文節

SAME AREA 文節は、ソート・ファイルまたはマージ・ファイルではない、複数のファイルが処理中に同じ主記憶域を使用することを指定します。 SAME AREA 文節は、構文検査されますが、文書化のための記述として処理されます。

SAME AREA 文節 - 形式



注:

1 構文検査だけ行われます。

SAME AREA 文節に指定される複数のファイルは同じ編成またはアクセスである必要はありません。

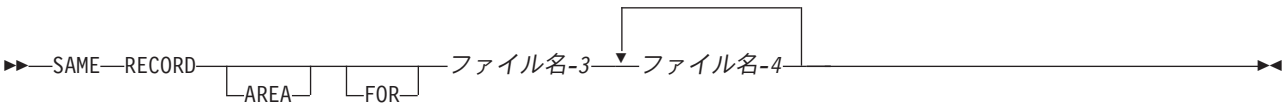
ファイル名-3、ファイル名-4 ...
I-O-CONTROL 段落と同じプログラムの FILE-CONTROL 段落に指定しなければなりません。これらは、外部ファイル・コネクターを参照できません。

SAME RECORD AREA 文節

SAME RECORD AREA 文節は、 複数のファイルが現行論理レコードの処理に同じ主記憶域を使用することを指定します。すべてのファイルを同時にオープンできます。

注: SAME RECORD AREA 文節は、主記憶域を効率的に使うことを意図しています。しかし、IBM i 仮想記憶アーキテクチャーではこの文節は不要になっており、この文節はパフォーマンスのためというより互換性を保つためにサポートされています。実際には、SAME RECORD AREA 文節を使用すると、パフォーマンスが低下してプログラム・サイズが大きくなります。

SAME RECORD AREA 文節 - 形式



共用記憶域にある論理レコードは、次の両方であると見なされます。

- SAME RECORD AREA 文節を使用してオープンされている各出力ファイルの論理レコード
- SAME RECORD AREA 文節を使用して最も新しく読み取られた出力ファイルの論理レコード

SAME RECORD AREA 文節によって、明示的なデータ操作をせずにあるファイルから別のファイルへデータを移動できます。これは指定されるファイルの入出力レコード域が同一であり、すべてがユーザーに使用できるからです。

1 つのプログラム内に複数の SAME RECORD AREA 文節を書くことができます。ただし、以下の事柄を考慮する必要があります。

- 特定のファイル名を複数の SAME RECORD AREA 文節の中で指定してはなりません。
- SAME RECORD AREA 文節の中に SAME AREA 文節のファイル名が複数ある場合は、その SAME AREA 文節のファイル名のすべてが、SAME RECORD AREA 文節の中にもなければなりません。ただし、その SAME AREA 文節に記入されていないファイル名を、その SAME RECORD AREA 文節の中に追加できます。
- 複数のファイルに SAME RECORD AREA が指定されていると、それらのファイルのレコード記述記入項目またはファイル記述記入項目には GLOBAL 文節を入れてはなりません。

SAME RECORD AREA 文節

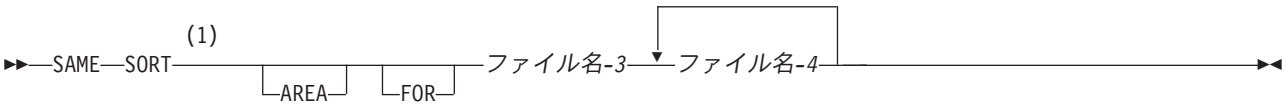
- SAME RECORD AREA 文節は、EXTERNAL ファイルと一緒には使えません。

SAME SORT AREA 文節

SAME SORT AREA 文節を使えば、特定の SORT ステートメントに対する記憶域の割り当てを最適化できます。

SAME SORT AREA 文節は構文検査されますが、文書化のための記述として処理されます。

SAME SORT AREA 文節 - 形式



注:

1 構文検査だけ行われます。

SAME SORT AREA 文節を指定する場合、指定するファイル名の少なくとも 1 つはソート・ファイルでなければなりません。ソート・ファイルでないファイルも指定できます。次の規則が適用されます。

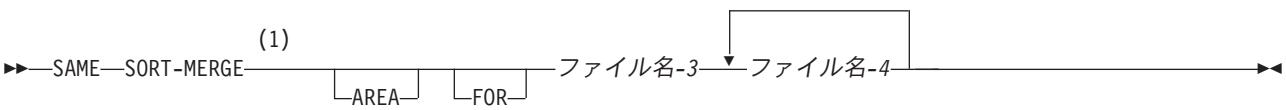
- SAME SORT AREA 文節を複数指定できますが、特定の 1 つのソート・ファイルをその複数の文節の中で指定してはなりません。
- ソート・ファイルでないファイルが、SAME AREA 文節と 1 つまたは複数の SAME SORT AREA 文節の両方で指定されている場合は、SAME AREA 文節内のすべてのファイルがその SAME SORT AREA 文節の中になければなりません。
- SAME SORT AREA 文節で指定される複数のファイルは、同じ編成またはアクセスである必要はありません。
- SAME SORT AREA 文節で指定された、ソート・ファイル以外のファイルは、ユーザーがそれらを SAME RECORD AREA 文節で指定しない限り、相互にストレージを共用することはありません。

SAME SORT-MERGE AREA 文節

SAME SORT-MERGE AREA 文節を使えば、特定の SORT または MERGE ステートメントに対する記憶域の割り当てを最適化できます。

SAME SORT-MERGE AREA 文節は構文検査されますが、文書化のための記述として処理されます。

SAME SORT-MERGE AREA 文節 - 形式



注:

1 構文検査だけ行われます。

SAME SORT-MERGE AREA 文節を指定する場合、指定するファイル名の少なくとも 1 つはソート・ファイルまたはマージ・ファイルでなければなりません。ソート・ファイルまたはマージ・ファイルでないファイルも指定できます。次の規則が適用されます。

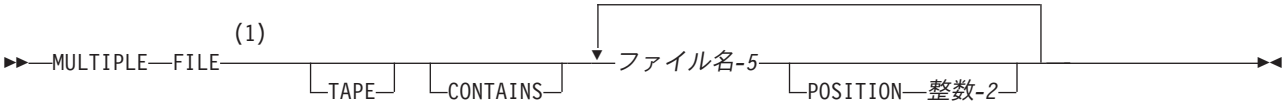
- SAME SORT-MERGE AREA 文節を複数指定できます。ただし、特定の 1 つのソート・ファイルまたはマージ・ファイルをそのような複数の文節の中で指名してはなりません。
- ソート・ファイルまたはマージ・ファイルでないファイルが、SAME AREA 文節と 1 つまたは複数の SAME SORT-MERGE AREA 文節の両方で指定されている場合は、SAME AREA 文節内のすべてのファイルがその SAME SORT-MERGE AREA 文節の中になければなりません。
- SAME SORT-MERGE AREA 文節で指定される複数のファイルは、同じ編成またはアクセスである必要はありません。
- SAME SORT-MERGE AREA 文節で指定された、ソート・ファイルまたはマージ・ファイル以外のファイルは、ユーザーがそれらを SAME RECORD AREA 文節で指定しない限り、相互にストレージを共用しません。

MULTIPLE FILE TAPE 文節

この文節は、複数のファイルが同じリール・テープを共用することを指定します。コマンド言語を使うことによって、システムからこの機能が提供されます。

MULTIPLE FILE TAPE 文節は、構文検査されますが、文書化のための記述として扱われます。 **IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『*プログラミング*』のセクション『*CL および API*』の CRTTAPF、CHGTAPF、および OVRTAPF コマンドを参照してください。

MULTIPLE FILE TAPE 文節 - 形式



注:

1 構文検査だけ行われます。

整数-2
符号のない整数でなければなりません。これは、テープでのファイルの相対位置を指定します。

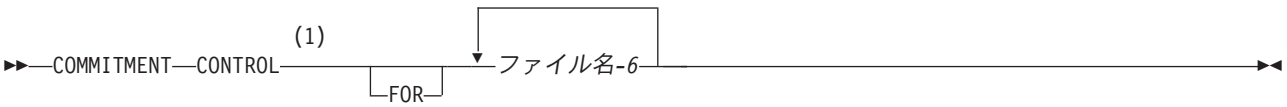
ファイル名-5
テープを共用するファイルを指定します。

COMMITMENT CONTROL 文節

IBM Extension

COMMITMENT CONTROL 文節は、オープン時にコミットメント制御のもとに置かれることになるファイルを指定します。

COMMITMENT CONTROL 文節 - 形式



COMMITMENT CONTROL 文節

注:

1 IBM 拡張

ファイル名-6 は、COMMITMENT CONTROL 文節のある I-O-CONTROL 段落と同じプログラムの FILE CONTROL 段落に指定しなければなりません。

そのように指定すると、これらのファイルは、COMMIT や ROLLBACK ステートメントによって影響を受けます。 COMMIT ステートメントにより、データベース・レコードに対する変更の同期化が可能になるとともに、COMMIT が完了するまでは別のジョブによってこれらのレコードを修正することができなくなります。 ROLLBACK ステートメントを使えば、データベース・ファイルに行われた変更を永続化してはならないときにそれを取り消せます。

COMMITMENT CONTROL 文節は、装置タイプが DATABASE と割り当てられたファイルにだけ指定できます。コミットメント制御下に置かれたファイルは、順次、相対、または索引付きの編成でもよく、それぞれの編成に許される任意のアクセス・モードが使用可能です。

システムは、コミットメント制御のもとに置かれたファイルの中に収められたレコードをアクセスする際、これらのレコードが他から使用されるのを禁止します。レコードは、COMMIT または ROLLBACK ステートメントによって解放されるまではロックされたままです。コミットメント下におけるファイルのレコード・ロックについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

注: 常に首尾一貫した方法でファイルを使用するようにして、レコードのロックで問題が起きたり、まだデータベースに対して永続的にコミットされていないレコードを読み取ったりしないようにします。通常、ファイルは常に、コミットメント制御のもとでアクセスするか、またはコミットメント制御のもとではアクセスしないかを明確にしておくことが必要です。

End of IBM Extension

データ部

データ部の概要

COBOL ソース・プログラムのデータ部は、オブジェクト・プログラムで処理されるすべてのデータおよび、物理レコードと論理レコードの関係を、構造化された方法で記述します。データ部は、COBOL ソース・プログラムではオプショナルです。

このセクションでは、データ部の構造を概略し、データ・タイプについて説明します。

データ部の構造

データ部は、DATA DIVISION という語で始め、その後にピリオドとスペースを続けなければなりません。

データ部は、次の 4 つのセクションに分けられます。

ファイル・セクション
外部的に保管されるデータ (ソート・マージ・ファイルも含む) を記述します。

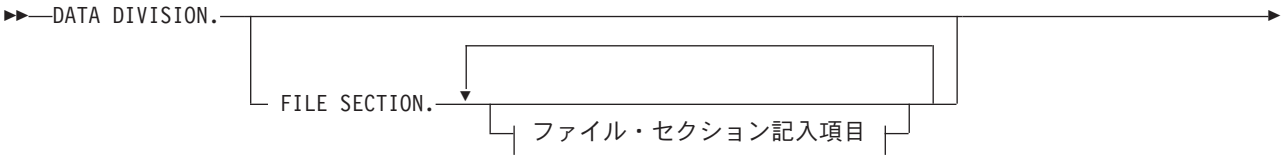
作業用ストレージ・セクション
内部データを記述します。

ローカル・ストレージ・セクション
呼び出しごとに割り振られる内部データを記述します。

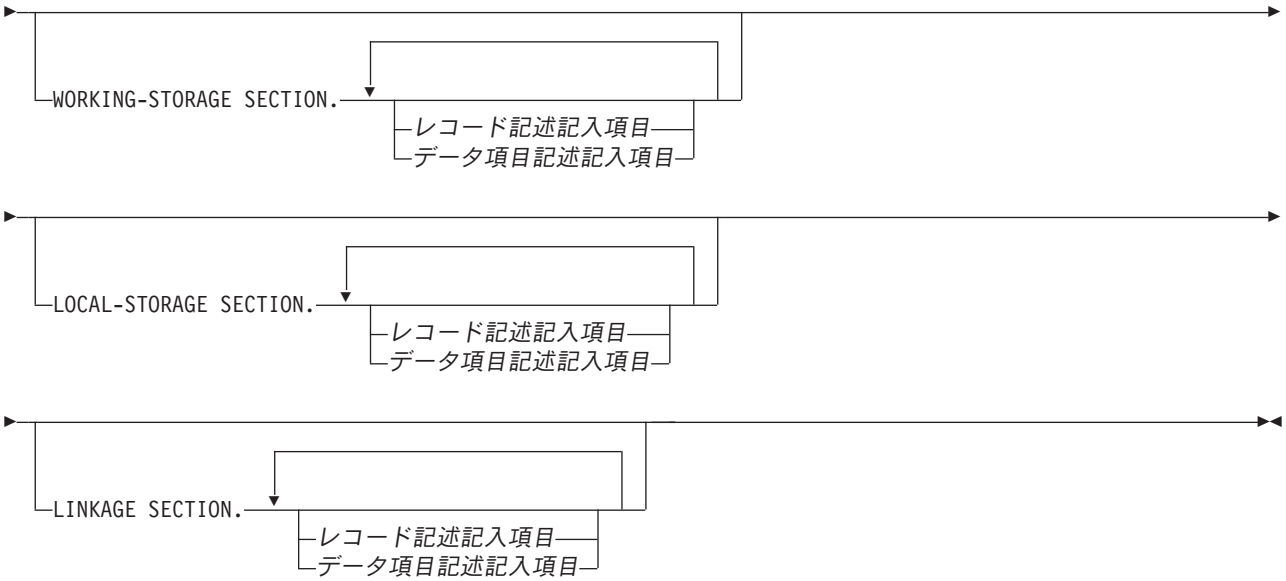
リンケージ・セクション
別のプログラムから使用可能にされるデータを記述します。このセクションは、呼び出し先プログラム内にあって、呼び出し側プログラムから提供されて呼び出し先プログラムによって参照されるデータ項目を記述します。呼び出し先プログラムは、ネストされたプログラムのこともあります。ネストされたプログラムについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

それぞれのセクションは、COBOL ソース・プログラム内に特定の論理機能を持ち、その論理機能が必要ではないときには、それぞれのセクションをソース・プログラムから省くことができます。セクションを含める場合は、以下に示す順序で書かなければなりません。

データ部 - 形式



データ部の構造



ファイル・セクション記入項目:

ファイル・セクション

ファイル・セクションは、次のファイルを記述します。

- 外部的に保管されるすべてのファイル
- 各ソート・マージ・ファイル

ファイル記述記入項目

ファイル・セクションの中の最高レベルの編成を表します。これは、ファイルの物理的な構造と ID についての情報を提供し、そのファイルに関連付けられるレコード名を示します。

ファイル記述記入項目で必要とされる形式と文節については 6-13 ページの『データ部 — ファイルとソート記述記入項目』を参照してください。

レコード記述記入項目

特定のファイル内に含まれている 1 つまたは複数の特定レコードを記述するか、あるいは (TYPEDEF 文節を使用することによって) タイプ名を記述するデータ記述記入項目のセットです。レコード記述記入項目で必要とされる形式と文節については 6-29 ページの『データ部 — データ記述記入項目』を参照してください。

レコード記述記入項目は複数指定できます。ただし、タイプ名を記述しない各記入項目は、それぞれが同じレコード・ストレージ域の代替記述になります。

ファイル・セクションで記述されるデータ域は、そのデータ域を含むファイルがオープンされていない限り、処理のために使用することはできません。 FILE SECTION で定義したタイプ名は、他のデータ項目を定義するために WORKING-STORAGE、 LOCAL-STORAGE、または LINKAGE SECTIONS 内で使用できます。

グループ項目 (テーブルを含む) の長さは、16 711 568 バイトに制限されます。

ファイル・セクション中のデータ項目の初期値は未定義です。

IBM Extension

ファイルのレコード記述記入項目は、形式 2 の COPY ステートメント (DD、 DDR、DDS、または DDSR オプション) を使って指定できます。これによってレコード形式のフィールド記述を、DDS での定義とまったく同じにすることができます。また、レコード形式記述がただ 1 か所に保持されるためプログラムがより書きやすくなります。 COPY ステートメントのこの形式の詳細については 8-1 ページの『コンパイラー指示ステートメント』を参照してください。

End of IBM Extension

作業用ストレージ・セクション

作業用ストレージ・セクションは、外部データ・ファイルの一部ではなく、そのプログラムが内部で作成および処理するデータ・レコードを記述します。タイプ名は WORKING-STORAGE SECTION 内で定義できます。

レコード記述記入項目

詳細は、 6-2 ページの『ファイル・セクション』を参照してください。作業用ストレージ・セクション内にあり、相互に一定の階層関係にあるデータ記入項目は、レベル番号によって構造化されるレコードにグループ化される必要があります。

データ項目記述記入項目

作業用ストレージ・セクション内にあり、相互に階層関係をもたない独立した項目は、それ以上、細分化する必要がない限り、レコードにグループ化する必要はありません。すなわち、それぞれがレベル番号 77 または 01 のいずれかで始まる別々のデータ項目記述記入項目の中で定義されます。データ項目記述記入項目で必要とされる形式と文節については 6-29 ページの『データ部 — データ記述記入項目』を参照してください。

作業用ストレージ・セクションにあるすべてのデータ項目 (指標データ項目を除く) の初期値は、VALUE 文節をその項目と関連付けることにより指定されます。任意の指標データ項目、すなわち VALUE 文節と関連のない任意のデータ項目の初期値は未定義です。

注: グループ項目 (テーブルを含む) には、最大 16 711 568 バイトが許されます。

ローカル・ストレージ・セクション

IBM Extension

ローカル・ストレージ・セクションは、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します。呼び出しごとに、ローカル・ストレージ・セクションで定義されたデータ項目は再度割り振られ、VALUE 文節で割り当てられた値に初期化されます。ローカル・ストレージ・セクションで定義されたデータ項目は、 EXTERNAL 文節を指定することはできません。ローカル・ストレージ・セクションは、分離文字ピリオドが後に付いたヘッダー LOCAL-STORAGE SECTION で始める必要があります。

レコード記述記入項目

詳細は、 6-2 ページの『ファイル・セクション』を参照してください。

データ項目記述記入項目

詳細は、『作業用ストレージ・セクション』を参照してください。

ローカル・ストレージ・セクションは、再帰的プログラムおよび非再帰的プログラムで指定できます。

End of IBM Extension

リンケージ・セクション

リンケージ・セクションは、CALL ステートメントを介して別のプログラムから使用可能にされるデータを記述します。また、このセクションは、ADDRESS OF 特殊レジスターを使ってアクセスされるデータの形式を記述するのにも使えます。例えば、リンケージ・セクション項目用の ADDRESS OF 特殊レジスターを、ILE のバインド可能 API を使って動的に割り振られるデータに設定できます。

レコード記述記入項目
詳細は、6-2 ページの『ファイル・セクション』を参照してください。

データ項目記述記入項目
詳細は、6-3 ページの『作業用ストレージ・セクション』を参照してください。

リンケージ・セクションのレコード記述記入項目とデータ項目記述記入項目は、データ項目の名前と記述を提供しますが、ストレージは提供しません。データ域は別の場所にあるので、ストレージはプログラム内に予約されません。タイプ名は LINKAGE SECTION で定義できます。

リンケージ・セクションでは任意のデータ記述文節を使用して項目を記述できますが、次のような例外があります。

- VALUE 文節は、レベル 88 の項目以外の項目に対して指定してはなりません。

  IBM Extension

  リンケージ・セクションでレベル 88 以外の項目に VALUE 文節を指定すると、それはコメントとして取り扱われます。

  End of IBM Extension

- EXTERNAL 文節は、リンケージ・セクションに指定することはできません。
- GLOBAL 文節は、リンケージ・セクションに指定することはできません。

IBM Extension

- GLOBAL 文節は、レベル番号 01 のときリンケージ・セクションのデータ名または条件名に指定できます。リンケージ・セクションのデータ項目に GLOBAL を指定すると、組み込まれたソース・プログラムは、データ項目名でその項目を直接参照できます。 LINKAGE セクションのコーディングについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

End of IBM Extension

注: グループ項目 (テーブルを含む) には、最大 16 711 568 バイトが許されます。

ADDRESS OF

IBM Extension

ADDRESS OF は、データ項目の計算アドレスを参照します。そのデータ項目は、修正済みまた添え字付き

参照にすることができます。タイプ名またはタイプ名の従属項目ではなく、そのレベル番号が 66 または 88 ではない任意のデータ部項目に対して ADDRESS OF を使用できます。そのようなアドレスを参照することはできますが、変更することはできません。

ADDRESS OF 項目は、USAGE IS POINTER として暗黙定義されています。

ADDRESS OF 特殊レジスター: ADDRESS OF 特殊レジスターは、データ構造の開始アドレスであり、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。

互いに再定義するレコードを除いて、リンケージ・セクション内のレコード (レベル番号 01 または 77) ごとにアドレスが存在します。再定義する場合、この特殊レジスターも同様に再定義されます。

特殊レジスターは、暗黙に USAGE IS POINTER として定義されますが、これは変更できます。

参照で ADDRESS OF ID を変更した場合、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。計算アドレスになります。

ユーザーは、ADDRESS OF または ADDRESS OF 特殊レジスターを、LENGTH 関数への引数として指定できます。 ADDRESS OF または ADDRESS OF 特殊レジスターを LENGTH 関数への引数として使用した場合、その結果は常に 16 となり、ADDRESS OF に指定された ID に依存しません。

関数 ID は、ADDRESS OF または ADDRESS OF 特殊レジスターのどちらにおいても使用できません。

日時データ項目は ADDRESS OF または ADDRESS OF 特殊レジスターを使用する式の中で使用できます。

End of IBM Extension

データのタイプ

ファイル・データとプログラム・データの 2 つのタイプのデータを処理できます。

ファイル・データ

ファイルとは、入出力装置上に存在するデータ・レコードの集合のことです。 (6-18 ページの『ファイル・セクション』を参照。) ファイルは、物理レコード・グループと見なすことも、論理レコード・グループと見なすこともできます。

物理レコードとは、 記憶装置へ (または記憶装置から) 移される際に 1 つのエンティティーとして取り扱われるデータの単位です。物理レコードの大きさは、それが収容される特定の入出力装置によって決まります。この大きさは、ファイルに含まれる論理情報の大きさや内容とは必ずしも直接的な関係はありません。

論理レコードとは、 データの細分化部分が互いに論理的な関係をもっているデータの単位です。論理レコードそのものが物理レコードである (つまり、データの 1 つの物理単位に完全に含まれる) ことも、複数の論理レコードが 1 つの物理レコードに含まれることも、または 1 つの論理レコードが複数の物理レコードに分かれて入っていることもあります。

ファイル記述記入項目は、 データの物理的な側面 (例えば、物理レコードと論理レコードの大きさの関係、論理レコードの大きさと名前など) を指定します。

レコード記述記入項目は、ファイル内の論理レコードを記述します。 例えば、論理レコードの各フィールド内にあるデータのカテゴリーや形式、データが割り当てられる種々の値を記述します。

物理レコードと論理レコードの間の関係が確立された後では、ユーザーは論理レコードだけを使用できます。したがって、本書では特に「物理レコード」という用語を使用しない限り、「レコード」という用語は論理レコードを指します。

プログラム・データ

プログラム・データとは、ファイルから読み取られるものではなく、プログラムそのものによって作成されるデータのことです。

論理レコードという概念は、ファイル・データだけでなくプログラム・データにも適用されます。したがって、プログラム・データを論理レコードにグループ化し、一連のレコード記述記入項目で定義できます。そのようにグループ化する必要のない項目は、独立データ項目記述記入項目に定義できます。

データの関係

プログラムで使用するすべてのデータの関係は、データ部で、レベル標識とレベル番号のシステムを使用して定義します。

レベル標識は、その記述記入項目とともに、プログラム内の各ファイルを識別します。レベル標識は、それに関連した任意のデータ階層の最高レベルを表します。すなわち、FD はファイル記述レベルの標識で、SD はソート・マージ・ファイル記述レベルの標識です。

レベル番号とその記述記入項目は、特定データの特性を示します。レベル番号は、データ階層を記述するために使用できます。例えば、このデータには特殊な目的があることを示すことができ、さらにレベル番号は、レベル標識に関連 (従属) させたり、単独で使えば内部データを記述したり複数のプログラムに共通したデータを記述したりすることもできます。 (レベル番号の規則については 6-36 ページの『レベル番号』を参照してください。)

データのレベル

レコードを定義した後に、分割されたレコードをさらに分割して、より詳しいデータ参照を行うようにできます。

例えば、百貨店のカスタマー・ファイルの場合に、1 人のカスタマーに関するすべてのデータを 1 つのレコードに完全に収容することができるとします。そのレコードを分割したものが、顧客名、顧客住所、顧客番号、販売部門番号、販売単価、販売金額、それまでの残高、およびその他の付随情報であったとします。

レコードの基本的な細分化項目 (それ以上分割されないフィールド) を**基本項目**といいます。 したがって、レコードは一連の基本項目で構成されている場合もあれば、レコードそれ自体が 1 つの基本項目である場合もあります。

一連の基本項目を参照する必要が生じた場合には、基本項目をまとめて**グループ項目**にすることができます。グループ自体を組み合わせて、1 つまたは複数のサブグループを含む、より大きいグループにすることもできます。したがって、データ項目から成る階層において、1 つの基本項目を複数のグループ項目に属させることができます。

レベル番号体系は、基本項目とグループ項目をレコードに編成する方法を指定するものです。 特殊レベル番号を使うこともでき、この番号で特殊目的に使うデータ項目が識別されます。

レコード記述記入項目のデータ・レベル: レコード内のグループ項目および基本項目には、それぞれ別々の記入項目が必要であり、また各項目ごとにレベル番号を割り当てなければなりません。

レベル番号は、01 ～ 49 の 1 桁または 2 桁の整数であるか、または 66、77、88 の 3 つの特殊レベル番号のうちの 1 つです。 次のレベル番号が、レコードを構造化するために使用されます。

01 このレベル番号は、レコードそのものを指定する最も包括的なレベル番号です。レベル 01 の記入項目は、グループ項目または基本項目のどちらでもかまいません。これは区域 A から始まらなければなりません。(TYPEDEF 文節を使用して定義した) タイプ名はレベル 01 の項目でなければなりません。

02 ～ 49
これらのレベル番号は、レコード内のグループ項目や基本項目を指定します。これらは、区域 A または区域 B から始められます。データ項目の包括性が低いほど、この一連の番号の中からより高い (必ずしも連続していない) レベル番号が割り当てられます。

1 つのグループ項目には、そのグループのレベル番号より小さいかあるいは等しいレベル番号が現れるまで、そのグループ項目の後ろにあるすべてのグループ項目と基本項目が含まれます。

ある 1 つのグループ項目のすぐ下にあるすべてのグループ項目と基本項目には、上のグループ項目のレベル番号より大きい同一のレベル番号を割り当てなければなりません。

タイプ名がグループ項目である場合、そのタイプ名を TYPE 文節で使用して新しいデータ項目を定義すると、その新規データ項目には使用されたタイプ名の従属項目と同じ名前、記述、および階層を持つ項目が従属します。新規データ項目が結果的に持つことになるレベルの数には制限がありません。その理由を以下に示します。

- TYPE 文節のサブジェクトが持つことができる最高レベル番号は 49 であり、タイプ名が記述できるグループ項目の 1 つ当たりの最大レベル数は 49 です。
- タイプ宣言は他のタイプ宣言を参照することができます。

コーディング例:

IBM Extension

ILE COBOL コンパイラーでは、同じレベルの他のものと同じでない非標準レベル番号を使えます。例えば、次の 2 つのデータ記述記入項目は同じです。

```
01   EMPLOYEE-RECORD.
     05  EMPLOYEE-NAME.
         10  FIRST     PICTURE  X(10).
         10  LAST      PICTURE  X(10).
     05  EMPLOYEE-ADDRESS.
         10  STREET    PICTURE  X(10).
         10  CITY      PICTURE  X(10).
01   EMPLOYEE-RECORD.
         05  EMPLOYEE-NAME.
             10  FIRST     PICTURE  X(10).
             10  LAST      PICTURE  X(10).
         04  EMPLOYEE-ADDRESS.
             08  STREET    PICTURE  X(10).
             08  CITY      PICTURE  X(10).
```

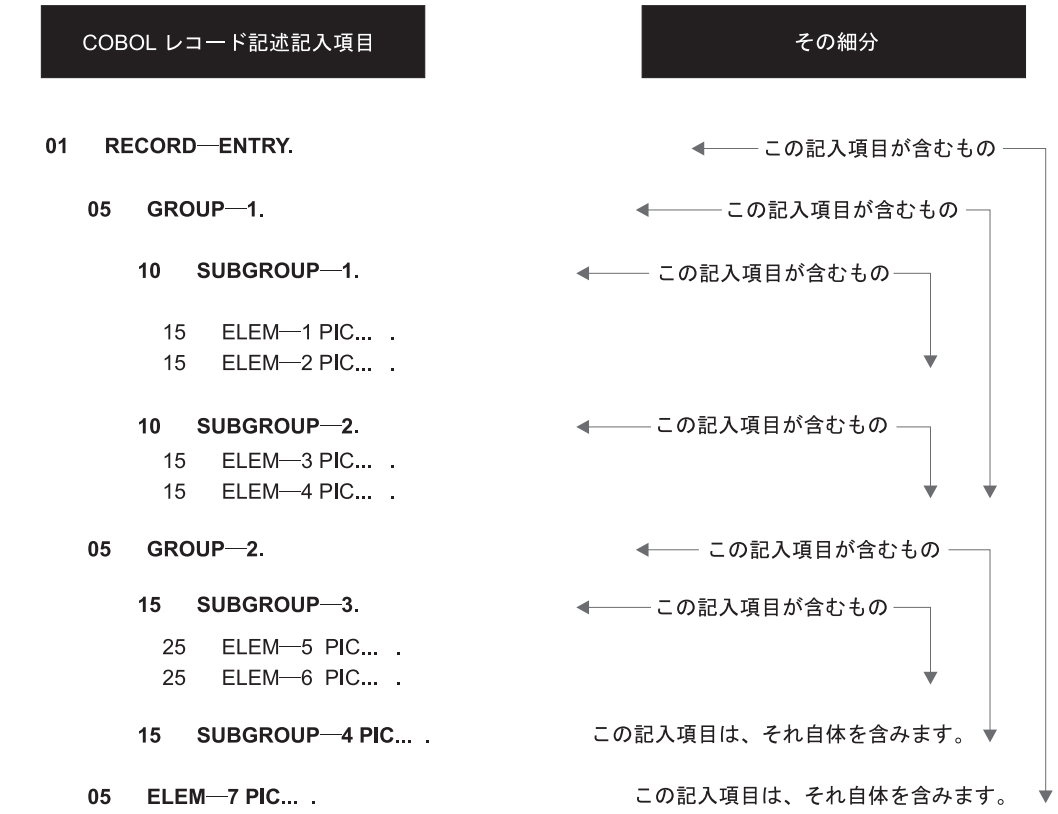
04 は 05 より小さいため、EMPLOYEE-NAME の下になりませんが、他方 01 より大きいため、EMPLOYEE-RECORD の下になります。 04 の代わりに 07 を使うと、EMPLOYEE-ADDRESS は EMPLOYEE-NAME の下になります (これは、この例では望ましくありません)。

このようなコーディングは推奨されません。また、この拡張機能は互換性のためだけに提供されています。

End of IBM Extension

概念例:

図 6-1 はこの概念を表したものです。レベル 01 の記入項目に直接従属するグループは、すべて同一のレベル番号をもっていることに注意してください。またあるサブグループの基本項目と別のサブグループの基本項目は必ずしも同じレベル番号をもっていないこと、基本項目は階層内のどのレベルでも指定することができることに注意してください。



以下は、レコード記述記入項目のストレージ配置を図解したものです。

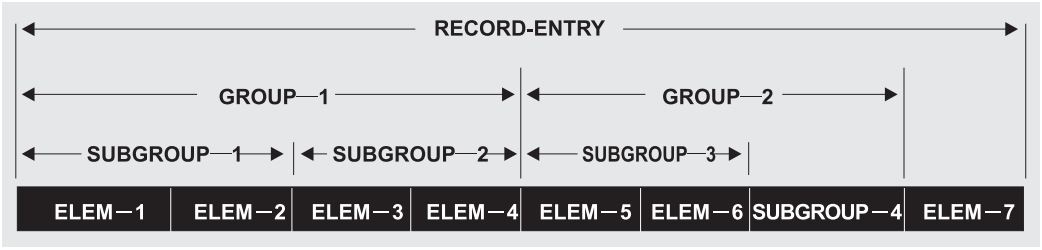


図 6-1. レコード記述のレベル

特殊なレベル番号: 特別なレベル番号は、レコードの構造を作らない項目を識別します。特別レベル番号は次のとおりです。

66 RENAMES 文節しか入れてはならない項目を識別します。このような項目は、すでに定義されているデータ項目をグループ化し直します。 (詳しくは 6-78 ページの『RENAMES 文節』を参照してください。)

77
作業用ストレージ・セクション、ローカル・ストレージ・セクションまたはリンケージ・セクションの

独立した項目 (他の項目の構成部分ではなく、それ自体分割されていないもの) であるデータ項目記述記入項目を識別します。レベル 77 の項目は区域 A から始めなければなりません。

88 条件変数の特定の特定の値と結び付いている任意の条件名記入項目を識別します。 条件名記入項目には、VALUE 文節しか入れてはなりません。 (詳しくは 6-102 ページの『VALUE 文節』を参照してください。)

注: プログラム内で参照される、作業用ストレージ・セクション、ローカル・ストレージ・セクションおよびリンケージ・セクション内のレベル 77 記入項目およびレベル 01 記入項目には、固有のデータ名を付けなければなりません。なぜなら、どちらも修飾することができないからです。プログラム内で参照される従属データ名は、固有名として定義するか、あるいは修飾によって固有名にしておく必要があります。参照されないデータ名は固有に定義する必要はありません。

字下げ: 連続するデータ記述記入項目は、先行する記入項目と同じ列で始めることも、字下げすることもできます。字下げは、文書化するときに役立ちますが、コンパイラーの処理には影響を及ぼしません。

データのクラスおよびカテゴリー

ポインター、プロシージャー・ポインター、および指標データ項目を除き、COBOL プログラムで使用するほとんどのデータは、クラスとカテゴリーに分けることができます。 プログラム内の各基本項目は、すべて 1 つのカテゴリーと 1 つのクラスに属しています。すべてのグループ項目は、その下の基本項目が別のクラスとカテゴリーに属していても、英数字クラスに属します。 6-10 ページの表 6-1 に、データのクラスとカテゴリーとの関係を示します。

ある 1 つの項目のデータ・カテゴリーは、PICTURE 文字ストリング、BLANK WHEN ZERO 属性、および USAGE 属性によって決定されます。詳細は 6-64 ページの『データ・カテゴリーと PICTURE の規則』を参照してください。

IBM Extension

項目のデータ・カテゴリーは、その項目用の FORMAT 文節から判別することもできます。 FORMAT 文節は、カテゴリーの日付、時刻、およびタイム・スタンプの項目を定義します。

表示画面の形式に関連する標識の値および外部記述のプリンター・ファイルを修正したり渡したりするための IBM 拡張として、ブール・データ・タイプが用意されています。ブール値の 0 は標識が**オフ**の状況で、ブール値の 1 は標識が**オン**の状況です。

ブール・リテラルには単一の 0 または 1 が入り、引用符で囲まれ、しかもその直前に識別用の B が付きます。ブール・リテラルは、B"0" または B"1"のいずれかとして定義されます。

ブール文字は 1 バイトを占めます。

表意定数 ZERO がブール・データ項目またはブール・リテラルと関連付けられると、それはブール・リテラル B"0" を表します。

予約語 ALL はブール・リテラルに有効です。

End of IBM Extension

組み込み関数であるデータ項目はすべて基本項目であり、英数字、数字、DBCS、国別、ブール、日付、時刻、またはタイム・スタンプのカテゴリーに属し、それに対応するクラスに属します。各組み込み関数のカテゴリーは、その関数の定義によって判別されます。

データのクラスおよびカテゴリー:

表 6-1. データ項目のクラス、カテゴリー、および使用法

基本データ項目のクラス[2]	カテゴリー	使用法
英字	英字	DISPLAY
英数字	数字編集 英数字編集 英数字	DISPLAY
数字	数字	DISPLAY (タイプ・ゾーン 10 進数) NATIONAL (タイプ国別 10 進数) PACKED-DECIMAL (タイプ内部 10 進数) COMP-3 (タイプ内部 10 進数) BINARY COMP COMP-4COMP-5
	内部浮動小数点[1]	COMP-1 COMP-2
	外部浮動小数点[1]	DISPLAY
ブール[1]	ブール[1]	DISPLAY
DBCS[1]	DBCS[1] DBCS 編集[1]	DISPLAY-1
国別[1]	国別[1]	NATIONAL
日付/時刻[1]	日付[1] 時刻[1]	DISPLAY PACKED-DECIMAL
	タイム・スタンプ[1]	DISPLAY
1. IBM 拡張 2. すべてのカテゴリーで、グループ項目のクラスは英数字です。		

位置合わせの規則

基本項目にデータを位置決めするときの標準位置合わせの規則は、受け入れ項目 (つまり、データを移動する先の項目で 7-140 ページの『基本移動』を参照) のカテゴリーによって異なります。

数字:

1. データは、**仮想小数点**の位置に合わせて置かれ、必要に応じて切り捨てられるかゼロを埋め込まれます。 (仮想小数点とは、PICTURE 文字の P または V のことであって、論理上の意味はあっても、実際の文字としてデータに存在しない小数点のことです。)
2. 仮想小数点が明示的に指定されていない場合は、受け入れ項目は、フィールドのすぐ右側に仮想小数点が指定されているものとして扱われます。そしてデータは上記の規則に従って扱われます。

数字編集: データは小数点の位置に合わせられ、(必要ならば) 右端または左端のいずれかが切り捨てられるか、またはゼロを埋め込まれます。ただし、編集の結果、先行するゼロが置き換えられる場合は別です。

ただし、当該データのデータ記述記入項目に PICTURE 文節の LOCALE 句が指定されている場合は 6-57 ページの『LOCALE 句』で説明しているように、調整およびゼロの埋め込みまたは切り捨てが行われます。

内部浮動小数点:

IBM Extension

小数点が、フィールドのすぐ左に想定されます。そしてデータが、小数点に続く左端の数字位置に合わせられ、指数もそれに合わせて調整されます。

End of IBM Extension

外部浮動小数点:

IBM Extension

データが左端数字位置に合わせられ、指数がそれに従って調整されます。

End of IBM Extension

英数字、英数字編集、英字:

- データは左端の文字位置に合わせられ、(必要ならば) 右端が切り捨てられるか、またはスペースを埋め込まれます。
- この受け入れ項目に JUSTIFIED 文節を指定した場合は、上記の規則は 6-43 ページの『JUSTIFIED 文節』で説明されているように修正されます。

IBM Extension

- DBCS 受け入れ項目の場合、データは左端の文字位置に合わせられ、(必要ならば) 右端が切り捨てられるか、または DBCS スペースが埋め込まれます。
- DBCS 受け入れ項目に JUSTIFIED 文節を指定した場合は、上記の規則は 6-43 ページの『JUSTIFIED 文節』で説明されているように修正されます。
- 国別データ受け入れ項目の場合、データは左端の文字位置に合わせられ、(必要ならば) 右端が切り捨てられるか、または国別 (UCS-2) スペースが埋め込まれます。
- 国別データ受け入れ項目に JUSTIFIED 文節を指定した場合は、上記の規則は 6-43 ページの『JUSTIFIED 文節』で説明されているように修正されます。

End of IBM Extension

日付、時刻、およびタイム・スタンプ:

IBM Extension

1. USAGE DISPLAY が指定されている日時クラス項目の場合は、データは左端の文字位置に合わせられ、(必要ならば) その右側にスペースが埋め込まれます。
2. USAGE PACKED-DECIMAL が指定されている日時クラス項目の場合は、データは右端の数字位置に合わせられ、(必要ならば) その左側にゼロが埋め込まれます。

End of IBM Extension

標準データ・フォーマット

ILE COBOL 言語の場合、デフォルトのデータ・フォーマットは EBCDIC 文字セットです。

文字ストリングおよび項目のサイズ

ユーザー・プログラムでは、基本項目のサイズは、その PICTURE 文字ストリングの中で指定する文字位置の個数によって決まります。しかし、ストレージ内では、その項目が実際に占めるバイト数 (PICTURE 文字ストリングと USAGE 文節の組み合わせによって判別される) によって項目のサイズが決められます。

USAGE DISPLAY で記述された項目 (カテゴリーは英字、英数字、英数字編集、数字編集、数字、および外部浮動小数点) の場合、項目の PICTURE 文字ストリングと SIGN IS SEPARATE 文節 (該当する場合) によって記述されたそれぞれの文字位置ごとに 1 バイトのストレージが予約されます。

IBM Extension

USAGE DISPLAY-1 で記述される項目 (カテゴリー DBCS) の場合は、項目の PICTURE 文字ストリングによって記述されたそれぞれの文字位置ごとに 2 バイトのストレージが予約されます。

USAGE NATIONAL で記述される項目 (カテゴリーは国別および数字) の場合、項目の PICTURE 文字ストリングと SIGN IS SEPARATE 文節 (符号付きの数字の場合) によって記述されたそれぞれの文字位置ごとに 2 バイトのストレージが予約されます。

LOCALE 句を含む PICTURE 文節を持つ基本項目のサイズは SIZE 句の整数-1 から判別されます。

内部浮動小数点項目については、ストレージ内の項目のサイズはその USAGE 文節によって決められます。 USAGE COMPUTATIONAL-1 は項目用に 4 バイトのストレージを予約し、USAGE COMPUTATIONAL-2 は 8 バイトのストレージを予約します。

日時クラスの基本項目のサイズは、FORMAT リテラルまたは SIZE 句内の整数から判別されます。

End of IBM Extension

通常、算術項目をあるフィールドからそれより短いフィールドに移すと、コンパイラーは短い方の項目の PICTURE 文字ストリングで表された文字数に合わせて、 データを切り捨てます。

例えば、PICTURE S99999 と指定され、+12345 という値が入っている送り出しフィールドから、PICTURE S99 と指定され、BINARY である受け入れフィールドにデータが移されるとすると、そのデータは切り捨てられて +45 になります。詳しくは 6-89 ページの『USAGE 文節』を参照してください。

符号付きデータ

COBOL 内で使用される代数符号には、演算符号と編集符号の 2 つのカテゴリーがあります。

演算符号 (+、-) は、符号付き数字項目と関連した符号であり、その代数特性を示します。代数符号の内部表現は、その項目の USAGE 文節、およびオプショナルの SIGN 文節によって決まります。ゼロは、演算符号にかかわらず、固有の値と見なされます。符号なしフィールドは、常に正またはゼロであると見なされます。

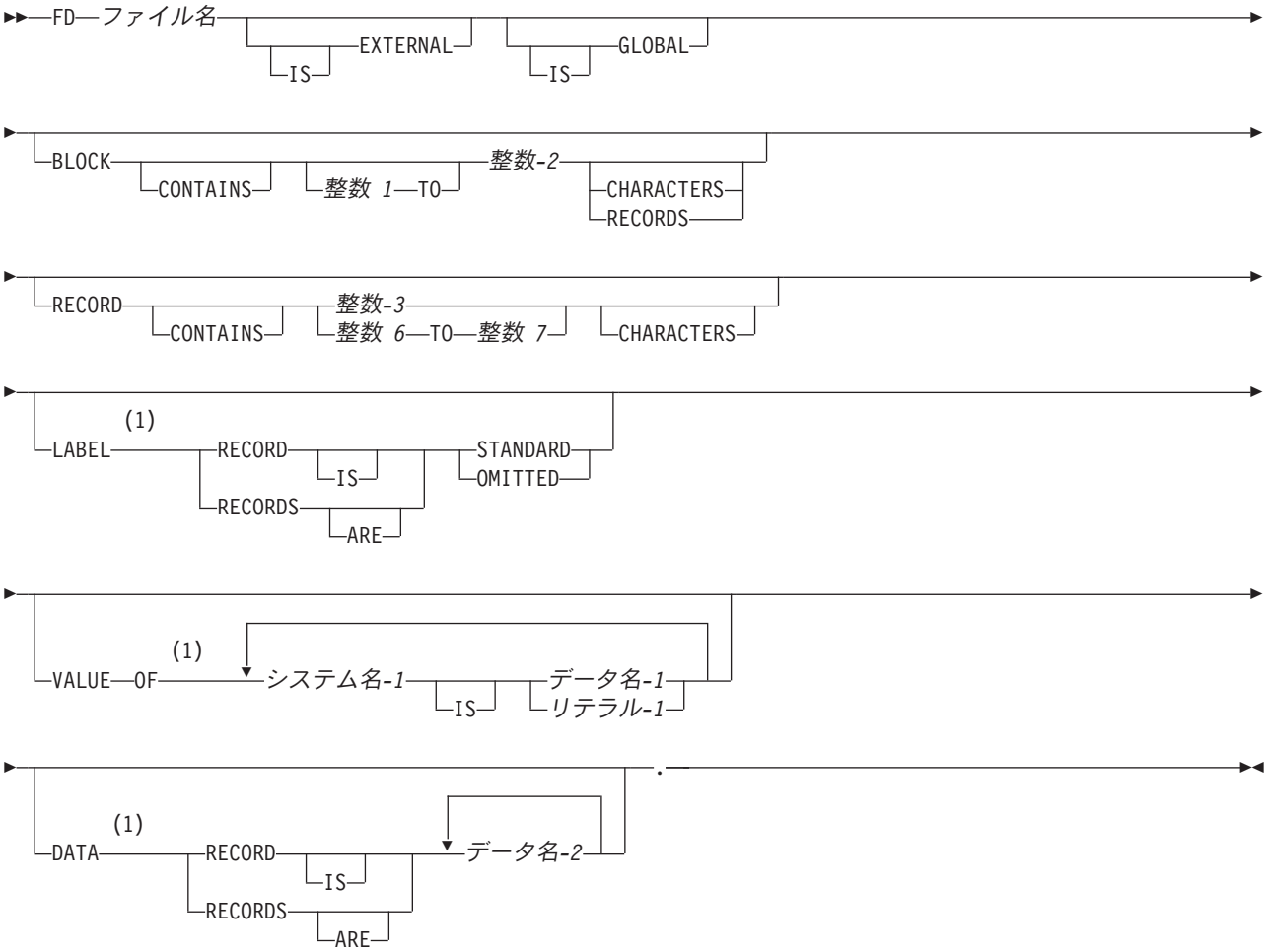
編集符号は、数字編集項目に関連した記号であり、編集出力内の項目の符号を識別する PICTURE 記号 (+、-、CR、DB) です。

データ部 — ファイルとソート記述記入項目

COBOL プログラムでは、**ファイル記述 (FD) 記入項目** (またはソート・マージ・ファイルの**ソート記述 (SD) 記入項目**) は、ファイル・セクションの中の最高レベルの編成を表しています。

ファイル記述記入項目 - 形式 1 - 順次ファイル

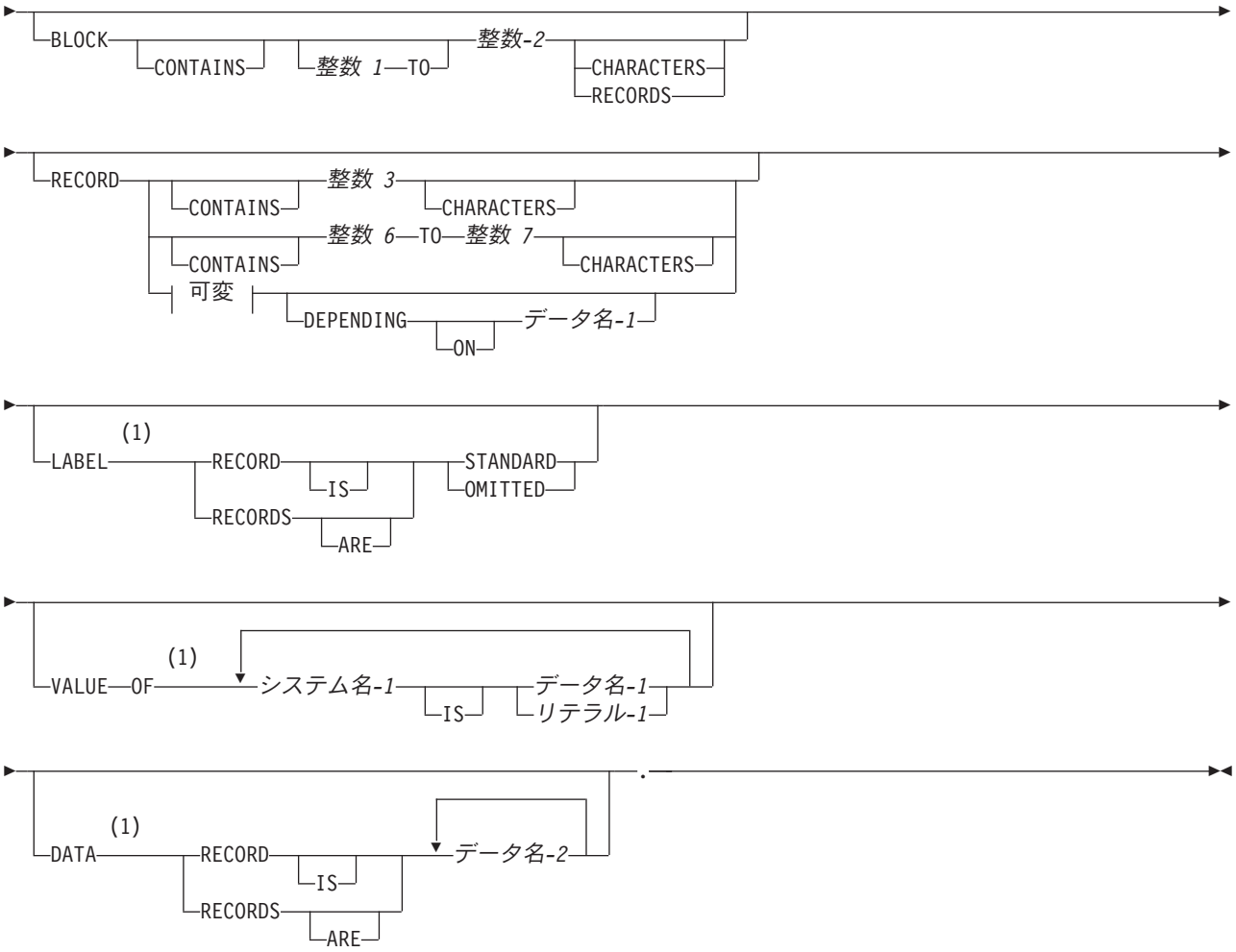
ファイル記述記入項目 - 形式 1a - 形式ファイル、データベース



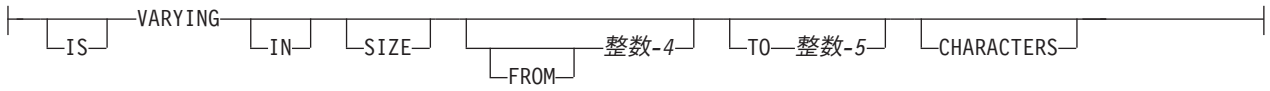
注:

1 構文検査だけ行われます。

ファイル記述記入項目 - 形式 1b - ディスク



可変:

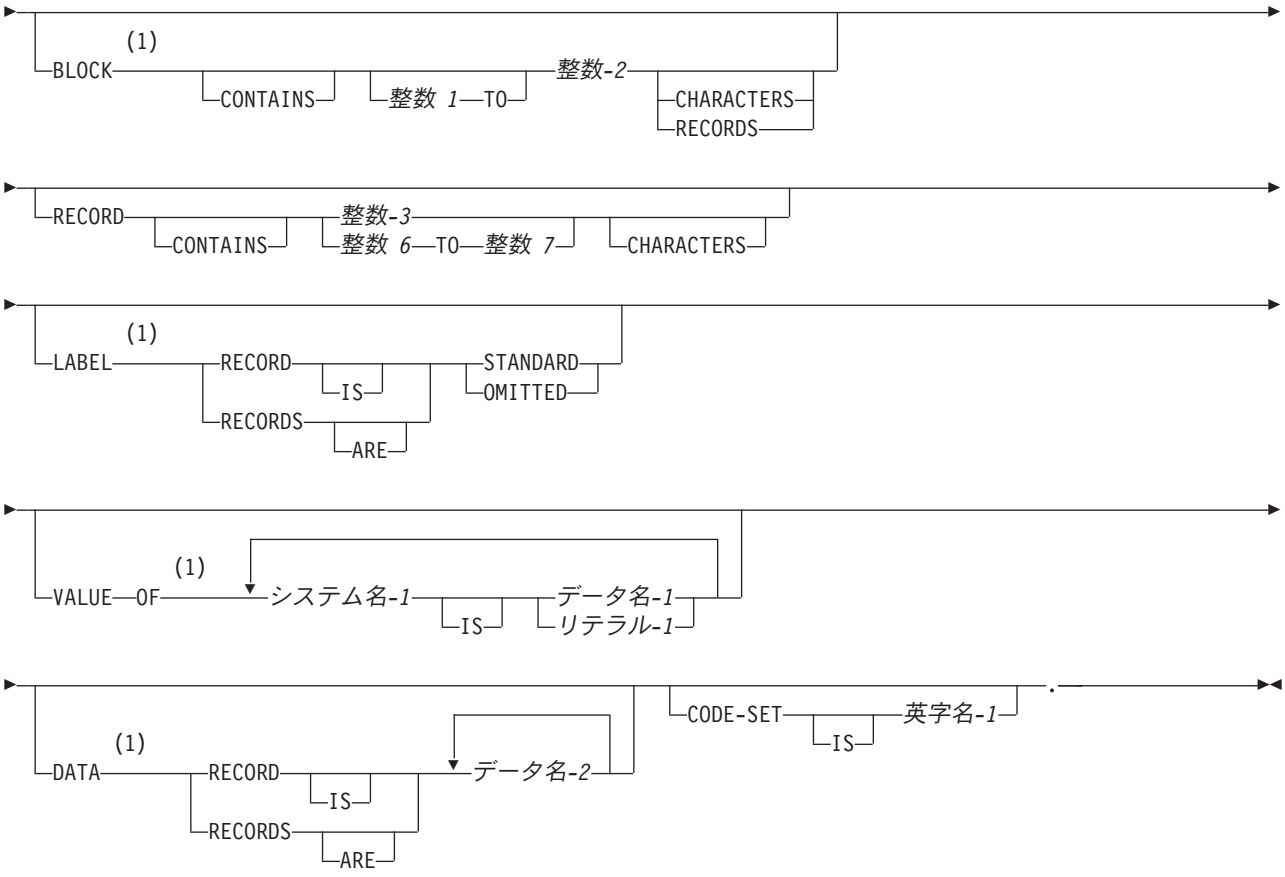


注:

1 構文検査だけ行われます。

ファイル記述記入項目 - 形式 2 - ディスケット・ファイル

ファイル記述記入項目 - 形式 2 - ディスケット

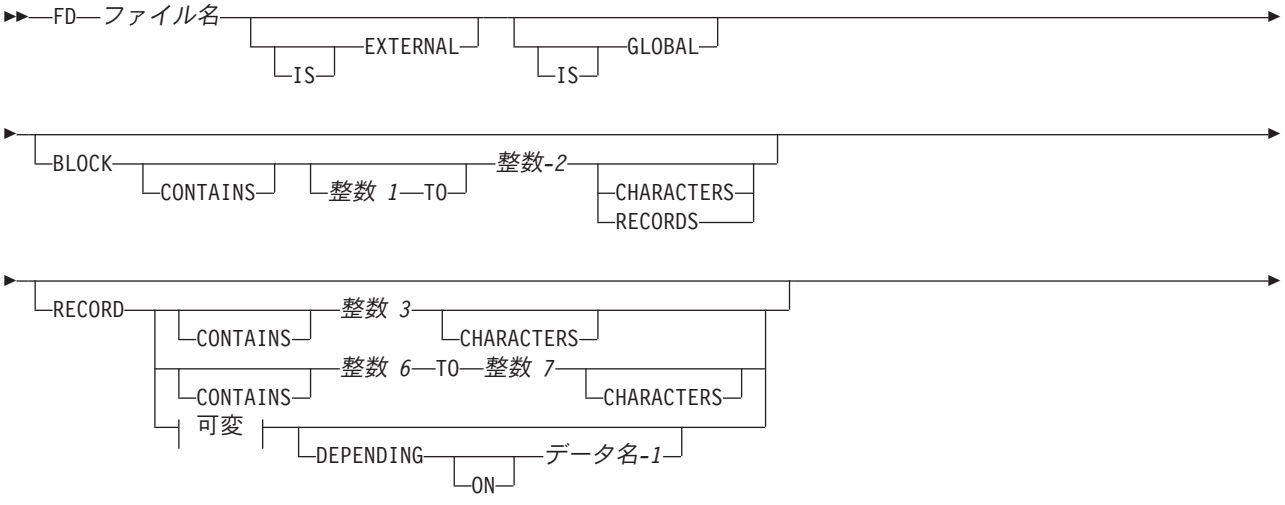


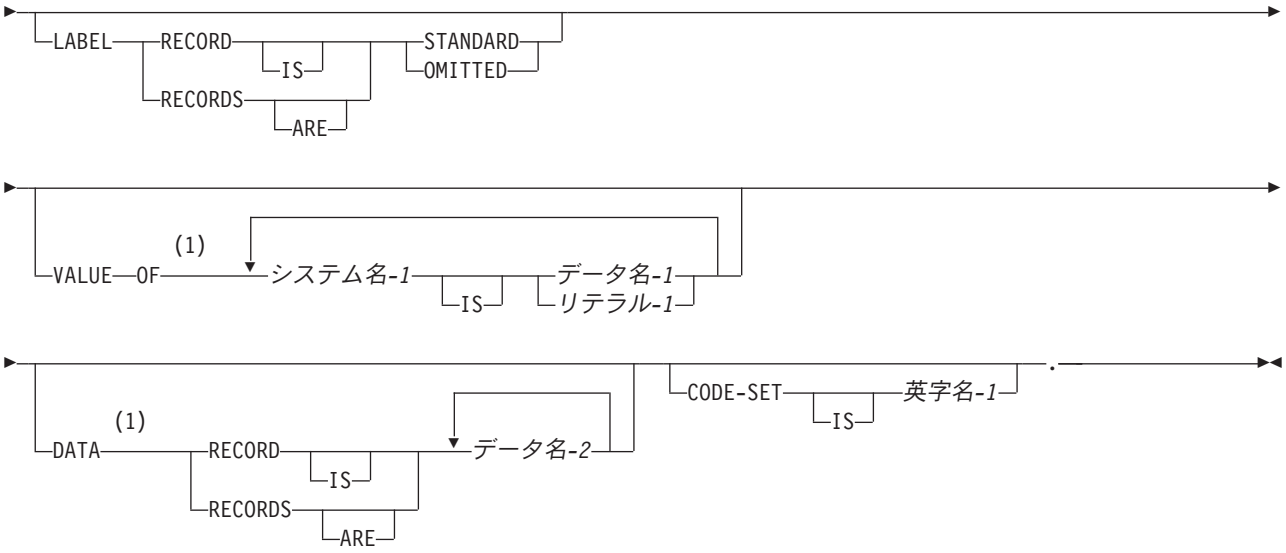
注:

1 構文検査だけ行われます。

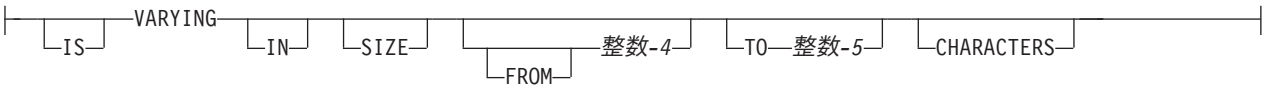
ファイル記述記入項目 - 形式 3 - テープ・ファイル

ファイル記述記入項目 - 形式 3 - テープ・ファイル





可変:

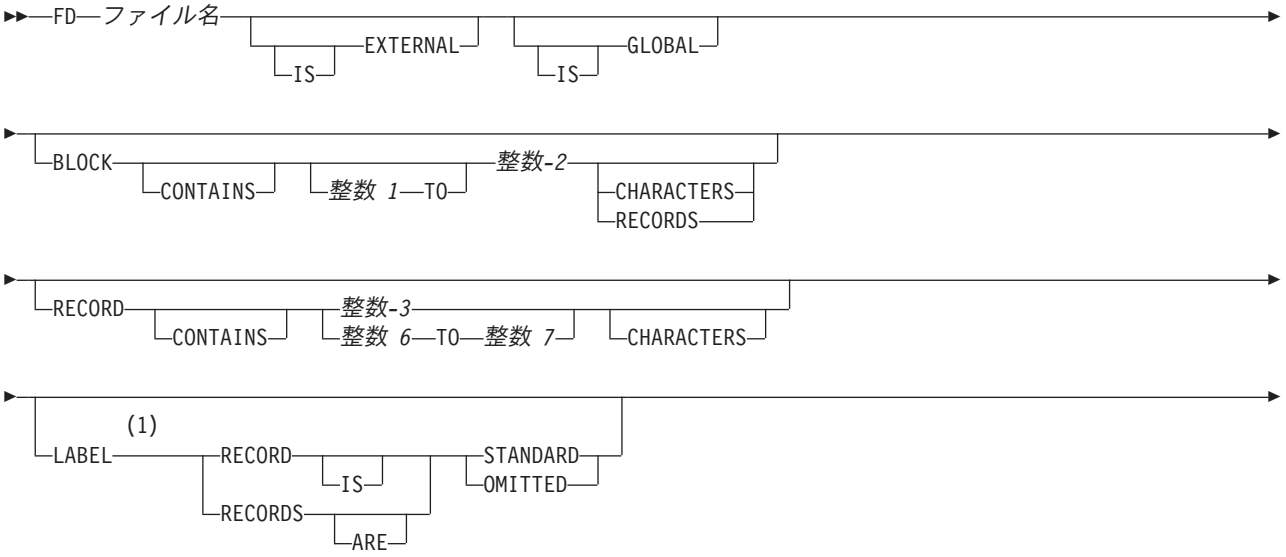


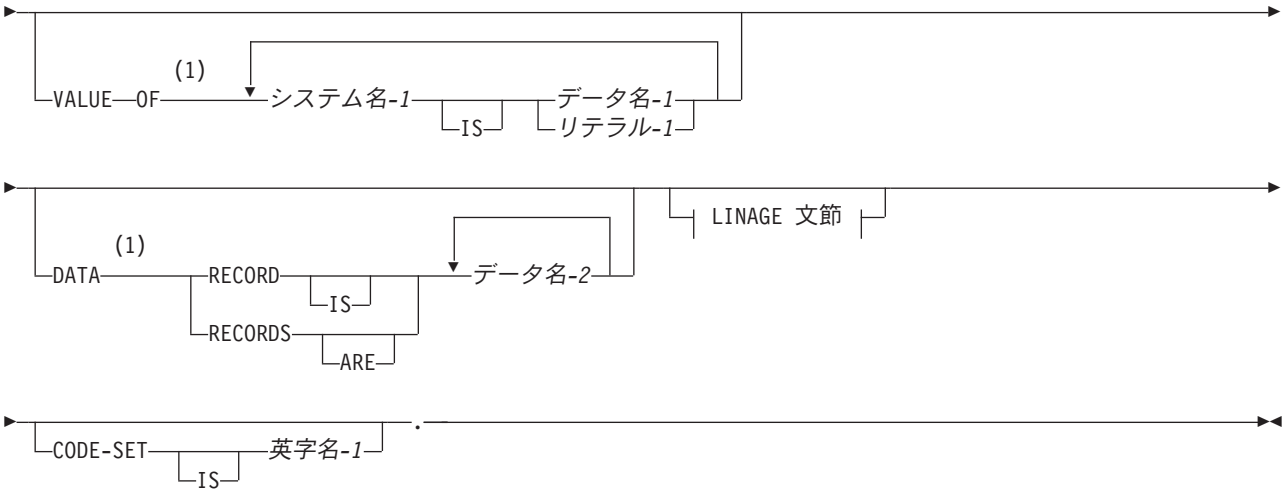
注:

1 構文検査だけ行われます。

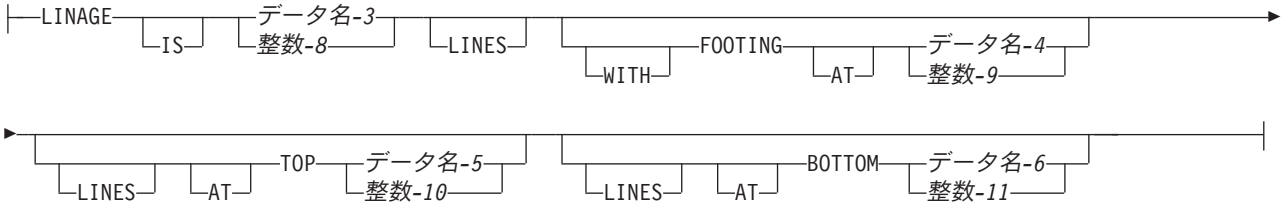
ファイル記述記入項目 - 形式 4 - プリンター・ファイル

ファイル記述記入項目 - 形式 4 - プリンター





LINAGE 文節:

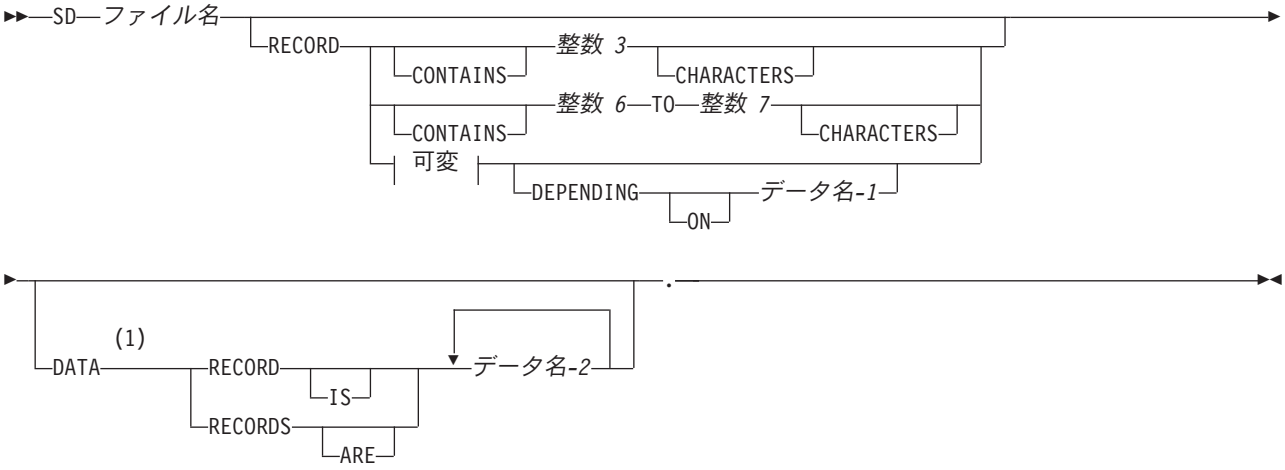


注:

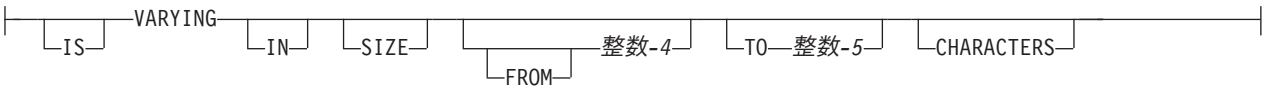
1 構文検査だけ行われます。

ソート記述記入項目 - 形式 5 - ソート・ファイルまたはマージ・ファイル

ファイル記述記入項目 - 形式 5 - ソート・マージ



可変:



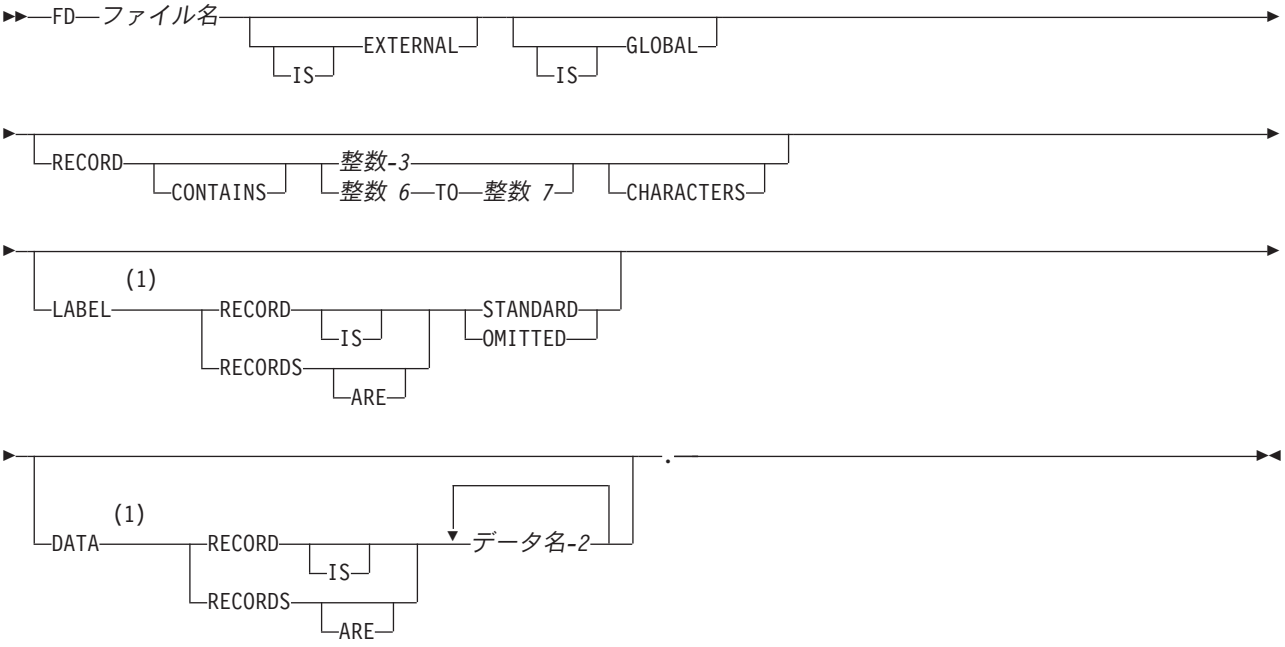
注:

1 構文検査だけ行われます。

ファイル記述記入項目 - 形式 6 - トランザクション・ファイル

IBM Extension

ファイル記述記入項目 - 形式 6 - トランザクション



注:

1 構文検査だけ行われます。

End of IBM Extension

ファイル・セクション

ファイル・セクションには、それぞれの入力または出力ファイルごとにレベル標識がなければなりません。

- ソート・マージ・ファイル以外のファイルの場合は、必ず FD 記入項目がファイル・セクションに含まれていなければなりません。 FD 記入項目の最後の文節のすぐ後に、分離文字ピリオドを付けなければなりません。
- ソート・ファイルまたはマージ・ファイルの場合は、SD 記入項目がファイル・セクションに含まれていなければなりません。 SD 記入項目の最後の文節のすぐ後に、分離文字ピリオドを付けなければなりません。

ファイル名

レベル標識 (FD または SD) の後に記入します。これに関連する SELECT 文節で指定するファイル名と同じでなければなりません。ファイル名は、最低限 1 つの文字が英字でなければならないという、ユーザー定義語の作成に対する規則を守らなければなりません。ファイル名は、当該プログラムの中で固有のものでなければなりません。

ファイル名の後には、1 つまたは複数のレコード記述記入項目が続かなければなりません。 レコード記述記入項目はタイプ名を記述できます。タイプ名ではない各記入項目は、それぞれ同じ記憶域の再定義を暗黙指定しています。

ファイル名の後に続く文節の指定はオプショナルであり、任意の順序で置くことができます。

EXTERNAL 文節

EXTERNAL 文節は、ファイル・コネクターが外部であることを指定します。

EXTERNAL 文節 - 形式

実行単位内に示される外部ファイルは 1 つだけです。外部ファイルは、実行単位内にあってしかもそのファイルを記述する任意の COBOL プログラムで参照できます。

ファイル・セクションでは、EXTERNAL 文節はファイル記述記入項目にしか指定できません。ファイル記述記入項目に入れるレコードは、対応する外部ファイル記述記入項目内に同じ名前をもっている必要はありません。さらに、そのようなレコードの数は、対応するファイル記述記入項目内のものと同じである必要もありません。しかし、対応する外部ファイル記述記入項目での最大レコード長は同じでなければなりません。

EXTERNAL ファイルの場合、対応する EXTERNAL ファイルのすべての BLOCK CONTAINS 文節の値は、実行単位内で互いに一致しなければなりません。一致が必要なのは文字位置に関してだけで、その値が CHARACTERS または RECORDS と指定されているかどうかは関係ありません。

実行単位内にあってしかも外部ファイル・コネクターと関連しているすべてのファイル記述記入項目には、次のものがなければなりません。

- いずれかのファイル記述記入項目に LINAGE 文節があれば、LINAGE 文節
- 整数-8、整数-9、整数-10、および整数-11 が指定されていれば、これらに対応した同じ値
- データ名-3、データ名-4、データ名-5、およびデータ名-6 が参照するものに対応した同じ外部データ項目

EXTERNAL 文節を使用しても、それに関連したファイル名がグローバル名であることを意味しません。

TYPEDEF 文節を EXTERNAL 文節と同じデータ記述記入項目に指定することはできません。ただし、TYPE 文節の場合は可能です。

外部ファイルについての考慮事項

一般的に、外部ファイルの定義はすべて同じでなければなりません。不一致があると、始動時に各種定義が比較されるとプログラムは失敗します。外部ファイルの属性は、次のように比較されます。

- ファイルに対応するいずれかの定義が外部記述されている (例えば、COPY ステートメントの形式 2 を使って) と、他の定義もすべて外部記述されていなければなりません。すべての定義に関連したレベル検査情報は、一致していなければなりません。

EXTERNAL 文節

- ASSIGN TO 文節に指定された名前は、すべての定義で同じでなければなりません。 これには、装置タイプも含まれます。
- ファイルのすべての定義で、ORGANIZATION モードと ACCESS モードは同じでなければなりません。
- OPTIONAL 句は、指定するなら、ファイルのすべての定義に対して指定しなければなりません。
- RELATIVE KEY 句に指定する外部データ項目は、ファイルのどの定義でも、同じ物理位置にあってしかも同じバイト数を占めていなければなりません。
- 関連したレコード内のレコード・キーの位置は、ファイルのどの定義でも同じでなければなりません。
- ファイルに関連したブロック化情報は、ファイルのどの定義でも同じでなければなりません。これには、ブロック化を実行するかどうかと、ブロック・サイズも含まれます。
- RECORD 文節の最大または最小文字数の値は、ファイルのどの定義でも同じでなければなりません。
- CODE-SET 文節に指定された文字セットは、ファイルのどの定義でも同じでなければなりません。
- DUPLICATES 句に指定された値は、ファイルのどの定義でも同じでなければなりません。
- LINAGE 文節に指定された値は、ファイルのどの定義でもすべて同じでなければなりません。
- ASSIGN 文節の属性の指定 (別々の標識) は、ファイルのどの定義でも同じでなければなりません。
- COMMITMENT CONTROL 文節の指定は、ファイルのどの定義でも同じでなければなりません。
- ファイルの定義の入ったすべてのモジュールで、*DUPKEYCHK または *INZDLT コンパイル時オプションの指定は同じでなければなりません。
- コマンド CRTCBLMOD または CRTBNDCBL 用の CCSID パラメーターの指定は、当該ファイルの定義を含むすべてのモジュールについて同一でなければなりません。

GLOBAL 文節

GLOBAL 文節は、ファイル名で指定されるファイル・コネクターがグローバル名であることを指定します。

GLOBAL 文節 - 形式

グローバル・ファイル名は、それを宣言するプログラム、およびそのプログラムに直接的または間接的に含まれるすべてのプログラムで使用できます。

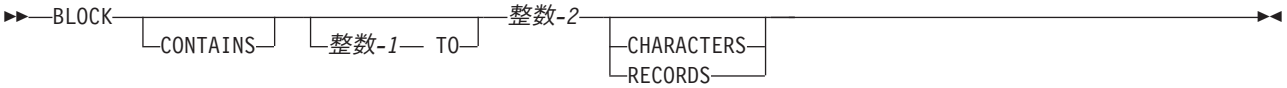
ファイル名は、そのファイル名のファイル記述記入項目に GLOBAL 文節が指定されていると、グローバル名になります。 レコード名がグローバルになるのは、そのレコード名を宣言したレコード記述記入項目に GLOBAL 文節が指定されているときか、またはファイル・セクションのレコード記述記入項目の場合は、そのレコード記述記入項目と関連したファイル名のファイル記述記入項目に GLOBAL 文節が指定されているときです。このようなレコード記述記入項目はタイプ名を記述できます。

GLOBAL 文節は TYPEDEF 文節と同じデータ記述記入項目に指定することができます。この文節の有効範囲が適用されるのはタイプ名に対してだけです。TYPE 文節でグローバル・タイプ名を使用して定義したデータ項目には適用されません。

BLOCK CONTAINS 文節

BLOCK CONTAINS 文節は、物理レコードのサイズを指定します。

BLOCK CONTAINS 文節 - 形式



したがってこの文節は、各物理レコードにただ 1 つの完全な論理レコードが入っているときは、省くことができます。

FORMATFILE ファイルとプリンター・ファイルでは、この文節は構文検査されます。この文節を他のファイル・タイプに対して活動化するには、OPTION パラメーター値の *BLK または PROCESS オプション BLK を使用してください。

整数-1、整数-2
ゼロ以外の符号なしの整数でなければなりません。この整数は、文字数またはレコード数を指定します。整数-2 が 0 の場合には、システムによってブロック・サイズが決定されます。

CHARACTERS
データ・レコード内の文字がどのような USAGE をもっているかには関係なく、物理レコードを保管するのに要する文字位置の数を指定します。

整数-2 だけを指定するならば、それは物理レコードの正確な大きさを示します。整数-1 と整数-2 の両方を指定するならば、それらはそれぞれ物理レコードの最小と最大の文字サイズを表します。

整数-1 と整数-2 には、物理レコードに含まれる制御バイトや埋め込みバイトも入れなければなりません。 (論理レコードには埋め込みバイトは含まれません。)

非テープ・ファイルの場合、整数-2 だけがブロック化因数を制御します。整数-2 が 0 の場合には、システム・デフォルトのブロック化因数が適用されます。

CHARACTERS 句がデフォルトです。 CHARACTERS は、物理レコードに埋め込みが行われているときに指定しなければなりません。

一般的に、RELEASE、REWRITE、または WRITE ステートメントの可変長レコードの長さは、データ名-1 (指定されている場合) で決まります。データ名-1 が指定されていなくて、レコード記述にテーブルが入っていないときは、長さはそのレコード記述内の文字数になります。データ名-1 は指定されていないけれど、レコードにテーブルが入っているときは、長さは、そのレコードの固定部分にその現行テーブル長を加算したものになります。

可変長レコードをディスク・ファイルに使うとき、BLOCK CONTAINS 文節はそのブロック・サイズを指定します。 READ 操作後、データ名-1 の中に実際のレコード・サイズが入ります。可変長レコードで WRITE を実行するには、データ名-1 をそのレコード長に設定しなければなりません。

テープ・ファイルの場合に、データをテープに転送するときは、それぞれの可変長レコードには 4 バイトのヘッダーが、またそれぞれのブロックにも 4 バイトのヘッダーが入ります。しかし、これらの 4 バイトのヘッダーはシステムにより提供され、可変長レコードの最大サイズは 32 764 に制限されていることを除いて、COBOL ユーザーには関係がありません。

テープ・ファイルに可変長レコードが使用されるとき、BLOCK CONTAINS 文節は最大物理レコード長を指定します。これに対して各レコードの論理レコード長は、コンパイラーによって WRITE ステートメントに使用されるレコード名から推測されます。明示的な長さが READ ステートメントの後に必要とされる場合は、ユーザーは I-O-FEEDBACK 簡略名によって、それを得ることができます。

RECORDS
各物理レコードに含まれる論理レコード数を指定します。

最大レコード・サイズは 32 767 で、最大ブロック・サイズは 32 767 です。これらの最大値には、可変長ブロック・レコードに必要な任意の制御バイトが含まれているため、可変長ブロック・レコードの最大サイズ・データ・レコードは 32 759 になります。

RECORD 文節

RECORD 文節は、固定長または可変長のレコードの文字位置の数を指定します。

RECORD 文節 - 形式 1

形式 1 は固定長レコードの文字位置の数を指定します。

RECORD 文節 - 形式 1

整数-3

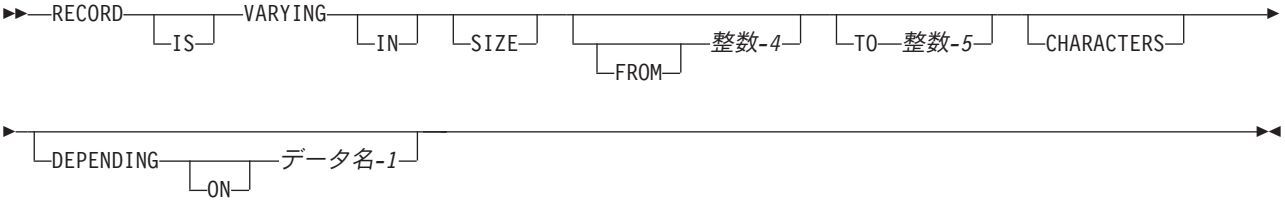
ファイル内の各レコードに入っている文字位置の数を指定する、符号なしの整数でなければなりません。

レコード記述記入項目で決定された最大レコード長が RECORD 文節に指定された長さと一致しないと、最大値が使われます。

RECORD 文節 - 形式 2

形式 2 は、可変長レコードを取り扱うときの推奨形式です。

RECORD 文節 - 形式 2



整数-4

ファイルの任意のレコードに含まれる文字位置の最小数を指定します。整数-4 を指定しないと、ファイルの任意のレコードに含まれる文字位置の最小数は、そのファイルのレコード用に記述された最小文字位置数になります。整数-4 を指定するなら、ゼロ以外で、整数-5 未満にしなければなりません。

整数-5

ファイルの任意のレコード内の文字位置の最大数を指定します。整数-5 を指定しないと、ファイルの任意のレコードに含まれる文字位置の最大数は、そのファイルのレコード用に記述された最大文字位置数になります。

データ名-1

符号のない基本整数でなければなりません。

データ名-1 が指定された場合

- ファイルに対して RELEASE、REWRITE、または WRITE ステートメントを実行する前に、レコード内の文字位置数を、データ名-1 が参照するデータ項目に入れておかなければなりません。

- DELETE、RELEASE、REWRITE、START、または WRITE ステートメントを実行しても、または READ または RETURN ステートメントが失敗しても、データ名-1 が参照するデータ項目の内容は変わりません。
- ファイルに対する READ または RETURN ステートメントの実行が正常に完了すると、データ名-1 が参照するデータの内容に、読み取られたばかりのレコード内の文字位置数が示されます。

レコード記述に関連した文字位置数は、すべての基本データ項目 (再定義と名前変更を除く) 内の文字位置数に、同期化に起因したすべての暗黙の FILLER を加算したもので判別されます。

テーブルが指定されていると、上記の合算にはレコードに記述されたテーブル・エレメントの最小数と最大数が使われて、レコード記述に関連した文字位置の最小数と最大数が判別されます。

書き込まれる論理レコード内の文字位置数が整数-4 以下または整数-5 以上であれば出力ステートメントは失敗し、RELEASE ステートメントの実行時を除いて、それに関連した入出力状況キーは、その条件の原因を示す値に設定されます。

RELEASE、REWRITE、または WRITE ステートメントの実行時のレコード内の文字位置数は、次のような条件で決まります。

- データ名-1 が指定されていれば、データ名-1 が参照するデータ項目の内容による。
- データ名-1 が指定されていなくて、レコードに可変長オカレンス・データ項目が入っていないときは、そのレコード内の文字位置数による。
- データ名-1 は指定されていないが、レコードに可変長オカレンス・データ項目が入っているときは、固定部分に、出力ステートメントの実行時のオカレンス数に記述されたテーブルの該当部分を加算したものによる。

READ ... INTO ステートメントまたは RETURN ... INTO ステートメントの実行時、暗黙の MOVE ステートメント内の送り出しデータ項目として関与する現行レコード内の文字位置数は、次のような条件で判別されます。

- データ名-1 が指定されていれば、データ名-1 が参照するデータ項目の内容による。
- データ名-1 が指定されていなければ、データ名-1 がもし指定されていればデータ名-1 が参照するデータ項目へ移動されたはずの値による。

RECORD 文節 - 形式 3

形式 3 は、固定長または可変長レコードのいずれかの文字位置の数を指定します。 (後者は、テープ・ファイルにだけ適用されます。)

RECORD 文節 - 形式 3

テープ・ファイルの場合

この場合、レコードは可変長であり、次のような記述が適用されます。

整数-6、整数-7

これらは、符号なし整数でなければなりません。整数-6 は、最小データ・レコード・サイズを指定し、整数-7 は最大データ・レコード・サイズを指定します。

他のすべてのファイルの場合

形式 3 をテープ以外のファイルに使用すると、レコードは最大データ・レコードのサイズの固定長レコードとして取り扱われます。論理レコードは、CRTxxxF CL コマンドに定義されたとおりのレコード長になるよう、切り捨てられるかまたは埋め込みが行われます。次の表のユーザー指定のレコード長は、そのファイルの中の最大長レコードを定義したもので、レコード記述によって指定したものです。

入出力タイプ	ユーザー指定長がファイルのレコード長より短い場合	ユーザー指定長がファイルのレコード長より長い場合
入力	切り捨て	ブランクの埋め込み
出力	ブランクの埋め込み	旧ファイル (空ではない) の場合は切り捨て。新ファイル (空ファイル) の場合、長い方のレコード長を使用

注: ファイルの最大レコード長は 32 767 です。

すべての形式についての一般考慮事項

RECORD 文節を使用するとき、レコード・サイズは、レコードを内部で保管するのに必要な文字位置数に指定しなければなりません。つまり、レコード内のデータ項目を表すのに使用される文字の個数ではなく、レコードの文字が内部で占有するバイト数を指定しなければなりません。レコードのサイズは、グループ項目のサイズを得る際の規則に従って決められます。

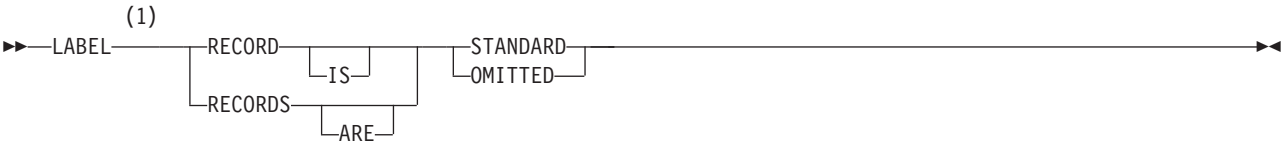
RECORD 文節を省略すると、コンパイラーはレコード記述からそのレコード長を判別します。レコード記述内の記入項目の 1 つに OCCURS DEPENDING ON 文節が含まれていると、コンパイラーは可変長項目の最大値を使用して、内部でレコードを保管するのに必要な文字位置数を計算します。

関連するファイル・コネクターが外部ファイル・コネクターである場合、実行単位内にあって、そのファイル・コネクターと関連するすべてのファイル記述記入項目は、同じ文字位置最大数を指定しなければなりません。

LABEL RECORDS 文節

LABEL RECORDS 文節は、ラベルの有無を示します。この文節が有効なのは FD - 形式 3 (TAPEFILE) の場合だけです。他のすべての FD 形式ではこの文節は構文検査されてから、文書化のための記述として扱われます。

LABEL RECORDS 文節 - 形式



注:

1 構文検査だけ行われます。

これがファイルに指定されない場合、ファイルのラベル・レコードはシステム・ラベル仕様に従わなければなりません。

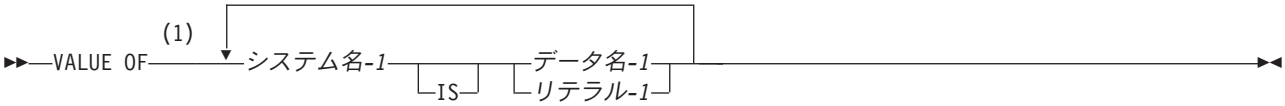
STANDARD
当該ファイルに、システム仕様に従うラベルが付いていることを示します。

OMITTED
ファイルにラベルが付いていないことを示します。

VALUE OF 文節

VALUE OF 文節は、当該ファイルと関連しているラベル・レコードの項目を記述します。文節は構文検査されますが、文書化のための記述と見なされます。

VALUE OF 文節 - 形式



注:

1 構文検査だけ行われます。

システム名-1
ユーザー定義語の形成に関する規則に従わなければなりません。

リテラル-1
任意のリテラルです。

IBM Extension

浮動小数点リテラルは使用できません。

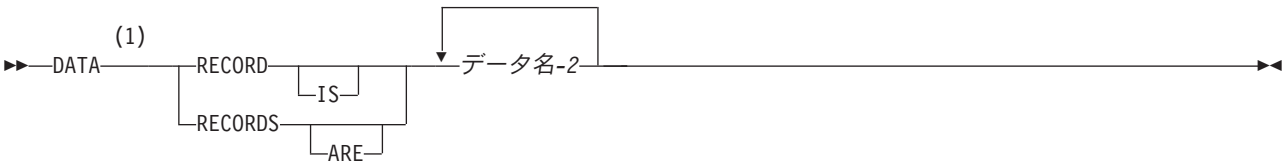
End of IBM Extension

データ名-1
必要なときは修飾されますが、添え字付きにはできません。これは、必ず作業用ストレージ・セクションに記述しなければならず、また、USAGE IS INDEX 文節とともに記述することはできません。

DATA RECORDS 文節

DATA RECORDS 文節は構文検査されますが、ファイルに関連するデータ・レコードの名前の文書化としてだけ使用できます。

DATA RECORDS 文節 - 形式



注:

1 構文検査だけ行われます。

データ名-2
ファイルに関連するレコード記述記入項目の名前。データ名-2 はタイプ名であってはなりません。

DATA RECORDS 文節

複数のデータ名を指定してある場合は、このファイルに複数のタイプのデータ・レコードが含まれていることを示します。このファイルの複数のレコード記述が、同じ記憶域を占めます。これらのレコードの記述または長さは、同じである必要はありません。データ名がリストされる順序は意味をもちません。

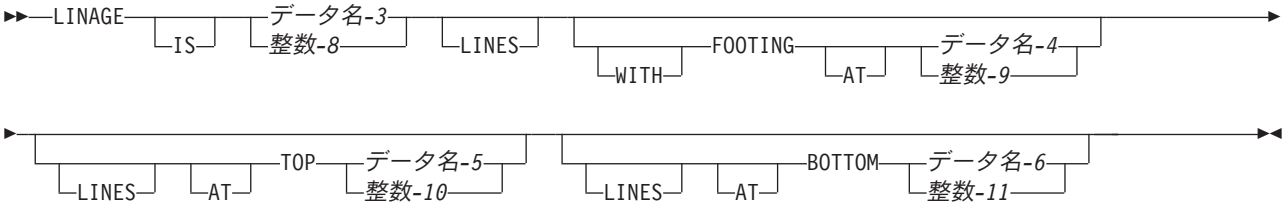
LINAGE 文節

LINAGE 文節は、1 論理ページの大きさを行数で指定します。オプショナルで、この文節はまたフッター域が始まる行番号、および論理ページの上下の端を指定します。 (論理ページと物理ページは同一サイズである必要はありません。)

LINAGE 文節は、装置 PRINTER に割り当てられたファイルにだけ指定できます。 5-29 ページの『ASSIGN 文節』を参照してください。

IBM i のプリンター・ファイルには、DDS を介して活用できる有用な機能がいくつか備わっています。このようなファイルは ILE COBOL プログラムで FORMATFILE と宣言します。プリンター・ファイルについて詳しくは、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) 内のカテゴリー『データベースおよびファイル・システム』のセクション『*DB2 for i*』を参照してください。

LINAGE 文節 - 形式



LINAGE 文節は、選択された装置ファイル内の行数に影響を与えることはなく、COBOL プログラム内の論理ページのメカニズムにだけ影響します。

実行時に、使用されるプリンター・ファイルによって物理ページ・サイズが判別されます。この情報は、LINAGE 文節に定義されたとおりの論理ページを作成するのに適した、スペース・コマンドおよび排出コマンドを出すために使用されます。このようにして、論理ページには複数の物理ページを入れたり、1 つの物理ページに複数の論理ページを入れることができます。

すべての整数は符号なしでなければなりません。すべてのデータ名は符号なしの整数データ項目として記述しなければなりません。

データ名-3、整数-8
この論理ページに書き出したり、またはスペース送りしたりすることのできる、あるいはその両方が可能な行数。これらの行が表すページ域は、**ページ本体**と呼ばれます。値はゼロより大きくなければなりません。

WITH FOOTING AT
整数-9 またはデータ名-4 の値は、ページ本体内のフッター域の最初の行番号を指定します。フッター行番号は、ゼロより大きく、かつページ本体の最終行を超えない値でなければなりません。フッター域は、これらの 2 つの行の間の範囲を占めます。

LINES AT TOP
整数-10 またはデータ名-5 の値は、論理ページの上部マージンの行数を指定します。この値はゼロでもかまいません。

LINES AT BOTTOM
整数-11 またはデータ名-6 の値は、論理ページの下部マージンの行数を示します。この値はゼロでもかまいません。

図 6-2 は、LINAGE 文節の各句の使用法を示しています。

LINAGE 文節の各種の句の図

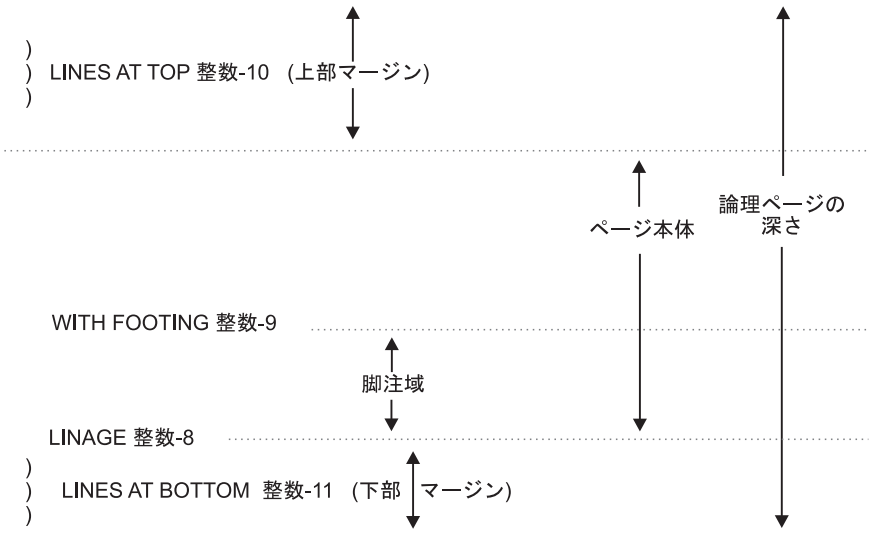


図 6-2. LINAGE 文節の各種の句

LINAGE 文節に指定される論理ページのサイズは、FOOTING 句を除く各句で指定する値の合計です。LINES AT TOP または LINES AT BOTTOM 句 (あるいはその両方) が省略された場合、上部および下部のマージンの値はゼロと見なされます。各論理ページのすぐ後に次の論理ページが続き、ページとページの間にスペースはとられません。

FOOTING 句が指定され、データ名-4 あるいは整数-9 の値がデータ名-3 または整数-8 の LINAGE 値と等しい場合、1 行 (論理ページの最終行) をフッター情報に使用できます。

FOOTING 句が省略される場合、ページ本体 (整数-8 またはデータ名-3) の値と同等と見なされます。

OPEN OUTPUT ステートメントの実行時に、整数-8、整数-9、整数-10、および整数-11 の値 (それらが指定されていれば) が使用されて、そのファイルの論理ページのページ本体、最初のフッター行、上部マージン、および下部マージンが決定されます。 図 6-2を参照してください。このあと、プログラムが実行されている間、そのファイルに関して印刷されるすべての論理ページに対して、これからの値が使用されます。

OUTPUT 句を持つ OPEN ステートメントがファイルに対して実行されるとき、データ名-3、データ名-4、データ名-5、およびデータ名-6 により、最初の論理ページだけに対してページ本体、最初のフッター行、上部マージン、および下部マージンが決定されます。

ADVANCING PAGE 句を持つ WRITE ステートメントの実行時、あるいはページ・オーバーフローの状態が起こったとき、データ名-3、データ名-4、データ名-5、およびデータ名-6 の値 (指定されている場合) を使用し、次の論理ページに対して、ページ本体、最初のフッター行、上部マージン、および下部マージンが決定されます。

LINAGE-COUNTER 特殊レジスター

LINAGE 文節を含む FD 記入項目ごとに、別々の LINAGE-COUNTER 特殊レジスターが生成されます (複数のレジスターが生成されたら、各 LINAGE-COUNTER をそれに関連したファイル名で修飾しなければなりません)。

LINAGE-COUNTER に関する暗黙の記述は、次のいずれかです。

- LINAGE 文節がデータ名-3 を指定すれば、LINAGE-COUNTER にはデータ名-3 と同じ PICTURE と USAGE が入ります。
- LINAGE 文節で整数-8 を指定した場合には、LINAGE-COUNTER は、整数-8 の 2 進数表現を十分に含むことができる 2 進数項目です。

ある時点の LINAGE-COUNTER の値は、現在のページ内の装置が位置付けられている行の行番号です。LINAGE-COUNTER を手続き部のステートメントで参照できますが、そのステートメントで修正することはできません。

そのファイルに対して OPEN ステートメントが実行されたときに、LINAGE-COUNTER は 1 に初期設定されます。

LINAGE-COUNTER は、そのファイルに対する任意の WRITE ステートメントによって自動的に修正されます。 (7-245 ページの『WRITE ステートメント』を参照。)

順次ファイルのファイル記述に LINAGE 文節と EXTERNAL 文節が含まれていると、LINAGE-COUNTER データ項目は外部データ項目になります。同様に、その記述に LINAGE 文節と GLOBAL 文節が含まれていると、LINAGE-COUNTER データ項目はグローバル・データ項目になります。

関数への整数の引数が認められているところではどこでも、LINAGE-COUNTER 特殊レジスターを指定できます。

CODE-SET 文節

CODE-SET 文節は、DISKETTE および TAPEFILE のデータを表すために使われる文字コードを指定します。

CODE-SET 文節 - 形式

CODE-SET 文節が指定されると、英字名は入出力装置でデータを表すのに使われる文字コード規則を識別します。

CODE-SET 文節はまた、入出力メディア上の文字コードを内部 EBCDIC 文字セットへ (またはその逆へ) 変換するためのアルゴリズムを指定します。

英字名-1 は、STANDARD-1 (ASCII エンコード・ファイルの場合)、STANDARD-2 (ISO の 7 ビット・エンコード・ファイルの場合)、NATIVE (EBCDIC エンコード・ファイルの場合)、または EBCDIC (EBCDIC エンコード・ファイルの場合) として SPECIAL-NAMES 段落に 定義しなければなりません。NATIVE を指定すると、CODE-SET 文節は構文検査はされますが、プログラムの実行にはまったく影響しません。

CODE-SET 文節がファイルに指定されると、そのファイル内のすべてのデータは USAGE DISPLAY をもたねばなりません。そして、符号付き数字データがある場合には、それを SIGN IS SEPARATE 文節で記述しなければなりません。

CODE-SET 文節を省略すると、EBCDIC 文字セットと見なされます。

注: テープ・ファイルおよびディスケット・ファイルに関しては、IBM i システムがサポートするのは ASCII と ISO だけです。したがって、CODE-SET 文節で、テープまたはディスケット・ファイルでないファイルに対して、STANDARD-1 (ASCII) または STANDARD-2 (ISO) の文字コード・セットを指定した場合には、警告メッセージが出され、EBCDIC 文字コードが使用されます。

IBM Extension

CODE-SET 文節を省略すると、ディスケット・ファイル作成 (CRTDKTF) またはテープ・ファイル作成 (CRTTAPF) CL コマンドの CODE パラメーターが使用されます。

CODE-SET 文節は、ディスケット・ファイル一時変更 (OVRDKTF) またはテープ・ファイル指定変更 (OVRTAPF) CL コマンドの CODE パラメーターを、プログラムの実行時に指定することによって変更できます。これらのコマンドについて詳しくは、**IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL および API*』を参照してください。

End of IBM Extension

データ部 — データ記述記入項目

データ記述記入項目は、データ項目の特性を指定します。データ項目には、**暗黙** (デフォルト値) または**明示**の 2 つの**属性**があります。

この章では、データ記述記入項目とレコード記述記入項目 (データ記述記入項目の集合を指す) のコーディングについて説明します。この章では、データ記述記入項目とレコード記述記入項目を指すときに、**データ記述記入項目**という 1 つの用語を使います。

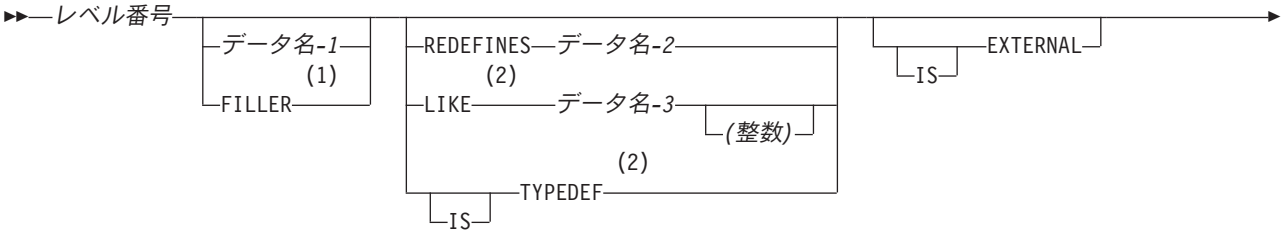
独立したデータ項目を定義するデータ記述記入項目は、レコードを構成しません。これらは**データ項目記述項目**といいます。

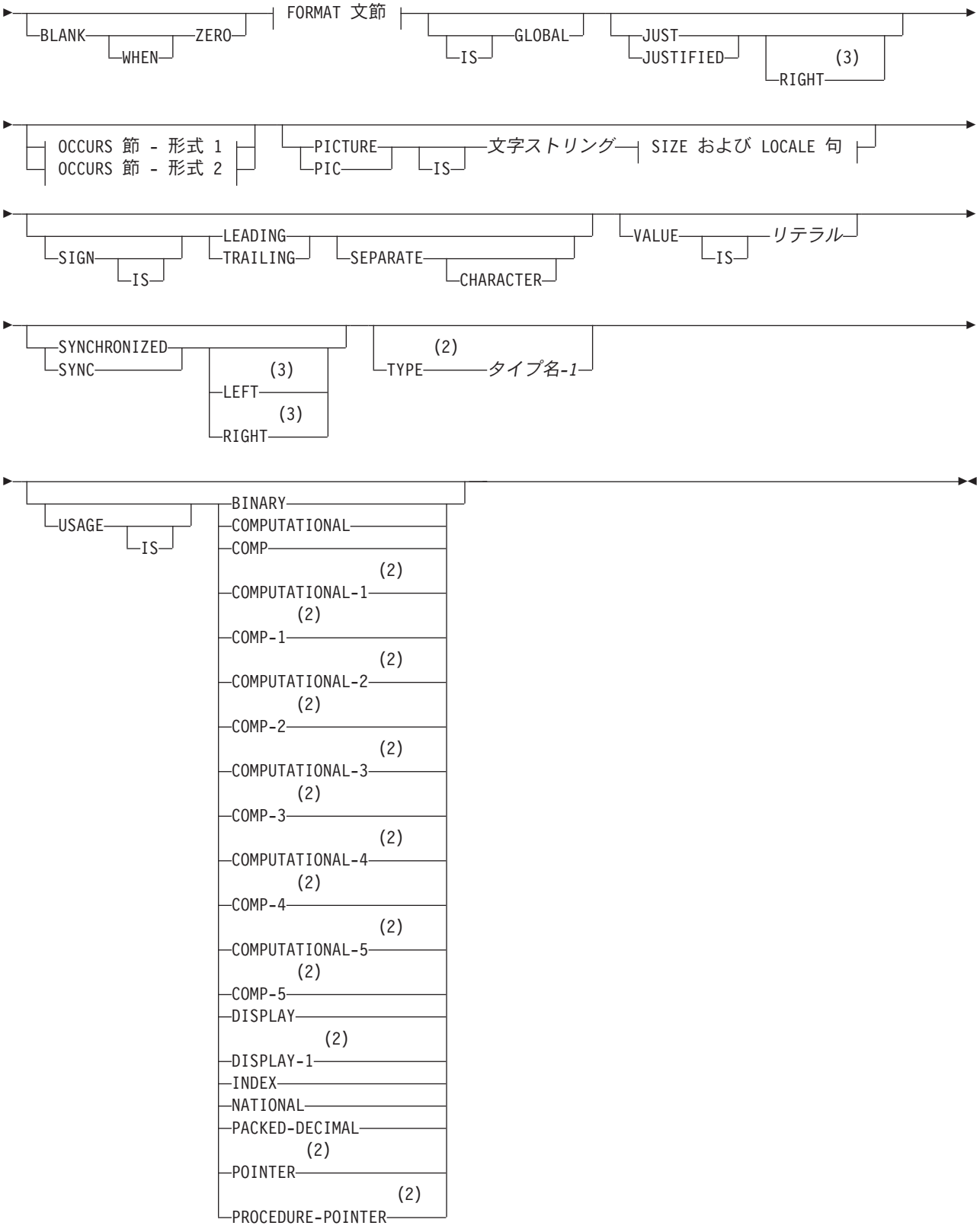
データ記述記入項目には、次ページ以降に示す 4 つの一般形式があります。

形式 1

形式 1 は、すべてのデータ部セクションのデータ記述記入項目に使われます。この形式内のレベル番号は、01 ～ 49、ならびに 77 のいずれかです。

データ記述記入項目 - 一般形式 1





注:

1 TYPEDEF 文節とともに使用することはできません。

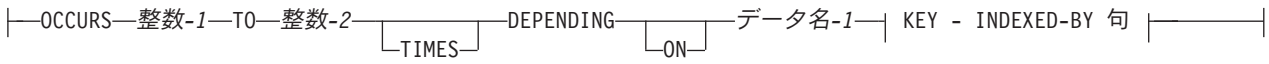
2 IBM 拡張

3 構文検査だけ行われます。

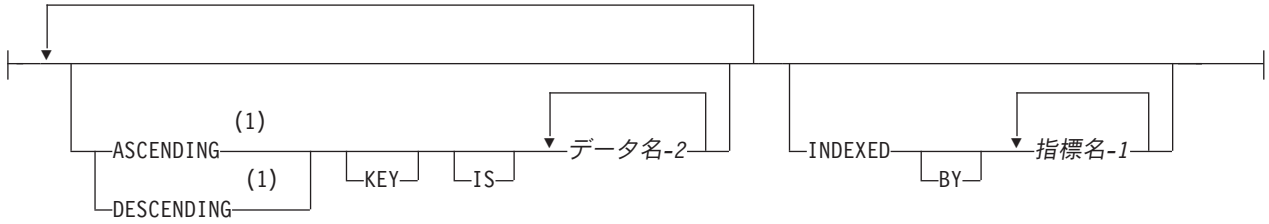
データ記述記入項目 - 一般形式 1 (続き)

OCCURS 文節 - 形式 1:

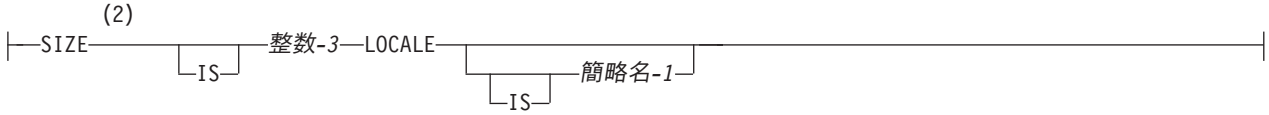
OCCURS 文節 - 形式 2:



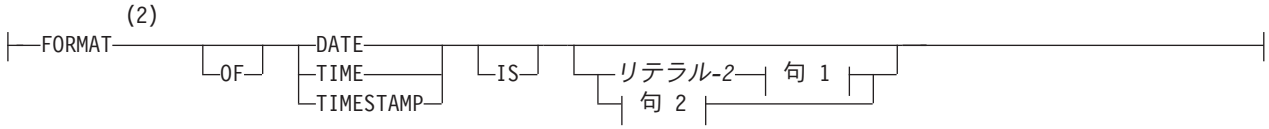
KEY - INDEXED-BY 句:



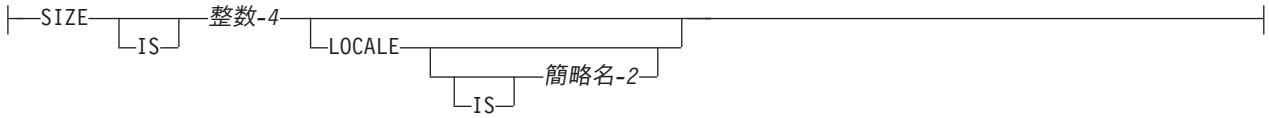
SIZE 句および LOCALE 句:



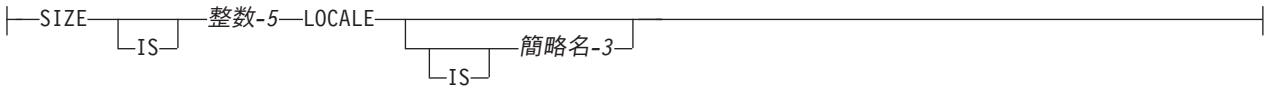
FORMAT 文節:



句 1:



句 2:



注:

1 ブール・データ・タイプとともに使用することはできません。

2 IBM 拡張

文節は、次の 3 つの例外を除いて任意の順序で書くことができます。

- データ名または FILLER が指定される場合、それらはレベル番号の直後に続かなければなりません。
- REDEFINES 文節を指定する場合は、データ名-1 または FILLER に続く最初の記入項目にしなければなりません。データ名-1 または FILLER を指定しない場合には、REDEFINES 文節は、レベル番号に続く最初の記入項目にしなければなりません。記述するデータ項目は、FILLER が指定されているのと同様に扱われます。
- TYPEDEF 文節を指定する場合は、データ名-1 に続く最初の記入項目にしなければなりません。TYPEDEF 文節を FILLER とともに指定することはできません。また、TYPEDEF 文節と REDEFINES 文節の両方をデータ名-1 に対して指定することはできません。

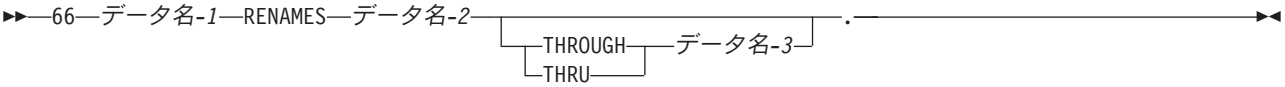
すべての文節が相互に互換性があるというわけではありません。詳細は、個々の文節の説明を参照してください。

文節は、スペース、分離文字のコンマ、または分離文字のセミコロンで区切らなければなりません。

形式 2

形式 2 はすでに定義済みの項目を再グループ化します。

データ記述記入項目 - 一般形式 2



レベル 66 の記入項目は、別のレベル 66 の記入項目や、レベル 01、レベル 77、レベル 88 の記入項目の名前を変更することはできません。

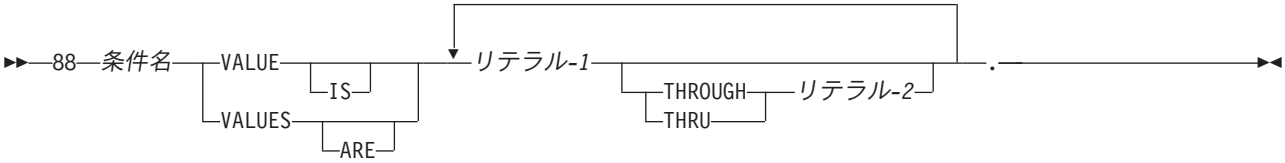
ある 1 つのレコードに関連したレベル 66 記入項目はすべて、そのレコードの最後のデータ記述記入項目のすぐ後になければなりません。

詳細は 6-78 ページの『RENAMES 文節』に説明があります。

形式 3

形式 3 は条件名を記述します。

データ記述記入項目 - 一般形式 3



条件名

ある 1 つの値、一連の値、またはある範囲の値を条件変数と関連付けるユーザー指定名。

条件変数とは、 1 つまたは複数の値を取ることのできるデータ項目であり、その後でそれらの値を条件名と関連付けることができます。

形式 3 は、基本およびグループ項目の両方の記述に使用できます。条件名項目についての詳細は 7-13 ページの『条件名条件』に説明があります。

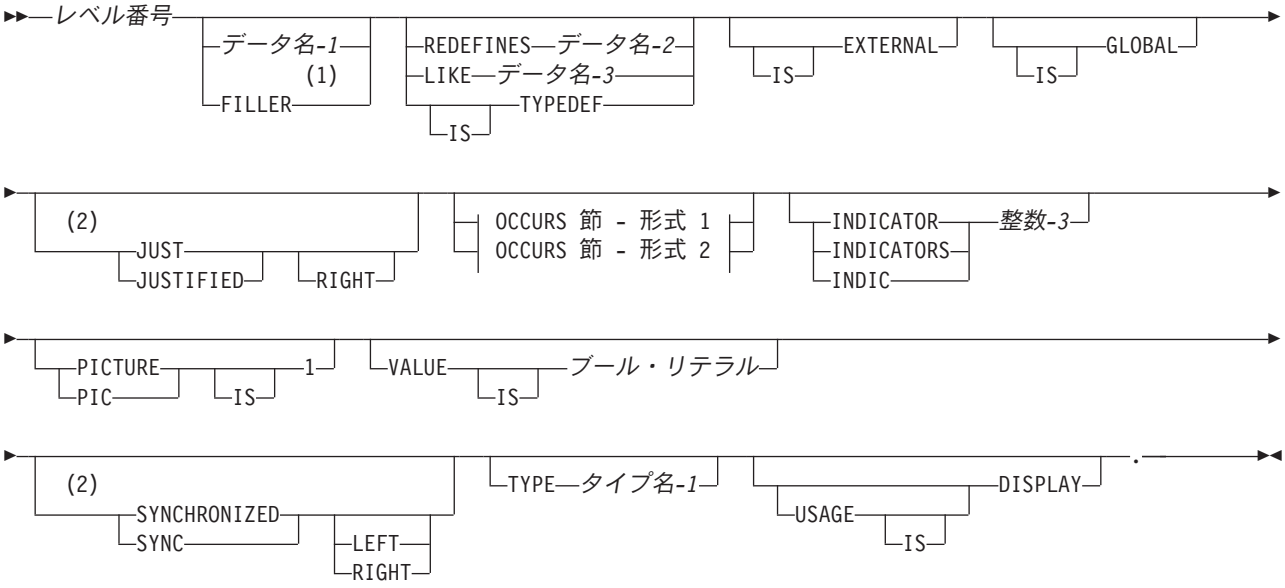
形式 4

IBM Extension

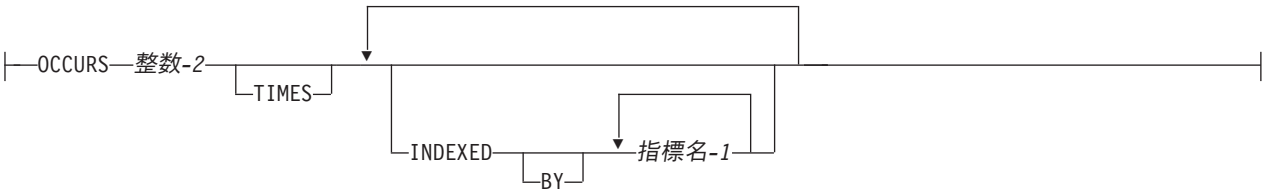
この形式は、ブール・データを記述します。**ブール・データ項目**は値を 1 または 0 に限定された項目です。

注: COBOL プログラムで標識を使用するときには、ブール・データのデータ記述記入項目を用いてそれらをブール・データ項目として記述しなければなりません。

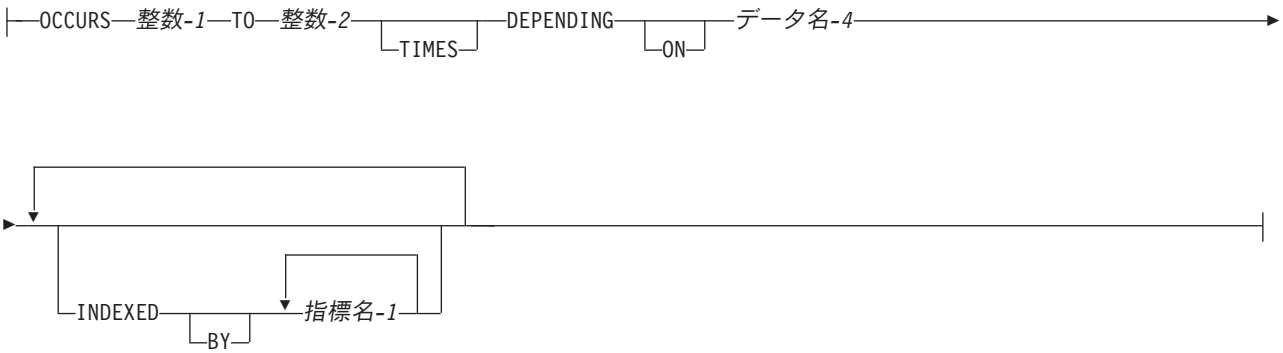
データ記述記入項目 - 形式 4 - ブール・データ



OCCURS 文節 - 形式 1:



OCCURS 文節 - 形式 2:



注:

1 TYPEDEF 文節とともに使用することはできません。

2 構文検査だけ行われます。

ブール・データとともに使用される文節の特別な考慮事項を説明します。文節のその他すべての規則は、他のデータの規則と同じです。

End of IBM Extension

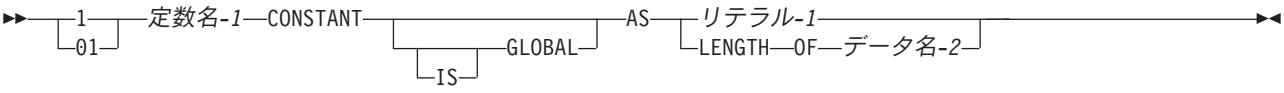
形式 5

IBM Extension

形式 5 は定数名を示します。定数名を記述できるのは、レベル 01 の記入項目に対してだけです。定数名項目についての詳細は 6-35 ページの『CONSTANT 文節』に説明があります。

End of IBM Extension

IBM Extension



End of IBM Extension

CONSTANT 文節

IBM Extension

CONSTANT 文節は、定数名とリテラルの関連付けに使用されます。定数名は、その後リテラルの代わりに使用することができます。 CONSTANT 文節を指定できるのは、基本定数名のレベル 01 の記入項目に対してだけです。 CONSTANT 文節を、事前に定義済みの別の定数名としても定義できます。

定数名は、使用する前に、CONSTANT 文節で定義しておく必要があります。定数名は、データ部および手続き部で使用でき、これらの部では、COPY ステートメントおよび TITLE ステートメントなどのコンパイラー指示ステートメント以外では、リテラルまたは整数が許可されています。

定数名-1 は、形式がクラスのリテラルおよび定数名-1 のカテゴリーを指定する場所であればどこでも使用できます。定数名-1 のクラスおよびカテゴリーは、リテラル-1 のクラスおよびカテゴリーと同じか、または LENGTH OF 句が指定されている場合は整数です。定数名-1 が整数の場合、ピクチャー・ストリングでの繰り返しの指定にも使用される場合があります。

リテラル-1 は、表意定数であってはなりません。

LENGTH OF 句が指定されている場合、定数名-1 の値は、LENGTH 組み込み関数で指定されているとおりに決定されます。ただし、例外として、データ名-2 が、OCCURS DEPENDING ON 文節で記述される可変長データ項目の場合は、データ項目の最大サイズが使用されます。

End of IBM Extension

LIKE 文節

この文節を使用してデータ項目の長さを変更することはできません。

OCCURS 文節

OCCURS 文節および INDICATOR 文節がともに基本レベルで指定されるとき、ブール・データ項目のテーブルは、外部標識に対応するテーブルの各エレメントとともに定義されます。このテーブルの最初のエレメントは、INDICATOR 文節に指定された標識番号に対応し、2 番目のエレメントは、INDICATOR 文節で指定された標識の後に順番どおりに続く標識に対応します。

例えば、次のようにコーディングした場合、

```
07   SWITCHES  PIC 1
               OCCURS 10 TIMES
               INDICATOR 16.
```

SWITCHES (1) は標識 16 に対応し、SWITCHES(2) は標識 17 に対応し、以下同様にして、SWITCHES (10) は標識 25 に対応します。

INDICATOR 文節

標識フィールドが、独立した標識域にある場合には、INDICATOR 文節は、DDS に定義された標識とブール・データ項目を関連付けます。標識フィールドがレコード域にある場合、INDICATOR 文節は構文検査されますが、文書化のための記述として取り扱われます。

整数-3 は、1 ～ 99 の値でなければなりません。

INDICATOR 文節は、基本レベルでのみ指定しなければなりません。

VALUE 文節

VALUE 文節は、ブール・データ項目の開始内容を指定します。ブール・リテラルの許容値は、B “0”、B “1”、およびゼロです。

レベル番号

レベル番号は、レコード内のデータの階層を指定し、また特別な目的を持つデータ項目を識別します。レベル番号は、データ記述記入項目、タイプ名、名前変更または再定義される項目、あるいは条件名記入項目を開始します。レベル番号は、1 ～ 49 の整数または特別レベル番号の 66、77、または 88 のうちのいずれかの値をもちます。

レベル番号

レベル番号の後には、分離文字のピリオドが続くか、または、スペースをあけてその後にそれに関連したデータ名-1、FILLER、または適切なデータ記述文節が続かなければなりません。レベル番号 01 および 77 は区域 A から始めなければなりません。レベル番号 77 の後には、スペースあけてその後にそれに関連したデータ名-1 が続かなければなりません。レベル番号 02 ～ 49、66、そして 88 は区域 A または B で始めます。

レベル番号 01 ～ 09 の代わりに、1 桁のレベル番号 1 ～ 9 を指定することもできます。

連続するデータ記述記入項目は、最初の項目と同じ桁から始めることも、あるいはレベル番号に合わせて字下げすることもできます。字下げしても、レベル番号の意味に影響はありません。

レベル番号を字下げすると、新しいレベル番号ごとに区域 A の右側に任意の数のスペースをあけてから始めることができます。右側へ字下げする範囲は、区域 B の幅による制限しか受けません。

詳しくは、 6-6 ページの『データのレベル』および 6-12 ページの『標準データ・フォーマット』を参照してください。

IBM Extension

1 つのグループ項目に直接従属する基本項目またはグループ項目に付けるレベル番号は個々に違っていてもかまいません。

End of IBM Extension

データ名-1

記述されるデータを明示的に識別します。

データ名-1 を指定した場合、データ名-1 はプログラムの中で使用されるデータ項目を識別します。このデータ項目は、レベル番号の後につづく最初の語でなければなりません。

プログラム実行時に、データ項目を変更できます。

次のものにデータ名-1 を指定しなければなりません。

- レベル 66、レベル 77、およびレベル 88 の項目
- 文節 GLOBAL、EXTERNAL、または TYPEDEF を含む記入項目
- GLOBAL 文節または EXTERNAL 文節の入ったファイル記述項目と関連したレコード記述記入項目

FILLER

これは、プログラムの中で明示的に参照されることのないデータ項目です。キーワード FILLER の指定はオプショナルです。指定する場合、FILLER はレベル番号の直後に指定しなければなりません。

条件変数にではなくそれが取ることのできる値にだけ明示参照が行われるなら、条件変数と一緒にキーワードの FILLER を使えます。 FILLER を条件名またはタイプ名とともに使用することはできません。

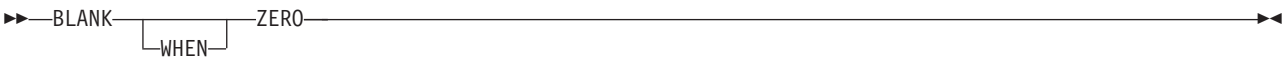
MOVE CORRESPONDING ステートメント、ADD CORRESPONDING、または SUBTRACT CORRESPONDING ステートメントにおいては、FILLER 項目は指定しても無視されます。
INITIALIZE ステートメントでは、基本 FILLER 項目は無視されます。

データ名-1 または FILLER 文節が省略された場合には、記述されているデータ項目は FILLER が指定されたものとして処理されます。

BLANK WHEN ZERO 文節

BLANK WHEN ZERO 文節は、項目の値がゼロのときはその項目にスペースだけが入ることを指定します。

BLANK WHEN ZERO 文節 - 形式



BLANK WHEN ZERO 文節は、基本数字項目または数字編集項目に関してのみ指定できます。これらの項目は、USAGE IS DISPLAY として暗黙にまたは明示的に記述されなければなりません。 BLANK WHEN ZERO 文節が数字項目に関して指定されると、その項目は数字編集項目と見なされます。

BLANK WHEN ZERO 文節は、レベル 66 またはレベル 88 の項目に対して指定してはなりません。

BLANK WHEN ZERO 文節は、PICTURE 記号 S または * を含む記入項目に指定してはなりません。

BLANK WHEN ZERO 文節を以下のものとともに使用することはできません。

- USAGE IS INDEX 文節

IBM Extension

- 日時クラスの項目
- 外部または内部の浮動小数点項目
- USAGE IS POINTER 文節
- USAGE IS PROCEDURE-POINTER 文節で記述された項目
- DBCS 項目
- 国別項目
- TYPE 文節

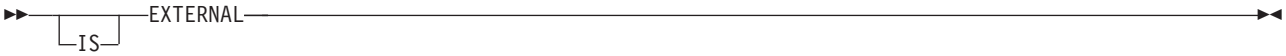
End of IBM Extension

EXTERNAL 文節

EXTERNAL 文節は、データ項目に関連したストレージを、実行単位内の特定のプログラムにではなく、実行単位に関連付けることを指定します。

EXTERNAL 文節

EXTERNAL 文節 - 形式



外部データ項目は、実行単位にあってしかもデータ項目を記述した任意のプログラムから参照できます。データ項目の別々の記述を使ってさまざまなプログラムから外部データ項目を参照しても、常に同一のデータ項目を対象とします。 1 つの実行単位では、外部データ項目は 1 回しか現れません。

EXTERNAL 文節は、作業用ストレージ・セクションの 01 レベル記入項目に、またはファイル記述記入項目に指定できます。同一のデータ部に同一のデータ名の付いた 2 つのデータ記述記入項目があると、そのうち 1 つの項目にしか EXTERNAL 文節を入れられません。外部データ・レコード内の指標名、条件名、および名前変更 (レベル 66) 項目は EXTERNAL 属性を持ちません。

データ名文節で指定されたレコードに含まれるデータは、外部データであり、そのデータを記述し、オプショナルで再定義する実行単位内の任意のプログラムからアクセスしたり処理したりできます。このデータには、次のような規則が適用されます。

- 実行単位内の複数のプログラムが同一の外部データ・レコードを記述するときは、それに関連したレコード記述記入項目のすべてのレコード名は同一でなければならず、それらのレコードは同一数の標準データ・フォーマット文字を定義しなければなりません。しかし、外部レコードを記述するプログラムには、その外部レコード全体を再定義する REDEFINES 文節を含めたデータ記述記入項目を入れることができ、その全体の再定義は、実行単位内の他のプログラムと同様に行う必要はありません。
- EXTERNAL 文節を使用しても、それに関連したデータ名がグローバル名であることを意味しません。
- EXTERNAL 文節を含むか、またはこの文節を含む記入項目に従属するデータ記述記入項目に、VALUE 文節を使ってはなりません。そのようなデータ記述記入項目に関連した条件名記入項目には、VALUE 文節を指定できます。
- TYPEDEF 文節は EXTERNAL 文節と同じデータ記述記入項目に指定することはできません。

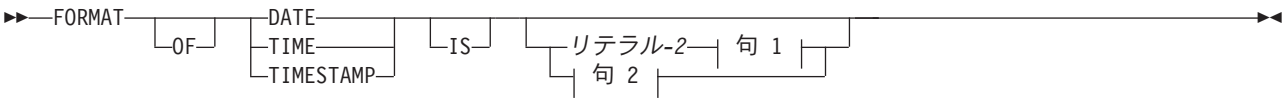
詳しくは、 2-26 ページの『データ参照と名前の有効範囲』を参照してください。

FORMAT 文節

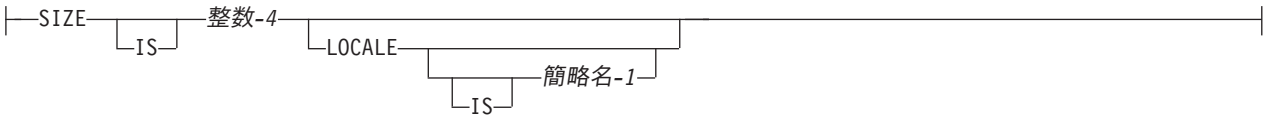
IBM Extension

FORMAT 文節では、日付、時刻、またはタイム・スタンプの基本項目の一般特性および編集要件を指定します。

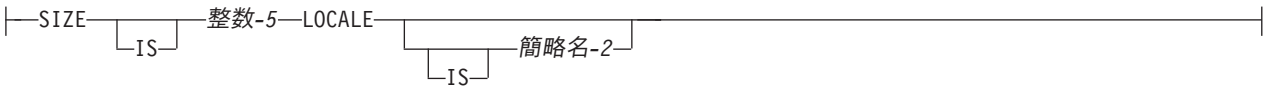
FORMAT 文節 - 形式



句 1:



句 2:



FORMAT 文節は、RENAMES 文節のサブジェクトを除き、日付、時刻、またはタイム・スタンプの各基本項目のそれぞれに対して指定する必要があります。

タイム・スタンプ項目に対して SIZE 句が指定されていない場合、サイズはデフォルトの 26 になります。指定されている場合、その値は 19、または 21 から 32 の間でなければなりません。

リテラル-2 および LOCALE 句は、タイム・スタンプ項目には指定できません。タイム・スタンプは固定形式であり、それはタイム・スタンプ項目のサイズに依存しています。

- SIZE 句が指定されていない場合、形式は、リテラル-2 の値の「@Y-%m-%d-%H.%M.%S.@Sm」と同等になります。
- 値 19 の SIZE 句が指定されている場合、形式は、リテラル-2 の値の「@Y-%m-%d-%H.%M.%S」と同等になります。
- 21 から 32 の値の SIZE 句が指定されている場合、形式は、リテラル-2 の値の「@Y-%m-%d-%H.%M.%S.」の後ろにタイム・スタンプの秒の小数部を付けたものと同等になります。例えば、サイズ 25 のタイム・スタンプは、値「2014-01-23-01.02.03.12345」を取る可能性があります。

リテラル-2 または LOCALE 句を日付項目または時刻項目に指定しないと、その項目の形式は SPECIAL-NAMES FORMAT 文節から判別されます。

日時クラスのデータ項目を参照変更することはできません。

FORMAT 文節を指定する場合は、以下の文節を指定することはできません。

- PICTURE 文節
- SIGN 文節
- BLANK WHEN ZERO 文節
- JUSTIFIED 文節
- LIKE 文節。ただし、LIKE 文節は、データ項目の FORMAT を定義する際に使用することができます。LIKE 文節で日付、時刻、またはタイム・スタンプの項目のサイズを変更することはできません。 LIKE 文節による日付、時刻、またはタイム・スタンプの項目への参照時には、継承された適切な FORMAT 文節情報からコメントが生成されます。
- TYPE 文節。

以下の一般規則が適用されます。

- 条件名を日時項目に関連付けることができる。この条件名の VALUE 文節は THRU 句を使用して指定できます。

- SYNCHRONIZED 文節は文書として扱われる。
- 文節 OCCURS、REDEFINES、および RENAMES は、日付、時刻、またはタイム・スタンプの項目と関連付けることができる。
- LIKE 文節を指定する場合は FORMAT 文節を指定することはできない。
- 関連付けられている VALUE 文節は、どれでも非数字リテラルを指定しなければならない。このリテラルはまさに指定したとおりに扱われます。フォーマット設定は一切行われません。

リテラル-2
日付項目または時刻項目の形式を指定します。リテラル-2 は最小でも 2 文字の非数字リテラルでなければなりません。リテラル-2 の内容は、分離文字および変換指定子から構成されます。有効な変換指定子のリストが 5-17 ページの表 5-4 に記載されていますので、そちらを参照してください。リテラル-2 の内容に関する規則の詳細は 5-16 ページの『FORMAT 文節』に記載されている、SPECIAL-NAMES 段落で使用する FORMAT 文節についての説明を参照してください。

SIZE 句

SIZE 句について、より詳細な説明が必要な場合は、5-19 ページの『SIZE 句』を参照してください。このセクションでは、この SIZE 句に指定するパラメーターについて説明します。

整数-3、整数-4
整数-3 および整数-4 から、日付項目または時刻項目の桁数でのサイズが判別されます。日付項目または時刻項目のサイズがコンパイル時に判別できない場合は、整数-3 または整数-4 を指定する必要があります。日付項目または時刻項目では、整数-3 および整数-4 の値は両方とも 4 以上でなければなりません。

簡略名-1、簡略名-2
簡略名-1 または簡略名-2 の詳細は 『LOCALE 句』および 5-19 ページの『LOCALE 句』に記載されている説明を参照してください。

日時クラスの項目の使用法

日時クラスの項目に USAGE 文節が指定されていない場合は、USAGE DISPLAY が想定されます。日時項目の場合は、USAGE DISPLAY か USAGE PACKED-DECIMAL (COMP-3) かを明示的に指定できます。日時クラスの項目に対して USAGE PACKED-DECIMAL を指定できるのは、リテラル-2 の内容が変換指定子だけの場合に限られます。これらの変換指定子は結果的に数字に置き換えられます。

FORMAT 文節と PICTURE 文節の類似点

FORMAT 文節は、暗黙の PICTURE 文節を定義します。日付項目または時刻項目を容易に記述できる PICTURE 文字ストリングは 1 つもないにもかかわらず、ある種の形式においては定義に近いものが存在します。例えば、FORMAT '%y,%m,%d' が指定されている日付項目は PICTURE 99/99/99 に類似します。ここで、'/' PICTURE 記号は ',' に置き換えられます。

LOCALE 句: LOCALE 句は、日付、時刻、タイム・スタンプ項目を各文化圏に適切な形式で指定します。

リテラル-2 を指定せずに LOCALE 句を指定すると、日付項目および時刻項目に使用される形式および分離文字は完全にロケールに基づくものとなります。

リテラル-2 を指定した上で LOCALE 句を指定すると、項目の形式はリテラル-2 から判別されますが、変換指定はロケールに基づく項目に置き換えられます。

簡略名-1、簡略名-2
簡略名を指定すると、日付項目または時刻項目のために使用されるロケールは SPECIAL-NAMES 段落

の LOCALE 文節の簡略名に関連付けられているロケールとなります。簡略名を指定しない場合は現行ロケールが使用されます。現行ロケールの判別方法について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

LOCALE OF 特殊レジスター: LOCALE OF 特殊レジスターは、指定したデータ項目に関連付けられているロケール簡略名と等価の値を戻します。そのデータ項目に関連付けられているロケールがない場合は、キーワード COBOL が戻されます。 LOCALE OF 特殊レジスターを変更することはできません。また、これを指定できるのは、ロケール簡略名が使用可能な PROCEDURE DIVISION においてだけです。

日時データ項目は LOCALE OF 特殊レジスターを使用する式の中で使用することができます。

DDS データ・タイプおよび FORMAT リテラルと等価の値: DDS では、日付データ・タイプの形式は DATFMT パラメーターで指定します。 *MDY および *ISO も有効な DATFMT パラメーターです。 DATSEP キーワードは、月、日、および年の値の間の分離文字として使用される 1 文字の値を指定するために DATFMT キーワードとともに使用します。 表 6-2 に、DATFMT パラメーター、それぞれの形式で使用可能な DDS 日付分離文字、およびこれらのパラメーターと等価の COBOL 形式リテラルの完全なリストを示します。

DDS では、時刻データ・タイプの形式は TIMFMT パラメーターで指定します。 *HMS および *ISO も有効な TIMFMT パラメーターです。 TIMSEP キーワードは、時、分、および秒の値の間の分離文字として使用される 1 文字の値を指定するために TIMFMT キーワードとともに使用します。 表 6-3 に、TIMFMT パラメーター、それぞれの形式で使用可能な DDS 時刻分離文字、およびこれらのパラメーターと等価の COBOL 形式リテラルの完全なリストを示します。

表 6-2. DDS 日付データ・タイプおよびそれらと等価の ILE COBOL の形式

IBM i 形式	COBOL が生成する形式	内容	フォーマット	有効な分離文字	長さ
*MDY	%m/%d/%y	月 / 日 / 年	mm/dd/yy	/-.,スペース	8
*DMY	%d/%m/%y	日 / 月 / 年	dd/mm/yy	/-.,スペース	8
*YMD	%y/%m/%d	年 / 月 / 日	yy/mm/dd	/-.,スペース	8
*JUL	%y/%j	ユリウス	yy/ddd	/-.,スペース	6
*ISO	@Y-%m-%d	ISO (国際標準化機構)	yyyy-mm-dd	-	10
*USA	%m/%d/@Y	IBM USA 標準規格	mm/dd/yyyy	/	10
*EUR	%d.%m.@Y	IBM 欧州標準規格	dd.mm.yyyy	.	10
*JIS	@Y-%m-%d	JIS (日本工業規格) 西暦	yyyy-mm-dd	-	10

表 6-3. DDS 時刻データ・タイプおよびそれらと等価の ILE COBOL の形式

IBM i 形式	COBOL が生成する形式	内容	フォーマット	有効な分離文字	長さ
*HMS	%H:%M:%S	時 : 分 : 秒	hh:mm:ss	:.,スペース	8
*ISO	%H.%M.%S	ISO (国際標準化機構)	hh.mm.ss	.	8
*USA	%I:%M @p	IBM USA 標準規格。AM および PM は、大文字小文字を任意に混合させることができます。	hh:mm AM または hh:mm PM	:	8

表 6-3. DDS 時刻データ・タイプおよびそれらと等価の ILE COBOL の形式 (続き)

IBM i 形式	COBOL が生成する形式	内容	フォーマット	有効な分離文字	長さ
*EUR	%H.%M.%S	IBM 欧州標準規格	hh.mm.ss	.	8
*JIS	%H:%M:%S	JIS (日本工業規格) 西暦	hh:mm:ss	:	8

FORMAT OF 特殊レジスター: PROCEDURE DIVISION の FORMAT OF 句は、FORMAT OF 特殊レジスターと呼ばれる暗黙的な特殊レジスターを作成します。このレジスターの内容は、ID によって参照されるデータ項目の FORMAT リテラルと等価です。FORMAT OF 特殊レジスターは、日時クラスのデータ項目に対してのみ指定することができます。この特殊レジスターの長さは、当該のデータ項目用の FORMAT 句内で指定されるリテラルまたはロケールによって決まります。

FORMAT OF 特殊レジスターには次のような暗黙の定義が入っています。

```
USAGE DISPLAY, PICTURE X(n)
where n equals the number of bytes of the implicit or explicit
FORMAT literal.
```

例として、以下の日付データ項目 date2 用のデータ記述記入項目について考えてみます。

```
05 date2 FORMAT DATE IS '%d,%m,%y'.
```

以下の MOVE ステートメントは、組み込み関数 CONVERT-DATE-TIME を使用して日付データ項目 date3 を日付データ項目 date2 の形式に変換します。 FORMAT OF 句は、その内容が %d,%m,%y となる暗黙特殊レジスターを作成します。

```
MOVE FUNCTION CONVERT-DATE-TIME(date3, DATE, FORMAT OF date2)
          TO alpha-num-date.
```

この例では、特殊レジスターの長さは 8 文字です。

次の規則が適用されます。

- FORMAT OF 特殊レジスターは、変更不能であり、FORMAT 非数字リテラルが使用可能な PROCEDURE DIVISION においてのみ指定可能。
- FORMAT OF 句を使って参照される ID ごとに別個の FORMAT OF 特殊レジスターが 1 つずつ存在する。

End of IBM Extension

GLOBAL 文節

GLOBAL 文節は、データ名また定数名を宣言するプログラムと、そのデータ名または定数名を宣言するプログラム内に含まれるすべてのプログラムが、そのデータ名または定数名を使えるよう指定します。しかし、含まれる側のプログラムでその名前を宣言しないことが前提条件です。グローバル名に従属するデータ名、グローバル名と関連した条件名または指標名は、すべてグローバル名です。

GLOBAL 文節 - 形式

GLOBAL 文節が、データ名を宣言したデータ記述記入項目またはデータ記述記入項目が従属である別の記入項目のいずれかで指定された場合、そのデータ名はグローバルです。

IBM Extension

GLOBAL 文節は、リンケージ・セクションおよびローカル・ストレージ・セクションに指定できますが、レベル番号が 01 のデータ記述記入項目にしか指定できません。

End of IBM Extension

同じデータ部の場合、同じデータ名を指定された 2 つのデータ項目のデータ記述記入項目に GLOBAL 文節を入れてはなりません。

グローバル名を記述したプログラムに直接または間接に含まれるプログラム内のステートメントは、もう一度その名前を記述しなくてもその名前を参照できます。

IBM Extension

TYPEDEF 文節を GLOBAL 文節とともに指定すると、GLOBAL 文節の有効範囲が当該のタイプ名、およびそのタイプ名に従属するすべてのデータ項目に対して適用されます。TYPE 文節内のグローバル・タイプ名を使用して定義したデータ項目が GLOBAL 属性を獲得することはありません。

End of IBM Extension

データの共用

実行単位内の 2 つのプログラムは、次のような場合に共通データを参照できます。

1. プログラムで外部データ・レコードを記述してさえいれば、その外部データ・レコードのデータ内容をそのプログラムから参照できます。
2. プログラムが別のプログラムに含まれているとき、どちらのプログラムからでも、含む側のプログラムに、またはその含む側のプログラムを直接または間接に含む任意のプログラムにグローバル属性を保有するデータを参照できます。
3. 参照のときに渡されるパラメーターは、呼び出し側プログラムと呼び出し先プログラムで共用できます。

JUSTIFIED 文節

JUSTIFIED 文節は、英字または英数字カテゴリーの受け入れ項目の標準位置合わせ規則を無効にします。

JUSTIFIED 文節 - 形式

注:

1 構文検査だけ行われます。

JUSTIFIED 文節は、基本レベルでのみ指定できます。 RIGHT はオプショナルの語であり、構文検査だけが行われ、プログラムの実行には影響を与えません。

JUSTIFIED 文節

JUSTIFIED 文節は、数字項目または数字編集項目に対して指定することができません。

IBM Extension

DBCS 項目、DBCS 編集項目、および国別項目に対して指定できます。

End of IBM Extension

JUSTIFIED 文節は、以下のものには許されません。

- レベル 66 (RENAMES) 記入項目
- レベル 88 (条件名) 記入項目
- USAGE IS INDEX 文節で記述された項目

IBM Extension

- USAGE IS POINTER 文節で記述された項目
- USAGE IS PROCEDURE-POINTER 文節で記述された項目
- 外部または内部の浮動小数点項目
- TYPE 文節が指定されている項目

End of IBM Extension

IBM Extension

JUSTIFIED 文節は、英数字編集項目に指定できます。

End of IBM Extension

JUSTIFIED 文節を省略すると、標準の位置合わせ規則が適用されます (6-10 ページの『位置合わせの規則』を参照)。

受け入れ項目に対して JUSTIFIED 文節を指定した場合、データはその受け入れ項目の右端の文字位置に位置合わせされます。また、以下のとおりです。

- 送り出し項目が受け入れ項目より大きいならば、左端の文字が切り捨てられます。
- 送り出し項目が受け入れ項目より小さいならば、左の使用されない文字位置がスペースで埋められます。

JUSTIFIED 文節は、VALUE 文節によって定められた初期値には影響を与えません。

LIKE 文節

IBM Extension

LIKE 文節を使用すると、すでに定義済みのデータ項目から PICTURE、USAGE、SIGN、および FORMAT をコピーすることにより、データ項目のこれらの特性を定義することができます。また、LIKE 文節を使用して、データ項目の長さを元の項目の長さとは異なるものに定義できます。

LIKE 文節 - 形式

注:

1 IBM 拡張

データ名-1
基本項目、グループ項目、指標名、またはタイプ名を参照することができます。データ名-1 によって参照される項目は、LIKE 文節のオブジェクト として認識されます。

整数
新しい項目と既存項目との長さの差を指定します。

これには符号を付けることができます。

整数の前にブランクまたは + がある場合、この新しい項目は長くなります。- が整数の前にある場合、新しい項目は短くなります。

整数オプションを使用して、以下に示す変更はできません。

- 編集項目の長さを変更すること。
- 指標、ポインター、またはプロシージャー・ポインター項目の長さを変更すること。
- データ項目内の小数点以下の桁数を変更すること。
- 内部浮動小数点データ項目または外部浮動小数点データ項目の長さを変更すること。
- 時刻、またはタイム・スタンプの項目の長さを変更すること。

項目の属性に BLANK WHEN ZERO が入っていると、その項目は編集項目として扱われることに注意してください。

LIKE 文節は、新しいデータ項目に既存データ項目の特性を継承させます。これらの特性は、既存の項目の PICTURE、USAGE、SIGN、BLANK WHEN ZERO、および FORMAT の属性です。

コンパイラーはコメントを生成して、新しい項目の特性を識別します。このコメントは、LIKE 文節を含むステートメントの後に現れます。

デフォルトの USAGE IS DISPLAY および SIGN IS TRAILING の特性はコメントとして印刷されないことに注意してください。

継承可能な FORMAT 特性には、以下のものがあります。

- 項目のカテゴリー (日付、時刻、またはタイム・スタンプ)
- FORMAT リテラル
- SIZE 句および LOCALE 句

FORMAT 文節の詳細については 6-38 ページの『FORMAT 文節』を参照してください。

継承した USAGE 特性に基づいて生成されるコメント

元の項目に別の USAGE 文節を指定できますが、その場合はコメントの数が制約されます。これについては 6-46 ページの表 6-4で説明しています。

表 6-4. 継承した USAGE 特性に基づいて生成されるコメント

継承される USAGE 文節	生成されるコメント
PACKED-DECIMAL COMPUTATIONAL COMPUTATIONAL-3	* USAGE IS PACKED-DECIMAL
COMP-1 COMUTATIONAL-1	* USAGE IS COMPUTATIONAL-1
COMP-2 COMUTATIONAL-2	* USAGE IS COMPUTATIONAL-2
BINARY COMP-4COMPUTATIONAL-4	* USAGE IS BINARY
COMP-5 COMPUTATIONAL-5	* USAGE COMP-5
INDEX	*USAGE IS INDEX
NATIONAL	*USAGE IS NATIONAL
DISPLAY	これはデフォルトの USAGE なのでコメントは生成されない
DISPLAY-1	* USAGE IS DISPLAY-1
POINTER	* USAGE IS POINTER
PROCEDURE-POINTER	* USAGE IS PROCEDURE-POINTER

LIKE 文節を使用して定義するデータ項目の特性は、ユーザーのコンパイル済みプログラムのリスト内に示されます。

規則と制約事項

LIKE 文節は、レベル番号 01 ～ 49、およびレベル番号 77 で使用できます。

データ名または FILLER 項目を指定する場合、その後の任意の位置に LIKE 文節を記述できます。その他の場合には、レベル番号の後の任意の位置にこれを記述できます。

LIKE 文節の前後には、次の文節の 1 つまたは複数を指定できます。

- JUSTIFIED
- SYNCHRONIZED
- BLANK WHEN ZERO
- VALUE
- OCCURS

BLANK WHEN ZERO は、これが以前に受け継がれなかったという場合にのみ指定できることに注意してください。

次の文節とともに LIKE 文節を使用することはできません。

- REDEFINES
- SIGN
- USAGE
- PICTURE

- FORMAT
- TYPE
- TYPEDEF

LIKE 文節に継承済みの文節を指定すると、重複エラーになります。

数字項目の場合、新しい項目の数字の総数はゼロにはできません。ただし、その項目に小数部分が入っている場合には、整数部分の数字がゼロになってもかまいません。

PICTURE 文節が英字、数字、または英数字を混合して指定しているときに LIKE 文節の長さを修正すると、新しい PICTURE 文節は英数字を指定します。

LIKE 文節を使用して、この文節内で名前を指定する項目に従属している項目を定義することはできません。

LIKE 文節のオブジェクトのデータ記述の中に TYPE 文節を含めることはできません。LIKE 文節のオブジェクトがグループ項目である場合は、このグループに従属する項目を TYPE 文節を使用して定義することはできません。ある LIKE 文節のオブジェクトが (レベル 01 の) グループ項目に従属しており、かつそのレベル 01 のグループ項目に従属する項目のいずれかに TYPE 文節が含まれている場合は、その TYPE 文節内で参照されるタイプ名は DATA DIVISION においてその LIKE 文節が使用されるポイントで完全に定義される必要があります。

コーディング例

データ項目 HEIGHT と同じ属性を持つデータ項目 DEPTH を作成するには、単純に次のように書きます。

```
DEPTH LIKE HEIGHT
```

データ項目 STATE と同じ属性をもち、ただし 1 バイトだけ長いデータ項目 PROVINCE を作成したい場合は、次のように書きます。

```
PROVINCE LIKE STATE (+1)
```

End of IBM Extension

OCCURS 文節

テーブルの処理に使用するデータ部の文節は OCCURS 文節と USAGE IS INDEX 文節です (USAGE IS INDEX の説明は 6-89 ページの『USAGE 文節』を参照してください)。 OCCURS 文節の形式 1 は、固定長テーブルを処理します。 OCCURS 文節の形式 2 は、可変長テーブルを処理します。

テーブル処理の概念

テーブルとは、一連の項目の集合の中で、各項目が他の項目と同じデータ記述を持つ論理的に連続する一連の項目です。 COBOL には、1 つのテーブルの全体または一部をエンティティーとして参照するようなデータ参照の使用法が備わっています。

COBOL では、テーブルはデータ記述の中で OCCURS 文節とともに定義されます。 OCCURS 文節は、指定された項目が示された回数だけ繰り返されることを指定します。そのように指定された項目は、テーブル・エレメントと見なされ、その名前および記述は、その項目が出現するたびに適用されます。オカレンスには固有のデータ名が与えられないので、特定のオカレンスへの参照は、エレメント内の希望する項目のオカレンス番号とともに、テーブル・エレメントのデータ名を指定することによってのみ可能です。

オカレンス番号は添え字とも呼ばれ、各テーブル・エレメントにオカレンス番号を与える方法は、添え字付けと呼ばれます。添え字付けは、後の節で説明します。

OCCURS 文節を含んだデータ項目のデータ名は、OCCURS 文節の**サブジェクト**と呼ばれます。 OCCURS 文節のサブジェクト (またはその下にある任意のデータ項目) が参照される場合、次のような場合を除いて、そのサブジェクトに添え字が付いているか、指標が付けられていなければなりません。

- OCCURS 文節のサブジェクトが、SEARCH ステートメントのサブジェクトとして使われるとき
- そのサブジェクト (またはそれに従属するデータ項目) が、ASCENDING/DESCENDING KEY 文節のオブジェクトであるとき
- 従属データ項目が、REDEFINES 文節のオブジェクトであるとき

OCCURS 文節のサブジェクトに添え字または指標を付けると、それはテーブル内の 1 つのオカレンスを表します。付けない場合、そのサブジェクトはテーブル全体を表します。

IBM Extension

用途が POINTER または PROCEDURE-POINTER である項目は、OCCURS 文節を含むことも、または OCCURS 文節を使って宣言された項目に従属することもあります。

ポインターまたはプロシージャー・ポインター・テーブル項目を含むテーブルは 6-100 ページの『ポインターの位置合わせ』で定義されているとおりのポインター位置合わせに従います。必要な場合には、コンパイラーが FILLER 項目を加えてテーブルの第 1 エレメントのポインターを位置合わせし、また同エレメントの最後には FILLER を加えて次のポインターを位置合わせします。これは、テーブル中のすべてのポインターが位置合わせされるまで続きます。

ブール、外部浮動小数点または内部浮動小数点、日付、時刻、またはタイム・スタンプの項目は、OCCURS 文節を含んでいることもあり、また、OCCURS 文節を使って宣言された項目に従属することもあります。

End of IBM Extension

制限

テーブルの処理時には、次の制約事項を理解しておかなければなりません。

- OCCURS 文節中の項目のオカレンス数は、最高 16 711 568 までとすることができます。
- 従属エレメントを含めたテーブル・エレメントのサイズ限度は、16 711 568 バイトです。
- OCCURS 文節は、次のようなデータ記述記入項目には入れられません。
  - レベル番号 01、66、77、または 88 を持つもの。
  - 再定義されたデータ項目を記述するもの。ただし、再定義された項目は、OCCURS 文節を含む項目に従属できます。

テーブルの定義

ILE COBOL コンパイラーでは、1 ～ 7 次元のテーブルを使用できます。

1 次元のテーブルを定義するには、1 つの OCCURS 文節を含むグループ項目をセットアップします。OCCURS 文節は、レベル番号が 01、66、77、または 88 であるデータ記述記入項目には入らないということに注意してください。

例えば、次のとおりです。

```
01 TABLE-ONE.
05 ELEMENT-ONE OCCURS 3 TIMES.
   10 ELEMENT-A PIC X(4).
   10 ELEMENT-B PIC 9(4).
```

TABLE-ONE は、テーブルを含むグループ項目です。 ELEMENT-ONE は、3 回発生する 1 次元テーブルのエレメントです。 ELEMENT-A および ELEMENT-B は、ELEMENT-ONE に従属する基本項目です。

3 次元テーブルを定義するには、別の 1 次元テーブルの各オカレンス内にそれ自体が含まれている 1 次元テーブルの各オカレンス内に 1 次元テーブルを定義します。例えば、次のとおりです。

```
01 TABLE-THREE.
05 ELEMENT-ONE OCCURS 3 TIMES.
   10 ELEMENT-TWO OCCURS 3 TIMES.
      15 ELEMENT-THREE OCCURS 2 TIMES
         PICTURE X(8).
```

TABLE-THREE は、テーブルを含むグループ項目です。 ELEMENT-ONE は、3 回発生する 1 次元テーブルのエレメントです。 ELEMENT-TWO は、ELEMENT-ONE の各オカレンス内に 3 回発生する 2 次元テーブルのエレメントです。 ELEMENT-THREE は、ELEMENT-TWO の各オカレンス内に 2 回発生する 3 次元テーブルのエレメントです。 図 6-3 は、TABLE-THREE のストレージ・レイアウトを示しています。

ELEMENT—ONE 3 回発生	ELEMENT—TWO 3 回発生	ELEMENT—THREE 2 回発生	バイト変位
ELEMENT—ONE (1)	ELEMENT—TWO (1, 1)	ELEMENT—THREE (1, 1, 1)	0
		ELEMENT—THREE (1, 1, 2)	8
	ELEMENT—TWO (1, 2)	ELEMENT—THREE (1, 2, 1)	16
		ELEMENT—THREE (1, 2, 2)	24
	ELEMENT—TWO (1, 3)	ELEMENT—THREE (1, 3, 1)	32
		ELEMENT—THREE (1, 3, 2)	40
ELEMENT—ONE (2)	ELEMENT—TWO (2, 1)	ELEMENT—THREE (2, 1, 1)	48
		ELEMENT—THREE (2, 1, 2)	56
	ELEMENT—TWO (2, 2)	ELEMENT—THREE (2, 2, 1)	64
		ELEMENT—THREE (2, 2, 2)	72
	ELEMENT—TWO (2, 3)	ELEMENT—THREE (2, 3, 1)	80
		ELEMENT—THREE (2, 3, 2)	88
ELEMENT—ONE (3)	ELEMENT—TWO (3, 1)	ELEMENT—THREE (3, 1, 1)	96
		ELEMENT—THREE (3, 1, 2)	104
	ELEMENT—TWO (3, 2)	ELEMENT—THREE (3, 2, 1)	112
		ELEMENT—THREE (3, 2, 2)	120
	ELEMENT—TWO (3, 3)	ELEMENT—THREE (3, 3, 1)	128
		ELEMENT—THREE (3, 3, 2)	136
			144

図 6-3. TABLE-THREE のストレージ・レイアウト

テーブル・エレメントの参照

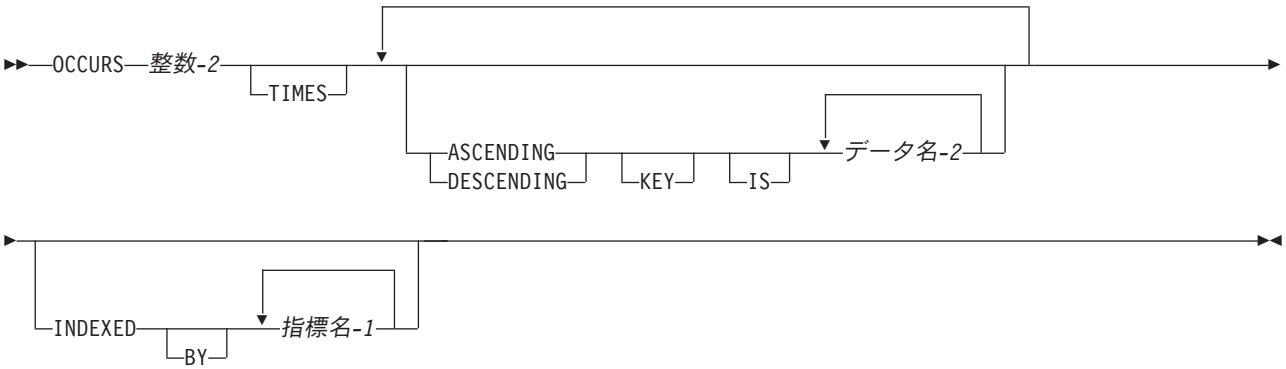
ユーザーがテーブル・エレメント、あるいはテーブル・エレメントに関連した項目を参照するときはいつでも、参照がどのオカレンスを指しているかを示さねばなりません。

1 次元テーブルでは、希望するエレメントのオカレンス番号によって、詳細な情報が提供されます。 2 次元以上のテーブルでは、各次元のオカレンス番号を指定する必要があります。例えば、前に定義されている 3 次元テーブルでは、ELEMENT-THREE への参照には、ELEMENT-ONE、ELEMENT-TWO、および ELEMENT-THREE のオカレンス番号を与えねばなりません。

固定長テーブル

固定長テーブルは、OCCURS 文節を使って指定されます。 7 つの添え字および指標が使用できるので、形式 1 の OCCURS 文節の 6 つのネストされたレベルと 1 番外側のレベルを 1 つ使用できます。形式 1 の OCCURS 文節は、OCCURS DEPENDING ON 文節の従属として定義してもかまいません。このように、7 次元までのテーブルを指定できます。

OCCURS 文節 - 形式 1 - 固定長テーブル



整数-2

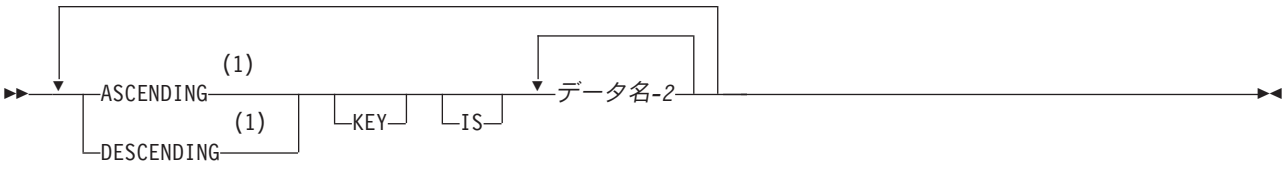
オカレンスの正確な数を指定します。これはゼロより大きくなければなりません。

ILE COBOL 言語では、整数-2 は 1 ～ 16 711 568 バイトでなければなりません。

ASCENDING/DESCENDING KEY 句

データは、データ名-2 に含まれる値にしたがって、昇順または降順に (指定されたキーワードに従って) 並べられます。データ名は、重要度の降順に記入されます。

ASCENDING/DESCENDING KEY 句 - 形式



注:

1 ブール・データ・タイプとともに使用することはできません。

順序は、オペランドの比較の規則に従って決められます (7-14 ページの『比較条件』を参照)。
ASCENDING および DESCENDING KEY データ項目は、OCCURS 文節およびテーブル・エレメントのバイナリー・サーチを行う SEARCH ALL ステートメントで使用されます。

データ名-2

サブジェクト記入項目の名前、またはサブジェクト記入項目に従属する記入項目の名前でなければなりません。

データ名-2 がサブジェクト記入項目を指定する場合は、その記入項目全体が ASCENDING/DESCENDING KEY となり、またこのテーブル・エレメントに対して指定することのできる唯一のキーとなります。

データ名-2 がサブジェクト記入項目を指定しない場合は、データ名-2 は、次の規則が適用されます。

- テーブル記入項目そのもののサブジェクトに従属しなければなりません。
- OCCURS 文節を含む他の記入項目に従属または後続してはなりません。
- OCCURS 文節を含んではなりません。

ASCENDING/DESCENDING KEY 句の規則

ASCENDING/DESCENDING KEY 句を指定するときは、次の規則が適用されます。

- キーは、重要度の降順に記入しなければなりません。
- 使用している照合順序に従って、ASCENDING または DESCENDING 順序でテーブル内にデータを配列しなければなりません。
- キーは、DISPLAY、BINARY、PACKED-DECIMAL、または COMPUTATIONAL に使用できます。

IBM Extension

- KEY 句は、DBCS 項目の OCCURS 文節に指定できます。
- # キーは、COMPUTATIONAL-1、COMPUTATIONAL-2、COMPUTATIONAL-3、 COMPUTATIONAL-4、
  # または COMPUTATIONAL-5 に使用できます。
- キーは、DISPLAY-1 を使用できます。
- キーは、日時クラスの項目であっても構いません。

End of IBM Extension

ASCENDING/DESCENDING KEY 句のコーディング例

次の例は、ASCENDING KEY データ項目の指定を示しています。

```
WORKING-STORAGE SECTION.
01  TABLE-RECORD.
  05  EMPLOYEE-TABLE OCCURS 100 TIMES
      ASCENDING KEY IS WAGE-RATE EMPLOYEE-NO
      INDEXED BY A,  B.
    10  EMPLOYEE-NAME                       PIC X(20).
    10  EMPLOYEE-NO                         PIC 9(6).
    10  WAGE-RATE                           PIC 9999V99.
    10  WEEK-RECORD OCCURS 52 TIMES
        ASCENDING KEY IS WEEK-NO INDEXED BY C.
      15  WEEK-NO                           PIC 99.
      15  AUTHORIZED-ABSENCES               PIC  9.
      15  UNAUTHORIZED-ABSENCES             PIC  9.
      15  LATE-ARRIVALS                     PIC  9.
```

EMPLOYEE-TABLE に関するキーは、その記入項目に従属し、一方 WEEK-RECORD に関するキーは、その従属記入項目に従属しています。

上記の例では、EMPLOYEE-TABLE の中のレコードは、WAGE-RATE の昇順に並べ、また WAGE-RATE の中では、EMPLOYEE-NO の昇順に並べなければなりません。 WEEK-RECORD の中のレコードは、WEEK-NO の昇順に並べなければなりません。そのように並んでいない場合に任意の SEARCH ALL ステートメントを実行すると、結果は予測できません。

INDEXED BY 句

INDEXED-BY 句は、そのテーブルで使用することができる指標を指定します。そのテーブル・エレメントを参照するために指標付けを使用する場合は、INDEXED BY 句を指定する必要があります。 2-33 ページの『指標名を使用した添え字付け (指標付け)』を参照してください。

指標の値は、指標データ項目の値を保管することによってプログラムへアクセスできるようになります。指標データ項目は、USAGE IS INDEX 文節を含んでいるデータ記述記入項目によってプログラムに記述されます。指標値は、SET ステートメントを実行することによって指標データ項目に移されます。

指標は、通常、テーブルを含むプログラムに関連した静的メモリーで割り振られます。このため、指標は、プログラムに再入したときの、最後に使われた状態になります。ただし、以下の場合、指標は、呼び出しごとに割り振られます。したがって、プログラムに入るたびに、以下のセクションのテーブルの指標に対して値を設定する必要があります。

- ローカル・ストレージ・セクション
- RECURSIVE 属性でコンパイルされるプログラムのリンケージ・セクション

INDEXED BY 句 - 形式

指標名-1

ユーザー定義語の形成規則に従っていなければなりません。少なくとも 1 文字は英字でなければなりません。

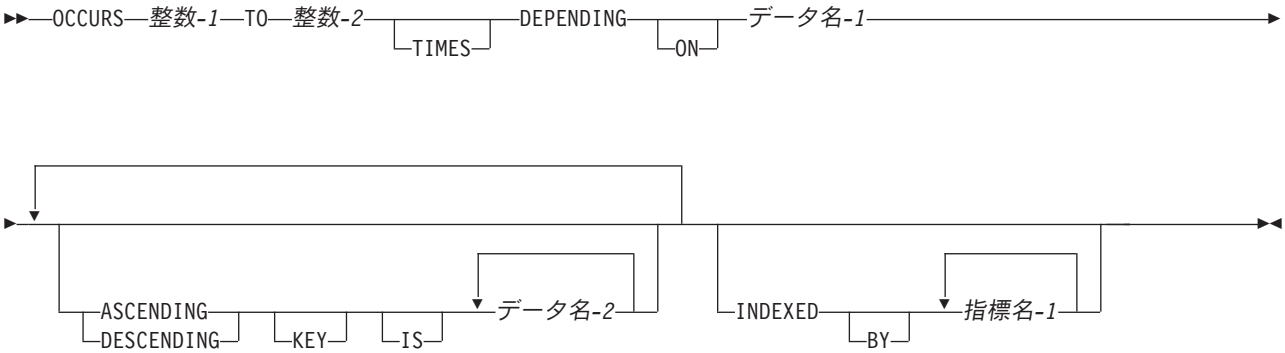
各指標名は、プログラムで使用するためにコンパイラーによって作成される指標を指定します。それらの指標名はデータ名ではなく、COBOL プログラムの他のどこでも識別されません。それらは、このオブジェクト・プログラムだけを使用するための専用の特殊レジスターと見なすことができます。そのため、それらの指標名はデータでもデータ階層の一部でもありません。そのため、おのおのが固有名でなければなりません。

GLOBAL 属性を持つデータ項目に、指標を使ってアクセスされるテーブルが含まれていると、そのテーブルに定義される指標も GLOBAL 属性をもちます。

可変長テーブル

可変長テーブルは、OCCURS 文節の形式 2 を使用して指定されます。

OCCURS 文節 - 形式 2 - 可変長テーブル



サブジェクト項目の**長さ**は固定されています。可変であるのは、そのサブジェクト項目の**反復回数**だけです。

整数-1
最小のオカレンス数。

整数-1 の値は、ゼロかそれ以上でなければならず、また、整数-2 の値より小さくなければなりません。

整数-2
最大オカレンス数。

データ名-1
OCCURS DEPENDING ON 文節の**オブジェクト**、つまり、サブジェクト項目の現行オカレンス数を表す現行値をもった (整数) データ項目を指定します。オカレンス番号がオブジェクトの値を超える項目の内容は、予測できません。

OCCURS DEPENDING ON 文節のオブジェクトは、どのテーブルの範囲内のどの記憶位置 (すなわち、テーブルの最初の文字位置からテーブルの最後の文字位置までのすべての記憶位置) も占めてはなりません。

OCCURS DEPENDING ON 文節のオブジェクトは、位置が可変でない場合があります。このオブジェクトは、OCCURS DEPENDING ON 文節を含んでいる項目の後に置くことはできません。

グループ項目 (あるいは、従属した OCCURS DEPENDING ON 項目を含んでいるデータ項目、または OCCURS DEPENDING ON 項目の後にあるが、この項目に従属していないデータ項目) を参照するときには、OCCURS DEPENDING ON 文節のオブジェクトの値は、整数-1 から整数-2 までの範囲内になければなりません。グループの位置が可変でないとすれば、参照されるグループが CALL BY REFERENCE ステートメントで使われているときには、この規則は適用されません。

EXTERNAL 文節の入ったレコード記述記入項目に含まれるデータ記述記入項目に OCCURS 文節を指定するときは、データ名-1 は、同じデータ部に記述された EXTERNAL 属性を持つデータ項目を参照しなければなりません。

データ記述記入項目が GLOBAL 文節の入ったものに従属するときは、データ名-1 はグローバル名でなければならず、同じデータ部に記述されたデータ項目を参照しなければなりません。

従属 OCCURS DEPENDING ON 項目を含むグループ項目が参照される場合、テーブル域のどの部分が処理に使用されるかは、次のように判別されます。

- オブジェクトがグループ外にある場合は、操作の開始時にオブジェクトによって指定されるテーブル域の部分だけが使用されます。

- オブジェクトが同一グループに含まれていて、かつグループ・データ項目が送り出し項目として参照される場合には、操作の開始時にオブジェクトの値によって指定されるテーブル域の部分だけが操作時に使用されます。
- オブジェクトが同一グループに含まれていて、かつグループ・データ項目が受け入れ項目として参照される場合には、操作時にはグループ項目の最大長が使用されます。

可変長テーブルの入ったグループ項目に対して参照変更を加えると、その参照変更で参照されるデータ項目から固有データ項目が作成されます。その参照されるデータ項目の長さは、まず前の規則を適用して判別されます。その後、参照変更の規則が適用され、その固有データ項目の長さが判別されます。

CALL ステートメントの USING 句で引数として可変長テーブルの入ったグループ項目が使用されると、呼び出し先プログラムの立場から見たそのパラメーターのストレージ・サイズは、その引数の渡し方によって異なります。引数が BY REFERENCE で渡されるとき、最大サイズは、呼び出し側プログラム内の引数のデータ記述によって記述されます。引数が BY CONTENT で渡されるとき、グループ項目は送り出し項目とみなされます。

グループ項目の後に非従属項目が続く場合、実際の長さ (最大長ではなく) が使用されます。項目のサブジェクト (あるいは、項目のサブジェクトに従属するデータ項目、または項目のサブジェクトの上位のデータ項目) を参照するときには、OCCURS DEPENDING ON 文節のオブジェクトの値は、整数-1 から整数-2 までの範囲内になければなりません。

OCCURS 文節の**サブジェクト**は、OCCURS 文節を含むデータ項目のデータ名です。OCCURS 文節のサブジェクトがタイプ名に従属する場合があります。 OCCURS 文節そのものを除いて、サブジェクトに対して使用されるデータ記述文節は、記述される項目のそれぞれのオカレンスに適用されます。

サブジェクトが SEARCH または USE FOR DEBUGGING 以外のステートメントで使用されているときはいつでも、サブジェクトが REDEFINES 文節のオブジェクトでないかぎり、添え字や指標を付ける必要があります。この場合、サブジェクトはテーブル・エレメントの中の 1 つのオカレンスを指します。

サブジェクトが SEARCH ステートメントで使用されている場合、USE FOR DEBUGGING ステートメントで使用されている場合、あるいはサブジェクトが REDEFINES 文節のオブジェクトの場合は、添え字および指標を付けてはなりません。この場合、サブジェクトはテーブル・エレメント全体を表します。

上記の 2 つの制限は LENGTH OF 特殊レジスターには適用されないことに注意してください。

1 つのレコード記述記入項目では、OCCURS DEPENDING ON 文節を含む記入項目の後には、それに従属する項目またはレベル 66 の項目だけしか続けることはできません。

OCCURS DEPENDING ON 文節は、別の OCCURS 文節に従属するものとして指定することはできません。

IBM Extension

複合 OCCURS DEPENDING ON は、以下のように構成されます。

- 従属項目は OCCURS DEPENDING ON 文節を含むことができます。
- OCCURS DEPENDING ON 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ます。ただし、非従属項目は、OCCURS DEPENDING ON 文節のオブジェクトであってはなりません。
- OCCURS DEPENDING ON 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、OCCURS DEPENDING ON オブジェクトの値による影響を受けます。

- OCCURS DEPENDING ON 文節のサブジェクトに従属する記入項目は、OCCURS DEPENDING ON 文節を含むことができます。
- ファイル記述 (FD) 記入項目内で暗黙の再定義が使用される場合、従属レベル項目は OCCURS DEPENDING ON 文節を含むことができます。
- OCCURS DEPENDING ON 文節を含む従属項目を持つテーブルには、INDEXED BY 句を指定することができます。

複合 OCCURS DEPENDING ON について詳しくは 9-46 ページの『付録 H. 複合 OCCURS DEPENDING ON』を参照してください。

End of IBM Extension

OCCURS 文節に使用するすべてのデータ名は修飾できますが、そのデータ名に添え字または指標を付けることはできません。

OCCURS または OCCURS DEPENDING ON 文節は、次のようなデータ記述記入項目には指定できません。

- レベル番号 01、66、77、または 88 を持つもの。
- 再定義されたデータ項目を記述するもの。(ただし、再定義された項目は、OCCURS 文節を含む項目に従属できます。) 6-75 ページの『REDEFINES 文節』を参照してください。

ASCENDING/DESCENDING および INDEXED BY 文節については 6-50 ページの『固定長テーブル』で説明されています。

注: OCCURS DEPENDING ON 文節を使用する場合には、テーブルに含まれるオカレンス数は、16,711,568 を超えてはならず、テーブル・エレメントの長さも 16,711,568 バイトを超えてはならず、さらにテーブル全体の長さも 16,711,568 バイトを超えてはなりません。

IBM Extension

複合 OCCURS DEPENDING ON は COBOL 85 標準の拡張としてサポートされます。コンパイラーによって許可される複合 ODO の基本形式を以下に示します。

- DEPENDING ON オプションが指定された OCCURS 文節によって記述され、非従属エレメントまたはグループが後に続いているデータ項目 (可変位置項目)。
- DEPENDING ON オプションが指定された OCCURS 文節によって記述され、DEPENDING ON オプションが指定された OCCURS 文節によって記述された非従属データ項目が後に続いているデータ項目 (可変位置テーブル)。
- DEPENDING ON オプションが指定された OCCURS 文節によって記述され、DEPENDING ON オプションが指定された OCCURS 文節によって記述された別のデータ項目内でネストしているデータ項目 (可変長エレメントを持つテーブル)。
- 可変長エレメントを持つテーブルの指標名。

複合 ODO は使用が容易でなく、コードの維持管理を困難にする可能性があります。ディスク・スペースを節約するために使用する場合には 9-46 ページの『付録 H. 複合 OCCURS DEPENDING ON』に示したガイドラインに従ってください。

End of IBM Extension

添え字付け

添え字付けとは、添え字を使用してテーブル参照を行う手法のことです。**添え字**とは、テーブル・エレメントのオカレンス番号を指定する正の整数値です。

添え字付けは、テーブル内の次元数を使って OCCURS 文節に関連付けられます。例えば 4 次元のテーブルには 4 つの添え字が必要です。添え字付けは、OCCURS 文節を介して定義された、多次元配列内のエレメントを識別するための COBOL での手法と見なすことができます。

詳細は 2-31 ページの『添え字付け』を参照してください。

RANGE オプションが指定あるいは暗示される場合には、システムによって添え字の値が有効であることが確認されます。 RANGE オプションが活動状態でない場合、添え字の値が有効であることを確認するのはユーザー の責任です。 RANGE オプションを使っても、その指標記入項目が有効かどうかをシステムが検査することはありません。指標の値が有効であることを確認するのはユーザー の責任です。

注: CRTCBLMOD コマンド、CRTBNDCBL コマンド、および PROCESS ステートメントに関する包括的な情報については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

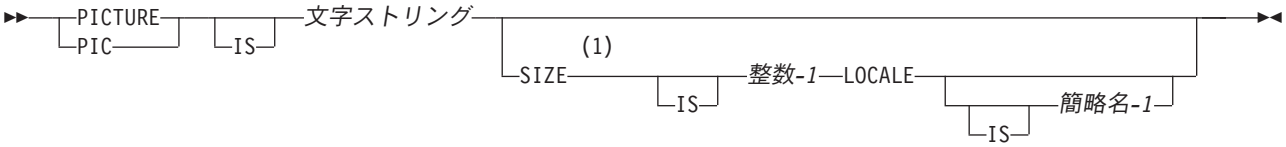
添え字付けの制約事項

1. データ名を添え字または修飾子として使用するときは、そのデータ名に添え字または指標を付けることはできません。
2. 指標は、PERFORM、SEARCH、または SET ステートメントによってのみ修正できます。
3. 添え字の中でリテラルを使用するときは、そのリテラルは正または符号なしの整数でなければなりません。
4. 相対添え字付けおよび指標付けでリテラルを使用するときは、そのリテラルは符号なしの整数でなければなりません。

PICTURE 文節

PICTURE 文節は、基本項目の一般的な特性や編集上の要件を指定します。

PICTURE 文節 - 形式



注:

1 IBM 拡張

PICTURE 文節は、指標データ項目、LIKE 文節のサブジェクト、RENAMES 文節のサブジェクト、または TYPE 文節のサブジェクトを除くすべての基本項目のそれぞれに対して指定する必要があります。

PICTURE 文字ストリングは、最大 90 文字までです。この文字ストリングは、記号として使用される特定の COBOL 文字で構成されます。基本データ項目のカテゴリーは、ロケール句が指定されている場合を除き、これらの文字の可能な組み合わせによって判別されます。 PICTURE 文節内の LOCALE 句は、カテゴリーの 1 つである数字編集項目を定義します。

DECIMAL-POINT IS COMMA を SPECIAL-NAMES 段落の中で指定すると、PICTURE 文字ストリングおよび数字リテラル内でピリオドとコンマの機能を交換できます。

通貨記号は、円記号 (\) または SPECIAL-NAMES 段落の CURRENCY SIGN 文節に指定された通貨記号のいずれかによって PICTURE 文字ストリングに示されます。

通貨記号は、通貨ストリングが置かれる 1 つまたは複数の文字位置を表します。

プログラムに複数の通貨記号が定義されている場合、同じ PICTURE 文字ストリング内ではいずれか 1 つの記号だけを繰り返すことができます。

ピクチャー・ストリング内に単一の通貨記号が現れる場合は、固定挿入記号になります。編集項目のサイズは、対応する通貨ストリングに含まれる文字数分増やされます。

複数の同じ通貨記号のストリングがピクチャー・ストリング内に現れる場合は、浮動挿入記号として使用されます。編集項目のサイズは、最初に現れる通貨記号に対応する通貨ストリングに含まれる文字数分、および PICTURE 文字ストリングの追加通貨記号ごとにそれ以降の文字分増やされます。

PICTURE 文節は、次のものには使えません。

- USAGE IS INDEX で記述された項目の記述

IBM Extension

- 内部浮動小数点 (USAGE IS COMP-1 または USAGE IS COMP-2) データ項目
- USAGE IS POINTER または USAGE IS PROCEDURE-POINTER データ項目
- 日付、時間、またはタイム・スタンプ項目
- TYPE 文節を含む項目の記述

End of IBM Extension

LOCALE 句

IBM Extension

LOCALE 句を PICTURE 文節に指定すると、編集はロケールの指定に従って行われます。次の規則が適用されます。

- BLANK WHEN ZERO 文節はロケール編集よりも高い優先順位を持つ。
- 簡略名-1 が指定されている場合は、項目の編集および編集解除には SPECIAL-NAMES 段落内の簡略名-1 に関連付けられているロケールが使用される。これ以外の場合は現行ロケールが使用されます。

  注: 編集ステージと編集解除ステージとでロケールを切り替えると、予測不能な結果を生じる可能性があります。項目の編集に使用されるロケールと項目の編集解除に使用されるロケールが同一であることを必ず確認する必要があります。
- 通貨記号 (cs) がピクチャー・ストリングに指定されている場合は、その通貨記号の位置、長さ、および文字ストリングはロケールから判別される。
- 小数点、3 桁ごとの分離文字、およびグループ化の方法はロケールから判別される。
- 小数点の位置合わせおよびゼロへの置き換えは 6-10 ページの『位置合わせの規則』で説明されている規則に従って行われる。

- PICTURE 文字ストリングに + が指定されている場合は、正数および負数の表示方法はロケールから判別される。
- 送信データは小数点の位置に合わせられ、(必要ならば) 受信データ項目の先頭または末尾の受信文字位置のいずれかにおいて切り捨てあるいはゼロの埋め込みが行われる。このデータに対しては右寄せも行われます。その際、ロケールの指定に従ってグループ化および分離文字の設定が行われます。先行ゼロはブランクに置き換えられます。

フォーマット設定後、PICTURE 文字ストリングに指定した桁数が受け入れ側の項目に合わず、送る側の項目に余分な桁が存在する場合は、左側の桁が切り捨てられ、オペレーティング・システムのエスケープ・メッセージが発行されます。

End of IBM Extension

PICTURE 文節で使用される記号

PICTURE 文節で使用される各記号の意味は、以下の表の中で定義されています。

- LOCALE 句が指定されていない 場合は 表 6-5 を参照。
- LOCALE 句が指定されている 場合は 6-61 ページの表 6-6 を参照。

PICTURE 文節の各記号を指定する際に守らなければならない順序は、以下の図に示されています。

- 6-62 ページの図 6-4 (LOCALE 句が指定されていない場合)
- 6-63 ページの図 6-5 (LOCALE 句が指定されている場合)

PICTURE の文節の記号の詳しい説明は図の後にあります。

PICTURE 文字ストリング内の句読点文字は、句読点文字とは見なされず、PICTURE 文字ストリング記号と見なされます。

OPTION オプション・パラメーター値 *NOMONOPIC、または PROCESS ステートメント・オプション NOMONOPIC が指定されている場合、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を区別します。すなわち、PICTURE 記号の A、B、C、D、E、G、N、P、R、S、V、X、および Z の大文字に対応する小文字は、PICTURE 文字ストリングでのそれらの大文字の表現と等価です。その他の小文字は、対応する大文字表現と等価ではありません。

OPTION パラメーター値 *MONOPIC、または PROCESS ステートメント・オプション MONOPIC が指定されている場合、PICTURE 文字ストリング内のすべての英字は大文字に変換されます。

表 6-5. LOCALE 句が指定されていない場合における PICTURE 文節の記号の意味

記号	意味
A	英字またはスペースだけを入れることができる文字位置。
B	スペースが非 DBCS データで 1 バイト、DBCS データで 2 バイトを占める文字位置。
E	**IBM Extension** 外部浮動小数点項目における指数の開始位置のマーク。マークは 1 バイトのストレージを占めます。 **End of IBM Extension**

表 6-5. LOCALE 句が指定されていない場合における PICTURE 文節の記号の意味 (続き)

記号	意味
P	仮想小数点の位取り位置。これは、データ項目内の数に小数点がない場合に仮想小数点の位置を指定するために使用します。位取り位置文字 P は、データ項目のサイズとしてはカウントされません。位取り位置文字は、数字編集項目、あるいは算術オペランドとして現れる項目の最大桁数 (63) を判別する際には、カウントされます。位取り位置文字 P は、PICTURE 文字ストリング内の左端または右端の桁位置に P の連続ストリングとしてのみ記入できます。位取り位置文字 P は仮想小数点を暗黙のうちに示す (一連の P が左端の PICTURE 文字ならば一連の P の左側、P が右端の PICTURE 文字ならば一連の P の右側) ので、仮想小数点記号 V は、PICTURE 記述などの中では、左端または右端のいずれかの文字が冗長となります。 PICTURE 文字ストリングに記号 P が入っているデータ項目を参照するある演算においては、データ項目の実際の文字表示よりむしろデータ項目の代数値が使用されます。この代数値は、記号 P により指定される数字位置の代わりにゼロ、および指図されたロケーションに小数点を想定します。値のサイズは、PICTURE 文字ストリングにより表される数字位置の数です。これらの演算は次のうちのいずれかです。 • 数字送り出しオペランドを必要とする演算 • 送り出しオペランドが数字で、その PICTURE 文字ストリングに記号 P が入っている MOVE ステートメント • 送り出しオペランドが数字編集で、その PICTURE 文字ストリングに記号 P が含まれ、受け入れオペランドが数字または数字編集の MOVE ステートメント • 両オペランドが数字である比較演算 その他すべての演算において、記号 P で指定される数字位置は無視され、オペランドのサイズにはカウントされません。
S	演算符号があることを示します。(ただし、演算符号の表現や、また必ずしもその位置を表すものではない。) これは PICTURE ストリングの左端の文字として書かなければなりません。演算符号は、演算に使用される項目の値が正であるか負であるかを示します。関連する SIGN 文節で SEPARATE CHARACTER 句を指定していないかぎり、記号 S は、基本項目のサイズの判別ではカウントされません。ハードウェア命令は符号を使用するので、可能なところでは PICTURE 文節に S を含めて パフォーマンスを向上させることができます。
V	仮想小数点の位置を示します。 これは文字ストリング内に、1 回だけ書くことができます。 V は、文字位置を表すものではないため、基本項目のサイズとしてはカウントされません。仮想小数点がストリングの右端記号の右側にあるとき、V は必要ありません。
X	EBCDIC 文字セットから任意の使用可能な文字を入れることができる文字位置。
Z	先行数字位置。その位置にゼロが入ると、そのゼロはスペース文字に置き換えられます。各 Z は項目のサイズとしてカウントされます。
9	数字が入る文字位置で、項目のサイズとしてカウントされます。
1	**IBM Extension** ブール値の B"1" または B"0" を含む文字位置。 DISPLAY として明示的あるいは暗黙に定義して使用しなければなりません。 **End of IBM Extension**
0	数字のゼロが挿入される文字位置。各ゼロは項目のサイズとしてカウントされます。
/	スラッシュが挿入される文字位置。各スラッシュは、項目のサイズとしてカウントされます。

表 6-5. LOCALE 句が指定されていない場合における PICTURE 文節の記号の意味 (続き)

記号	意味
,	コンマが挿入される文字位置。この文字は項目のサイズとしてカウントされます。コンマ挿入文字が PICTURE 文字ストリングの最後の記号の場合には、PICTURE 文節がデータ記述記入項目の最後の文節で、直後に分離文字ピリオドが続かなければなりません。
.	位置合わせ用の小数点を表す編集記号。さらに、これはピリオドが挿入される文字位置を表します。この文字は項目のサイズとしてカウントされます。ピリオド挿入文字が、PICTURE 文字ストリング内の最後の記号である場合には、PICTURE 文節がデータ記述記入項目の最後の文節であり、直後に分離文字ピリオドが続かなければなりません。 **注:** ある種のプログラムでは、SPECIAL-NAMES 段落で DECIMAL-POINT IS COMMA 文節が指定されている場合、ピリオドとコンマの機能が交換されます。 その場合は、PICTURE 文節の中にピリオドやコンマがどこにあったとしても、ピリオドの規則がコンマに適用され、コンマの規則がピリオドに適用されます。
+ - CR DB	編集符号制御記号。それぞれ、編集符号制御記号が入れられる文字位置を表します。 1 つの文字ストリングの中でこれらの記号を併用することはできません。記号の中で使用された各文字は、データ項目のサイズを判別する際にカウントされます。
*	金額変造防止シンボル。これは先頭数字位置であり、この位置にゼロが含まれているときにはアスタリスクが入れられます。それぞれのアスタリスク (*) は、項目のサイズとしてカウントされます。
$	通貨記号が入る文字位置。文字ストリング内の通貨記号は、記号 \ で表されるか、または環境部の SPECIAL-NAMES 段落の CURRENCY SIGN 文節で指定された単一の文字で表されます。通貨記号は、項目のサイズとしてカウントされます。
G	**IBM Extension** DBCS 位置。ストレージの 2 バイトを占めます。1 文字としてカウントされます。非 DBCS 項目には指定できません。 USAGE は、DISPLAY-1 として明示的に定義しなければなりません。 **End of IBM Extension**
N	**IBM Extension** • 使用法が NATIONAL として明示的に定義されている場合には、国別 (UCS-2 またはユニコード) 文字位置。 • 使用法が DISPLAY-1 として明示的に定義されている場合には、1 文字として 2 バイトのストレージを占有する DBCS 位置。 • 基本項目に、またはデータ項目が属しているグループに、USAGE 文節が指定されていない場合には、以下の規則が適用されます。 – NATIONAL コンパイラー・オプションが有効であれば、USAGE NATIONAL が暗黙指定されます。 – その他の場合は、USAGE DISPLAY-1 が暗黙指定されます。 **End of IBM Extension**

表 6-6. LOCALE 句が指定されている場合における PICTURE 文節の記号の意味

記号	意味
9	数字が入る文字位置であり、編集項目内に含めることができる数字としてカウントされます。
.	位置合わせ用の小数点を表す編集記号。ピリオド挿入文字が、PICTURE 文字ストリング内の最後の記号である場合には、PICTURE 文節がデータ記述記入項目の最後の文節であり、直後に分離文字ピリオドが続かなければなりません。実行時に使用される小数点文字はロケールからとられます。 **注:** ある種のプログラムでは、SPECIAL-NAMES 段落で DECIMAL-POINT IS COMMA 文節が指定されている場合、ピリオドとコンマの機能が交換されます。その場合は、PICTURE 文節の中にピリオドやコンマがどこにあったとしても、ピリオドの規則がコンマに適用され、コンマの規則がピリオドに適用されます。
+	編集符号制御記号。記号 + は、指定されたロケールに従って編集項目に符号を付けることを指示します。記号 + が指定されていない場合は、編集項目は符号なしとなります。
cs	文字ストリング内の通貨記号は、指定されたロケールと関連付けられている通貨ストリングを編集項目に含めることを指示します。

6-62 ページの図 6-4 は、LOCALE 句を指定しない 場合において、PICTURE 文節の各記号を指定する際に従う必要のある順序を示しています。図の最後の注を参照してください。 6-63 ページの図 6-5 は、LOCALE 句を指定する 場合において、PICTURE 文節の各記号を指定する際に従う必要のある順序を示しています。

最初の記号 ＼ 2番目の記号		固定挿入記号										浮動挿入記号						その他の記号								
		B	0	/	,	.	{+ −}	{+ −}	{CR DB}	$	E	{Z *}	{Z *}	{+ −}	{+ −}	$	$	9	A X	S	V	P	P	1	G	N
固定挿入記号	B	X	X	X	X	X	X			X		X	X	X	X	X	X	X	X		X		X		X	
	0	X	X	X	X	X	X			X		X	X	X	X	X	X	X	X		X		X			
	/	X	X	X	X	X	X			X		X	X	X	X	X	X	X	X		X		X			
	,	X	X	X	X	X	X			X		X	X	X	X	X	X	X			X		X			
	.	X	X	X	X		X			X		X		X		X		X								
	{+ −}																									
	{+ −}	X	X	X	X	X				X	X	X	X			X	X	X			X	X	X			
	{CR DB}	X	X	X	X	X				X		X	X			X	X	X			X	X	X			
	$						X																			
	E				X	X												X			X					
浮動挿入記号	{Z *}	X	X	X	X		X			X		X														
	{Z *}	X	X	X	X	X	X			X		X	X								X		X			
	{+ −}	X	X	X	X					X				X												
	{+ −}	X	X	X	X	X				X				X	X						X		X			
	$	X	X	X	X		X									X										
	$	X	X	X	X	X	X									X	X				X		X			
その他の記号	9	X	X	X	X	X	X			X		X		X		X		X	X	X	X		X			
	A X	X	X	X														X	X							
	S																									
	V	X	X	X	X		X			X		X		X		X		X		X		X				
	P	X	X	X	X		X			X		X		X		X		X		X		X				
	P						X			X										X	X		X			
	1																									
	G	X																							X	
	N																									X

図 6-4. LOCALE 句を指定しない場合における PICTURE 文節の記号の順序

図 6-4 に関する注:

1. 交差するところにある X は、最上段にある記号が、ある文字ストリングの中で、左欄にある記号の左側であればどこにあってもよいことを示します。
2. \ 文字は、該当する文字セットで表されますが、通貨記号のデフォルト値です。

3. A、X、Z、9、または * 記号のうちの少なくとも 1 つ、または +、-、あるいは \ 記号のうちの少なくとも 2 つが PICTURE ストリングになければなりません。

IBM Extension

4. 記号 G または N は、PICTURE 文字ストリングに単独で現れます。

End of IBM Extension

5. 固定挿入記号 + と -、浮動挿入記号 Z、*、+、-、$、および記号 P は、上記の PICTURE 文字優先テーブルに 2 回現れています。各記号の最左端の列と最上段は、小数点の左側にあるときの用法を示しています。表の中で 2 番目に出てくる記号は、それぞれ小数点の右側にあるときの用法を示しています。({ }) は、相互に排他的な項目を示しています。
6. 中括弧 ({}) は、相互に排他的な項目を示しています。

IBM Extension

最初の記号 / 2 番目の記号		記号			
		9	CS	.	+
記号	9	X	X	X	X
	CS				X
	.	X	X		X
	+				

図 6-5. LOCALE 句を指定する場合における PICTURE 文節の記号の順序

End of IBM Extension

文字ストリングの表現法

次の記号は、1 つの PICTURE 文字ストリングで複数回表すことができます。

```
A  B  P  X  Z  9  0  /  ,  +  -  *  $
```

IBM Extension

```
G     N
```

End of IBM Extension

次の記号は、1 つの PICTURE 文字ストリングで 1 回だけ表すことができます。

```
S V . CR DB
```

IBM Extension

E 1

End of IBM Extension

LOCALE 句が指定されている場合に複数回現れることができる記号は 9 だけです。 LOCALE 句が指定されている場合に 1 つの PICTURE 文字ストリング内で 1 回だけ現れることができる記号を以下に示します。

. + cs

PICTURE 文字ストリング内に複数回現れることができる記号のすぐ後にある小括弧で囲まれた整数は、その記号が連続発生する回数を指定します。整数は、定数名によって指定される場合があります。連続オカレンス数が 16 711 568 を超えることはできません。

例えば、次の 2 つの PICTURE 文節の指定は同等です。

```
PICTURE IS $99999.99CR

PICTURE IS $9(5).9(2)CR
```

上記の記号が文字ストリングに現れるごとに、それはその文字または許容される文字セットがデータ項目に 1 回現れることを示します。

データ・カテゴリーと PICTURE の規則

PICTURE 記号の可能な組み合わせによって、項目のデータ・カテゴリーが決まります。

- 英字項目
- 数字項目
- 数字編集項目
- 英数字項目
- 英数字編集項目
- ブール項目

IBM Extension

- DBCS 項目
- DBCS 編集項目
- 国別項目
- 外部浮動小数点項目

End of IBM Extension

注: LOCALE 句が PICTURE 文節に指定されている場合は、PICTURE 文節によって定義されるデータのカテゴリーは数字編集のみとなります。

英字項目:

- PICTURE 文字ストリングには記号 A だけ含むことができます。
- 標準データ・フォーマットの項目の内容は、英字とスペース文字の中の任意の文字で構成されていなければなりません。
- USAGE DISPLAY を指定するか、または暗黙指定する必要があります。

- 関連する VALUE 文節には、英字だけ、または表意定数 SPACE が入っている非数字リテラルを指定しなければなりません。

数字項目:

- 数字項目のタイプには、次のものがあります。
  - 2 進数
  - パック 10 進数 (内部 10 進数)
  - ゾーン 10 進数 (外部 10 進数)
  - 国別 10 進数 (外部 10 進数)
- PICTURE 文字ストリングには、記号 9、P、S、および V を入れることができます。
- 数字の桁数は、1 以上 18 以下でなければなりません。

IBM Extension

- パック 10 進数およびゾーン 10 進数の項目の場合、数字の桁数は 1 ～ 63 の範囲にできます。

End of IBM Extension

- 符号なしの場合、 標準データ・フォーマットの項目の内容には、アラビア数字 0 から 9 の組み合わせが入っていなければなりません。符号付きの場合、 +、-、あるいは演算符号のその他の表現を入れることができます。
- 項目の USAGE は、DISPLAY、BINARY、COMPUTATIONAL、または PACKED-DECIMAL とすることができます。

IBM Extension

- 項目の USAGE は、COMPUTATIONAL-3、COMPUTATIONAL-4、または NATIONAL とすることができます。

USAGE NATIONAL で記述された符号付き数字項目の場合は、SIGN IS SEPARATE 節を指定または暗黙指定する必要があります。

End of IBM Extension

- 基本数字項目に関連付けられている VALUE 文節は、数字リテラルまたは表意定数 ZERO を指定しなければなりません。基本数字項目から成るグループ項目に関連付けられている VALUE 文節は、**非数字**リテラルまたは表意定数を指定しなければなりません。なぜなら、そのようなグループは英数字であると見なされるからです。上記のどちらの場合でも、リテラルは指定されたとおりに扱われ、編集は行われません。

数字項目の例を示します。

```
PICTURE          有効な値の範囲
9999                  0 ～ 9999
S99                 -99 ～ +99
S999V9            -999.9 ～ +999.9
PPP999                0 ～ .000999
S999PPP            -1000 ～ -999000 および
                   +1000 ～ +999000 または 0
```

数字編集項目:

- PICTURE 文字ストリングには次の記号を入れることができます。

  ```
  B  P  V  Z  9  0  /  ,  .  +  -  CR  DB  *  $
  ```

  入れられる記号の組み合わせ方は、PICTURE 文節の記号の許される順番 (6-62 ページの図 6-4 を参照) および編集規則 (6-69 ページの『PICTURE 文節の編集』を参照) で決まります。その他に、次の規則も適用されます。

  – BLANK WHEN ZERO 文節を項目に指定するか、ストリングに次の記号のうちの少なくとも 1 つが含まれていなければなりません。

    ```
    B  /  Z  0  ,  .  *  +  -  CR  DB  $
    ```

  – 文字ストリングで表される数字の桁数は、1 ～ 18 の範囲になければなりません。

  IBM Extension

  – 文字ストリングで表される数字の桁数は、1 ～ 63 の範囲になければなりません。

  End of IBM Extension

  – 結果的な文字位置の全体の長さは、127 以下でなければなりません。

- 標準データ・フォーマットで数字を表す文字位置の内容は、10 個のアラビア数字のいずれかでなければなりません。
- USAGE DISPLAY を指定するか、または暗黙指定する必要があります。
- 関連付けられている VALUE 文節では、非数字リテラルまたは表意定数を指定しなければなりません。リテラルは指定したとおりに扱われ、編集は行われません。

LOCALE 句を指定する場合:

IBM Extension

- PICTURE 文字ストリングには次の記号を入れることができます。

  ```
  9  .  + cs (LOCALE 用通貨記号)
  ```

  これらの記号のうち、次のものは 1 回だけしか使用することができません。

  ```
  .  + cs
  ```

- 使用可能な文字位置の数は SIZE 句内の整数-1 で指定します。
- USAGE DISPLAY を指定するか、または暗黙指定する必要があります。
- 関連付けられている VALUE 文節では、非数字リテラルまたは表意定数を指定しなければなりません。リテラルは指定したとおりに扱われ、編集は行われません。
- 受信項目が数字編集データ項目であり、PICTURE 文節の LOCALE 句がそのデータ記述記入項目内に指定されている場合、そのデータは 6-57 ページの『LOCALE 句』で説明されている規則に従って位置合わせされます。

End of IBM Extension

英数字項目:

- PICTURE 文字ストリングは、次のいずれかで構成する必要があります。
  – 記号 X

  - 記号 A、X、および 9 の組み合わせ (すべて A またはすべて 9 の入った文字ストリングは、英数字項目を定義しません)。
- 項目は、文字ストリングが記号 X のみを含んでいるかのように扱われます。
  - 標準データ・フォーマットの項目の内容は、EBCDIC 文字セットで許容される任意の文字とすることができます。
  - USAGE DISPLAY を指定するか、または暗黙指定する必要があります。
  - 関連付けられている VALUE 文節では、非数字リテラルまたは表意定数を指定しなければなりません。

英数字編集項目:

- PICTURE 文字ストリングには次の記号を入れることができます。

  ```
  A X 9 B 0 /
  ```

- ストリングは、少なくとも A または X が 1 つと、少なくとも 1 つの B または 0 (ゼロ) または / が含まれていなければなりません。
- 標準データ・フォーマットの項目の内容は、EBCDIC 文字セットで許容される任意の文字とすることができます。
- 結果的な文字位置の全体の長さは、127 以下でなければなりません。
- USAGE DISPLAY を指定するか、または暗黙指定する必要があります。
- 関連付けられている VALUE 文節では、非数字リテラルまたは表意定数を指定しなければなりません。リテラルは指定したとおりに扱われ、編集は行われません。

ブール項目:

IBM Extension

次の規則が適用されます。

1. PICTURE 文字ストリングには、記号 1 だけ含めることができます。
2. ただ 1 つの文字 1 が指定できます。
3. 項目の USAGE は、DISPLAY だけになり得ます。
4. 関連する VALUE 文節では、ブール・リテラル (B"1" または B"0") あるいはゼロを指定しなければなりません。
5. 次の文節は、ブール項目に指定できません。
   - SIGN 文節
   - BLANK WHEN ZERO 文節
   - ASCENDING/DESCENDING KEY 文節
6. INDICATOR 文節を指定できます。

(標識について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください)。

End of IBM Extension

DBCS 項目:

IBM Extension

1. PICTURE 文字ストリングには、記号 G または N を含むことができます。
2. G または N のおのおのは、単一の DBCS 文字位置 (2 バイト) を表現します。
3. PICTURE 文節記号 G を使用した場合、USAGE DISPLAY-1 を指定しなければなりません。
4. PICTURE 文節記号 N を使用した場合、USAGE DISPLAY-1 を暗黙的または明示的に指定しなければなりません。
5. 関連する VALUE 文節は、DBCS リテラルまたは表意定数 SPACE/SPACES を指定しなければなりません。

End of IBM Extension

DBCS 編集項目:

IBM Extension

1. PICTURE 文字ストリングは、G および B と少なくともおのおの 1 つとの組み合わせです。
2. G および B のおのおのは、単一の DBCS 文字位置 (2 バイト) を表現します。
3. USAGE DISPLAY-1 を指定しなければなりません。
4. 関連する VALUE 文節は、DBCS リテラルまたは表意定数 SPACE/SPACES を指定しなければなりません。

End of IBM Extension

国別項目:

IBM Extension

1. PICTURE 文字ストリングは、記号 N を含むことができます。
2. 項目の内容は UCS-2 文字でなくてはならず、複数のエンコード単位を必要とする文字であってはいけません。
3. 各 N は単一の UCS-2 文字を表します。
4. PICTURE 文節記号 N を使用した場合、USAGE NATIONAL を暗黙的または明示的に指定しなければなりません。
5. 関連する VALUE 文節では、英数字リテラル、非数字リテラル、国別リテラル、または以下の表意定数のいずれか 1 つを指定しなければなりません。
   - SPACE/SPACES
   - ZERO/ZEROS/ZEROES
   - ALL *英数字リテラル*
   - ALL *国別リテラル*

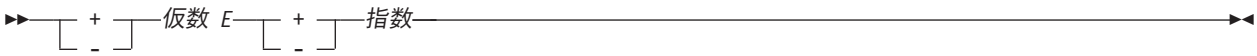
End of IBM Extension

外部浮動小数点項目:

IBM Extension

- PICTURE ストリングは次の形式に従わなくてはなりません。

  フォーマット



  + または -
  符号文字は、仮数および指数の直前になくてはなりません。 + 符号は、正の値を表すために正符号が出力で使用されていることを示し、負符号が負の値を表していることを示しています。 - 符号は、正の値を表すためにブランクが出力で使用されていることを示し、負符号が負の値を表していることを示しています。各符号桁は 1 バイトのストレージを占めます。

  仮数
  仮数には記号を入れることができます。

  ```
  9 . V
  ```

  実小数点はピリオド (.) で表示され、仮想小数点は V で表示されます。仮数には、実小数点または仮想小数点のいずれかが存在しなければなりません。小数点は、先頭にあっても、途中にあっても、最後にあってもかまいません。仮数には 1 ～ 16 桁の数字を入れることができます。 . および V は、この数字の桁数としてはカウントされません。

  E 仮数と指数の区切りを示すために使用されます。これは必要です。

  指数
  指数は記号 `99` または `999` で構成されている必要があります。

- 文節 OCCURS、REDEFINES、LIKE、RENAMES、および TYPEDEF は外部浮動小数点項目と関連付けることができます。
- SIGN 文節は、文書として受け入れられ、その符号の表示に対してどのような影響も与えません。
- SYNCHRONIZED 文節は、文書として扱われます。
- 次の文節は、外部浮動小数点項目に指定できません。
  - BLANK WHEN ZERO
  - JUSTIFIED
  - VALUE

End of IBM Extension

PICTURE 文節の編集

PICTURE 文節の中で編集を行う一般的な方法が 2 つあります。

- 挿入による編集
  - 単純挿入
  - 特別挿入
  - 固定挿入
  - 浮動挿入

- 消去と置き換えによる編集
  - アスタリスクを使ったゼロの消去と置き換え
  - スペースを使ったゼロの消去と置き換え

項目に許容される編集のタイプは、**データ・カテゴリー**によって異なります。各カテゴリーに有効な編集のタイプが以下に示されています。

表 6-7. データ・カテゴリーに有効な編集

カテゴリー	**編集のタイプ**
英字	なし
IBM Extension ブール **End of IBM Extension**	なし
IBM Extension DBCS **End of IBM Extension**	なし
IBM Extension 国別 **End of IBM Extension**	なし
IBM Extension DBCS 編集 **End of IBM Extension**	単純挿入
数字	なし
英数字	なし
英数字編集	単純挿入
数字編集	すべて
IBM Extension 外部浮動小数点 **End of IBM Extension**	特別挿入

単純挿入による編集: この編集タイプは、英数字編集項目および数字編集項目に有効です。

IBM Extension

この編集タイプは、DBCS 編集項目に有効です。

End of IBM Extension

各挿入記号は、項目のサイズとしてカウントされ、それに同等の文字が挿入される項目内の位置を表します。

表 6-8. 単純挿入による編集 - 各データ・カテゴリーごとの正しい挿入記号

カテゴリー	正しい挿入記号
英字	なし
IBM Extension ブール **End of IBM Extension**	なし
IBM Extension DBCS **End of IBM Extension**	なし
IBM Extension 国別 **End of IBM Extension**	なし
IBM Extension DBCS 編集 **End of IBM Extension**	B
数字	なし
英数字	なし
英数字編集	B 0 / - . , &
数字編集	B 0 / ,

単純挿入による編集の例

```
PICTURE          データの値          編集結果
X(10)/XX         ALPHANUMER01       ALPHANUMER/01
X(5)BX(7)        ALPHANUMERIC       ALPHA NUMERIC
99,B999,B000     1234               01, 234, 000
99,999           12345              12,345
GGBBGG           D1D2D3D4           D1D2    D3D4
```

特別挿入による編集: このタイプの編集は、次の項目に対してのみ有効です。

- 数字編集項目

IBM Extension

- 外部浮動小数点項目

End of IBM Extension

ピリオド (.) は特別挿入記号ですが、位置合わせに使う実小数点も表します。

ピリオド挿入記号は、項目のサイズとしてカウントされ、実小数点が挿入される項目内の位置を表します。

1 つの PICTURE 文字ストリングの中に、実小数点と、仮想小数点としての記号 V の両方ではなく、いずれか 1 つを指定しなければなりません。

特別挿入による編集の例

```
PICTURE          データの値          編集結果
999.99              1.234           001.23
999.99             12.34            012.34
999.99            123.45            123.45
999.99           1234.5             234.50
+999.99E+99     12345               +123.45E+02
```

固定挿入による編集: このタイプの編集は、数字編集項目に対してのみ有効です。次の挿入記号を使用します。

- 通貨記号 (例えば \)
- + - CR DB (編集符号制御記号)

固定挿入による編集では、1 つの PICTURE 文字ストリングの中には 1 つの通貨記号と 1 つの編集符号制御記号だけを指定できます。

通貨記号は、通貨符号が表示される位置を表します。通貨符号を通貨記号自体にしたり、SPECIAL-NAMES 段落の CURRENCY SIGN 文節に指定されている長さの 1 文字または複数の文字の通貨ストリングにしたりできます。編集項目のサイズは、対応する通貨ストリングに含まれる文字数分増やされます。

+ または - 記号が前に付いている場合を除き、通貨記号 (\) は文字ストリングの最初の文字でなければなりません。

+ または - を記号として使用する場合、それは文字ストリングの最初かあるいは最後の文字でなければなりません。

CR または DB を記号として使用する場合、それは文字ストリングの右端の 2 文字の位置を占めなければなりません。これらの 2 文字位置に記号 CR または DB が含まれる場合には、大文字が挿入文字です。

編集符号制御記号によって得られる結果は、次に示すように、データ項目の値に応じて異なります。

PICTURE PICTURE 内 文字ストリング	結果: データ項目 正またはゼロ	結果: データ項目 負
+	+	-
-	スペース	-
CR	2 スペース	CR
DB	2 spaces	DB

固定挿入による編集の例です。

PICTURE	データの値		編集結果
999.99+	+6555.556	555.55+	
+9999.99	-6555.555	-6555.55	
9999.99	+1234.56	1234.56	
$999.99	-123.45	$123.45	
U999.99	-123.45	EUR123.45	1
-$999.99	-123.456	-$123.45	
-u999.99	-123.456	-USD123.45	2
-$999.99	+123.456	$123.45	
$9999.99CR	+123.45	$0123.45	
$9999.99DB	-123.45	$0123.45DB	

1 通貨符号が次のように定義されている場合: CURRENCY SIGN IS "EUR" PICTURE SYMBOL "U"

2 通貨符号が次のように定義されている場合: CURRENCY SIGN IS "USD" PICTURE SYMBOL "u"

注: 符号を切り捨てると、貸方として負の金額が示されることに注意してください。

浮動挿入による編集: このタイプの編集は、数字編集項目に対してのみ有効です。次の記号を使用します。

- 通貨記号 (例えば \)
- + -

1 つの PICTURE 文字ストリングの中では、これらの記号は浮動挿入文字として同時に指定することはできません。

浮動挿入による編集は、PICTURE 文字ストリングに連続した複数の浮動挿入記号を組み込むことによって指定されます。

通貨記号は通貨符号を表します。通貨符号は通貨記号自体にしたり、SPECIAL-NAMES 段落の CURRENCY SIGN 文節に指定されている長さの 1 文字または複数の文字の通貨ストリングにしたりできます。編集項目のサイズは、最初に現れる通貨記号に対応する通貨ストリングに含まれる文字数分、および PICTURE 文字ストリングの追加通貨記号ごとにそれ以降の文字分増やされます。

浮動挿入記号が単一文字を表す場合には、この記号を使用して、対応する文字が挿入されるすべての文字位置を表します。文字ストリング内の左端の浮動挿入記号は、そのデータ項目の中に文字が現れる左端の限界を表します。右端の浮動挿入記号は、文字が現れる右端の限界を表します。

浮動挿入記号が複数文字の通貨ストリングを表す場合には、この記号を使用して、通貨ストリングの最終文字が挿入されるすべての位置を表します。文字ストリング内の左端の浮動挿入記号は、そのデータ項目の中に通貨ストリングの最終文字が現れる左端の限界を表します。右端の浮動挿入記号は、通貨ストリングの最終文字が現れる右端の限界を表します。

文字ストリングの左端から 2 番目の浮動挿入記号は、データ項目の中で数字データが現れる左端の限界を表します。この限界の右端にある、または右端の隣の浮動挿入記号は、数字位置を表します。これは、ゼロ以外の数字が前に付いた数字で始まる数値データによって置き換えることができます。

浮動挿入記号のストリング内か、またはそのすぐ右側にある単純挿入記号 (B 0 / ,) は、浮動文字ストリングの一部と見なされます。ピリオド (.) の特別挿入記号が浮動ストリングに入っていると、文字ストリングの一部と見なされます。

PICTURE 文字ストリングでは、浮動挿入による編集を表す方法は 2 とおりあります。それにより、次のように編集が行われます。

1. 小数点の左端にある任意の、またはすべての先行数字位置を浮動挿入記号で表します。編集が行われると、データ内の最初の非ゼロ数字または小数点 (この 2 つのうち、より左側にある方) のすぐ左に、単一の浮動符号挿入記号 (+ または -) あるいは通貨符号が 1 つ置かれます。挿入記号または通貨記号の左側の未使用の位置はスペースで埋められます。
2. すべての数字位置を浮動挿入記号で表します。編集が行われると、次のようになります。
   - データの値がゼロならば、データ項目全体にスペースが入ります。
   - データの値がゼロでなければ、結果は規則 1 の場合と同じになります。

切り捨てが行われるのを避けるために、PICTURE 文字ストリングの最小サイズは、次の数の合計でなければなりません。

- 送り出し項目の文字位置の数
- 受け入れ項目の固定挿入の記号の数
- 浮動挿入記号の文字の数

浮動挿入による編集の例です。

```
   PICTURE                 データの値          編集結果
$$$$.99                  .123              $.12
       $$$9.99                 .12               $0.12
   $,$$$,999.99           -1234.56            $1,234.56
   U,UUU,UU9.99-          -1234.56           EUR1,234.56-
   u,uuu,uu9.99            1234.56           USD1,234.56
   +,+++,999.99         -123456.789          -123,456.78
  $$,$$$,$$$.99CR       -1234567           $1,234,567.00CR
  ++,+++,+++.+++            0000.00
```

注: 符号を切り捨てると、貸方として負の金額が示されることに注意してください。

ゼロの消去と置き換えによる編集: このタイプの編集は、数字編集項目に対してのみ有効です。 ゼロ消去による編集では、記号 Z と * を使用します。これらの記号は、1 つの PICTURE 文字ストリングにおいては相互に排他的です。

次の記号は、1 つの PICTURE 文字ストリングの中では浮動記号として同時に指定することはできません。

```
Z  *  +  -  通貨記号 (例えば、$)
```

ゼロ消去および置き換えによる編集を指定するには、許容されている記号を 1 つまたは複数含むストリングを使用して、ゼロ消去と置き換えによる編集を行うことのできる左端の文字位置を表します。

浮動編集記号のストリング内か、またはそのすぐ右側にある単純挿入記号 (B 0 /、) は、そのストリングの一部と見なされます。ピリオド (.) の特別挿入記号が浮動編集ストリングに入っていると、文字ストリングの一部と見なされます。

PICTURE 文字ストリングでは、ゼロ消去を表す方法は 2 とおりあります。それにより、次のように編集が行われます。

- 小数点の左側にある任意の、またはすべての先行数字位置を消去記号で表します。編集が行われると、消去記号と同じ文字位置に現れるデータ内の先行ゼロは、置き換え文字で置き換えられます。消去は、次の文字のうち、最も左側にある文字で終わります。
  - 消去記号に対応しない文字
  - ゼロ以外のデータが入っている文字
  - 小数点
- PICTURE 文字ストリング内のすべての数字位置を消去記号で表します。編集が行われると、データの値がゼロでなければ、結果は前述の規則の場合と同じです。データの値がゼロのときは、次のようになります。
  - Z が指定されていれば、データ項目全体にスペースが入ります。
  - * が指定されていれば、データ項目全体 (実小数点は除く) にアスタリスクが入ります。

注: 同じ記入項目に、消去記号としてのアスタリスク (*) と BLANK WHEN ZERO 文節の両方を指定してはなりません。

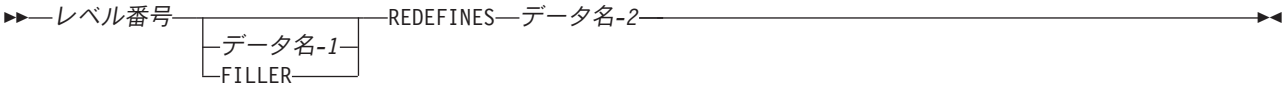
ゼロ消去と置き換えによる編集の例です。

```
   PICTURE                 データの値          編集結果
****.**           0000.00                ****.**
          ZZZZ.ZZ           0000.00
          ZZZZ.99           0000.00                  .00
          ****.99           0000.00               ****.00
          ZZ99.99           0000.00                 00.00
         Z,ZZZ.ZZ+          +123.456                123.45+
         *,***.**+          -123.45                **123.45-
     **,***,***.**       +12345678.9         12,345,678.90+
     $Z,ZZZ,ZZZ.ZZCR       +12345.67         $   12,345.67
    $B*,***,***.**BBDB     -12345.67        $ ***12,345.67  DB
```

REDEFINES 文節

REDEFINES 文節によって、異なるデータ記述記入項目を使って、同じコンピューター・ストレージ域を記述できます。

REDEFINES 文節 - 形式



REDEFINES 文節を指定する場合は、データ名-1 または FILLER に続く最初の記入項目にしなければなりません。データ名-1 または FILLER を指定しない場合には、REDEFINES 文節は、レベル番号に続く最初の記入項目にしなければなりません。記述するデータ項目は、FILLER が指定されているのと同様に取り扱われます。

データ名-1 およびデータ名-2 のレベル番号は同じでなければならず、レベル 66 またはレベル 88 であってはなりません。

データ名-1/FILLER
同じ区域に関する代替記述を識別します。これは**再定義する**項目または **REDEFINES サブジェクト**です。

データ名-2

再定義される項目または **REDEFINES オブジェクト**です。これは、REDEFINES 文節の**サブジェクト**であるデータ名-1 と対照的です。

IBM Extension

データ名-1 とデータ名-2 のどちらも、ポインター、プロシージャー・ポインター、外部浮動小数点データ項目または内部浮動小数点データ項目、DBCS 項目、国別項目、日付項目、時刻項目、またはタイム・スタンプ項目を指定できます。

End of IBM Extension

REDEFINES 文節をコーディングするときは、次のような規則が適用されます。

FD 記入項目に従属する複数のレベル 01 記入項目を書くと (さらにそのレベル 01 記入項目がタイプ名ではない場合は)、暗黙の再定義と呼ばれる条件が生じます。すなわち、2 番目のレベル 01 記入項目は、最初の記入項目に割り当てられた記憶を暗黙のうちに再定義します。このようなレベル 01 記入項目の中では REDEFINES 文節および TYPE 文節を指定してはなりません。さらに、TYPE 文節は、レベル 01 記入項目に従属するいかなる項目の中でも指定してはなりません。

再定義プロセス

再定義は、データ名-1 から始まり、データ名-1 のレベル番号と同じかそれより低いレベル番号が検出された時点で終了します。データ名-1 とデータ名-2 のレベル番号の間には、それら 2 つより数字の上で低いレベル番号を持つ記入項目があってはなりません。例えば、次のとおりです。

```
05    A PICTURE X(6).
05    B REDEFINES A.
  10 B-1                    PICTURE X(2).
  10 B-2                    PICTURE 9(4).
05   C                      PICTURE 99V99.
```

この例では、A が再定義される項目で、B が再定義する項目です。再定義は B から始まって、2 つの従属項目 B-1 と B-2 を含んでいます。レベル 05 の項目 C を検出すると、再定義が終了します。

再定義される項目のデータ記述記入項目には、OCCURS 文節を含めることができません。しかし、再定義される項目は、データ記述記入項目に OCCURS 文節を含んでいる項目に従属することはできます。この場合、REDEFINES 文節内の再定義される項目への参照は、添え字付けされてはなりません。元の項目、再定義項目、およびこれらの項目に従属するすべての項目は、OCCURS DEPENDING ON 文節を含むことができません。

REDEFINES 文節の入ったデータ記述記入項目に GLOBAL 文節を使うと、その文節のサブジェクトだけがグローバル属性をもちます。 EXTERNAL 文節は、REDEFINES 文節と同じデータ記述記入項目に指定してはなりません。オブジェクトが GLOBAL または EXTERNAL であれば、サブジェクトは属性を継承しません。

データ名-1 (再定義する項目) はデータ名-2 (再定義される項目) より小さくてもかまいません。データ名-1 (再定義する項目) がデータ名-2 (再定義される項目) より大きくてもかまわないのは、再定義される項目が 01 のレベル番号で指定され、外部データ・レコードであると宣言されていない場合だけです。

同じ記憶域を何回でも再定義できます。その記憶域の新しい記述を与える項目は同じセクションになくてはならず、再定義される記憶域の直後になければならず、新しい文字位置を定義する項目を間にはさんではなりません。複数の再定義がある場合、それらはすべてのその記憶域を定義した元の項目のデータ名を使用しなければなりません。例えば、次のとおりです。

```
05    A                     PICTURE 9999.
05    B REDEFINES A         PICTURE 9V999.
05    C REDEFINES A         PICTURE 99V99.
```

再定義する項目 (データ名-1 によって識別される)、および従属する項目には、VALUE 文節を含めてはなりません再定義する記入項目に TYPEDEF 文節を含めることはできません。 再定義する記入項目、再定義される記入項目、および各種の従属記入項目には TYPE 文節を入れてはなりません。

REDEFINES 文節についての考慮事項

ある区域内のデータ項目を、その長さを変えずに再定義できます。例えば、次のとおりです。

```
05  NAME-2.
  10  SALARY                PICTURE XXX.
  10  SO-SEC-NO             PICTURE X(9).
  10  MONTH                 PICTURE XX.
05  NAME-1 REDEFINES NAME-2.
  10  WAGE                  PICTURE XXX.
  10  EMP-NO                PICTURE X(9).
  10  YEAR                  PICTURE XX.
```

データ項目の長さおよびタイプを再定義される区域内で変更することもできます。例えば、次のとおりです。

```
# 05  NAME-2.
#   10  SALARY                PICTURE XXX.
#   10  SO-SEC-NO             PICTURE X(9).
#   10  MONTH                 PICTURE XX.
# 05  NAME-1 REDEFINES NAME-2.
#   10  WAGE                  PICTURE 999V999.
#   10  EMP-NO                PICTURE X(6).
#   10  YEAR                  PICTURE XX.
```

ある区域を再定義しても、その区域のすべての記述は常に有効のままです。つまり、再定義によってどのデータも、消去されたりそれ以前の記述を置き換えたりしません。したがって、B REDEFINES C と指定した場合、2 つのプロシージャー・ステートメント MOVE X TO B と MOVE Y TO C はどちらも、プログラム内の任意のポイントで実行できます。

最初のステートメントの場合、B と記述された区域は X の値と形式をとります。 2 番目のステートメント場合、同じ物理域 (今度は C と記述されます) は Y の値と形式をとります。 2 番目のステートメントが最初のステートメントの直後に実行されると、1 つの記憶域で X の値は Y の値に置き換わります。

再定義するデータ項目の USAGE は、再定義される項目の USAGE と同じである必要はありません。ただし、同じでないときは、既存のデータに何の変化も起こりません。例えば、次のとおりです。

```
05  B                     PICTURE  99 USAGE DISPLAY VALUE 8.
05  C REDEFINES B         PICTURE S99 USAGE COMPUTATIONAL-4.
05  A                     PICTURE S99 USAGE COMPUTATIONAL-4.
```

DISPLAY 値 8 のビット構成は、次のとおりです。

```
1111 0000 1111 1000.
```

B を再定義しても、記憶域内のデータのビット構成は変わりません。したがって、次の 2 つのステートメントを実行した場合、異なる結果になります。

```
ADD B TO A
ADD C TO A
```

前者のステートメントでは、値 8 が A に加算されます (B が USAGE DISPLAY をもっているため)。後者のステートメントでは、記憶域内のビット構成 (2 桁の 10 進数に切り捨てられる) は 2 進数の -48 になり、-48 という値が A に加算されます (C が USAGE COMPUTATIONAL-4 をもっているため)。

前記の例は、再定義を適切に使用しないと、予期しないまたは正しくない結果となることを示します。

コーディング例

REDEFINES 文節は、再定義される区域の範囲内にある項目 (すなわち、再定義される項目に従属する項目) に関して指定できます。例えば、次のとおりです。

```
05  REGULAR-EMPLOYEE.
  10  LOCATION                      PICTURE A(8).
  10  GRADE                         PICTURE X(4).
  10  SEMI-MONTHLY-PAY              PICTURE 9999V99.
  10  WEEKLY-PAY REDEFINES SEMI-MONTHLY-PAY
                                    PICTURE 999V999.
05  TEMPORARY-EMPLOYEE REDEFINES REGULAR-EMPLOYEE.
  10  LOCATION                      PICTURE A(8).
  10  FILLER                        PICTURE X(6).
  10  HOURLY-PAY                    PICTURE 99V99.
```

再定義する項目に従属する項目に関しても、REDEFINES 文節を指定できます。例えば、次のとおりです。

```
05  REGULAR-EMPLOYEE.
  10  LOCATION                      PICTURE A(8).
  10  GRADE                         PICTURE X(4).
  10  SEMI-MONTHLY-PAY              PICTURE 999V999.
05  TEMPORARY-EMPLOYEE REDEFINES REGULAR-EMPLOYEE.
  10  LOCATION                      PICTURE A(8).
  10  FILLER                        PICTURE X(6).
  10  HOURLY-PAY                    PICTURE 99V99.
  10  CODE-H REDEFINES HOURLY-PAY    PICTURE 9999.
```

不確定の結果

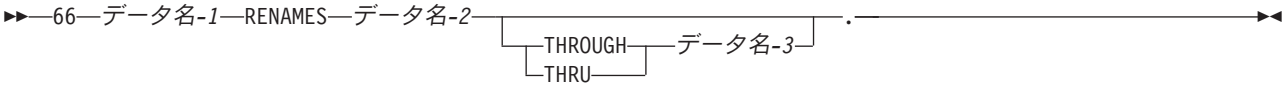
次のときには、不確定の結果が起こることがあります。

- 再定義される項目が再定義する項目に移されたとき (すなわち、B REDEFINES C であり、MOVE B TO C というステートメントを実行したとき)。
- 再定義する項目が再定義される項目に移されたとき (すなわち、B REDEFINES C であり、MOVE C TO B というステートメントを実行したとき)。

RENAMES 文節

RENAMES 文節は、いくつかの基本データ項目の代替グループ化 (重複する可能性がある) を指定します。

RENAMES 文節 - 形式



1 つの論理レコードに対し、1 つまたは複数の RENAMES 記入項目を書くことができます。 1 つの論理レコードに関連する RENAMES 記入項目はすべて、そのレコードの最後のデータ記述記入項目の直後に書かなければなりません。

データ名-1

データ項目の代替グループ化を識別します。

レベル 66 の記入項目は、レベル 01、レベル 77、レベル 88、または別のレベル 66 の記入項目を名前変更することはできません。

データ名-1 を修飾子に使うことはできません。これは、レベル標識記入項目またはレベル 01 記入項目の名前でしか修飾できません。

IBM Extension

データ名-2 が DBCS データ項目を指定し、THROUGH 句が指定されていない場合、データ名-1 は DBCS データ項目を指定できます。

End of IBM Extension

IBM Extension

データ名-2 が国別データ項目を指定し、THROUGH 句が指定されていない場合、データ名-1 は国別データ項目を指定できます。

End of IBM Extension

IBM Extension

データ名-2 が次のいずれかのデータ項目を参照し、かつ THROUGH 句が指定されていない場合、データ名-1 は次のいずれかのタイプのデータ項目になります。

- DBCS
- 国別
- ポインターまたはプロシージャー・ポインター
- 内部浮動小数点または外部浮動小数点
- 日付、時刻、またはタイム・スタンプ

End of IBM Extension

データ名-2、データ名-3

基本データ項目の元のグループ化を識別します。つまりそれらの項目は、関連したレベル 01 記入項目内の基本項目またはグループ項目でなければならず、同じデータ名であってはなりません。データ名は双方とも修飾されていてもかまいません。

データ名-2 とデータ名-3 のデータ記入項目、またはそれらが従属するグループ記入項目に対するデータ記入項目の中では、OCCURS 文節を指定してはなりません。また、データ名-2 とデータ名-3 の間で定義されたどの項目に対しても、OCCURS DEPENDING ON 文節を指定してはなりません。

IBM Extension

TYPE 文節は、データ名-2、データ名-3 のデータ記述内で指定してはなりません。同様に、データ名-2 とデータ名-3 の間で定義された項目またはこれらの項目に従属するどの項目でも指定してはなりません。データ名-2、データ名-3、またはデータ名-2 とデータ名-3 の間で定義された項目が TYPE 文節を使用して定義したグループ項目に従属している場合は、データ名-1 はその同じグループ項目に従属し

ていなければなりません。

End of IBM Extension

データ名-3 を指定すると、データ名-1 は、次のすべての基本項目を含むグループ項目として扱われます。

- データ名-2 (これが基本項目の場合)、またはデータ名-2 (これがグループ項目の場合) の中の最初の基本項目で始まる。
- データ名-3 (これが基本項目の場合)、またはデータ名-3 (これがグループ項目の場合) の最後の基本項目で終わる。

データ名-3 内の左端の文字は、データ名-2 の左端の文字の前にあってはなりません。データ名-3 の右端の文字は、データ名-2 の右端の文字の後に続かなければなりません。このことは、データ名-3 がデータ名-2 に従属することはできないことを意味します。

データ名-3 を指定しないと、データ名-2 のすべてのデータ属性がデータ名-1 のデータ属性になります。つまり、以下のとおりです。

- データ名-2 がグループ項目のときは、データ名-1 はグループ項目として扱われます。
- データ名-2 が基本項目のときは、データ名-1 は基本項目として扱われます。

6-81 ページの図 6-6 に RENAMES 文節の有効な指定と無効な指定を図示します。

有効および無効な RENAMES 文節の指定の図示

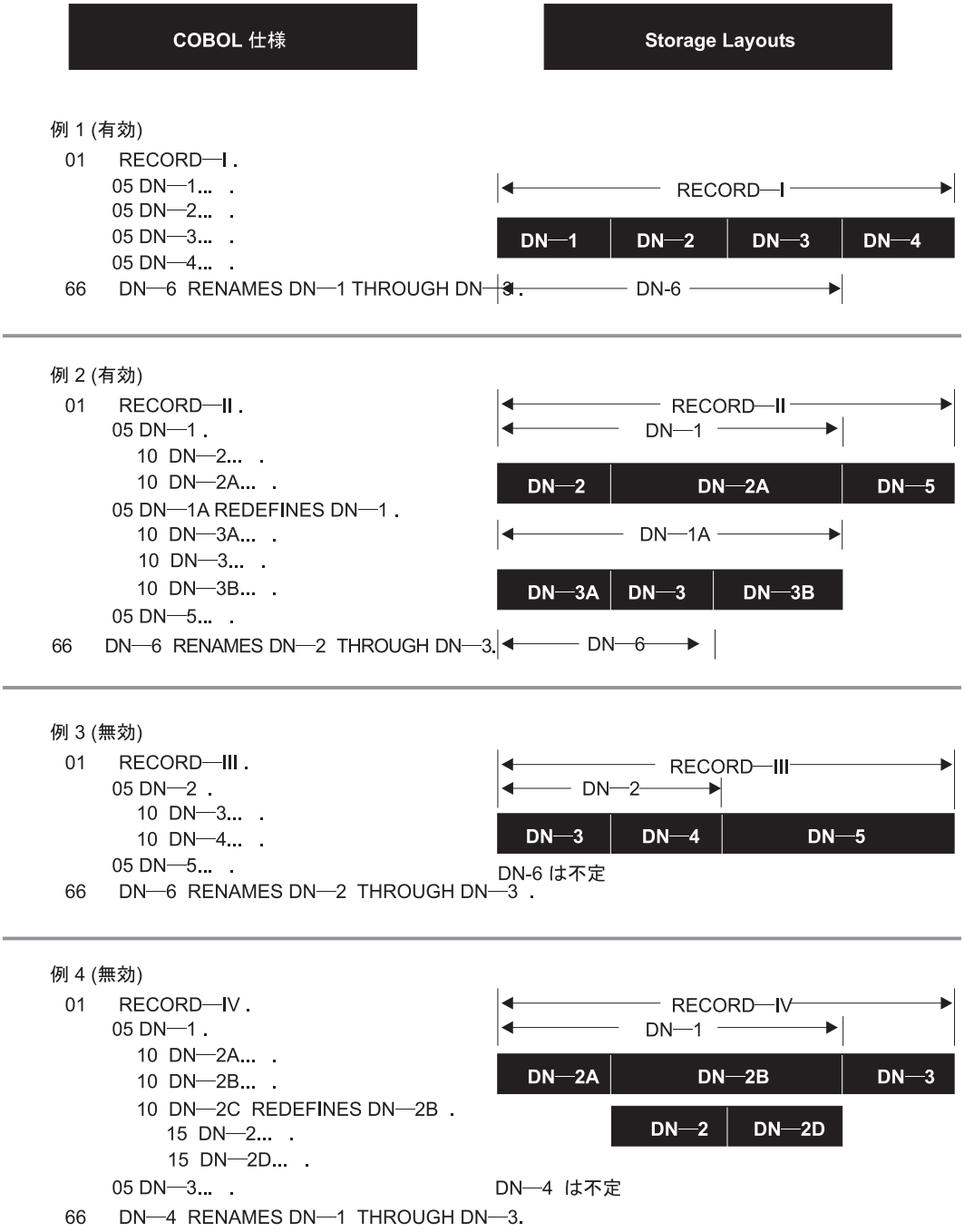


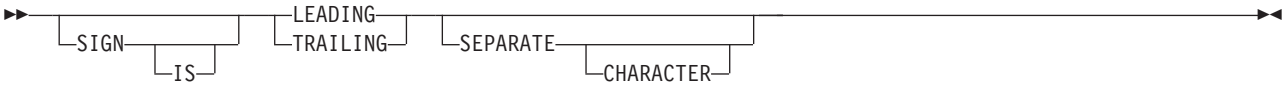
図 6-6. RENAMES 文節 — 有効な指定と無効な指定

SIGN 文節

SIGN 文節は、数字記入項目のための、演算符号の表現の位置とモードを指定します。

SIGN 文節

SIGN 文節 - 形式



SIGN 文節は、符号付き数字データ記述記入項目 (すなわち、PICTURE 文字ストリングに S が含まれている記入項目)、またはその種の基本記入項目を少なくとも 1 つ含んでいるグループ項目に対してだけ指定できます。 USAGE IS DISPLAY または USAGE IS NATIONAL を明示的にまたは暗黙のうちに指定する必要があります。

SEPARATE 句を指定しないで SIGN 節を指定する場合、USAGE DISPLAY を明示的または暗黙的に指定する必要があります。SIGN IS SEPARATE を指定している場合は、USAGE DISPLAY または USAGE NATIONAL のいずれかを指定できます。

SIGN 文節が必要であるのは、演算符号の性質や位置を明示的に記述する必要がある場合だけです。

SIGN 文節 (指定するならば) は、演算符号が適用される数字データ記述記入項目、または演算符号が適用されるグループに従属するそれぞれの符号付き数字データ記述記入項目に関して、演算符号の位置と表示モードを定義します。

SIGN 文節が指定されるグループ項目に従属する、グループ記入項目または基本記入項目のいずれかに SIGN 文節を指定した場合、従属記入項目の SIGN 文節は、従属記入項目に優先します。

FD 記入項目に CODE-SET 文節を指定した場合、そのファイル記述記入項目に関連する符号付き数字データ記述記入項目は、SIGN IS SEPARATE 文節で記述しなければなりません。

記号 S を含む PICTURE 文字ストリングを持つ数字データ記述記入項目はどれも、符号付き数字データ記述記入項目です。そのような記入項目に対して SIGN 文節も指定されており、しかも計算や比較の際に変換が必要であれば、変換は自動的に行われます。

IBM Extension

SIGN 文節は、外部浮動小数点項目の文書として扱われます。内部浮動小数点項目では、SIGN 文節は無効です。

SIGN 文節は、FORMAT 文節が指定されている場合には指定できません。

TYPE 文節は SIGN 文節と同じデータ記述記入項目に指定することはできません。

End of IBM Extension

SEPARATE CHARACTER

SEPARATE CHARACTER 句を指定しない 場合は、次のようになります。

- 演算符号は、基本数字データ項目の LEADING または TRAILING 文字位置 (いずれか、指定されている方) に関連付けられているものと見なされます。(この場合、SIGN IS TRAILING を指定することは、コンパイラーによる標準の処理と同等です。)
- PICTURE 文字ストリング内の文字 S は、データ項目のサイズ (標準データ・フォーマットの文字数) を判別する際にはカウントされません。

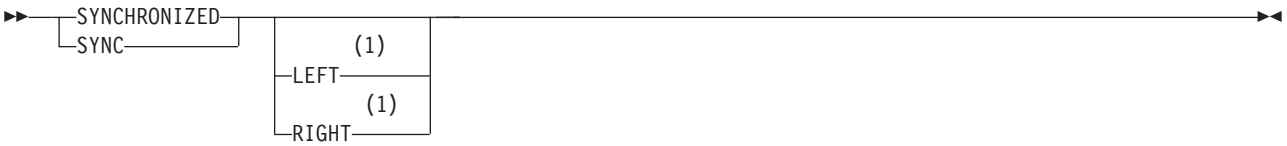
SEPARATE CHARACTER 句を指定する 場合は、次のようになります。

- 算術演算符号は、基本数字データ項目の LEADING または TRAILING 文字位置 (いずれか、指定されている方) に関連付けられているものと見なされます。この文字位置は数字位置ではありません。
- PICTURE 文字ストリング内の文字 S は、データ項目のサイズ (標準データ・フォーマットの文字数) を判別する際にカウントされます。
- + は、正の演算符号用に使われる文字です。
- - は、負の演算符号用に使用される文字です。

SYNCHRONIZED 文節

SYNCHRONIZED 文節は、基本項目をストレージ内の正しい位置に合わせるように指定します。SYNCHRONIZED 文節を使用するには、CRTCBLMOD または CRTBNDCBL コマンドに *SYNC コンパイラー・オプションを指定します。

SYNCHRONIZED 文節 - 形式



注:

1 構文検査だけ行われます。

指定した場合、LEFT 句と RIGHT 句は構文検査されますが、プログラムの実行には何の影響も与えません。

同期化を指定しないと、データは充てん文字スペースなしに連続して配置されます。同期化を指定すると、データは、1、2、4、8、または 16 バイト (許される場合。 6-85 ページの表 6-9 を参照) で整数除算できるアドレスに沿って位置合わせされます。その前のデータ項目が各境界間のすべてのバイトを使いきっていなければ、同期化の指定には、充てん文字スペースを (暗黙に) 使用する必要があるかもしれません。

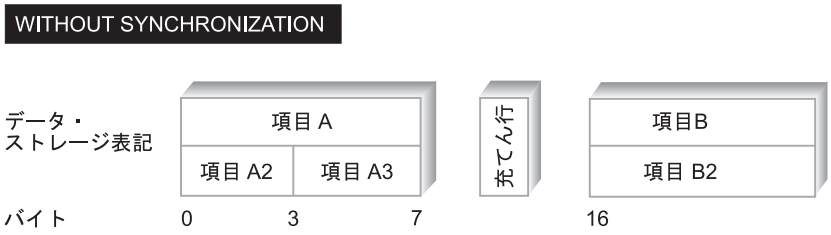
同期化データの利点

データを同期させるとどのような利点があるのでしょうか。データを同期した場合の利点として、アクセス可能度の点でパフォーマンスが向上します。欠点は、レコード・サイズが大きくなる (充てん文字スペースがレコードの一部を占める) ために一部のストレージが無駄になることです。

同期化を指定してもしなくても、01 レベルの項目とポインターは、常に 16 バイトの境界上に位置合わせされます。同期化は、基本項目にしか指定できません。グループ項目には、この文節を使うことはできません。

6-84 ページの図 6-7 はこの概念を表したものです。

```
01      A
   05       A2  PIC  X(3)
   05       A3  PIC  9(5)  BINARY
01      B
   05       B2  PIC  X(16)
```



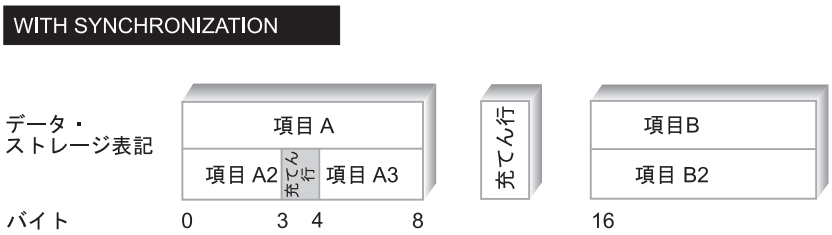


図 6-7. 同期化したときとしないときのデータ・ストレージ図

図 6-7 は、A と B は常に 16 バイト境界に位置合わせされることを示しています。同期化を指定しないと、A2 と A3 は、サイズに関係なく連続して保管されます。同期化を指定すると、4 バイト境界が選択され (A3 のタイプが原因で)、A3 はそれに応じて位置合わせされます。 A2 と A3 の間には、1 バイトの充てん文字があります。しかし A3 はより迅速にアクセスされるようになります。

同期化とオフセット: 上図では、A と B は、実際のストレージ内で互いに連続しているとは限らないことに注意してください。言い替えると、B は A の開始点から 16 バイト後、または 48 バイト (16 x 3) 後、または 16 x N バイト後のどれで開始するかユーザーには分からないということです。オフセットを指定することによって、同期化データを取り出そうとしてはなりません。

IBM Extension

SYNCHRONIZED 文節は、ポインター・データ項目とプロシージャー・ポインター・データ項目に対しては暗黙指定されます。リンケージ・セクションで宣言されたポインター・データ項目とプロシージャー・ポインター・データ項目は同期化されません。

SYNCHRONIZED 文節は TYPE 文節と同じデータ記述記入項目に指定することはできません。

DBCS、国別、外部浮動小数点、日付、時刻、またはタイム・スタンプのデータ項目に対する SYNCHRONIZED 文節は無視されます。

COMPUTATIONAL-1 データ項目の SYNCHRONIZED 文節は、フルワード境界でデータの位置合わせを行います。

COMPUTATIONAL-2 データ項目の SYNCHRONIZED 文節は、ダブルワード境界でデータの位置合わせを行います。

End of IBM Extension

項目に指定される USAGE によっては、SYNCHRONIZED 文節が特定の効果を持つ場合があります。表 6-9 に、ある項目の USAGE でどのように SYNCHRONIZED 文節がそれに影響を与えるかを示しています。

表 6-9. データ項目 USAGE および SYNCHRONIZED 文節

USAGE の指定	**SYNCHRONIZED 文節の処理**
DISPLAY	構文検査されますが、実行には影響しません
DISPLAY-1 (DBCS)	無視されます
NATIONAL	無視されます
PACKED-DECIMAL	構文検査されますが、実行には影響しません
COMPUTATIONAL-1	フルワード境界でデータの位置合わせをします
COMPUTATIONAL-2	ダブルワード境界でデータの位置合わせをします
COMPUTATIONAL-3	構文検査されますが、実行には影響しません
BINARY: PIC S9(1) ～ PIC S9(4)	レコードの先頭から 2 の倍数位置にデータ項目を位置合わせします
BINARY: PIC S9(5) ～ PIC S9(9)	レコードの先頭から 4 の倍数位置にデータ項目を位置合わせします
BINARY: PIC S9(10) ～ PIC S9(18)	レコードの先頭から 8 の倍数位置にデータ項目を位置合わせします
COMPUTATIONAL-4	USAGE BINARY の場合と同様に機能します
# COMPUTATIONAL-5	USAGE BINARY の場合と同様に機能します
COMPUTATIONAL	構文検査されますが、実行には影響しません
INDEX	使えません
POINTER	レコードの先頭から 16 の倍数の位置にデータ項目を位置合わせします
PROCEDURE-POINTER	USAGE POINTER の場合と同様に機能します

基本項目の長さが、SYNCHRONIZED 文節によって影響を受けることはありません。

OCCURS 文節を使った SYNCHRONIZED 文節の指定

OCCURS 文節の有効範囲内にある項目に SYNCHRONIZED 文節を指定すると、その項目が現れるごとに同期化が行われます。

REDEFINES 文節を使った SYNCHRONIZED 文節の指定

REDEFINES 文節が入っている項目に SYNCHRONIZED 文節を指定する場合には、再定義されるデータ項目には、再定義するデータ項目の正しい境界位置合わせが入っていなければなりません。 REDEFINES 文節の入っている項目には、埋め込み文字は追加されません。例えば、次のように書いた場合、データ項目 A はレコードの始まりからの位置が 4 の倍数バイトから始まらなければなりません。

```
02 A                    PICTURE X(4).
02 B REDEFINES A        PICTURE S9(9) BINARY SYNC.
```

REDEFINES 文節が入っている項目に従属している最初の基本項目である 2 進数項目に SYNCHRONIZED 文節を指定する場合には、その項目は、未使用の文字位置の追加を必要とするものであってはなりません。

FILLER 項目

FILLER 項目は、前の項目と同じレベル番号を持つ項目であるかのように扱われます。この暗黙の FILLER 項目のサイズは、次のように計算されます。

- 位置合わせされた項目の前にあるすべての基本データ項目によって占有される文字の総数が、前に追加された暗黙の FILLER 項目とともに合計されます。
- この合計が、上記の位置合わせの計算で乗数 (2、4、8、または 16) として使用された係数 m で除算されます。
- この除算の剰余 r がゼロである場合には、暗黙の FILLER 項目は必要ありません。剰余がゼロでない場合には、暗黙の FILLER 項目のサイズは m - r となります。

暗黙の FILLER 項目のサイズは、それが入っているグループ項目のサイズには含まれません。

グループ項目は通常、英数字として定義されます。すべての FILLER 項目はスペースで初期設定されます。その後、SYNCHRONIZED 文節を介して生成される暗黙の FILLER 項目も、(デフォルトの) *STDINZ コンパイラー・オプションのもとでスペースを使って初期設定されます。 *NOSTDINZ または STDINZHEX00 オプションのもとでは、これら暗黙の FILLER 項目に 16 進数のゼロが含まれることになります。

グループ項目が、OCCURS 文節で定義され、位置合わせされることになっているデータ項目が入っている場合には、暗黙の FILLER 項目も、コンパイラーによって追加される場合があります。暗黙の FILLER を追加するかどうかを決定するために、次の処置がとられます。

- 必要なすべての暗黙 FILLER 項目を含めて、コンパイラーはグループ項目のサイズを計算します。
- この合計が、グループ内の任意の基本項目で必要とされる最大の m によって除算されます。
- r がゼロの場合には、暗黙の FILLER 項目は必要ありません。 r がゼロでない場合には、サイズが m - r の暗黙の FILLER 項目を追加しなければなりません。

暗黙の FILLER 項目は、OCCUR 文節が入っているグループ項目が現れるたびに、その終わりに挿入できます。これは、後続のグループ項目のオカレンスを同期させるために行うものです。

レベル 01 または 77 の項目は、次のような規則に従って位置合わせされます。

区域	レベル番号	境界合わせ
作業用ストレージ・セクション	01 77	16 バイト 16 バイト
ローカル・ストレージ・セクション	01 77	16 バイト 16 バイト
ファイル・セクション	01	コンパイラーは、項目を同期化するために 16 バイト境界を想定します。
リンケージ・セクション	01 77	コンパイラーは、項目を同期化するために 16 バイト境界を想定します。ポインター・データ項目とプロシージャー・ポインター・データ項目は同期されません。

暗黙 FILLER の例

次の COBOL データ記述によって 6-87 ページの図 6-8 に示されるコンピューター・ストレージが割り振られることになります。

```
01 UNSYNCHRONIZED-RECORD
  02 UNSYNCHRONIZED-DATA-1 PIC 9(3) DISPLAY.
  02 UNSYNCHRONIZED-DATA-2 PIC X(2).
01 COMPOUND-REPEATED-RECORD.
  02 ELEMENTARY-ITEM-1 PIC X (2).
  02 GROUP-ITEM OCCURS 3 TIMES.
    03 ELEMENTARY-ITEM-2 PIC X.
    03 ELEMENTARY-ITEM-3 PIC S9(2) BINARY SYNC.
    03 ELEMENTARY-ITEM-4 PIC S9(4) V9(2) BINARY SYNC.
    03 ELEMENTARY-ITEM-5 PIC X (5).
```

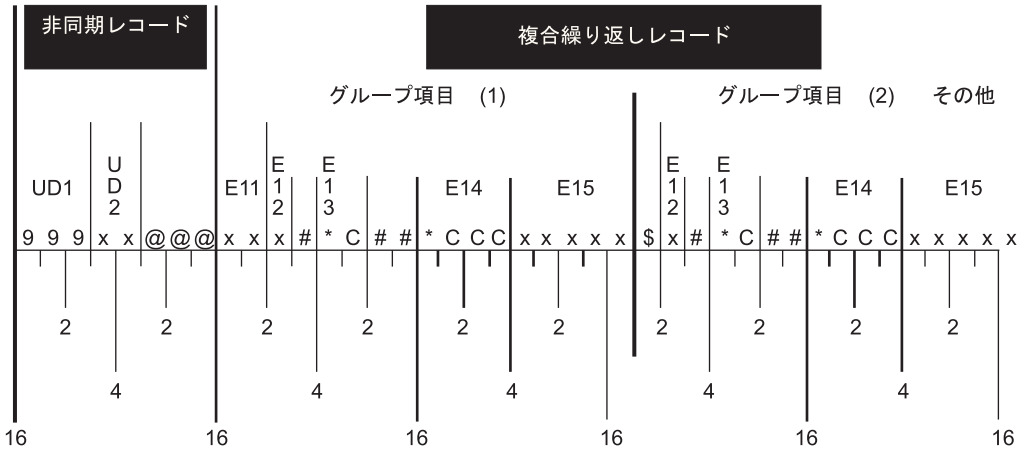


図 6-8. コンピューター・ストレージの割り振り

@ 自動同期化またはレコード (01 レベル) 記述のために割り振られる暗黙の FILLER バイトを示します。

後続のデータ項目が明示的に同期化されるときに割り振られる暗黙 FILLER バイトを示します。

***** 同期化された BINARY 項目の最初のバイトを示します。

$ 非基本項目が OCCURS 文節に従属しているときに割り振られる暗黙 FILLER バイトを示します。

9 数字の DISPLAY 文字に割り振られるバイト数を示します。

X 英数字の DISPLAY 文字に割り振られるバイト数を示します。

C BINARY データ・ストレージに割り振られるバイト数を示します。

TYPE 文節

IBM Extension

TYPE 文節は、記入項目のサブジェクトのデータ記述がユーザー定義データ・タイプによって指定されることを指示します。ユーザー定義データ・タイプは TYPEDEF 文節を使用して定義します。これについては 6-88 ページの『TYPEDEF 文節』で説明しています。

以下の一般規則が適用されます。

- (TYPEDEF 文節を使用して定義した) タイプ名-1 がグループ項目を記述している場合は、TYPE 文節のサブジェクトはグループ項目となり、その従属エレメントはタイプ名-1 の従属エレメントと同じ名前、記述、および階層をもちます。

  注: TYPE 文節のサブジェクトが持つことができる最高レベル番号は 49 であり、さらにタイプ名-1 はグループ項目である場合に最大 49 のレベルを持つことができるため、この階層のレベル数が 49 を超えることがあります。タイプ名の記述では他のタイプ名の参照が認められているため、この階層のレベル数には事実上制限はありません。
- TYPE 文節のサブジェクトのデータ記述内に VALUE 文節を指定すると、この記入項目に関しては、タイプ名-1 の記述内に指定されている VALUE 文節はすべて無視されます。
- タイプ名の有効範囲に関する規則は、データ名の有効範囲に関する規則と同様です。
- TYPE 文節のサブジェクトとなっている基本項目を参照変更することはできません。
- (タイプ名-1 を参照する) TYPE 文節のサブジェクトを参照する LIKE 文節または TYPE 文節のサブジェクトが従属しているグループ項目をタイプ名-1 の記述に入れることはできません。この規則はタイプ名-1 の従属データ項目についても同様です。
- (タイプ名-1 を参照する) TYPE 文節がそのサブジェクトの従属先のレコードを参照する場合は、その TYPE 文節をタイプ名-1 の記述に入れることはできません。この規則はタイプ名-1 の従属データ項目についても同様です。

  例えば、A は TYPEDEF 文節を使用して定義したグループ項目であるとします。 B も同様に TYPEDEF 文節を使用して定義したグループ項目であるが、B は TYPE A の従属項目も含んでいるとします。このような場合、A のタイプ定義に TYPE B の項目を含めることはできません。
- TYPE 文節のサブジェクトの名前は、全体でも一部でも変更することはできません。
- TYPE 文節のサブジェクトは、明示的にも暗黙的にも再定義することはできません。
- TYPE 文節のサブジェクトがグループ項目に従属している場合は、そのグループ項目のデータ記述に USAGE 文節を含めることはできません。
- TYPE 文節は、文節 BLANK、WHEN ZERO、FORMAT、JUSTIFIED、LIKE、PICTURE、REDEFINES、RENAMES、SIGN、SYNCHRONIZED、または USAGE が指定されているデータ記述記入項目内で使用することはできません。
- TYPE 文節は、文節 EXTERNAL、GLOBAL、OCCURS、TYPEDEF、および VALUE が指定されているデータ記述記入項目内に指定することができます。

TYPE 文節および TYPEDEF 文節の使用について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

End of IBM Extension

TYPEDEF 文節

IBM Extension

TYPEDEF 文節は、新規の**ユーザー定義データ・タイプ** (タイプ名) を作成します。この新しいユーザー定義データ・タイプの名前が TYPEDEF 文節のサブジェクトです。データ名-1 は TYPEDEF 文節を使用して指定する必要があります。したがって、FILLER は使用できません。 TYPEDEF 文節は、データ名-1 の直後に続けて指定する必要があります。 TYPEDEF 文節で新規データ・タイプを定義した後は、TYPE 文節を使用してデータ項目をこの新規データ・タイプとして宣言できます。 TYPE 文節の詳細は 6-87 ページの『TYPE 文節』を参照してください。

TYPEDEF 文節を指定できるのはレベル 01 の記入項目に対してだけです。対象とする記入項目がグループ項目であっても構いません。グループ項目を指定する場合、そのグループのすべての従属項目はタイプ宣言の一部となります。タイプ宣言用にストレージが割り振られることはありません。

TYPEDEF 文節は、以下の文節と同じデータ記述記入項目に指定することはできません。

- EXTERNAL
- REDEFINES
- LIKE

上記以外のすべてのデータ記述文節は、指定すると、(TYPE 文節内で) ユーザー定義データ・タイプを使用して定義したあらゆるデータ項目に対して指定されたものと想定されます。

TYPEDEF は、複合 OCCURS DEPENDING ON とともに使用することはできません。これは、TYPEDEF の一部であるテーブル内では、OCCURS DEPENDING ON 文節を指定できないことを意味します。詳細については 9-46 ページの『付録 H. 複合 OCCURS DEPENDING ON』を参照してください。

TYPEDEF 文節を指定できるのは、プログラムの作業用ストレージ・セクション、ローカル・ストレージ・セクション、リンケージ・セクション、またはファイル・セクションの中だけです。

TYPE 文節は TYPEDEF 文節と同じデータ記述記入項目に指定することができます。

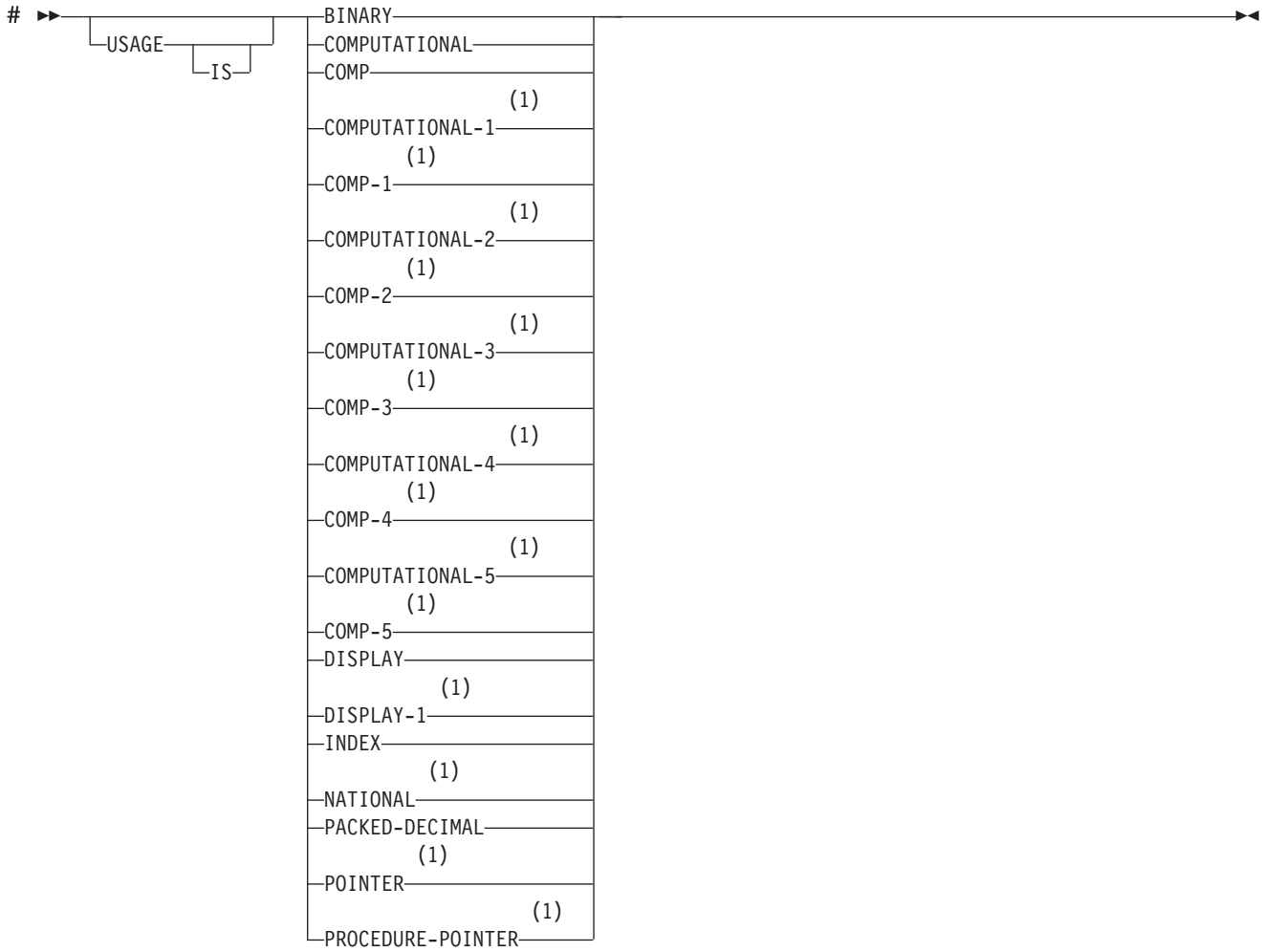
End of IBM Extension

USAGE 文節

USAGE 文節は、データがストレージ内で表現される形式を指定します。この形式は、ある特定の手続き部ステートメントが使用される場合には、制約を受けることがあります。

USAGE 文節 - 形式

USAGE 文節



注:

1 IBM 拡張

次の表は、USAGE 文節で指定するさまざまなデータ項目に使用する句について概説しています。

表 6-10. USAGE 文節のデータ項目

データ項目	**USAGE 文節の句**
2 進数 (計算用項目)	BINARY または COMPUTATIONAL-4[1] または COMP-4[1]
# ネイティブ 2 進数 (計算用項目)	COMPUTATIONAL-5[1] または COMP-5[1]
パック 10 進数 / 内部 10 進数 (計算用項目)	PACKED-DECIMAL または COMPUTATIONAL または COMP または COMPUTATIONAL-3[1] または COMP-3[1]
内部浮動小数点 (計算用項目)	COMPUTATIONAL-1[1] または COMP-1[1] (4 バイト) COMPUTATIONAL-2[1] または COMP-2[1] (8 バイト)

表 6-10. USAGE 文節のデータ項目 (続き)

データ項目	USAGE 文節の句
数字 DISPLAY 項目 外部 10 進数 (ゾーン 10 進数) 外部浮動小数点[1]	DISPLAY
非数字 DISPLAY 項目 英字 英数字 英数字編集 数字編集項目 ブール[1] 日付、時刻、およびタイム・スタンプ [1]	DISPLAY
DBCS[1] DBCS 編集[1]	DISPLAY-1
国別[1]	NATIONAL
指標	INDEX
ポインター[1]	POINTER
プロシージャー・ポインター[1]	PROCEDURE-POINTER
注: 1. IBM 拡張	

USAGE 文節は、どのレベル (66 または 88 を除く) の記入項目に対しても指定できます。ただし、グループ・レベルで指定した場合は、グループそのものにではなく、そのグループ内の各基本項目にこの文節が適用されます。基本項目の使用法は、その基本項目が属するグループ項目に指定された使用法に矛盾するものであってはなりません。

USAGE 文節をグループ・レベルか基本レベルのいずれかで指定しなければ使用状況は DISPLAY と見なされます。

IBM Extension

TYPE 文節と USAGE 文節とを同じデータ記述記入項目に指定することはできません。

TYPE 文節が指定されているデータ記述記入項目は、USAGE 文節を含んでいるデータ記述記入項目に従属することはできません。例えば、以下の記述は誤りです。

```
01 FLAGS   USAGE   DISPLAY.
   05 F-STATUS    TYPE CHAR.
   05 FLAG-ACTIVE TYPE CHAR.
```

End of IBM Extension

計算用項目

計算用項目は、算術演算で使用される値です。計算用項目は数字でなければなりません。これには、2 進数、パック 10 進数、および内部浮動小数点のデータ項目が入ります。

USAGE 文節

グループ項目の USAGE が計算用項目のいずれかを指定して記述されている場合、その使用状況を持つのはグループ内の基本項目です。グループそれ自体は非数字と見なされるため、数値演算に使用することはできません。ただし、CORRESPONDING 句を使用するグループの場合は例外です (7-31 ページの『CORRESPONDING 句』を参照してください)。

計算用項目の最大の長さは、10 進数で 18 桁です。

IBM Extension

パック 10 進数の計算用項目の最大の長さは、10 進数で 31 桁です。

End of IBM Extension

計算用項目の PICTURE に含めることができるのは、次のものに限ります。

9 1 つまたは複数の数字位置

S 1 つの演算符号

V 1 つの暗黙の小数点

P 1 つまたは複数の 10 進数位取り位置

IBM Extension

他の計算用項目とは異なり、COMPUTATIONAL-1 および COMPUTATIONAL-2 項目 (内部浮動小数点) には PICTURE ストリングを入れることができません。

End of IBM Extension

BINARY 句

BINARY 句は 2 進数データ項目に指定されます。この項目には、10 進数の数字 0 ～ 9 と符号から構成された 10 進数と同等の 10 進数を入れることができます。

2 進数項目によって占有される記憶容量は、その PICTURE 文節で定義される 10 進数の桁数によって異なります。

PICTURE 文節の桁数
占有ストレージ

1 ～ 4 2 バイト

5 ～ 9 4 バイト

10 ～ 18
8 バイト

記憶域の左端のビットが演算符号です。

PACKED-DECIMAL 句

PACKED-DECIMAL 句は内部 10 進数項目に対して指定されます。この項目は、ストレージの中ではパック 10 進数形式で表されます。各文字位置に 2 桁ずつ入りますが、後書き文字位置には、後続の文字位置をのぞいて、低位の数字と符号が入ります。そのような項目には、10 進数で 18 桁を超えない値を表す、0 ～ 9 までの数字と符号を入れることができます。符号の表記は 6-97 ページの図 6-9 に示されています。

IBM Extension

パック 10 進数の計算用項目の最大の長さは、10 進数で 63 桁です。

PACKED-DECIMAL は、FORMAT リテラルの内容が変換指定子だけの日付項目および時刻項目にも指定できます。これらの変換指定子に数字以外のものを入れてはなりません。

End of IBM Extension

COMPUTATIONAL または COMP 句

COMPUTATIONAL または COMP 句は内部 10 進数項目に対して指定されます。この項目はストレージの中で 1 バイトに 2 桁が入り、右端バイトの右端 4 ビットに符号が入れられます。内部 10 進項目には、0 から 9 の数字と符号を入れることができます。内部 10 進項目の PICTURE に S が入っていない場合には、符号位置は正と解釈されるビット構成により占有されます。 USAGE COMP は、操作上のパフォーマンスという点では、すべての USAGE の中で最も効率的です。

ILE COBOL コンパイラーの場合、COMPUTATIONAL 句は以下と同義です。

- オプション COMPASBIN が指定されている場合には、USAGE COMP-4 (2 進数)
- その他の場合には、PACKED-DECIMAL

COMPUTATIONAL-1 または COMP-1 句

IBM Extension

COMPUTATIONAL-1 または COMP-1 句は内部浮動小数点項目 (単精度) に対して指定されます。
COMP-1 項目は 4 バイト長です。符号は左端バイトの最初のビットに入っており、指数は次の 8 ビットに入っています。最後の 23 ビットには、仮数が入ります。条件式の場合、クラス条件を COMP-1 または COMPUTATIONAL-1 の内部浮動小数点データ項目に使用することはできません。

End of IBM Extension

COMPUTATIONAL-2 または COMP-2 句

IBM Extension

COMPUTATIONAL-2 または COMP-2 句は内部浮動小数点項目 (倍精度) に対して指定されます。
COMP-2 項目は 8 バイト長です。符号は左端バイトの最初のビットに入っており、指数は次の 11 ビットに入っています。残りの 52 ビットには仮数が入ります。条件式の場合、クラス条件を
COMPUTATIONAL-2 または COMP-2 の内部浮動小数点データ項目に使用することはできません。

End of IBM Extension

COMPUTATIONAL-3 または COMP-3 句 (内部 10 進数)

IBM Extension

これは、PACKED-DECIMAL と同じです。

コンパイル・パフォーマンスを向上させるには、COMP-3 (パック 10 進数) 項目の PICTURE 文節に奇数の数字位置を指定します。内部的には、パック 10 進数項目の右端のバイトには、数字が 1 つと符号が 1

つ入り、それ以外のバイトには数字が 2 つずつ入ります。よりパフォーマンスの良い構成を使用している場合には、脱落している数字をコンパイラーから供給する必要はなくなります。

桁数が偶数であるパック 10 進数項目に割り振られるストレージの左端 (未使用) 数字位置の内容は、その項目の値が変更されるときに変更可能です。したがって、例えばこの項目が再定義されたり、グループ・フィールドの一部を形成したり、索引付きファイルに対するキーとして使用される場合、予期しない値が含まれている可能性があります。

End of IBM Extension

COMPUTATIONAL-4 または COMP-4 句 (2 進数)

IBM Extension

これは、BINARY と同じです。

End of IBM Extension

COMPUTATIONAL-5 または COMP-5 句 (2 進数)

IBM Extension

このタイプのデータ項目は、ストレージで 2 進データとして表されます。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。

*NOSTDTRUNC コンパイラー・オプションまたは NOSTDTRUNC PROCESS オプションを使用すると、
BINARY データ項目 (USAGE BINARY、COMP-4) は、USAGE COMP-5 を宣言しているかのように処理
されます。唯一異なる点として、符号なし BINARY データ項目には常に符号ビットが含まれるため、符号
なし BINARY データ項目の最大値および最小値は、符号付き BINARY データ項目の場合と同じになりま
す。

下表に、USAGE COMP-5 で記述されたデータ項目のいくつかのピクチャー文字ストリング、結果のストレ
ージ表現、および値の範囲を示します。

表 6-11. COMP-5 データ項目のストレージ表現

#	ピクチャー	ストレージ表現	数値
#	S9(1) ～ S9(4)	2 進数ハーフワード (2 バイト)	-32768 ～ +32767
#	S9(5) ～ S9(9)	2 進数フルワード (4 バイト)	-2,147,483,648 ～ +2,147,483,647
# #	S9(10) ～ S9(18)	2 進数ダブルワード (8 バイト)	-9,223,372,036,854,775,808 ～ +9,223,372,036,854,775,807
#	9(1) ～ 9(4)	2 進数ハーフワード (2 バイト)	0 ～ 65535
#	9(5) ～ 9(9)	2 進数フルワード (4 バイト)	0 ～ 4,294,967,295
#	9(10) ～ 9(18)	2 進数ダブルワード (8 バイト)	0 ～ 18,446,744,073,709,551,615

#

COMP-5 データ項目のピクチャーは、位取り係数 (つまり、小数点以下の桁数または暗黙の整数位置) を指
定できます。この場合、上の表にリストされた最大容量は、適切に位取りする必要があります。例えば、
PICTURE S99V99 COMP-5 で記述されたデータ項目は、ストレージで 2 進数ハーフワードとして表さ
れ、 -327.68 から +327.67 の範囲の値をサポートします。

使用上の注意: ON SIZE ERROR 句を算術ステートメントで使用した場合にレシーバーが USAGE
COMP-5 で定義されていると、レシーバーに入れることができる最大値は、項目の 10 進 PICTURE 文字
ストリングによって暗黙指定される値になります。この最大値を超える値を格納しようとすると、サイズ・
エラー状態が生じます。

End of IBM Extension

DISPLAY 句

データ項目は文字形式 (8 ビットのバイトに対して 1 文字) で保管されます。これは印刷出力の際に用いられる形式に対応します。 DISPLAY は明示的または暗黙に指定できます。

USAGE IS DISPLAY は、次のタイプの項目に対して有効です。

- 英字
- 英数字
- 英数字編集項目
- 数字編集
- 外部 10 進数 (数字) 項目

IBM Extension

- ブール
- 日付、時刻、およびタイム・スタンプの項目
- 外部浮動小数点項目

End of IBM Extension

IBM Extension

条件式の場合、クラス条件は、USAGE DISPLAY を持つ外部浮動小数点データ項目で使用することはできません。

End of IBM Extension

英字、英数字、英数字編集、および数字編集の項目については 6-64 ページの『データ・カテゴリーと PICTURE の規則』で説明しています。

ゾーン項目の PICTURE 文字ストリングに入れることのできるものは、9、算術演算符号記号 S、仮想小数点 V、および 1 つまたは複数の P だけです。

外部 10 進数 (数字): 外部 10 進数項目は、**ゾーン 10 進数項目**とも呼ばれます。数の各桁は 1 バイトで表現されます。各バイトの高位 4 ビットは、ゾーン・ビットです。下位バイトの高位 4 ビットは、その項目の符号を表します。数字が正の場合には、これらの 4 ビットには 16 進数 F が入ります。数字が負の場合には、これらの 4 ビットに 16 進数 D が入ります。各バイトの下位 4 ビットにはその桁の値が入ります。

外部 10 進数項目の最大の長さは 18 桁です。

IBM Extension

外部 10 進数項目の最大の長さは 63 桁です。

End of IBM Extension

外部浮動小数点 (数字):

IBM Extension

外部浮動小数点項目は、次の形式の文字ストリングです。

仮数符号
+ または - (必須)

仮数 1 ～ 16 桁の長さの数値。明示的に小数点を示す場合はピリオド (.) を、暗黙的に小数点を示す場合は V のいずれかを含む場合があります。小数点の記号は、仮数の任意の位置に入れることができます。

E 指数が入った定数。

指数符号
+ または - (必須)

指数 2 または 3 桁の数値。

End of IBM Extension

数字項目の内部表現: 6-97 ページの図 6-9 は、USAGE 文節で指定される数字項目の内部表現を示しています。数字 DISPLAY 項目には、外部 10 進数および外部浮動小数点のデータ項目が入ります。また、計算用数字項目 (2 進数、内部 10 進数、および内部浮動小数点のデータ項目) もこの図に示しています。

ITEM	DESCRIPTION	VALUE	INTERNAL REPRESENTATION*
外部 10 進数	PIC S9999 DISPLAY	+1234 —1234 1234	F1 F2 F3 F4 F1 F2 F3 D4 F1 F2 F3 F4
	PIC 9999 DISPLAY	+1234 —1234 1234	F1 F2 F3 F4 F1 F2 F3 F4 F1 F2 F3 F4
	PIC S9999 DISPLAY SIGN LEADING	+1234 —1234 1234	F1 F2 F3 F4 D1 F2 F3 F4 F1 F2 F3 F4
	PIC S9999 DISPLAY SIGN TRAILING SEPARATE	+1234 —1234 1234	F1 F2 F3 F4 4E F1 F2 F3 F4 60 F1 F2 F3 F4 4E
	PIC S9999 DISPLAY SIGN LEADING SEPARATE	+1234 —1234 1234	4E F1 F2 F3 F4 60 F1 F2 F3 F4 4E F1 F2 F3 F4
内部 10 進数	PIC S9999 {COMP } {COMP—3}	+1234 —1234	01 23 4F 01 23 4D
	PIC 9999 {COMP } {COMP—3}	+1234 —1234	01 23 4F 01 23 4F
2 進数	PIC S9999 COMP—4	+1234 —1234	04 D2 FB 2E
	PIC 9999 COMP—4	+1234 —1234	04 D2 04 D2
内部 浮動 小数点	COMP—1	+1234 —1234	44 9A 40 00 C4 9A 40 00
内部 浮動 小数点	COMP—2	+1234 —1234	40 93 48 00 00 00 00 00 C0 93 48 00 00 00 00 00
外部 浮動 小数点	PIC +9(2).9(2)E+99 DISPLAY	+1234 —1234	4E F1 F2 4B F3 F4 C5 4E F0 F2 60 F1 F2 4B F3 F4 C5 4E F0 F2

* 各バイトの内部表記は、16 進数 2 桁で表示されます。
各桁のビット構成は以下のとおりです。

16 進数の数字	ビット構成	16 進数の数字	ビット構成
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

注:
1. 2 進数の左端は符号を表し、0 ならば正、1 ならば負です。
2 負の 2 進数は、2 の補数の形式で表されます。
3 16 進数 4E は EBCDIC 文字の + を表し、16 進数 60 は EBCDIC 文字の — を表します。
4 SIGN TRAILING の仕様 (SEPARATE CHARACTER オプションなし) は、コンパイラーの標準のアクションに相当するものです。

注: ネイティブ 2 進数 COMP-5 数値項目の内部表現は、2 進数 COMP-4 数値項目の内部表現と同じになります。

図 6-9. 数字項目の内部表現

表 6-12. 数字国別項目の内部表現

項目	内容	値	内部表現
国別 10 進数	PIC 9999 NATIONAL	1234	00 31 00 32 00 33 00 34
	PIC S9999 NATIONAL SIGN LEADING SEPARATE	+1234	00 2B 00 31 00 32 00 33 00 34
		-1234	00 2D 00 31 00 32 00 33 00 34
	PIC S9999 NATIONAL SIGN TRAILING SEPARATE	+1234	00 31 00 32 00 33 00 34 00 2B
		-1234	00 31 00 32 00 33 00 34 00 2D

DISPLAY-1 句

IBM Extension

DISPLAY-1 句は、DBCS または DBCS 編集として項目を定義します。

End of IBM Extension

INDEX 句

INDEX 句で定義されるデータ項目は、**指標データ項目**です。

指標データ項目は、指標名の値を後の参照に備えて保管するために使用することのできる 4 バイトの基本項目 (必ずしもテーブルと関連付けられる必要はない) です。 SET ステートメントによって、指標データ項目に指標名の値を割り当てることができます。

指標名の値は変位で、テーブル内のオカレンス番号に対応しています。指標名の値は次の値と等価です。

(オカレンス番号 - 1) * 記入項目の長さ

指標名を 999 999 999 以上の値に設定すると、指標名の値は未定義のままになります。

SEARCH ステートメント、SET ステートメント、比較条件、手続き部ヘッダーの USING 句、または CALL ステートメントの USING 句の中でだけ、指標データ項目を直接的に参照できます。

指標データ項目は、MOVE ステートメントまたは入出力ステートメントで参照されるグループ項目の一部とすることができます。

指標データ項目は、テーブル・オカレンスを表す値を保管しますが、それ自体は必ずしもテーブルの一部として定義する必要はありません。したがって、SEARCH または SET ステートメントで直接的に、あるいは MOVE ステートメントまたは入出力ステートメントで間接的に指標データ項目を参照した場合、そのステートメントの実行時に値は変換されません。

USAGE IS INDEX 文節は、どのレベルでも書くことができます。 USAGE IS INDEX 文節でグループ項目が記述されると、そのグループ内の基本項目は指標データ項目になりますが、グループそのものは指標データ項目にはならず、そのグループ名を SEARCH ステートメントや SET ステートメント、または比較条件で使うことができなくなることが あります。基本項目の USAGE 文節は、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。

指標データ項目を条件変数とすることはできません。

文節 JUSTIFIED、PICTURE、BLANK WHEN ZERO、SYNCHRONIZED、TYPE、VALUE、または FORMAT を使用して USAGE IS INDEX 文節で記述されているグループ項目または基本項目を記述することはできません。

ソース・プログラムを他のシステムへ移送できるようにする場合、そのプログラムが外部レコードに保管されるときには、指標データ項目の内容に依存するものであってはなりません (その内容はシステム固有のものだからです)。

NATIONAL 句

IBM Extension

NATIONAL 句は、国別として項目を定義します。対応するデータ項目のピクチャー・ストリングは、1 つの N または複数の N のみを含むことができます。

End of IBM Extension

POINTER 句

IBM Extension

USAGE IS POINTER 文節を使って定義されたデータ項目は、ポインター・データ項目です。

ポインター・データ項目とは、基底アドレッシングに使用できる 16 バイト基本項目のことです。ポインター・データ項目は、等しいかどうかを比較したり、他のポインター項目に移したりすることができます。

ポインター・データ項目は、以下でしか使用できません。

- SET ステートメント (形式 5 および形式 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
- ADDRESS OF または LENGTH OF を含む式
- 組み込み関数の引数

USAGE IS POINTER 文節は、66 と 88 を除くすべてのレベルで書くことができます。

グループ項目が USAGE IS POINTER 文節で記述されているなら、そのグループ内の基本項目はポインター・データ項目です。ただし、そのグループ自身はポインター・データ項目ではなく、ポインター・データ項目が許可されている構文中では使えません。

ポインター・データ項目は、MOVE ステートメントまたは I/O ステートメントの中で参照されているグループの一部になることがあります。しかし、ポインター・データ項目がグループの一部である場合、ステートメント実行時にポインター値が別の内部表記に変換されることはありません。

ポインター・データ項目は、REDEFINES 文節のサブジェクトまたはオブジェクトになることがあります。

ポインター・データ項目の VALUE 文節は、NULL または NULLS だけを含めることができます。

ポインター・データ項目はクラスまたはカテゴリーに属しておらず、条件変数として使用することはできません。

文節 JUSTIFIED、PICTURE、SIGN、TYPE、BLANK WHEN ZERO、および FORMAT を使用して USAGE IS POINTER 文節で定義されているグループ項目または基本項目を記述することはできません。

ポインター・データ項目は、CORRESPONDING 操作においては無視されます。

ポインター・データ項目をファイルに書くことができますが、そのポインター・データ項目を含むレコードを後で読んだ場合、その項目はもはや有効なアドレスを表すことにはなりません。

USAGE IS POINTER は、 ADDRESS OF 特殊レジスターに対して暗黙的に指定されます。

ILE COBOL ポインター・データ項目を通常の数字として扱うことはできません。

ポインターの位置合わせ: この節のポインターの位置合わせについての説明では、ポインターという用語は、ポインター・データ項目とプロシージャー・ポインター・データ項目の両方を指します。

ポインターが参照されたり REDEFINES 文節のサブジェクトであるときは、そのオブジェクト項目は**位置合わせ**されていなければなりません。言い換えると、レコードの始まりからの位置が 16 バイトの倍数になるオフセットに位置指定されていなければなりません。

作業用ストレージ・セクション、ローカル・ストレージ・セクションまたはファイル・セクションにおいてポインターとして記述されたデータ項目は位置合わせされます。ポインターが、レベル番号 01 から始まる構造の一部であれば、コンパイラーはその構造の先頭に位置合わせします。その後、コンパイラーはポインターの前に FILLER 項目を配置し、データ項目も確実に位置合わせされるようにします。コンパイラーは、これらの FILLER 項目を追加するときに警告を出します。

リンケージ・セクションでは、以下のとおりです。

- 処理オプション NOLSPTRALIGN が有効であれば、コンパイラーは FILLER 項目を構造に追加しません。コンパイラーは、ユーザーが 01 レベル項目を位置合わせしたという前提で処理していることを示す警告を出します。
- 処理オプション LSPTRALIGN が有効であれば、ポインターとして記述されたデータ項目も位置合わせされます。

ポインターがリンケージ・セクション中の REDEFINES 文節のサブジェクトで、その文節のオブジェクトがポインターでない場合、ユーザーは、ポインター位置合わせを保持する必要があるという警告を受け取ります。作業用ストレージ・セクション、ローカル・ストレージ・セクションまたはファイル・セクションでの同様な状況においては、ユーザーが文節のオブジェクトを位置合わせしないと、エラーが発生します。

SYNCHRONIZED 文節は USAGE IS POINTER または USAGE IS PROCEDURE-POINTER 文節とともに指定できますが、この文節はポインター用にすでに暗黙指定されています。

ポインターがテーブルの一部であれば、そのテーブルの最初の項目は位置合わせされ、そのポインターのすべてのオカレンスも必ず同様に位置合わせされるよう、そのテーブルの終わりに充てん文字項目が追加されることがあります。

データ構造に FILLER 項目が追加されないようにするには、構造の先頭にポインターを配置します。

End of IBM Extension

PROCEDURE-POINTER 句

IBM Extension

PROCEDURE-POINTER 句で定義されたデータ項目は、**プロシージャー・ポインター・データ項目**です。それは、次のような、ILE プロシージャーまたはプログラム・オブジェクトへの入り口点のアドレス (*PGM) の入った 16 バイトの基本項目です。

- コンパイル単位内にあってしかも PROGRAM-ID ステートメントで定義された、最も外側の ILE COBOL プログラム (ILE プロシージャー) の入り口点
- ILE C の関数 (ILE プロシージャー) などの、COBOL 以外のプログラムの入り口点
- プログラムの入り口点 (*PGM)

プロシージャー・ポインター・データ項目は、以下のものだけに使用できます。

- SET ステートメント
- 関係条件
- CALL ステートメントの USING 句または手続き部ヘッダー
- ADDRESS OF および LENGTH OF を含む式
- ターゲットとしての CALL ステートメント
- 組み込み関数の引数

ポインター・データ項目と同様、プロシージャー・ポインター・データ項目も位置合わせしなければなりません。

使用規則:

- USAGE IS PROCEDURE-POINTER 文節は、レベル 88 では記述できません。
- USAGE IS PROCEDURE-POINTER 文節内で記述されたグループ項目では、そのグループ内の基本項目はプロシージャー・ポインター・データ項目になります (グループそのものはプロシージャー・ポインターにはなりません)。
- 基本項目の USAGE 文節は、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。
- プロシージャー・ポインター・データ項目は、MOVE ステートメントまたは入出力ステートメントで参照されるグループの一部であってもかまいません。しかし、そのステートメントの実行時に値の変換は行われません。
- プロシージャー・ポインター・データ項目は、ファイルに書き込むことはできますが、プロシージャー・ポインターを含んだ同じレコードを後で読み出すと、その項目はもう正しいアドレスを示しません。
- GLOBAL、EXTERNAL、OCCURS、SYNCHRONIZED、および LIKE 文節を、USAGE IS PROCEDURE-POINTER と一緒に使うことができます。
- プロシージャー・ポインターは、REDEFINES 文節のサブジェクトになることもオブジェクトになることもあります。
- プロシージャー・ポインター・データ項目の VALUE 文節には、NULL または NULLS しか含められません。
- 文節 JUSTIFIED、PICTURE、TYPE、BLANK WHEN ZERO、および FORMAT は、USAGE IS PROCEDURE-POINTER 文節で定義されたグループ項目または基本項目を記述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、条件変数になることも、どのクラスまたはカテゴリーに属することもできず、CORRESPONDING 操作では無視されます。

End of IBM Extension

VALUE 文節

VALUE 文節は、データ項目の最初の内容、または条件名と関連する値を指定します。

VALUE 文節の用法は、この文節をデータ部のどのセクションで指定するかによって異なります。

IBM Extension

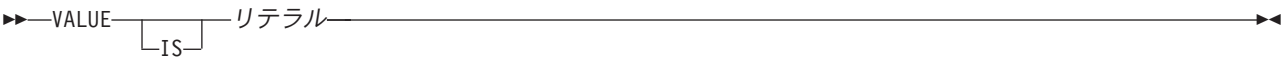
リンケージ・セクションでは、条件名以外の記入項目で使用される VALUE 文節は、コメントとして扱われます。

End of IBM Extension

ファイル・セクションおよびリンケージ・セクションでは、VALUE 文節は条件名とタイプ名の記入項目の中だけで使用するようにしなければなりません。作業用ストレージ・セクションおよびローカル・ストレージ・セクションでは、VALUE 文節は条件名記入項目およびタイプ名記入項目の中で使用できます。また、任意のデータ項目の初期値の指定に使用することが可能です。そのデータ項目は、プログラムの実行時の初めに指定された値を取ることになります。 初期値を明示的に指定しなかった場合は、どのような値をとるかは予期できません。

VALUE 文節 - 形式 1 - リテラル値

VALUE 文節 - 形式 1 - リテラル値



形式 1 は、データ項目の初期値を指定します。初期設定は、BLANK WHEN ZERO または JUSTIFIED 文節の指定とは無関係に行われます。

OCCURS 文節を含むか、あるいはそれに従属するデータ記述記入項目に指定された形式 1 の VALUE 文節は、それに関連するデータ項目の各オカレンスに、指定された値を割り当てます。 OCCURS 文節の DEPENDING ON 句を含むすべての構造は、VALUE の初期設定のための最大数のオカレンスを含むものと見なされます。

EXTERNAL 文節または REDEFINES 文節を含む記入項目が入っているか、またはそれに従属するデータ記述記入項目に、VALUE 文節を指定してはなりません。この規則は、条件名記入項目には適用されません。

VALUE 文節をグループ・レベルで指定する場合、リテラルは NULL または複数の NULLS 以外の非数字リテラルまたは表意定数でなければなりません。そのグループ内の従属記入項目は考慮されずに、グループ域が初期設定されます。また、このグループ内の従属記入項目に対して VALUE 文節を指定してはなりません。

グループ記入項目の場合、その記入項目に USAGE (USAGE DISPLAY を除く) 文節も含まれていれば VALUE 文節を指定してはなりません。

VALUE 文節は、データ記述記入項目の中にある他の文節や、この記入項目の階層のデータ記述の中にある他の文節と矛盾するものであってはなりません。

IBM Extension

COMPUTATIONAL-1 または COMPUTATIONAL-2 (内部の浮動小数点) 項目と関連付けられている任意の VALUE 文節では、浮動小数点リテラルを指定しなければなりません。条件名 VALUE 句も、浮動小数点リテラルを指定しなければなりません。さらに、表意定数 ZERO と、ゼロ・リテラルの整数および 10 進小数点形式は、浮動小数点 VALUE 文節または条件名 VALUE 句に指定できます。

浮動小数点リテラル値の詳細については 2-17 ページの『浮動小数点リテラル』を参照してください。

VALUE 文節は、外部浮動小数点項目に対して指定することができません。

DBCS 項目に関連付けられている VALUE 文節には、DBCS リテラルまたは表意定数 SPACE または SPACES が含まれていなければなりません。

国別文字 (PIC N) 項目に関連付けられている VALUE 文節には、非数値リテラル、国別リテラル、あるいは表意定数 SPACE または SPACES が含まれていなければなりません。

国別数字 (PIC 9) 項目に関連付けられている VALUE 文節には、数字リテラルまたは表意定数 ZERO/ZEROS/ZEROES が含まれていなければなりません。

国別リテラルを指定する VALUE 文節は、国別クラスのデータ項目にのみ関連付けることができます。

DBCS リテラルを指定する VALUE 文節は、DBCS クラスのデータ項目にのみ関連付けることができます。

VALUE 文節は、タイプ名のデータ記述記入項目内に指定できます。このような VALUE 文節は、このようなタイプ名を参照する TYPE 文節を使用して定義した任意の (タイプ名ではない) データ名を初期設定するために使用されます。 TYPE 文節のサブジェクトのデータ記述内に VALUE 文節を指定すると、この記入項目に関しては、関連付けられているタイプ名の記述内に指定されている VALUE 文節はすべて無視されます。

先のデータ項目に DEPENDING ON 句を伴う OCCURS 文節が含まれている場合、データ項目は VALUE 句を含むことができません。位置が固定でない項目は VALUE 文節を含むことができません。

日付、時刻、またはタイム・スタンプの項目に関連付けられている VALUE 文節は、非数値リテラルでなければなりません。このリテラルは位置合わせ規則に従って位置合わせされます。変換指定子または LOCALE の定義に合わせるためにリテラルのフォーマット設定が行われることはありません。ただし、当該の項目の USAGE に PACKED-DECIMAL が指定されている場合は例外であり、非数字リテラルはパック 10 進数に変換されます。

End of IBM Extension

リテラル値の規則:

- リテラルが指定されているところは、表意定数で置き換えることもできます。
- 項目が数字の場合は、VALUE 文節のリテラルはすべて数字でなければなりません。リテラルが作業用ストレージ・セクションの項目の値を定義している場合は、そのリテラルは数字の移動の規則によって境界合わせがされますが、1 つだけ制約があります。つまり、そのリテラルは、ゼロ以外の数字の切り捨

てを必要とする値を持てないということです。符号付きリテラルの場合は、関連する PICTURE 文字ストリングに符号 (S) が含まれていなければなりません。

- 1 つの例外はありますが、項目の VALUE 文節内の数字リテラルは、その項目の PICTURE 文節によって表される値の範囲内にある値でなければなりません。例えば PICTURE が 99PPP の場合、リテラルは 1 000 ～ 99 000 の範囲内にあるか、またはゼロでなければなりません。 PICTURE が PPP99 の場合は、リテラルは 0.000 00 ～ 0.000 99 の範囲内になければなりません。

  例外は以下のとおりです。

  – ピクチャー・シンボル P を PICTURE 文節に持たない、USAGE COMP-5 で記述されたデータ項目。

  – *NOSTDTRUNC コンパイラー・オプションが有効の場合、つまり、ピクチャー・シンボル P を PICTURE 文節に持たない BINARY または COMP-4 を使用して記述されたデータ項目である場合です。

  これらの項目に対する VALUE 文節は、固有の 2 進数表記の容量までの値を持っています。

- 項目がグループ項目、または英字、英数字、英数字編集、または数字編集の基本項目である場合は、VALUE 文節のリテラルはすべて非数字リテラルでなければなりません。リテラルは位置合わせ規則どおりに位置合わせされますが、その他の制約事項が 1 つあります。つまり、リテラル内の文字数はその項目のサイズを超えてはならないということです。

IBM Extension

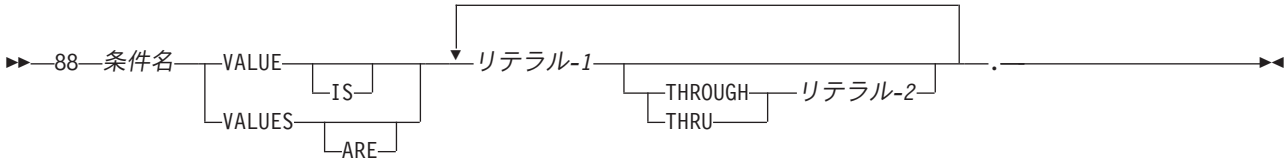
項目がブールの場合、VALUE 文節はブール・リテラルでなければなりません。

End of IBM Extension

- 記述された項目の最初の状況を判別する際には、PICTURE 文節内の編集文字の機能は無視されます。ただし項目のサイズを判別する際には編集文字も含められます。したがって、編集文字があれば、それをリテラルに含めなければなりません。例えば、項目が PICTURE +999.99 と定義されており、その値が +12.34 ならば、VALUE 文節は、VALUE '+012.34' と指定する必要があります。
- 単一の VALUE 文節を使って、最大 32 767 バイトを初期設定できます。単一のプログラムに含まれる **すべての** VALUE 文節で、最大 65 472 バイトを初期設定できます。

VALUE 文節 - 形式 2 - 条件名値

VALUE 文節 - 形式 2 - 条件名値



この形式は、1 つの値、複数個の値、値の範囲 (複数の範囲も可) などを条件名に関連付けます。そのような条件名はそれぞれ、別々のレベル 88 記入項目を必要とします。レベル番号 88 および条件名は、形式 2 VALUE 文節そのものの一部ではありません。これらは明確にするために形式に入れられています。

条件名

ある値を条件変数に関連付けるユーザー指定の名前。関連付けられる条件変数に添え字または指標が必要な場合は、その条件名を手続き部で参照するたびに、条件変数に必要なものと同様の添え字または指標を付けなければなりません。

条件名は、プロシージャー上、条件名条件の中でテストされます (7-10 ページの『条件式』を参照)。

リテラル-1

リテラル-1 だけを指定したときは、条件名は単一の値と関連付けられます。

リテラル-1 THROUGH リテラル-2

条件名は、少なくとも 1 つの値の範囲と関連します。 THROUGH 句が使用されるときはいつも、リテラル-1 はリテラル-2 より小さくなければなりません。

IBM Extension

関連付けられている条件変数が DBCS データ項目である場合、THROUGH 句に指定されるリテラルはすべて DBCS リテラル (または表意定数 SPACE、SPACES) でなければなりません。 THROUGH 句に指定される DBCS リテラルの範囲は、DBCS 文字の 16 進数値の 2 進数照合順序に基づきます。

関連付けられている条件変数が国別データ項目である場合、THROUGH 句に指定されるリテラルはすべて、非数値リテラルまたは国別リテラル (あるいは表意定数 SPACE、SPACES) でなければなりません。THROUGH 句に指定されるリテラルの範囲は、国別文字の 16 進数値の 2 進数照合順序に基づきます。

End of IBM Extension

条件名値の規則:

- VALUE 文節は、条件名記入項目内で必要であると同時に、その記入項目内の唯一の文節でなければなりません。各条件名記入項目は、先行の条件変数と関連付けられます。したがって、レベル 88 記入項目の前には必ず、条件変数に関する記入項目、あるいは 1 つの条件変数にいくつもの条件名が適用されている場合、別のレベル 88 記入項目がなければなりません。そのようなレベル 88 記入項目はそれぞれ、その条件変数の PICTURE 特性を暗黙指定されます。
- 特定の条件変数に関連した条件名記入項目は、その条件変数記入項目のすぐ後になければなりません。条件変数は、次に示すもの以外の任意のデータ記述記入項目とすることができます。
  - レベル 66 項目 (RENAMES 文節)
  - USAGE IS INDEX を指定されたデータ項目
  - USAGE IS POINTER または PROCEDURE-POINTER を指定された項目
- 条件名は、グループ項目データ記述記入項目と関連付けることができます。この場合、以下のとおりです。
  - 条件名の値は、非数字リテラルまたは表意定数として指定しなければなりません。
  - 条件名の値のサイズは、グループ内すべての基本項目のサイズの合計を超えてはなりません。
  - グループ内のどのエレメントにも、JUSTIFIED または SYNCHRONIZED 文節を含めることはできません。
  - DISPLAY 以外の USAGE をグループ内で指定することはできません。
- 条件名は、グループ・レベルと、そのグループ内の従属レベルの両方で指定できます。
- グループ・レベルでの条件名の定義によって暗示される比較テストは、グループ内の基本項目の性質とは無関係に、非数字オペランドの比較の規則に従って行われます。

IBM Extension

- VALUE 文節は、内部浮動小数点データ項目に対して使用できます。
- VALUE 文節は、DBCS 項目に対して使用できます。DBCS データ項目の比較テストは、DBCS 項目の比較の規則に従って実行されます。
- VALUE 文節は、国別項目に対して使用できます。国別データ項目の比較テストは、国別項目の比較の規則に従って実行されます。
- 条件名は、日付、時間、またはタイム・スタンプの項目に関連付けることができます。この場合、以下のとおりです。
  - 条件名の値は非数字リテラルとして指定しなければなりません。
  - 各条件名は、それぞれ条件変数の FORMAT 特性を暗黙的に持っています。したがって、この条件名を使用する比較テストは、すべて日時クラスの項目の比較に関する規則にしたがって行われます。
  - THROUGH 句は、条件変数が日時クラスの条件変数であるときに指定することができます。この場合、リテラル-1 の時刻または日付はリテラル-2 よりも小さくなければなりません。

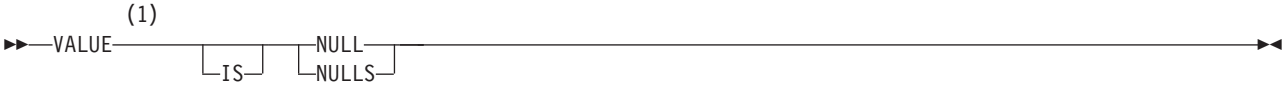
End of IBM Extension

- スペース、分離文字コンマ、あるいは分離文字セミコロンで、連続するオペランドを区切らなければなりません。
- 各記入項目は、分離文字ピリオドで終わる必要があります。
- 条件名記入項目のリテラルのタイプは、その条件変数のデータ・タイプと矛盾していてはなりません。

VALUE 文節 - 形式 3 - NULL 値

IBM Extension

VALUE 文節 - 形式 3 - NULL 値



注:

1 IBM 拡張

この形式は、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・データ項目に対して有効ではないアドレスを割り当てます。 NULL 値は未定義です。

VALUE IS NULL は、暗黙的にまたは明示的に USAGE IS POINTER または USAGE IS PROCEDURE-POINTER として記述された基本項目に対してだけ指定できます。

End of IBM Extension

手続き部

手続き部の概要

手続き部は、COBOL ソース・プログラムではオプショナルです。手続き部はオプショナルの宣言で構成され、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。

手続き部の構造は、以下のとおりです。

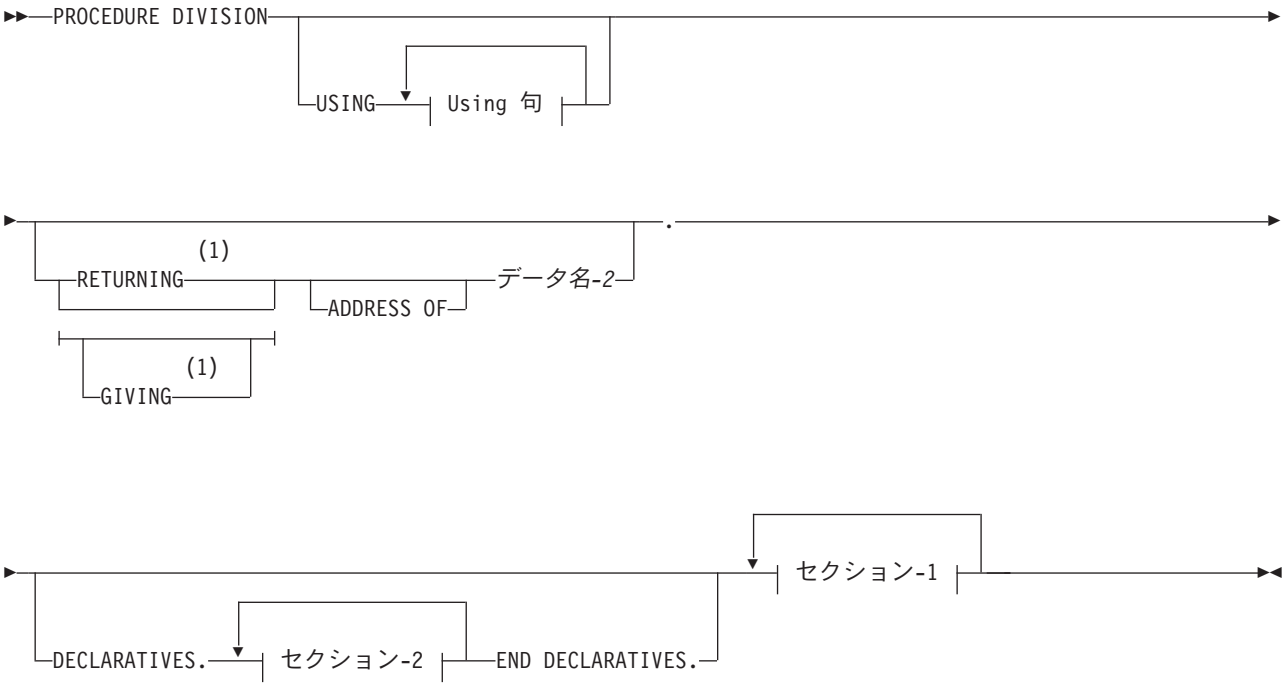
- 形式 1 - セクションと段落あり
- 形式 2 - 段落のみ

実行は、手続き部の宣言以外の最初のステートメントから始まります。ステートメントは、規則によって別の実行順序が決まる場合を除いて、コンパイルに渡される順序で実行されます。

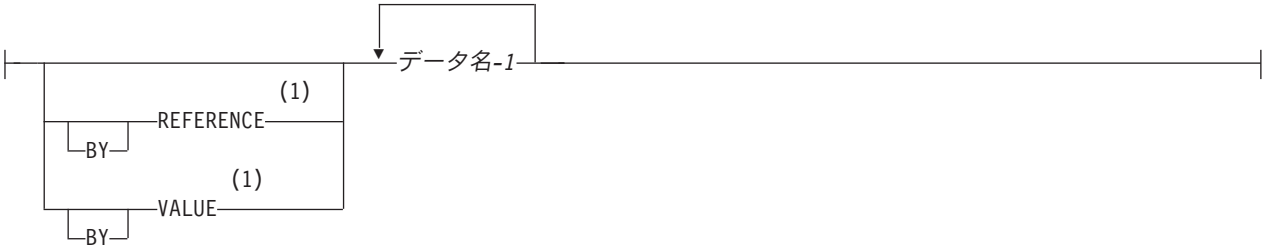
手続き部は、END PROGRAM ヘッダーの位置、すなわち次の COBOL ソース・プログラムの先頭の手前、またはプログラムの物理的な終わりで終了します。プログラムの物理的な終わりとは、ソース・プログラムにおいてこれ以上ステートメントが現れない物理的な位置のことです。

形式 1 - セクションと段落あり

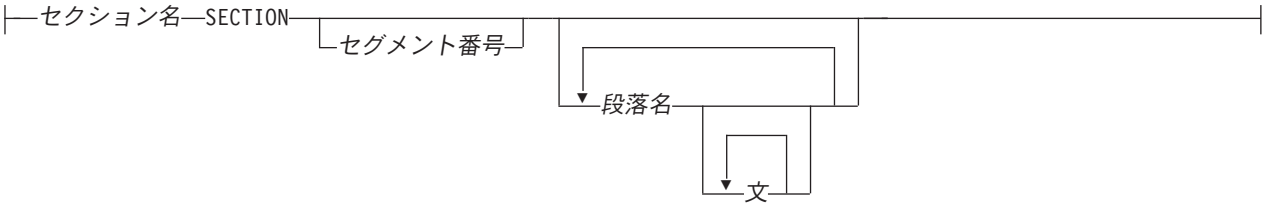
手続き部 - 形式 1



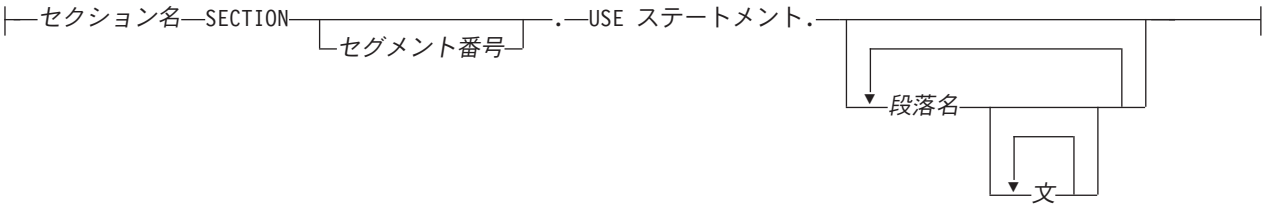
USING 句:



セクション-1:



セクション-2:

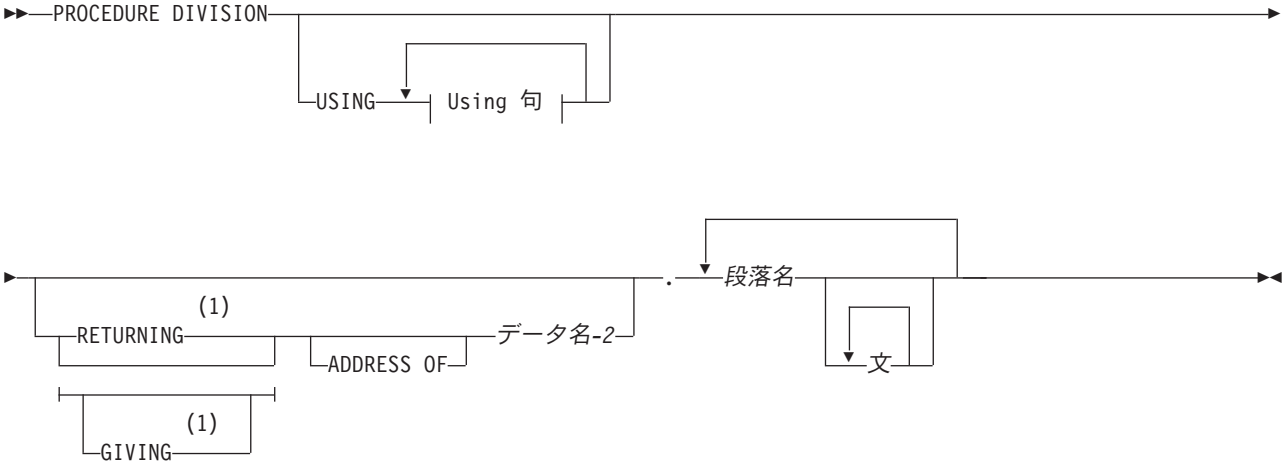


注:

1 IBM 拡張

形式 2 - 段落のみ

手続き部 - 形式 2



注:

1 IBM 拡張

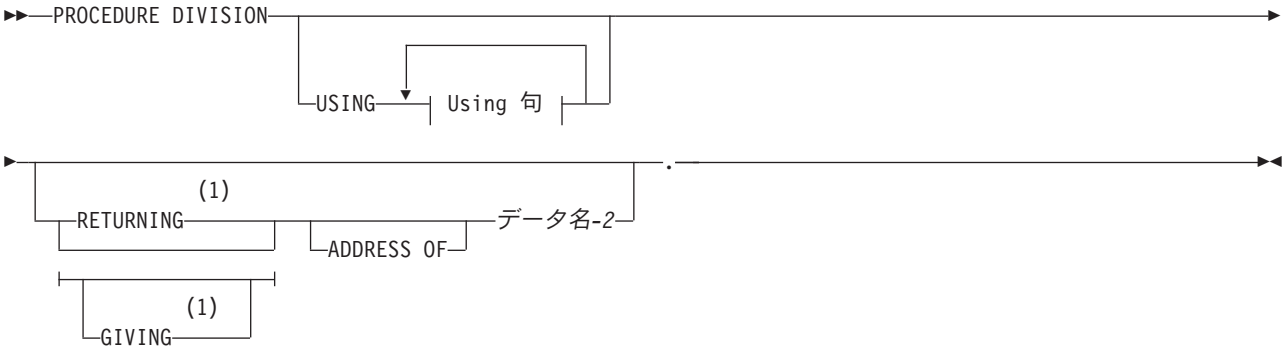
```
SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..


 004010 PROCEDURE DIVISION.
 004020 DECLARATIVES.
 004030 SECTION-NAME SECTION.
 004040 PARAGRAPH-NAMES.
 004050     PROGRAMMING STATEMENTS.
 004060*    COMMENTS.
 004070 END DECLARATIVES.
 004080 SECTION-NAME SECTION.
 004090 PARAGRAPH-NAME.
 004100     PROGRAMMING STATEMENTS.
```

図 7-1. 手続き部の構成を示すコーディング例

手続き部ヘッダー

手続き部が指定される場合には、次のヘッダーによって識別されます。

手続き部 - ヘッダー - 形式



注:

1 IBM 拡張

USING 句は、オブジェクト・プログラムが CALL ステートメントによって呼び出され、そのステートメントに USING 句が入っている場合にだけ必要とされます。

USING 句

USING 句は、呼び出し側プログラムで定義されているデータ項目を、呼び出し先サブプログラムで使用可能にします。

USING 句に関する以下の規則は、呼び出し側プログラムと呼び出し先プログラムが COBOL でコーディングされていることを前提にしています。

- USING 句を手続き部ヘッダーで指定するのは、このプログラムが USING 句を含んでいる CALL ステートメントによって呼び出されるサブプログラムである場合に限られます。すなわち、呼び出し側プログラム中の CALL USING ステートメントごとにそれに対応する USING 句が、呼び出し先サブプログラムに指定されている必要があります。

- USING 句は、呼び出し先サブプログラムの手続き部ヘッダーの中で有効です。
- 各 USING ID は、呼び出し先サブプログラムのリンケージ・セクションの中でレベル 01 またはレベル 77 の項目として定義されなければなりません。
- USING ID に REDEFINES 文節を含めることはできません。
- 特定のユーザー定義語をデータ名-1 として複数回表示することはできません。
- 呼び出し側プログラムにおいては、USING 句は CALL ステートメントに対して有効となります。つまり、各 USING ID はデータ部にレベル 01、レベル 77、または基本項目のいずれかとして定義されている必要があります。
- プログラムがプログラムの LINKAGE TYPE で呼び出される場合のデータ名の指定可能な最大数は 255 です。また、プロシージャーの LINKAGE TYPE で呼び出されたプログラムについては、データ名の最大数は 400 です。
- 呼び出し側サブプログラムおよび呼び出し先サブプログラムの両方で USING ID の現れる順序によって、両方のプログラムで使用可能なデータの対応付けが行われます。対応は位置によるものであり、名前によるものではありません。対応する ID は、文字数が同じでなければなりませんが、データ記述が同じである必要はありません。さらに指標名の場合、対応付けはなされません。これは、呼び出し側プログラムと呼び出し先プログラムにおける指標名が常に別々の指標を参照するためです。
- CALL USING ステートメントに指定される ID は、呼び出し側プログラムで使用可能なデータ項目の名前となり、呼び出し先プログラムの中でそのデータ項目を参照できます。また、ある特定の ID は複数回現れます。これらの項目は、データ部の任意のセクションで定義しておきます。
- GLOBAL 文節を含む USING ID は、コンパイル単位のただ 1 つの「手続き部」のヘッダーで指定することができます。

IBM Extension

- ID は、手続き部の USING 句で複数回現れることがあります。その場合には、CALL USING ステートメントによって渡される最後の値が使用されます。

End of IBM Extension

- 呼び出し先プログラムのリンケージ・セクションに定義されたデータ項目は、 そのプログラムの手続き部の中で参照できます。ただし、それは以下のいずれかの条件を満たしている場合に限られます。
  - 手続き部ヘッダーの USING 句のオペランドである。
  - REDEFINES または RENAMES 文節で定義され、そのオブジェクトが上の条件を満たす。

    IBM Extension

  - ADDRESS OF 特殊レジスターの引数として使用されている。

    End of IBM Extension

  - 上記の規則の条件を満たす項目に従属している。
  - 上記の条件のいずれかを満たしているデータ項目に関連した条件名または指標名である。

BY REFERENCE

IBM Extension

BY REFERENCE 句は、別の BY REFERENCE または BY VALUE 句により指定変更されるまで、その後に続くすべてのパラメーターに適用されます。

CALL 引数が BY CONTENT または REFERENCE により渡される場合、BY REFERENCE を PROCEDURE DIVISION ヘッダーでの対応する形式のパラメーターに対して指定するか、または暗黙に指定する必要があります。

BY REFERENCE と BY VALUE のどちらも指定されていない場合には、BY REFERENCE がデフォルトとなります。

BY REFERENCE 句を使用して、内部浮動小数点または外部浮動小数点、DBCS、日付、時刻、あるいはタイム・スタンプのデータ項目を渡すことができます。

End of IBM Extension

BY VALUE

IBM Extension

BY VALUE 句は、別の BY VALUE または BY REFERENCE 句により指定変更されるまで、その後に続くすべてのパラメーターに適用されます。

BY VALUE 句を使用して、内部浮動小数点または外部浮動小数点、日付、時刻、あるいはタイム・スタンプのデータ項目を渡すことができます。

End of IBM Extension

GIVING/RETURNING 句

IBM Extension

GIVING と RETURNING は同等です。

データ名-2: データ名-2 は、出力専用のパラメーターです。プログラムの結果として戻されるデータ項目を指定します。データ名-2 は、リンケージまたは作業用ストレージ・セクションに定義しなければなりません。添え字を付けたり参照変更することはできません。

データ名-2 は、内部浮動小数点または外部浮動小数点、DBCS、日付、時刻、あるいはタイム・スタンプのデータ項目のうちのどれであってもかまいません。

プログラムがその呼び出し側に戻されると、データ名-2 の値は CALL RETURNING 句に指定された ID に暗黙に保管されます。

RETURNING 句の存在は、RETURN-CODE 特殊レジスターの設定には影響しません。

呼び出し側プログラムが COBOL の場合、CALL ステートメントの GIVING/RETURNING 句を指定する必要があります。さらに、その呼び出し側プログラム内のデータ名 2 とそれに対応する CALL RETURNING ID は、同じ数の文字位置で、かつ同じ USAGE 文節、SIGN 文節およびカテゴリーのものでなければなりません。

PROCEDURE DIVISION RETURNING 句は主プログラムでは使用しないでください。予期しない結果となります。PROCEDURE DIVISION RETURNING 句を指定できるのは、呼び出し先サブプログラムだけです。主プログラムでは、操作環境に値を戻す RETURN-CODE 特殊レジスターを使用してください。

PROCEDURE DIVISION ヘッダーの RETURNING/GIVING 句に TYPE 句を含めることはできません。

ADDRESS OF 特殊レジスター: この特殊レジスターについては 6-5 ページの『ADDRESS OF 特殊レジスター』 ページを参照してください。

End of IBM Extension

宣言

宣言には、例外条件が発生した場合に実行する 1 つまたは複数の特別な目的のセクションを記述します。

宣言セクションを指定するときは、手続き部の始めに一連の宣言セクションとしてまとめて指定しなければならず、また手続き部の全体もいくつかのセクションに分割する必要があります。

各宣言セクションの先頭にはそのセクションの関数を示す USE 文があり、それ以降の部分に記述される一連のプロシージャーは、例外条件が発生した場合に取るべき処置を指定するものです。それぞれの宣言セクションは、別のセクション名とその後の USE 文で終わるか、または END DECLARATIVES というキーワードで終わります。 USE ステートメントの詳細は 8-32 ページの『USE ステートメント』を参照してください。 GLOBAL 句の使用については 8-34 ページの『ネストされたプログラムについての優先順位規則』を参照してください。

宣言セクションのグループ全体の前にはキーワード DECLARATIVES が、 手続き部ヘッダーの下の行に指定されています。また、このグループの終わりにはキーワード END DECLARATIVES が指定されています。キーワード DECLARATIVES と END DECLARATIVES は区域 A から始め、キーワードの後には分離文字としてピリオドを指定する必要があります。同じ行に他のテキストがあってはなりません。

手続き部の宣言部分において、各セクション・ヘッダー (オプショナルであるセグメント番号のある) の後には分離文字のピリオド、USE 文、そして分離文字のピリオドの順に指定する必要があります。同じ行に他のテキストがあってはなりません。

USE 文自体は実行されることはありません。その代わり、USE 文にはその後ろにあるプロシージャーの段落を実行する条件を定義し、取るべき処置を指定します。このプロシージャーの実行後に、それを実行したルーチンに制御が戻されます。

宣言プロシージャーの中では、非宣言のプロシージャーに対する参照があってはなりません。

USE ステートメントに関連したプロシージャー名は、別の宣言セクションで参照したり、または PERFORM ステートメントだけが指定されている非宣言プロシージャーで参照できます。

宣言は別の宣言からの、または COBOL プログラムの非宣言からの別個の呼び出しとして実行されます。「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」のエラー処理の章にある宣言の使用に関するセクションを参照してください。

宣言プロシージャーの中では、以前に呼び出されてまだ呼び出しルーチンに制御を戻していない USE プロシージャーを実行させるようなステートメントがあってはなりません。

宣言プロシージャーは最後のステートメントが実行されると、そのプロシージャーは終了します。

プロシージャー

手続き部の中の**プロシージャー**は、次のものから構成されます。

- 1 つのセクションまたはセクションのグループ
- 1 つの段落または段落のグループ

注: COBOL プロシージャーと ILE プロシージャー (ILE COBOL のソース・プログラム) とを混同しないでください。

プロシージャー名は、セクションまたは段落を識別するユーザー定義名です。

セクション

セクションは、1 つのセクション・ヘッダーで構成され、その後にオプションで、 1 つまたは複数の段落が続きます。**セクション・ヘッダー**はセクション名であり、その後に、SECTION というキーワード、セグメント番号 (指定はオプション)、および分離文字のピリオドが続きます。セクション・ヘッダーは、区域 A で開始する必要があります。セグメント番号については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」で説明されています。

セクション名とは、セクションを識別するためのユーザー定義の語のことです。セクション名を参照する場合、セクション名は修飾できないので、そのセクション名が定義されているプログラム内で固有なものでなければなりません。

セクションは、次のセクション・ヘッダーの直前か、手続き部の終わり、あるいは宣言の中ではキーワード END DECLARATIVES で終わります。

段落

段落は、段落名の後に分離文字のピリオドがあり、 場合によってはその後にさらに 1 つまたは複数の文が続くという構成になっています。

段落名とは、段落を識別するためのユーザー定義語のことです。段落名は修飾することができるので、固有なものにする必要はありません。段落名は区域 A で始めなければなりません。段落は次の段落名かセクション・ヘッダーの直前で、あるいは手続き部の終わりで終了します。また宣言においては、段落が終了するのは次の段落、次の USE ステートメント、あるいはキーワード END DECLARATIVES の直前です。プログラム中にある段落がセクションに含まれている場合、そのプログラムのすべての段落はセクションに含まれていなければなりません。

文: **文**は、分離文字のピリオドで終わる 1 つまたは複数のステートメントで構成されています。

ステートメント: **ステートメント**とは、ID や記号 (リテラル、関係演算子など) の構文的に有効な組み合わせのことで、その先頭に COBOL verb が指定されているものです。

実行は、手続き部の宣言以外の最初のステートメントから始まります。ステートメントは、規則によって別の実行順序が決まる場合を除いて、コンパイルに渡される順序で実行されます。

手続き部は、プログラムの物理的な終わり、つまりソース・プログラムにおいてこれ以上ステートメントが現れない物理的な位置で終わります。

ID: **ID** とは、データ項目を指名する構文的に正しい組み合わせのデータ名に、修飾子、添え字、および固有な参照ができるよう必要に応じて参照修飾子が指定されたものです。手続き部の参照では必ず (クラス・テストまたはテスト組み込み関数内の関数引数の場合は除く)、ID の内容と PICTURE 文節または FORMAT 文節によって指定されたクラスと互換性がなければなりません。そうでないと、予想し得ない結果になります。

手続き部のサンプル・ステートメント

```
. 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

PROCEDURE DIVISION.
DECLARATIVES.
ERROR-IT SECTION.
    USE AFTER STANDARD ERROR PROCEDURE ON INPUT-DATA.
ERROR-ROUTINE.
    IF CHECK-IT = "30" ADD 1 TO DECLARATIVE-ERRORS.
END DECLARATIVES.
BEGIN-NON-DECLARATIVES SECTION.
100-BEGIN-IT.
    OPEN INPUT INPUT-DATA OUTPUT REPORT-OUT.
110-READ-IT.
    READ INPUT-DATA RECORD
       AT END MOVE "Y" TO EOF-SW.
    IF EOF-SW NOT = "Y" ADD 1 TO RECORDS-IN.
200-MAIN-ROUTINE.
    PERFORM PROCESS-DATA UNTIL EOF-SW = "Y".
    PERFORM FINAL-REPORT THRU FINAL-REPORT-EXIT.
    DISPLAY "TOTAL RECORDS IN = " RECORDS-IN
       UPON WORK-STATION.
    DISPLAY "DECLARATIVE ERRORS = " DECLARATIVE-ERRORS
       UPON WORK-STATION.
STOP RUN.
PROCESS-DATA.
    IF RECORD-ID = "G"
        PERFORM PROCESS-GEN-INFO
    ELSE
        IF RECORD-CODE = "C"
           PERFORM PROCESS-SALES-DATA
        ELSE
           PERFORM UNKNOWN-RECORD-TYPE.
```

算術式

値を計算する式のことで、計算された値は条件ステートメントや算術ステートメントでオペランドとして使用されることがあります。算術式は、厳密な階層と優先順位に従ってオペランドと演算子から構成されます。

一般に、算術式は以下のものを指します。

1. 以下のような基本数字項目
   - 数字リテラル (整数または小数)
   - 基本数字項目として記述された ID
   - 表意定数 ZERO (ZEROS、ZEROES)
   - 数字関数
2. 括弧で囲まれた算術式
3. 単項演算子 (+、-) の後にある算術式
4. 2 項算術演算子 (+、-、*、/、**) が間にある 2 つの算術式

算術式の中に現れる ID とリテラルは、算術を行うことのできる基本数字項目または数字リテラルのいずれかを表していなければなりません。

指数式

指数式が正の数と負の数の両方として評価される場合、結果は常に正の数になります。例えば 4 の平方根は、常に +2 となります。

累乗される式の値がゼロである場合は、指数はゼロより大きい値を持たなければなりません。そうでない場合は、サイズ・エラー条件になります。式の結果が実数にならない場合は、いずれもサイズ・エラー条件になります。

値 2 のリテラル整数が指数でない限り、指数演算の結果は 13 桁目の小数部までで切り捨てられます。指数が非整数の場合、指数演算の結果は 7 桁まで正確です。

算術演算子

算術式では、5 つの 2 項算術演算子と 2 つの単項算術演算子を使用できます。それらは特殊文字によって表され、特殊文字の前後にはそれぞれ 1 つのスペースがなければなりません。

2 項演算子

	意味
+	加算
-	減算
*****	乗算
/	除算
****	累乗

単項演算子

	意味
+	+1 を乗算
-	-1 を乗算

括弧を使用することにより、複合式における評価の順序を強調または変更できます。これは、読みやすさと保守しやすさの両方を向上させるものとなります。

括弧は左右を組み合わせて使用する必要があり、また左括弧はそれに対応する右括弧より前になければなりません。

括弧内の式が最初に評価され、括弧のついた組み合わせが他の組み合わせの中でネストされていることもあります。評価は、最も内側にある組み合わせから順に外側に向かって行われていきます。

評価の順序が括弧により明示されていないなら、以下に示す階層に従って左から右の順に式が評価されます。

1. 単項演算子
2. 累乗
3. 乗算および除算
4. 加算および減算

算術式

算術式は、左括弧、単項演算式、またはオペランド (ID やリテラル) でだけ始めることができます。また、右括弧またはオペランドでだけ終わらせることができます。算術式では、ID またはリテラルが少なくとも 1 回は参照されていなければなりません。

算術式の最初の演算子が単項演算子の場合、その算術式が ID あるいは別の算術式の直後にある場合には、その演算子は左括弧のすぐ後に来なければなりません。

表 7-1 は、対にできる算術記号を示しています。 算術記号の対とは、そのような 2 つの記号を連続して組み合わせたものです。以下の表では、次のように表記しています。

Yes 対にできることを示します。

No 対にできないことを示します。

表 7-1. 算術記号の有効な対

最初の記号	2 番目の記号				
	ID またはリテラル	*** / ** + -**	**単項 + または単項 -**	**(**	**)**
ID またはリテラル	No	Yes	No	No	Yes
*** / ** + -**	Yes	No	Yes	Yes	No
単項 + または単項 -	Yes	No	No	Yes	No
(	Yes	No	Yes	Yes	No
)	No	Yes	No	No	Yes

条件式

条件式を使用すると、オブジェクト・プログラムは、テストの真理値に従って制御の代替パスを選択できます。条件式は、EVALUATE、IF、PERFORM、または SEARCH ステートメントの中で指定します。

条件式は以下のいずれかとして指定できます。

- 単純条件
- 複合条件

単純条件と複合条件は両方とも任意の数の括弧の対で囲むことができますが、括弧によって、単純条件であるか複合条件であるかが変更されることはありません。

単純条件

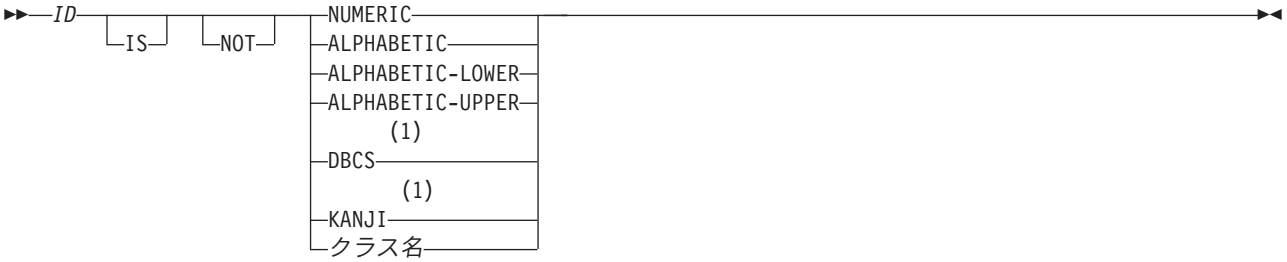
次の 5 つの単純条件があります。

- クラス条件
- 条件名条件
- 比較条件
- 符号条件
- スイッチ状況条件

単純条件は、真か偽のいずれかの真理値をとります。

クラス条件: クラス条件は、データ項目の内容が英字、英小文字、英大文字、数字であるか、あるいは環境部の SPECIAL NAMES 段落で定義されているように CLASS 文節によって指定された一連の文字の中の文字だけを含んでいるのかを識別します。

クラス条件 - 形式



注:

1 IBM 拡張

ID 使用法が DISPLAY であるデータ項目を参照する必要があります。 ID が長さゼロのグループ項目であり、さらにそれがクラス条件で指定されていないと、結果は常に真となります。 NOT が指定されていなければ、結果は常に偽となります。

ID が関数 ID である場合は、その参照対象は英数字、DBCS、または日時の関数でなければなりません。

NOT
これが使用された場合、NOT および次のキーワードは真理値によって実行されるクラス・テストを定義します。例えば、NOT NUMERIC は、データ項目が非数字かどうかを判別する真偽テストです。

NUMERIC
このデータ項目は全体として 0 から 9 の文字で構成されており、演算符号は含まれていることもいないこともあります。

テストされる項目の PICTURE に演算符号が含まれていないなら、内容が数字で演算符号が存在しない場合にだけ、その項目は数字であると判別されます。

テストされる項目の PICTURE に演算符号が含まれているときは、その項目が基本項目であり、内容が数字で、しかも有効な演算符号が存在する場合にだけ、その項目は数字であると判別されます。

EBCDIC 文字セットにおいて有効な正の組み込み演算符号は、16 進数の F、C、E、A です。また、負の符号は 16 進数の D と B です。なお、推奨する正の符号は 16 進数の F、負の符号は 16 進数の D となっています。 SIGN IS SEPARATE 文節に記述される項目の場合、有効な演算符号は + (16 進 4E) と - (16 進 60) です。

IBM Extension

数字および日時のデータ項目の場合、テストされている ID は、USAGE DISPLAY、USAGE PACKED-DECIMAL、USAGE COMP、または USAGE COMP-3 として明示的あるいは暗黙に記述できます。

End of IBM Extension

ALPHABETIC
ID が参照するデータ項目は、A ～ Z の大小の英文字とスペースの組み合わせで全体が構成されます。

ALPHABETIC-LOWER
ID が参照するデータ項目は、a ～ z の英小文字とスペースの組み合わせで全体が構成されます。

ALPHABETIC-UPPER
ID が参照するデータ項目は、A ～ Z の英大文字とスペースの組み合わせで全体が構成されます。

クラス名
ID が参照するデータ項目は、SPECIAL-NAMES 段落のクラス名の定義に列挙される文字で全体が構成されます。

クラス名テストは、数字として記述された ID に使用してはなりません。

IBM Extension

DBCS
ID は、以下の規則に従って DBCS 文字で全体が構成されます。

- DBCS データ項目の場合、テストされている ID は USAGE DISPLAY-1 として明示的あるいは暗黙に記述しなければなりません。
- DBCS 文字表示に有効な項目のデータ位置で、範囲検査が行われます。有効範囲はいずれのバイトとも X'41'～ X'FE' です。X'4040' は DBCS ブランクです。

KANJI
ID は、以下の規則に従って DBCS 文字で全体が構成されます。

- DBCS データ項目の場合、テストされている ID は USAGE DISPLAY-1 として明示的あるいは暗黙に記述しなければなりません。
- DBCS 文字表示に有効な項目のデータ位置で、範囲検査が行われます。有効範囲は、最初のバイトは X'41' ～ X'7F'、2 番目のバイトは X'41' ～ X'FE' です。 X'4040' は DBCS ブランクです。

End of IBM Extension

クラス・テストは、使用法が INDEX、POINTER、PROCEDURE-POINTER のいずれかである項目では有効ではありません。これらの項目はどのクラスやカテゴリーにも属さないからです。

IBM Extension

クラス条件は、外部の浮動小数点項目 (USAGE DISPLAY) または内部浮動小数点項目 (USAGE COMP-1 および USAGE COMP-2) には使用できません。

End of IBM Extension

表 7-2 に、クラス・テストの有効な形式を示します。

表 7-2. クラス・テストの有効な形式

ID のタイプ	クラス・テストの有効な形式	
英字	ALPHABETIC	NOT ALPHABETIC
	ALPHABETIC-LOWER	NOT ALPHABETIC-LOWER
	ALPHABETIC-UPPER	NOT ALPHABETIC-UPPER
	クラス名	NOT クラス名
英数字、 英数字編集、 または数字編集	ALPHABETIC	NOT ALPHABETIC
	ALPHABETIC-LOWER	NOT ALPHABETIC-LOWER
	ALPHABETIC-UPPER	NOT ALPHABETIC-UPPER
	NUMERIC	NOT NUMERIC
	クラス名	NOT クラス名

表 7-2. クラス・テストの有効な形式 (続き)

ID のタイプ	クラス・テストの有効な形式	
外部 10 進数 内部 10 進数	NUMERIC	NOT NUMERIC
IBM Extension DBCS DBCS 編集 **End of IBM Extension**	DBCS KANJI	NOT DBCS NOT KANJI
IBM Extension 日時 **End of IBM Extension**	NUMERIC クラス名	NOT NUMERIC NOT クラス名

条件名条件: 条件名条件は、条件変数をテストして、条件変数の値が条件名に関連した任意の値に等しいかを判別します。

条件名条件 - 形式

条件名は、比較条件のための略語として条件の中に使用されます。条件変数と条件名の値を比較する際の規則は、比較条件に関して示されている規則と同じです。

条件名が値の範囲 (またはいくつかの範囲) に結び付けられている場合、条件変数は、その値が範囲内 (両端の値も含めて) に入っているかどうかがテストされます。このテストの結果が真となるのは、条件名に対応する値の 1 つが、関連する条件変数の値に等しい場合です。

IBM Extension

条件名には、浮動小数点値および DBCS 値が使用できます。

End of IBM Extension

次の例は、条件変数および条件名の使用を図示しています。

```
01  NUMBER              PIC  99.
    88  FIVE            VALUE 5.
    88  ONE-DIGIT-EVEN   VALUE 0, 2, 4, 6, 8
    88  TWO-DIGIT-NUMBER VALUE 10 THRU 99
```

NUMBER は条件変数であり、FIVE、ONE-DIGIT-EVEN、TWO-DIGIT-NUMBER は条件名です。

次の IF ステートメントは、上記の例に追加して、特定のレコードの年齢グループを判別できます。

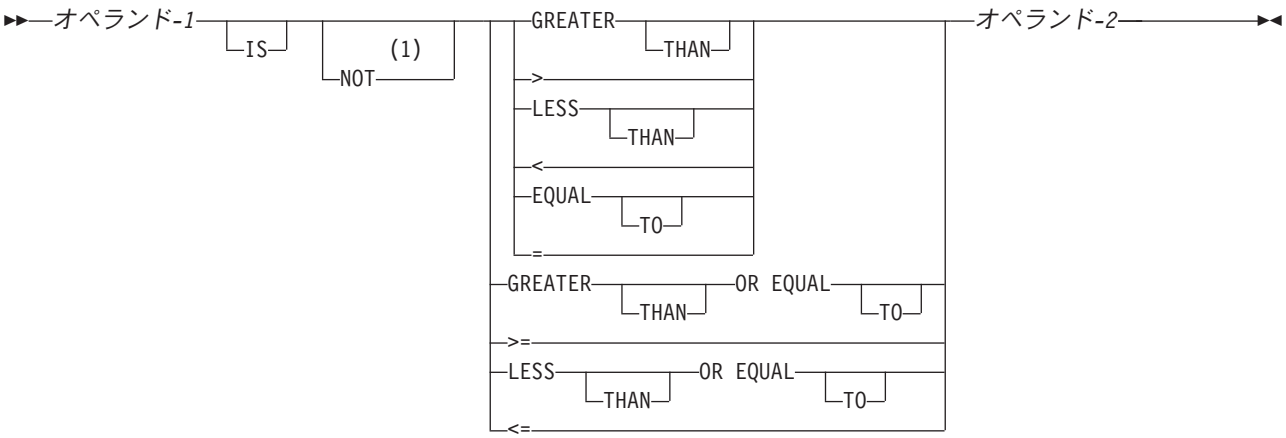
```
IF FIVE...            (値 5 のテスト)
IF ONE-DIGIT-EVEN     (値 0, 2, 4, 6, 8 のテスト)
IF TWO-DIGIT-NUMBER   (値 10 ～ 99 のテスト)
```

条件式

この条件名条件のテスト結果によって、オブジェクト・プログラムがいずれかの実行代替パスに進みます。

比較条件: 比較条件は 2 つのオペランドを比較し、どちらが ID、リテラル、算術式、指標名、または関数 ID にできるかを調べます。比較条件は少なくとも 1 つの ID を参照しなければなりません。

比較条件 - 形式



注:

1 NOT GREATER THAN OR EQUAL TO、NOT >=、NOT LESS THAN OR EQUAL TO、および NOT <= は IBM 拡張です。

オペランド-1
比較条件のサブジェクト。 ID、リテラル、関数 ID、算術式、または指標名です。

オペランド-2
比較条件のオブジェクト。 ID、リテラル、関数 ID、算術式、または指標名です。

関係演算子は、比較の種類を指定します。各関係演算子の前後には、スペースが 1 つずつなければなりません。

関係演算子
次のように書ける

IS GREATER THAN
IS >

IS NOT GREATER THAN
IS NOT >

IS LESS THAN
IS <

IS NOT LESS THAN
IS NOT <

IS EQUAL TO
IS =

IS NOT EQUAL TO
IS NOT =

IS GREATER THAN OR EQUAL TO
IS **>=**

IS LESS THAN OR EQUAL TO
IS **<=**

IBM Extension

IS NOT GREATER THAN OR EQUAL TO
IS NOT **>=**

IS NOT LESS THAN OR EQUAL TO
IS NOT **<=**

End of IBM Extension

DBCS 項目:

IBM Extension

DBCS データ項目およびリテラルは、すべての関係演算子で使用できます。比較 (2 つの DBCS 項目間のみ) は、DBCS 文字の 16 進数値の 2 進数照合順序に基づいています。比較する 2 つの項目の長さが異なる場合、小さいほうの項目の右側に DBCS スペースが埋め込まれます。

End of IBM Extension

ポインター・データ項目:

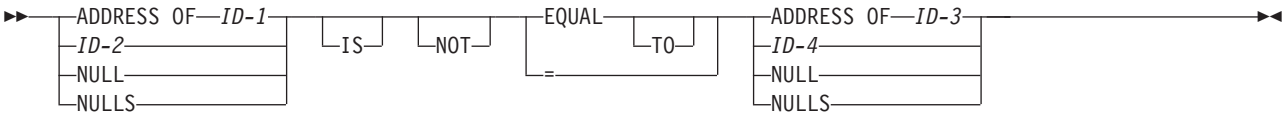
IBM Extension

ポインター・データ項目は、USAGE IS POINTER として明示的に定義された項目です。そのように定義されなければ、ADDRESS OF データ項目または ADDRESS OF 特殊レジスターであり、それらは暗黙に USAGE IS POINTER として定義されています。

ポインター・データ項目を指定すると、EQUAL および NOT EQUAL だけが関係演算子として使用できます。比較において使用される 2 つのアドレスがどちらも同じ保管場所にある場合、オペランドは等しくなります。

この比較条件は、IF、PERFORM、EVALUATE、および SEARCH 形式 1 のステートメントの中で認められています。 SEARCH 形式 2 (SEARCH ALL) ステートメントでは認められていません。それは、ポインター・データ項目に適用できる、意味のある配列がないからです。

ADDRESS 比較 - 形式



ID-1、ID-3
データ部セクションに定義されている任意のレベル項目 (レベル 66 と 88 を除く) を指定できます。

ID-2、ID-4
これは、USAGE IS POINTER として記述されていなければなりません。

NULL(S)
他のオペランドが以下のいずれかであるときだけ使用できます。

- 使用法が POINTER である項目
- ADDRESS OF 項目
- ADDRESS OF 特殊レジスター

つまり、NULL=NULL は使用できません。

End of IBM Extension

プロシージャー・ポインター・データ項目:

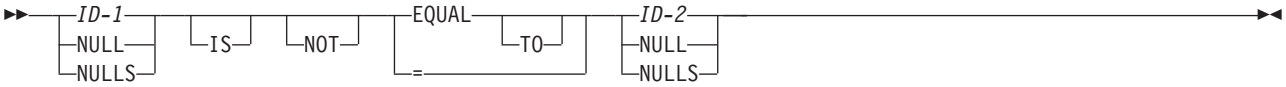
IBM Extension

プロシージャー・ポインター・データ項目は、USAGE IS PROCEDURE-POINTER として明示的に定義されます。

プロシージャー・ポインター・データ項目を指定すると、EQUAL および NOT EQUAL だけが関係演算子として使用できます。比較において使用される 2 つのアドレスがどちらも同じ保管場所にある場合、オペランドは等しくなります。

この比較条件は、IF、PERFORM、EVALUATE、および SEARCH 形式 1 のステートメントの中で認められています。しかし、SEARCH 形式 2 (SEARCH ALL) ステートメントでは認められていません。それは、プロシージャー・ポインター・データ項目に適用できる、意味のある配列がないからです。

プロシージャー・ポインター比較 - 形式



ID-1、ID-2
USAGE IS PROCEDURE-POINTER として記述する必要があります (詳しくは 6-101 ページの『PROCEDURE-POINTER 句』を参照)。

NULL(S)
他のオペランドが USAGE IS PROCEDURE-POINTER として定義されている場合に限り使用されます。 NULL=NULL は使用できません。

End of IBM Extension

数字オペランドおよび非数字オペランドの比較

数字と非数字の比較の規則は、次の表にあります。オペランドのいずれかがグループ項目の場合、非数字比較の規則が適用されます。

7-17 ページの表 7-3 は、*非数字オペランド* と比較できる項目を要約したものです。

7-18 ページの表 7-4 は、数字オペランド と比較できる項目を要約したものです。

表 7-3 および 7-18 ページの表 7-4 で使用されている記号は次のとおりです。

- NN = 非数字オペランドの比較
- NU = 数字オペランドの比較
- NL = 国別オペランドの比較
- NLN = 国別オペランドと非国別オペランドの比較
- DT = 日時オペランドの比較
- ブランク = 比較できない

表 7-3. 非数字の第 2 オペランドと比較できる項目

第 1 オペランド	**第 2 オペランド**												
	GR	**AL**	**AN**	**ANE**	**NE**	**FC[1]**	**NNL**	**DB**	**DBE**	**DA**	**TI**	**TS**	**NL**
非数字オペランド													
グループ (GR)	NN	NN	NN	NN	NN	NN	NN			NN	NN	NN	NN
英字 (AL)	NN	NN	NN	NN	NN	NN	NN						NLN
英数字 (AN)	NN	NN	NN	NN	NN	NN	NN			NN	NN	NN	NLN
英数字編集 (ANE)	NN	NN	NN	NN	NN	NN	NN			NN	NN	NN	
数字編集 (NE)	NN	NN	NN	NN	NN	NN	NN			NU	NU	NU	
表意定数 (FC[1])	NN	NN	NN	NN	NN								NL[4]
非数字リテラル (NNL)	NN	NN	NN	NN	NN					NN	NN	NN	NLN
DBCS 項目 (DB)[3]								NN	NN				NLN
DBCS 編集項目 (DBE)[3]								NN	NN				
日付 (DA)[3]	NN		NN	NN	NU		NN			DT		DT	
時刻 (TI)[3]	NN		NN	NN	NU		NN				DT	DT	
タイム・スタンプ (TS)[3]	NN		NN	NN	NU		NN			DT	DT	DT	
国別 (NL)	NN	NLN	NLN			NL[4]	NLN	NLN					NL
数字オペランド													
表意定数 ZERO (ZR)	NN	NN	NN	NN	NN								
数字リテラル (NL)	NN	NN	NN	NN	NN					NU	NU	NU	
外部 10 進数 (ED)[2]	NN	NN	NN	NN	NN	NN	NN			NU	NU	NU	NN
2 進数 (BI)										NU	NU	NU	
算術式 (AE)										NU	NU	NU	
ブール・データ項目または ブール・リテラル (BO)[3]													
内部 10 進数 (ID)										NU	NU	NU	
内部浮動小数点 (IFP)[3]													
外部浮動小数点 (EFP)[3]	NN[3]	NN[3]	NN[3]	NN[3]	NN[3]	NN[3]	NN[3]						
浮動小数点リテラル (FPL)[3]													
国別 10 進数 (ND) [2 3]	NN	NN	NN	NN	NN	NN	NN			NU	NU	NU	NN

表 7-4. 数字の第 2 オペランドと比較できる項目

第 1 オペランド	第 2 オペランド									
	ZR	NL	ED、ND	BI	AE	BO	ID	IFP[3]	EFP[3]	FPL[3]
非数字オペランド										
グループ (GR)	NN	NN	NN[2]						NN[3]	
英字 (AL)	NN	NN	NN[2]						NN[3]	
英数字 (AN)	NN	NN	NN[2]						NN[3]	
英数字編集 (ANE)	NN	NN	NN[2]						NN[3]	
数字編集 (NE)	NN	NN	NN[2]						NN[3]	
表意定数 (FC[1])			NN[2]						NN[3]	
非数字リテラル (NNL)			NN[2]						NN[3]	
日付 (DA)		NU	NU[2]	NU	NU		NU			
時刻 (TI)		NU	NU[2]	NU	NU		NU			
タイム・スタンプ (TS)		NU	NU[2]	NU	NU		NU			
数字オペランド										
表意定数 ZERO (ZR)			NU	NU	NU	NU[3]	NU	NU[3]	NU[3]	
数字リテラル (NL)			NU	NU	NU		NU	NU[3]	NU[3]	
外部 10 進数 (ED)	NU	NU	NU	NU	NU		NU	NU[3]	NU[3]	NU[3]
2 進数 (BI)	NU	NU	NU	NU	NU		NU	NU[3]	NU[3]	NU[3]
算術式 (AE)	NU	NU	NU	NU	NU		NU	NU[3]	NU[3]	NU[3]
ブール・データ項目またはブール・リテラル (BO)[3]	NU[3]					NU[3]				
内部 10 進数 (ID)	NU	NU	NU	NU	NU		NU	NU[3]	NU[3]	NU[3]
内部浮動小数点 (IFP)[3]	NU[3]	NU[3]	NU[3]	NU[3]	NU[3]		NU[3]	NU[3]	NU[3]	NU[3]
外部浮動小数点 (EFP)[3]	NU[3]	NU[3]	NU[3]	NU[3]	NU[3]		NU[3]	NU[3]	NU[3]	NU[3]
浮動小数点リテラル (FPL)[3]			NU[3]	NU[3]	NU[3]		NU[3]	NU[3]	NU[3]	
国別 10 進数 (ND) [2 3]	NU	NU	NU	NU	NU		NU	NU[3]	NU[3]	NU

7-17 ページの表 7-3 および 表 7-4 に関する注:

1 ZERO と NULL を除くすべての表意定数を含む

2 整数項目のみ

3 IBM 拡張

4 SPACE の場合のみ

数字オペランドの比較: 数字オペランドの代数値が比較されます。

- オペランドの長さ (桁数) は無関係です。
- 符号なし数字オペランドは正であると見なされます。
- ゼロは、符号には関係なく固有値であると見なされます。
- 数字オペランドどうしは、それぞれに指定されている USAGE のタイプには関係なく、比較できます。

非数字オペランドの比較: 非数字オペランドの比較は、使用されている文字セットの照合順序について行われます。

PROGRAM COLLATING SEQUENCE 文節が OBJECT-COMPUTER 段落に指定されている場合、SPECIAL-NAMES 段落の英字名文節に関係づけられている照合順序が使用されます。そうでなければ、固有 EBCDIC 文字セットが使用されます。

各オペランドのサイズは、そのオペランドの文字の合計数です。 以下の 2 つの場合が考えられます。

両オペランドのサイズが等しい場合

2 つのオペランドの対応する位置にある文字どうしが、左端の文字から始めて右端の文字まで比較されます。

対応する文字どうしが最後まですべて等しければ、両オペランドは等しいと見なされます。

等しくない文字の組が検出されると、それら 2 つの文字の照合順序での相対位置が判定されます。照合順序での位置が高い方の文字を含むオペランドが、より大きいオペランドであると見なされます。

両オペランドのサイズが等しくない場合

2 つのオペランドのサイズが等しくないときは、あたかも短い方のオペランドが、長い方のオペランドのサイズに合わせてその右端がスペースで拡張されているかのようにして、比較されます。

数字および非数字オペランドの比較: 上記で説明した非数字比較の規則が適用されます。 さらに、数字オペランドと非数字オペランドを比較するときは、その USAGE は同じでなければなりません。そのような比較においては、以下の規則も適用されます。

- 数字オペランドは整数リテラルまたは整数データ項目として記述されていなければなりません。
- 非整数のリテラルやデータ項目を非数字オペランドと比較してはなりません。

IBM Extension

- 外部浮動小数点項目は非数字オペランドと比較できます。

End of IBM Extension

いずれかのオペランドがグループ項目であれば、上述の非数字比較の規則が適用されます。そのほかに、次の規則があります。

- 非数字オペランドがリテラルまたは基本データ項目である場合、 数字オペランドはあたかもそれが同じサイズの英数字基本データ項目に移され、その後でこの英数字データ項目の内容が非数字オペランドと比較されるかのように取り扱われます。
- 非数字オペランドがグループ項目である場合、数字オペランドはあたかもそれが同じサイズのグループ項目に移され、その後でこのグループ項目の内容が非数字オペランドと比較されるかのように取り扱われます。

(7-139 ページの『MOVE ステートメント』を参照。)

ブール・オペランドの比較:

IBM Extension

ブール・オペランドは、[NOT] EQUAL TO 比較条件においてのみ使用されます。ブール・オペランドは、非ブール・オペランドと比較することはできません。ブール・データ項目およびリテラルは、1 文字位置の長さでなければなりません。 2 つのブール・オペランドが共にブール 1 またはブール 0 の値を持つ場合

には、この 2 つのオペランドは同等です。

End of IBM Extension

DBCS オペランドの比較:

IBM Extension

DBCS または DBCS 編集オペランドの比較の規則は、非数字オペランドの比較の規則と同じです。その比較は、DBCS 文字の 16 進数値の 2 進数照合順序に基づいています。 OBJECT-COMPUTER 段落の PROGRAM COLLATING 文節は、この比較には影響を及ぼしません。

End of IBM Extension

国別オペランドの比較:

IBM Extension

クラスが NATIONAL であるオペランドは、クラスが NATIONAL である別のオペランドと比較することができます。このような比較の結果は、UCS-2 文字セットの 16 進数値の 2 進数照合順序に基づきます。2 つのオペランドのサイズが異なる場合、短い方のオペランドの右側が、埋め込み文字 (Padding Character) コンパイル・オプションまたは等価の処理オプションに指定されている埋め込み文字によって埋められます。デフォルトは UCS-2 の 2 バイト文字のスペース文字 (NX"3000") です。

End of IBM Extension

国別オペランドと非国別オペランドの比較:

IBM Extension

クラスが NATIONAL であるオペランドは、英字データ項目、英数字データ項目、DBCS データ項目、非数字リテラル、または DBCS リテラルと比較することができます。

国別項目以外のデータ項目またはリテラルは、MOVE ステートメントの規則に従って、クラスが国別で、同じ論理的な長さである基本データ項目に移動された場合と同様に処理されます。それから、変換後の値が国別オペランドと比較されます。項目の長さが異なる場合、短い方の項目の右側が、埋め込み文字 (Padding Character) コンパイル・オプションまたは等価の処理オプションに指定されている埋め込み文字によって埋められます。デフォルトは、非国別オペランドが 1 バイト文字の項目である場合には、UCS-2 の 1 バイト文字のスペース文字であり、非国別オペランドが 2 バイト文字の項目である場合には、UCS-2 の 2 バイト文字のスペース文字 (NX"3000") です。

End of IBM Extension

日時オペランドの比較:

IBM Extension

日時クラスの項目と非数字オペランド (数字編集オペランドを除く) との比較の場合は、日時項目は非数字項目であるかのように扱われます。

日時クラスの項目と数字編集オペランドまたは数字オペランドとの比較時には、日時項目は編集解除されます。編集解除の結果、この項目は 1 つの整数値となります。その後、この整数値は他のオペランドと数字として比較されます。

1 つの日時項目と別の日時項目との比較時には、これらの項目は最初に共通の日付形式、時刻形式、またはタイム・スタンプ形式に変換されてから比較されます。形式リテラルの一部であるが変換指定子ではない文字 (例えば / または - などの文字) は、日時比較にまったく影響を及ぼしません。

日時項目とタイム・スタンプ項目との比較時には、タイム・スタンプの日付の部分だけが考慮されます。時刻項目とタイム・スタンプ項目との比較時には、タイム・スタンプの時刻の部分だけが考慮されます。

End of IBM Extension

指標名と指標データ項目の比較: 指標名または指標データ項目 (またはその両方) を含んだ比較は、次の規則に従います。

- 2 つの指標名の比較の際には、対応するオカレンス番号の比較がなされます。
- 指標名とデータ項目 (指標データ項目以外) の比較、あるいは指標名とリテラルの比較では、指標名のオカレンス番号がデータ項目かリテラルと比較されます。

IBM Extension

- 指標名と算術式の比較では、指標名の値に対応するオカレンス番号と、算術式が比較されます。

  算術式を使用できる場合であれば整数関数が使用できるため、この拡張によって、指標名と整数または数字関数を比較できます。

End of IBM Extension

- 指標データ項目と指標名または別の指標データ項目との比較では、変換されることなく実際の値が比較されます。比較データ項目に関してその他の比較を行った場合、どのような結果になるかわかりません。

表 7-5 は、指標名および指標データ項目に関する有効な比較を示しています。

表 7-5. 指標名と指標データ項目についての比較

比較されるオペランド	**指標名**	**指標データ項目**	**データ名**	**リテラル**	**算術式**
指標名	オカレンス番号を比較	変換せずに比較	オカレンス番号とデータ名を比較	オカレンス番号とリテラルを比較	オカレンス番号と算術式を比較
指標データ項目	変換せずに比較	変換せずに比較	使用不可	使用不可	使用不可

符号条件

符号条件は、数字オペランドの代数値がゼロより大きいか、小さいか、または等しいかを判別します。

符号条件 - 形式

オペランド

数字 ID として定義されているか、または少なくとも 1 つの ID を参照している算術式として定義されていなければなりません。

IBM Extension

オペランドは、浮動小数点 ID として定義できます。

End of IBM Extension

符号のないオペランドは POSITIVE または ZERO です。

NOT

代数テストが実行され、符号条件の真理値を確かめます。例えば、テストされたオペランドの値が正または負のときは、NOT ZERO は真であると見なされます。

スイッチ状況条件

スイッチ状況条件は UPSI スイッチの状況がオン、オフのどちらであるかを識別するもので、条件名に関連した値をテストすることにより識別が行われます。 (条件名に関係付けられた値は、英数字であると見なされています。) このテストの結果が真であるのは、UPSI スイッチが条件名に対応する値 (0 または 1) にセットされている場合です。

スイッチ状況条件 - 形式

条件名

SPECIAL-NAMES 段落に、UPSI スイッチの ON または OFF の値に関連させて定義する必要があります。 (5-4 ページの『SPECIAL-NAMES 段落』を参照。)

複合条件

複合条件は、単純条件、結合条件、複合条件のうち、任意のものを論理演算子により結合するか、または論理否定することによって作られます。

それぞれの論理演算子の前後には、スペースが 1 つずつなければなりません。次の表は、論理演算子とその意味を示しています。

表 7-6. 論理演算子とその意味

論理演算子	名前	意味
AND	論理積	両方の条件が真であるときは、真理値は**真**である。
OR	包含 OR (包含論理和)	いずれか一方または両方の条件が真であるときに、**真**である。
NOT	論理否定	真理値の逆 (条件が偽であれば、真理値は**真**である)。

括弧で囲まれていない限り、次の規則が (番号の順で) 優先します。

1. 算術演算
2. 単純条件
3. NOT
4. AND

5. OR

複合条件 (括弧で囲まれているかどうかに関係なく) の真理値は、次の値のいずれかに関して、記述されたすべての論理演算子が相互に作用した結果の真理値です。

- 個々の単純条件の真理値
- 論理結合または論理否定された条件の中間真理値

複合条件は、次のいずれかとすることができます。

- 否定単純条件
- 結合条件 (否定が可能)

否定単純条件: 単純条件を否定するには、論理演算子 NOT を使用します。否定単純条件は、単純条件の反対の真理値を与えます。

否定単純条件 - 形式

結合条件: 複数の条件を論理的に結合して、1 つの結合条件を作成できます。

結合条件 - 形式

次のどの条件でも結合できます。

- 単純条件
- 否定単純条件
- 結合条件
- 否定結合条件 (すなわち、NOT 論理演算子の後に、括弧で囲まれた結合条件を続けたもの)
- 前述の条件を組み合わせたもので、以下の表に示す規則に従って指定されます。

表 7-7. 結合条件 — 許されるエレメントの順序

結合条件のエレメント	左端	左端ではない場合、以下のものがその直前に指定可能。	右端	右端ではない場合、以下のものがその直後に指定可能。
単純条件	Yes	OR NOT AND (	Yes	OR AND)
OR AND	No	単純条件)	No	単純条件 NOT (
NOT	Yes	OR AND (	No	単純条件 (

表 7-7. 結合条件 — 許されるエレメントの順序 (続き)

結合条件のエレメント	左端	左端ではない場合、以下のものがその直前に指定可能。	右端	右端ではない場合、以下のものがその直後に指定可能。
(	Yes	OR NOT AND (	No	単純条件 NOT (
)	No	単純条件)	Yes	OR AND)

1 つの結合条件の中で AND または OR のいずれか一方 (両方ではなく) を排他的に使用するときは、括弧は不要です。ただし、演算子およびオペランドの正しい論理関係を維持するために暗黙の優先規則を変更する場合には、括弧が必要になることがあります。

左括弧と右括弧は対になっていなければならず、それぞれの左括弧はそれに対応する右括弧より左側になければなりません。

以下の表は、論理演算子と 条件 C1 と C2 の関係を示したものです。

表 7-8. 結合条件の論理演算子と演算結果

C1 の値	C2 の値	C1 AND C2	C1 OR C2	NOT (C1 AND C2)	NOT C1 AND C2	NOT (C1 OR C2)	NOT C1 OR C2
真	真	真	真	偽	偽	偽	真
偽	真	偽	真	真	真	偽	真
真	偽	偽	真	真	偽	偽	偽
偽	偽	偽	偽	真	偽	真	真

条件式の評価: 括弧を使用した場合には、結合条件の論理演算は次の順序で行われます。

1. 括弧内の条件が最初に評価されます。
2. 複数の括弧でネストされている場合には、最も内側の条件から最も外側の条件の順で評価が行われます。

括弧を使用しない場合 (あるいは、括弧のレベルが同順位でない場合) には、結合条件は次の順序で評価されます。

1. 算術式
2. 単純条件 (次の順序で)
   a. 比較条件
   b. クラス
   c. 条件名
   d. スイッチ状況
   e. 符号条件
3. 否定単純条件 (項目 2 と同じ順序)
4. 結合条件 (次の順序で)

   a. AND
   b. OR
5. 否定結合条件 (次の順序で)
   a. AND
   b. OR
6. 同じ評価順序レベルの連続オペランドは、左から右へ評価されます。ただし、結合条件の真理値は、コンポーネントのすべての条件の真理値を評価せずに決定されることがあります。

結合条件のコンポーネントである条件は、左から右へ評価されます。ある条件の真理値が、結合条件の前のエレメントの評価によって影響を受けない場合には、これらのエレメントの評価は行われません。ただし、その条件の真理値は、この段落の始めに述べたように、常に同じになります (あたかもその条件全体が演算されているかのように)。

値は、それらを含む条件が評価されるとき、算術式および関数に設定されます。同様に否定条件は、それらが示す複合条件を評価する必要が生じたとき、評価されます。

例えば、次のとおりです。

```
NOT A IS GREATER THAN B OR A + B IS EQUAL
          TO C AND D IS POSITIVE
```

この例は、次のように括弧で囲まれているかのように評価されます。

```
(NOT (A IS GREATER THAN B)) OR (((A+B) IS EQUAL
          TO C) AND (D IS POSITIVE))
```

この例での評価順序は次のとおりです。

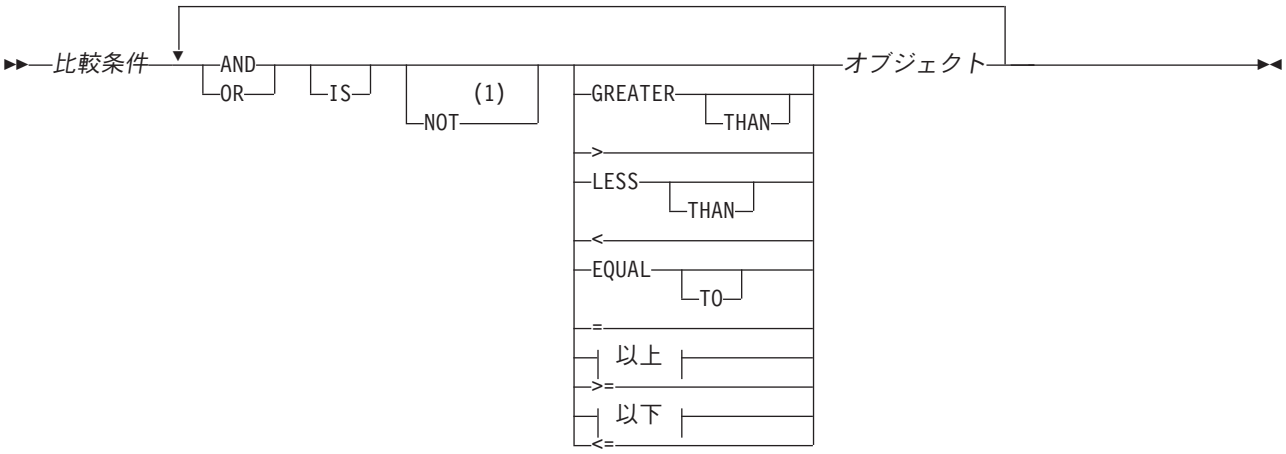
1. (NOT (A IS GREATER THAN B)) が評価されます。これが真の場合には、式が真なので、条件の残りの部分は評価されません。
2. (A+B) が評価され、中間結果 x が求められます。
3. (x IS EQUAL TO C) が評価されます。これが偽の場合には、式が偽なので、条件の残りの部分は評価されません。
4. (D IS POSITIVE) が評価され、式の最終的な真理値が求められます。

省略された結合比較条件: 比較条件が括弧を間に入れずに連続して記述されている場合、 最初の比較条件の後の任意の比較条件を、以下の 2 つの方法のいずれかで省略することが可能です。

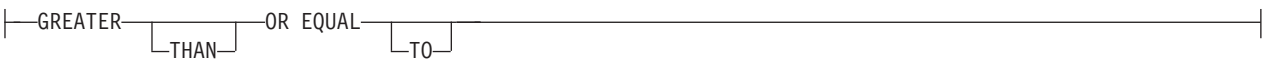
- サブジェクトの省略
- サブジェクトと関係演算子の省略

省略された結合比較条件 - 形式

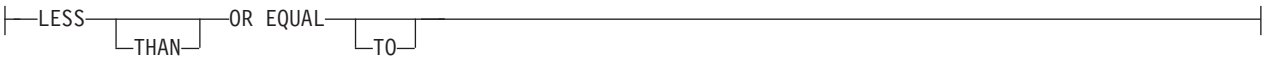
条件式



「以上」条件:



「以下」条件:



注:

1 NOT GREATER THAN OR EQUAL TO、NOT >=、NOT LESS THAN OR EQUAL TO、および NOT <= は IBM 拡張です。

オブジェクトとは、その前にある比較条件のサブジェクトと比較できる、任意のデータ項目または式のことです。

連続する一連の比較条件では、両方の省略形を指定できます。省略形の条件は次のように評価されます。

1. 最後に記述されたサブジェクトが省略されたサブジェクトであるものとして。
2. 最後に記述された関係演算子が、省略された関係演算子であるものとして。

その結果の結合条件は 表 7-9 に示したような、結合条件でのエレメントの順序の規則に従っていなければなりません。

NOT という語は、NOT GREATER THAN、NOT >、NOT LESS THAN、NOT <、 NOT EQUAL TO、および NOT = の形式の関係演算子の一部であると見なされます。

他の位置にある NOT は、論理演算子であると見なされます (したがって、否定比較条件となります)。

以下の例では、省略された結合比較条件に括弧が付いたものと付いていないもの、およびそれらと同じものを省略していない形が示されています。

表 7-9. 省略された結合比較条件

省略された結合比較条件	同等の内容
A = B AND NOT < C OR D	((A = B) AND (A NOT < C)) OR (A NOT < D)

表 7-9. 省略された結合比較条件 (続き)

省略された結合比較条件	同等の内容
A NOT > B OR C	(A NOT > B) OR (A NOT > C)
NOT A = B OR C	(NOT (A = B)) OR (A = C)
NOT (A = B OR < C)	NOT ((A = B) OR (A < C))
NOT (A NOT = B AND C AND NOT D)	NOT ((((A NOT = B) AND (A NOT = C)) AND (NOT (A NOT = D))))

ステートメント・カテゴリー

COBOL のステートメントには、次の 4 種類があります。

- 命令
- 条件
- 範囲区切り
- コンパイラー指示

命令ステートメント: 命令ステートメントは、プログラムが実行する無条件処置を指定するか、明示範囲終了符号によって終了する条件ステートメントになるかのいずれかです (7-29 ページの『範囲区切りステートメント』を参照してください)。 1 つの命令ステートメントを指定できるときにはいつでも、一連の命令ステートメントを指定できます。

表 7-10 に COBOL 命令ステートメントをリストします。

表 7-10. 命令ステートメントのタイプ

タイプ	命令ステートメント
算術計算	ADD[1] COMPUTE[1] DIVIDE[1] INSPECT (TALLYING) MULTIPLY[1] SUBTRACT[1]
データ操作	ACCEPT (DATE, DAY, DAY-OF-WEEK, TIME) INITIALIZE INSPECT (CONVERTING) INSPECT (REPLACING) MOVE SET STRING[2] UNSTRING[2] **IBM Extension** XML GENERATE[6] XML PARSE[6] **End of IBM Extension**

表 7-10. 命令ステートメントのタイプ (続き)

タイプ	命令ステートメント
終了	STOP RUN EXIT PROGRAM **IBM Extension** GOBACK **End of IBM Extension**
入出力	ACCEPT[6] ID CLOSE DELETE[3] DISPLAY[6] OPEN READ[4] REWRITE[3] SET (UPSI スイッチの場合) START[3] STOP リテラル WRITE[5] **IBM Extension** ACQUIRE COMMIT DROP ROLLBACK **End of IBM Extension**
順序付け	MERGE RELEASE RETURN SORT
プロシージャーの分岐	ALTER EXIT GO TO PERFORM
サブプログラム・リンケージ	CALL[7] CANCEL
テーブルの処理	SET

7-27 ページの表 7-10 に関する注:

1 ON SIZE ERROR または NOT ON SIZE ERROR 句が指定されていないもの

2 NOT ON OVERFLOW または ON OVERFLOW 句が指定されていないもの

3 INVALID KEY または NOT INVALID KEY 句が指定されていないもの

4 AT END、NOT AT END、INVALID KEY、NO DATA、または NOT INVALID KEY 句が指定されていないもの

5 INVALID KEY、NOT INVALID KEY、END-OF-PAGE、または NOT END-OF-PAGE 句が指定されていないもの

6 ON EXCEPTION または NOT ON EXCEPTION 句が指定されていないもの

7 ON OVERFLOW、ON EXCEPTION、または NOT ON EXCEPTION 句が指定されていないもの

条件ステートメント: **条件ステートメント**は、条件の真理値を判別すること、そしてオブジェクト・プログラムの次に取るべき処置がこの真理値によって決まることを指定します。 (7-10 ページの『条件式』を参照。)

図 7-2 には、条件ステートメントである COBOL ステートメント、または条件が含まれていて (例 : ON SIZE ERROR または ON OVERFLOW)、かつそのステートメントが明示範囲終了符号によって終了しないときに条件ステートメントになる COBOL ステートメントを列挙しています。

```
Arithmetic                      Ordering
ADD...ON SIZE ERROR             RETURN...AT END
ADD...NOT ON SIZE ERROR         RETURN...NOT AT END
COMPUTE...ON SIZE ERROR
COMPUTE...NOT ON SIZE ERROR
DIVIDE...ON SIZE ERROR
DIVIDE...NOT ON SIZE ERROR
MULTIPLY...ON SIZE ERROR
MULTIPLY...NOT ON SIZE ERROR
SUBTRACT...ON SIZE ERROR
SUBTRACT...NOT ON SIZE ERROR
データの操作                    サブプログラム・リンケージ
STRING...ON OVERFLOW            CALL...ON OVERFLOW
STRING...NOT ON OVERFLOW        CALL...ON EXCEPTION
UNSTRING...ON OVERFLOW          CALL...NOT ON EXCEPTION
UNSTRING...NOT ON OVERFLOW
XML GENERATE...ON EXCEPTION
XML GENERATE...NOT ON EXCEPTION
XML PARSE...ON EXCEPTION
XML PARSE...NOT ON EXCEPTION
判断                            テーブル処理
IF                              SEARCH...WHEN
EVALUATE
Input/Output
ACCEPT...ON EXCEPTION                 READ...NO DATA
ACCEPT...NOT ON EXCEPTION             REWRITE...INVALID KEY
DELETE...INVALID KEY                  REWRITE...NOT INVALID KEY
DELETE...NOT INVALID KEY              START...INVALID KEY
DISPLAY...ON EXCEPTION                START...NOT INVALID KEY
DISPLAY...NOT ON EXCEPTION            WRITE...AT END~OF~PAGE
READ...AT END                         WRITE...NOT AT END~OF~PAGE
READ...NOT AT END                     WRITE...INVALID KEY
READ...INVALID KEY                    WRITE...NOT INVALID KEY
READ...NOT INVALID KEY
```

図 7-2. 条件ステートメント

範囲区切りステートメント: 範囲区切りステートメントは、明示範囲終了符号を使用して、 条件ステートメントを命令ステートメントに変えます。この命令ステートメントは、その後でネストされます。さらに、明示範囲終了符号は命令ステートメントの範囲の終了を示すのに使用されることもあります。明示範囲終了符号は、条件句を持つことができるすべての COBOL verb に提供されています。

特に明示的に指定されていない限り、言語の規則によって命令ステートメントを指定できる場合はいつでも、範囲区切りステートメントを指定できます。

明示範囲終了符号: 明示範囲終了符号は、特定の手続き部ステートメントの終わりを示します。明示範囲終了符号によって区切られる条件ステートメントは、命令ステートメントと見なされ、命令ステートメントの規則に従わなければなりません。

明示範囲終了符号は次のとおりです。

```
END-ACCEPT              END-PERFORM
END-ADD                 END-READ
END-CALL                END-RETURN
END-COMPUTE             END-REWRITE
END-DELETE              END-SEARCH
END-DISPLAY             END-START
END-DIVIDE              END-STRING
END-EVALUATE            END-SUBTRACT
END-IF                  END-UNSTRING
END-MULTIPLY            END-WRITE
```

IBM Extension

END-XML

End of IBM Extension

暗黙範囲終了符号: 文の終わりにある区切りのピリオドは暗黙範囲終了符号であり、まだ終了していない前のステートメントすべての範囲を終了させます。あるステートメントが別のステートメントの中に含まれているとき、含まれる側のステートメントの後にある含む側のステートメントの次の句が、含まれる側のステートメントの範囲を終了させる暗黙範囲終了符号となります。

範囲終了符号によって終了していない条件ステートメントを、別のステートメントの中に入れることはできません。

IF ステートメントの中でネストしている条件ステートメントを除いて、ネストされるステートメントは命令ステートメントでなければならず、命令ステートメントの規則に従わなければなりません。条件ステートメントをネストすべきではありません。

コンパイラー指示ステートメント: 指定した処置を行うようコンパイラーに指示するステートメントのことです。これらのステートメントについては 8-1 ページの『コンパイラー指示ステートメント』で説明してあります。

タイプ	コンパイラー指示ステートメント
ライブラリー	COPY
宣言	USE
文書化	ENTER
コンパイラー・オプション	PROCESS
ソース・テキスト	REPLACE

タイプ	コンパイラー指示ステートメント
ソース・リスト	**IBM Extension** *CBL *CONTROL EJECT SKIP1 SKIP2 SKIP3 TITLE **End of IBM Extension**

ステートメントによる操作

COBOL ステートメントは、次の種類の操作を行います。

- 算術計算
- データ操作
- 入出力
- 順序付け
- サブプログラム・リンケージ
- テーブル処理
- プロシージャー分岐

共通の句および概念: 以下の算術ステートメントとデータ操作ステートメントには、いくつかの共通した句と概念があります。

- CORRESPONDING 句
- GIVING 句
- ROUNDED 句
- SIZE ERROR 句
- オーバーラップするオペランド

CORRESPONDING 句: 同じ名前の基本データ項目が属しているグループ項目が指定されている場合には、CORRESPONDING 句 (CORR) を使用することにより、それらの基本データ項目に対して ADD、SUBTRACT、および MOVE 操作を実行できます。

キーワード CORRESPONDING の後にある ID は両方とも、グループ項目でなければなりません。ここの説明では、これらの ID を ID-1 および ID-2 とします。

ID-1 から 1 つ、また ID-2 から 1 つのデータ項目の対 (従属項目) は、以下の条件が真である場合に対応します。

- ADD または SUBTRACT ステートメントで、両方のデータ項目は基本数字データ項目である。これ以外のデータ項目は無視されます。
- MOVE ステートメントで、データ項目のうち少なくとも 1 つが基本項目であり、その移動は移動規則に従って行われる。

- 両方の従属項目が同じ名前をもち、ID-1 および ID-2 までの (ID-1 および ID-2 は含まない) 同じ修飾子をもっている。
- 従属項目が、キーワード FILLER によって識別されない。
- ID-1 と ID-2 のどちらもレベル 66 またはレベル 88 の項目として記述されておらず、INDEX、POINTER、PROCEDURE-POINTER のどの項目の使用法でもない。 ID-1 と ID-2 のどちらも修正済みとして参照することができません。暗黙の修飾子のアプリケーションの後では、データ項目の名前は固有のものでなければなりません。
- 従属項目の記述には、REDEFINES、RENAMES、OCCURS、USAGE IS INDEX、USAGE IS POINTER、または USAGE IS PROCEDURE-POINTER のいずれかの文節は含まれない。その従属項目がグループであれば、その項目に従属する項目も無視されます。

  ただし、ID-1 および ID-2 は、それ自体が記述に REDEFINES 文節や OCCURS 文節を含む項目を含んでいたり、あるいはそれらに従属することがあります。
- ID-1 と ID-2 は、FILLER 項目に従属することがある。

  例えば、2 つのデータ階層が次のように定義されているとします。

```
05  ITEM-1 OCCURS 6 INDEXED BY X.
  10  ITEM-A PIC S9(3).
  10  ITEM-B PIC 99V9.
  10  ITEM-C PIC X(4).
  10  ITEM-D REDEFINES ITEM-C PIC 9(4).
  10  ITEM-E PIC 9(4) USAGE COMP.
  10  ITEM-F USAGE INDEX.
  10  ITEM-G PIC X(4).
05  ITEM-2.
  10  ITEM-A PIC 99.
  10  ITEM-B PIC 9V9.
  10  ITEM-C PIC A(4).
  10  ITEM-D PIC 9(4).
  10  ITEM-E PIC 9(9) USAGE COMP.
  10  ITEM-F USAGE INDEX.
  10  ITEM-G PIC X(4).
```

  その後、ADD CORR ITEM-2 TO ITEM-1(X) が指定されていると、以下のようになります。
  - ITEM-A および ITEM-A(X); ITEM-B および ITEM-B(X); ITEM-E および ITEM-E(X) は対応していると見なされて加えられます。
  - ITEM-C および ITEM-C(X); ITEM-G および ITEM-G(X) は数字ではないので含まれていません。
  - ITEM-D および ITEM-D(X) は加えられません。その理由は、ITEM-D(X) がそのデータ記述の中に REDEFINES 文節を含んでいるからです。
  - ITEM-F および ITEM-F(X) は、USAGE IS INDEX として定義されているので加えられません。

(デフォルトの) *PRTCORR コンパイラー・オプションまたは PROCESS ステートメントの PRTCORR オプションを使用すると、コンパイラーは、CORRESPONDING 句が入っている各ステートメントの後のコンパイラー・リスト内にコメント行を挿入します。これらのコメント行は、次の有効なソース・ステートメントの直前で印刷され、指定されたグループ内部で影響を受ける基本項目を識別します。

GIVING 句: GIVING という語の後にある ID により参照されるデータ項目は、算術演算の計算結果に設定されます。 この ID は、それ自体は計算に関与しないため、数字編集項目とすることができます。

GIVING 句 - 形式

ROUNDED 句: 小数点位置合わせが行われた後、算術演算結果の小数部の桁数と、結果の ID の小数部に用意されている桁数とが比較されます。

結果の小数部のサイズが、それを記憶するために用意された桁数を超えているときは、ROUNDED が指定されていない限り切り捨てが行われます。 ROUNDED が指定されている場合は、結果の ID の最下位の桁は、超過した部分の最上位桁が 5 以上の場合は 1 つ増加します。正確に丸めることができる最大の桁数は 62 です。

結果の ID が右端に P (複数個) を持つ PICTURE 文節によって記述されており、また計算結果が指定された整数桁数を超える場合、ストレージが割り振られている最右端の整数桁に対して、丸めまたは切り捨てが行われます。

IBM Extension

浮動小数点算術演算では、ROUNDED 句はどのような影響も与えません。浮動小数点算術演算の結果は*常に* 丸められます。浮動小数点演算式について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

End of IBM Extension

SIZE ERROR 句:

SIZE ERROR 句 - 形式

サイズ・エラー条件は、次の 3 つの状況で起こる可能性があります。

- 小数点位置合わせが行われた後の算術演算の結果の絶対値が、結果のフィールドに収容できる最大値を超えたとき
- ゼロ除算が行われたとき
- 指数式では、以下の場合
  - 指数 0 で累乗されたゼロ
  - 負の指数で累乗されたゼロ
  - 分数の指数で累乗された負数

サイズ・エラー条件は最終結果に適用され、中間結果には適用されません。

結果の ID が USAGE IS BINARY、COMP-4、または COMP-5 で定義される場合、それに収容できる最大
値は、関連した 10 進数の PICTURE 文字ストリングで暗黙に指定される最大値です。

ROUNDED 句が指定されている場合、サイズ・エラー検査の前に丸めが行われます。

サイズ・エラーが起こった場合、プログラムの次の処置は、ON SIZE ERROR 句が指定されているかどうかによって異なります。

ON SIZE ERROR 句が指定されて**いる**ときにサイズ・エラー条件が起こると、サイズ・エラーの影響を受ける結果の ID の値はそのまま変わりません。すなわち、誤った結果が受け入れ ID の中に入れられることはありません。その他の結果の ID の値は、それらの ID に関してサイズ・エラーが起こっていなければ影響を受けることはありません。算術演算の実行完了後、ON SIZE ERROR 句の中の命令ステートメントが実行されます。 ON SIZE ERROR 句に指定された命令ステートメントの完了時に明示的な制御の転送が実行されない場合、制御は算術ステートメントの最後に渡され、NOT ON SIZE ERROR 句が指定されていてもそれは無視されます。

ON SIZE ERROR 句が指定されておらず、算術ステートメントで指定した算術演算の実行後にサイズ・エラー条件が存在する場合は、その影響を受けた、結果の ID の値は未定義となります。その他の結果の ID の値は、それらの ID に関してサイズ・エラーが起こっていなければ影響を受けることはありません。算術演算の完了後、制御は算術ステートメントの最後に渡され、NOT ON SIZE ERROR 句が指定されていてもそれは無視されます。

ADD CORRESPONDING と SUBTRACT CORRESPONDING ステートメントの場合は、個々の算術演算でサイズ・エラー条件が起こっても、それぞれの加算または減算のすべてが完了するまで、ON SIZE ERROR 命令ステートメントは実行されません。

NOT ON SIZE ERROR: NOT ON SIZE ERROR 句が指定されていて、算術演算の実行後にサイズ・エラー条件が存在しない場合、NOT ON SIZE ERROR 句が実行されます

ON SIZE ERROR 句と NOT ON SIZE ERROR 句の両方が指定され、実行される句の中のステートメントに明示的な制御の転送が入っていないとき、必要ならば、暗黙の制御の転送が、算術ステートメントの終わりの句の実行後に行われます。

オーバーラップするオペランド: ステートメントでの項目のやり取りが記憶域の一部または全部を共用していながら、それらが同一のデータ記述項目で定義されていない場合には、そのようなステートメントの実行の結果は予測できません。さらに、項目のやり取りが同一のデータ記述項目により定義されているステートメントの中に、結果が予測不可能となるものがあります。これらの例は、前述のステートメントに関連した一般規則で言及されています。

算術ステートメント: 演算ステートメントは計算のために使用します。 個々の演算は、ADD、SUBTRACT、MULTIPLY、および DIVIDE ステートメントによって指定されます。これらの演算は、COMPUTE ステートメントを使って 1 つの式の中で記号によって組み合わせることができます。

算術ステートメントのオペランド: 算術ステートメントの各オペランドのデータ記述は、同じでなくてもかまいません。 計算の間、コンパイラーは必要なデータ変換や小数点の位置合わせを行います。

オペランドのサイズ: 各オペランドの最大サイズは、10 進数で 18 桁です。

IBM Extension

ゾーン 10 進数または内部 10 進オペランドの最大サイズは 10 進数で 63 桁です。

End of IBM Extension

各オペランドから、**オペランドの合成**での 10 進数の桁数を決定できます。オペランドの合成は、小数点の位置にオペランドを合わせて、さらにそれらオペランドを重ね合わせた結果得られる仮想のデータ項目です。

例えば、各項目がデータ部で次のように定義されているとします。

```
A  PICTURE 9(7)V9(5).
B  PICTURE 9(11)V99.
C  PICTURE 9(12)V9(3).
```

次のステートメントを実行すると、オペランドの合成は 17 桁の 10 進数になります。

```
ADD A B TO C
```

これは、次のような暗黙の記述を持っています。

```
Composite-of-Operands PICTURE 9(12)V9(5).
```

オペランドを合成したものが 18 桁以下であれば、実行時に有効数字が失われないように十分な桁が保たれます。

IBM Extension

(デフォルト) コンパイラー・オプション *NOEXTEND または PROCESS ステートメント・オプション NOEXTEND が指定されている場合、オペランドの合成は 10 進数で最大 30 桁までにできます。

算術計算モード・コンパイラー・オプション *EXTEND31 または PROCESS ステートメント・オプション EXTEND31 が指定されている場合、オペランドの合成は 10 進数で最大 31 桁まで可能です。

| 算術計算モード・コンパイラー・オプション *EXTEND31FULL または PROCESS ステートメント・オプション EXTEND31FULL が指定されている場合、オペランドの合成は 10 進数で最大 34 桁まで可能です。

算術計算モード・コンパイラー・オプション *EXTEND63 または PROCESS ステートメント・オプション EXTEND63 が指定されている場合、オペランドの合成は 10 進数で最大 63 桁まで可能です。

注: オペランドの合成が指定された最大値を超えると、実行中に有効数字が失われる場合があります。

End of IBM Extension

以下の表は、算術ステートメント内でオペランドの合成に使用できる 10 進数の最大桁数を示しています。

コンパイラー・オプション / 処理ステートメント	合成の最大長 (10 進数)
*NOEXTEND/NOEXTEND	18 **IBM Extension** 30 **End of IBM Extension**
*EXTEND31/EXTEND31	18 **IBM Extension** 31 **End of IBM Extension**

*EXTEND31FULL/EXTEND31FULL	**IBM Extension** 34 **End of IBM Extension**
*EXTEND63/EXTEND63	**IBM Extension** 63 **End of IBM Extension**

以下のリストには、算術ステートメントにおけるオペランドの合成の決め方が示されています。

ステートメント
オペランドの合成の決め方

ADD 与えられたステートメントの中のすべてのオペランド (GIVING という語の後に続くものは除く) を重ね合わせることによって決める。

COMPUTE
制約は適用されない。

DIVIDE
REMAINDER データ項目以外のすべての受け入れデータ項目を重ね合わせることによって決める。

MULTIPLY
すべての受け入れデータ項目を重ね合わせることによって決める。

SUBTRACT
与えられたステートメントの中のすべてのオペランド (GIVING という語の後に続くものは除く) を重ね合わせることによって決める。

どの算術ステートメントの場合にも、データを定義する際には、所要の精度の最終結果が得られるよう、十分な桁数と小数部分を指定することが重要です。演算の精度の詳細については 9-3 ページの『付録 B. 中間結果と算術精度』を参照してください。

複数の結果: 1 つの算術ステートメントが複数の結果を持つとき、実行は概念的に次のように進められます。

- このステートメントは、すべての算術演算を行って受け入れ項目に入れるべき 1 つの結果を出し、その結果を一時的な場所に保管します。
- 一連のステートメントは、この一時結果の値を、それぞれの受け入れフィールドに移すか、またはそれぞれの受け入れフィールドと組み合わせます。これらのステートメントは、複数の結果がリストされているのと同じ左から右への順序で書かれているものと見なされます。

例えば、次のステートメントを実行するとします。

```
ADD A, B, C, TO C, D(C), E.
```

これは、次の一連のステートメントを実行するのと同じです。

```
ADD A, B, C GIVING TEMP.
ADD TEMP TO C.
ADD TEMP TO D(C).
ADD TEMP TO E.
```

上の例では、TEMP はコンパイラーによって与えられる一時的な結果フィールドです。 D(C) との加算が行われるとき、添え字 C に C の新しい値が入ります。

注: 算術ステートメントの実行中に生成される中間結果はシステム特有のものであり、プログラムの可搬性に影響します。算術ステートメントとして COMPUTE の代わりに ADD、SUBTRACT、MULTIPLY、および DIVIDE を独立して使用すれば、不整合な結果が生じる危険性を少なくすることができます。

データ操作ステートメント: 以下の COBOL ステートメントはデータを移動および検査します。
ACCEPT、INITIALIZE、INSPECT、MOVE、READ、RELEASE、RETURN、REWRITE、SET、STRING、UNSTRING、および WRITE。

IBM Extension

XML PARSE、XML GENERATE。

End of IBM Extension

入出力ステートメント: COBOL 入出力ステートメントは、外部メディアに保管されているファイルへ(または、ファイルから) データを転送し、また入出力装置から得られる (または入出力装置へ送られる) 少量のデータを制御します。

COBOL では、プログラムで使用できるファイル・データの単位はレコードなので、ユーザーはそのようなレコードに対してだけ関心をもてばよいことになります。バッファーや内部記憶装置へのデータの移動、妥当性検査、エラー訂正 (可能な場合)、ブロック化と非ブロック化、ボリューム切り替え手順などの処理は、自動的に行われます。

環境部とデータ部でファイルがどのように記述されているかによって、手続き部で使用できる入出力ステートメントが決まります。

ファイル構成サポートの要約については 9-23 ページの『付録 F. ファイル構造サポートの要約およびファイル状況キーの値』を参照してください。

次の節の説明では、**ボリューム**および**リール**という用語を使用します。**ボリューム**とは、ユニット・レコード装置以外のすべての入出力装置を指します。**リール**は、テープ装置だけに適用されます。順次アクセス・モードでの直接アクセス装置の取り扱い方法は、テープ装置の取り扱い方法と論理的には同じです。

共通の処理機能: 共通の処理機能の中には、複数の入出力ステートメントに適用されるものがいくつかあります。共通の処理機能は次のとおりです。

- 状況キー
- INVALID KEY 条件
- INTO/FROM ID 句
- ファイル位置標識

状況キー: FILE STATUS 文節が FILE-CONTROL 項目に指定されている場合、そのファイルに対する要求の実行時に、指定した状況キー (FILE STATUS 文節で指名した 2 文字のデータ項目) の中に値が入れられます。 入れられる値はその要求の状況を示すものです。値が状況キーに入れられた後で、その要求に関連のある EXCEPTION/ERROR 宣言または INVALID KEY/AT END 句が実行されます。

状況キーの最初の文字は状況キー 1 (上位桁)、また 2 番目の文字は状況キー 2 (下位桁) として知られています。考えられる値の組み合わせとその意味は 9-28 ページの表 9-9 に示されています。

INVALID KEY 条件: INVALID KEY 条件は、START、READ、WRITE、 REWRITE、または DELETE ステートメントの実行中に発生することがあります。

(条件の発生理由の詳細は 7-41 ページの『手続き部のステートメント』を参照してください。) 無効キー条件が発生すると、その条件を引き起こした入出力ステートメントは正常に実行されません。無効キー条件が入出力操作の後に存在している場合、以下の処置がとられます。

1. 該当するファイル状況文節 (該当する USE プロシージャーではない) が存在している場合は、ファイル状況は変更され、制御がプログラムに戻されます。
2. INVALID KEY 句が指定されている場合には、その句の命令ステートメントに制御が渡されます。
3. ファイルに明示または暗黙の EXCEPTION/ERROR プロシージャーが指定されている場合は、それが実行されます。そのようなプロシージャーが指定されていない場合には、結果は予測できません。
4. ファイル状況文節、USE プロシージャー、またはエラー処理の INVALID KEY 句が存在しないときは、実行時メッセージが出されて、プログラムを終了させるかプログラムに戻るかのオプションがユーザーに与えられます。

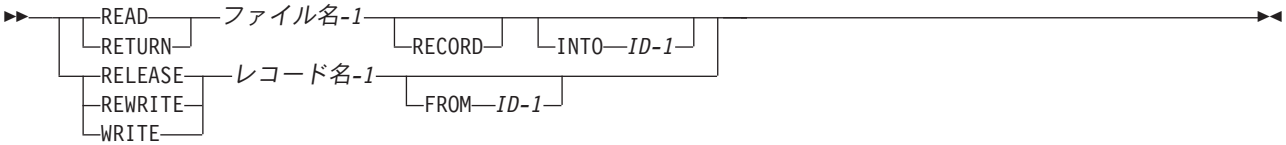
入出力操作の後に無効キー条件が存在しない場合、指定された INVALID KEY 句は無視され、次の処置がとられます。

1. 無効キー条件ではない例外条件が存在する場合、制御は、USE AFTER EXCEPTION プロシージャーが実行された後に、USE ステートメントの規則に従って転送されます。
2. 例外条件が存在しない場合には、NOT INVALID KEY 句に指定される入出力ステートメントまたは命令ステートメントの終わりにこれらが指定されれば、制御が転送されます。

エラー処理および INVALID KEY 句の役割について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の例外処理およびエラー処理に関する章を参照してください。

INTO/FROM ID 句: この句は READ、RETURN、RELEASE、REWRITE、および WRITE ステートメントに有効です。指定する ID は、 作業用ストレージ・セクション、ローカル・ストレージ・セクション、またはリンケージ・セクションの項目の名前、あるいは以前にオープンした他のファイルのレコード記述の名前でなければなりません。レコード名と ID は、同じ記憶域を参照してはなりません。

INTO/FROM ID 句 - 形式



INTO 句を指定して READ または RETURN ステートメントを実行すると、その結果は、次に示す規則が (この順序どおりに) 適用されたのと同じことになります。

1. INTO 句を指定していない同じ READ または RETURN ステートメントを実行する。
2. CORRESPONDING 句が指定されていない MOVE ステートメントの規則に従って、現在のレコードをレコード域から ID-1 に移動させる。現在のレコードの大きさは、RECORD 文節内に指定されている規則によって決定されます。ファイル記述記入項目に RECORD IS VARYING 文節が含まれている場合には、暗黙の MOVE はグループの移動となります。 READ または RETURN ステートメントが正常に実行されなかった場合には、暗黙の MOVE ステートメントは実行されません。レコードが読み取ら

れるか戻された後で、かつそのレコードがデータ項目に転送される直前に、ID-1 と関連する添え字付け、または参照変更の値が求められます。レコードは、レコード域と ID-1 の両方で使用できます。

FROM 句が指定されている RELEASE、REWRITE、または WRITE ステートメントが実行されると、その結果は、次に示すステートメントが (この順序どおりに) 実行されたのと同じことになります。

1. 以下のステートメントを

   ```
   MOVE identifier-1 TO record-name-1
   ```

   MOVE ステートメントに指定された規則に従って実行する。
2. FROM 句が指定されていない同じ RELEASE、REWRITE、または WRITE ステートメントを実行する。

RELEASE、REWRITE または WRITE ステートメントの実行が完了すると、ID-1 内の情報が使用可能になりますが、 SAME RECORD AREA 文節として指定されているものを除いて、レコード名-1 内の情報は使用することはできません。

ファイル位置標識: ファイル位置標識は概念的なエンティティーであり、本書では、入出力操作における特定の順序の中で所定のファイル内でアクセスされる次のレコードを正確に把握しやすくするために使用しています。ファイル位置標識の概念は、出力または拡張モードでオープンするファイルには使用できません。ファイル位置標識の設定は、次のような OPEN、READ、RETURN、ROLLBACK および START ステートメントの影響だけを受けます。

- OPEN ステートメントは、ファイル位置標識をファイルの最初のレコードに位置付けます。

IBM Extension

ファイル位置標識は、データベース・ファイルで変更 (OVRDBF) コマンドの POSITION パラメーターを使用することにより、ファイル内の任意のレコードに置くことができます。

End of IBM Extension

- 順次アクセス READ ステートメント、または動的アクセス READ NEXT ステートメントについては、次の考慮事項が適用されます。
  - OPEN または START ステートメントでファイル位置標識を位置付けた場合、ファイル位置標識によって識別されたレコードが使用できます。このレコードがすでに存在していないと、存在する次のレコードが使用可能になります。
  - 前の READ ステートメントがファイル位置標識を位置付けた場合、ファイル位置標識はファイル内の次の既存レコードを指すように更新され、そのレコードが使用可能になります。

IBM Extension

- 動的アクセス READ FIRST ステートメントでは、ファイル位置標識はファイルの最初のレコードを示す位置に置かれ、そのレコードは使用可能になります。
- 動的アクセス READ LAST ステートメントでは、ファイル位置標識はファイルの最後のレコードを示す位置に置かれ、そのレコードは使用可能になります。
- 動的アクセス READ PRIOR ステートメントでは、ファイル位置標識はファイルの前の既存レコードを示す位置に置かれ、そのレコードは使用可能になります。

End of IBM Extension

- RETURN ステートメントでは、次の考慮事項が適用されます。

- 最初の RETURN ステートメントは、ファイル位置標識をファイルの最初のレコードに位置付け、そのレコードが使用可能になります。
- 前の RETURN ステートメントがファイル位置標識を位置付けた場合、ファイル位置標識はファイル内の次の既存レコードを指すよう更新され、そのレコードが使用可能になります。

IBM Extension

- ROLLBACK ステートメントでは、コミットメント制御下にあるファイルに次の考慮事項が適用されます。
  - ROLLBACK ステートメントは、ファイル位置標識を先のコミットメント境界のポインターの位置に設定します。順次処理を行っている場合には、このことを覚えておくことが大切です。
  - ファイル位置標識は、ファイルがオープンされてから COMMIT ステートメントが出されていない場合には、OPEN のポインターの位置に設定されます。
  - ファイル位置標識は、コミットメント制御下にはあるがオープンされていないファイルには定義されません。

End of IBM Extension

- START ステートメントは、START ステートメントで指定された暗黙または明示の比較条件を満足させるファイル内の最初のレコードに、ファイル位置標識を位置付けます。

ファイル位置標識の概念は、ランダムのアクセス・モードのファイルまたは TRANSACTION ファイルには使用できません。

DB-FORMAT-NAME 特殊レジスター:

IBM Extension

入出力ステートメントの実行後、FORMATFILE または DATABASE ファイルについては、DB-FORMAT-NAME 特殊レジスターが次の規則に従って修正されます。

- READ、WRITE、REWRITE、START、または DELETE 操作が正常に完了した後、I-O 操作で使用されるレコード形式名は、暗黙に特殊レジスターへ移動されます。
- 入出力操作が正常に行われなかった後、DB-FORMAT-NAME には、最後に正常に行われた入出力操作に使用されたレコード形式名が入ります。
- DB-FORMAT-NAME は、最も外側のプログラムにおいて、PICTURE X(10) および GLOBAL として暗黙に定義されます。

英数字の引数が認められている場合はいつでも、DB-FORMAT-NAME 特殊レジスターを関数内で指定できます。

End of IBM Extension

プロシージャー分岐ステートメント: 手続き部のステートメント、文、および段落は順番に実行されますが、(以下に列挙した) プロシージャー分岐ステートメントが使用されている場合は例外です。

- ALTER
- EXIT
- GO TO
- PERFORM

手続き部のステートメント

ACCEPT ステートメント

ACCEPT ステートメントの実行によって、指定した ID へデータが転送されます。入力データに対して編集またはエラー・チェックは行われません。

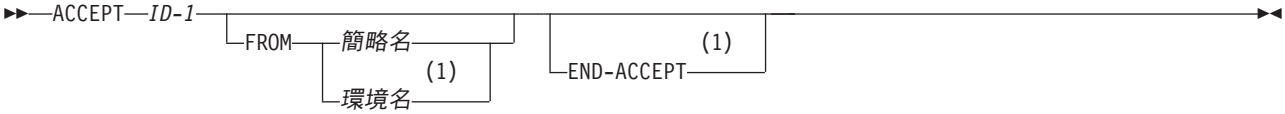
IBM Extension

- 形式 3 - フィードバック
- 形式 4 - 内部データ域
- 形式 5 - PIP データ域
- 形式 6 - 属性データ域
- 形式 7 - ワークステーション I/O
- 形式 8 - セッション I/O
- 形式 9 - データ域

End of IBM Extension

形式 1 - データの転送

ACCEPT ステートメント - 形式 1 - データ 転送



注:

1 IBM 拡張

形式 1 は、入力装置から ID-1 へデータを転送するために使用します。受け入れデータは、文字位置を左端そろえにした文字ストリングとして転送されます。データ変換は行われません。 ID-1 のサイズが入力装置のレコード長よりも大きい場合には、1 レコードの転送が完了した後で追加データが要求されます。追加データは、以前に装置から転送された最後の文字のすぐ右側の位置から ID-1 に転送されます。この処理は、ID-1 が満杯になるまで継続されます。転送中に、装置レコードが ID-1 を埋めるのに必要な文字数以上を保持すると、あふれたデータは切り捨てられます。

すべてのデータは文字ストリングとして転送されるため、ID-1 は通常、USAGE に DISPLAY を指定して明示的または暗黙的に定義されます。ただし、ACCEPT ステートメントは、入力装置上で ID-1 の内部表現に対応する形式でデータを入力できる場合に、他の形式でデータを処理します。

プログラム内にオペレーターの介入 (所定のメッセージ、コード、または例外標識の提示) を必要とする例外状況が発生した場合には、形式 1 を使用すると便利です。もちろん、この場合は、応答用の適切なメッセージがオペレーターに与えられていなければなりません。

ID-1
受け入れデータ項目。

IBM Extension

ID-1 の説明に TYPE 文節が含まれる場合は、その文節で参照されるタイプ名は基本項目でなければなりません。

End of IBM Extension

IBM Extension

ID-1 は USAGE で DISPLAY-1 を定義する必要があります。すなわち、DBCS または DBCS 編集の項目でもかまいません。この場合さらに、入力装置上のデータはシフトアウトおよびシフトイン文字で区切る必要があります。これらの文字はデータが転送されるときに除去されます。

End of IBM Extension

IBM Extension

ID-1 は NATIONAL 項目としても定義できます。受け入れられたデータは、ジョブの現行 CCSID によって指定されるコード・セットから変換されます。

End of IBM Extension

IBM Extension

ID-1 は、日付、時刻、タイム・スタンプの項目にはできません。

End of IBM Extension

簡略名

SPECIAL-NAMES 段落に指定する必要があります。この段落は、入力装置を参照する環境名と関連付けられています。入力装置は、対話式ジョブで使用されるワークステーション、バッチ・ジョブのジョブ入力ストリーム、またはシステム・オペレーターのコンソールにできます。

IBM Extension

環境名

簡略名の代わりに環境名 CONSOLE または SYSIN を指定できます。

End of IBM Extension

入力データ・ソース: 入力データ・ソースは、次のようにプログラム開始の方式によって異なります。

プログラム開始の方法	データ・ソース (入力装置)		
	環境名に関連付けられた場合		FROM 句を省略した場合
	CONSOLE または SYSTEM-CONSOLE	SYSIN または REQUESTOR	
BATCH	システム・オペレーターのメッセージ・キュー	ジョブ入力ストリーム	ジョブ入力ストリーム
INTERACTIVE	システム・オペレーターのメッセージ・キュー	ワークステーション	ワークステーション

ジョブ入力ストリームは CL コマンドに付随するデータから構成されます。入力ストリームの中にデータがない場合、または ID-1 に入れるにはデータが不十分である場合には、例外条件が発生します。

入力が**ジョブ入力ストリーム**からのものである場合には、次の規則が適用されます。

- 入力レコードの大きさは 80 文字と見なされます。
- ID-1 の長さが 80 文字以下である場合には、入力データは入力レコード内の先頭文字として現れなければなりません。 ID-1 の長さを超える文字は切り捨てられます。
- ID-1 が 80 文字よりも長い場合には、ID-1 の記憶域が満たされるまで引き続いて入力レコードが読み取られます。 ID-1 の長さ 80 文字の整数倍になっていない場合には、最後の入力レコードが切り捨てられます。

装置が**ワークステーション**の場合には、入力レコードの大きさは 100 です。また装置が**システム・オペレーターのメッセージ・キュー**の場合には、入力レコードの大きさは 58 です。次のステップが発生します。

1. プログラム名、テキスト “AWAITING REPLY POSITION(S)”、および開始位置と終了位置が含まれているシステム作成照会メッセージが、システム・オペレーターのメッセージ・キュー、またはワークステーションのオペレーターにあてて自動的に送られます。 ACCEPT 画面上に前の表示内容が現れることがあります。
2. 実行が延期されます。
3. 応答が ID-1 の中へ転送され、オペレーターがステップ 1 の照会に対する応答を行った後で処理が再開されます。応答の値が大文字または小文字で入力されたとおりにプログラムで使用可能となります。
4. ID-1 が入力レコード・サイズよりも長い場合には、ID-1 の長さに達するまで引き続いて入力レコードが読み取られます (ステップ 1 ～ 3)。

着信応答が ID-1 よりも長い場合には、ID-1 の長さを超えた文字位置で切り捨てられます。

注: READ ステートメントに使用した装置と同じ装置を使用した場合には、実行結果は予測できません。

コーディング例: 次に示すのは、ジョブ入力ストリームを含んでいるバッチ・ジョブ・ファイル・メンバーの例です。

```
//BCHJOB   JOB(ADD021) JOBD(QUSER/ACCTEST)
CALL       PGM(QSYS/ACCPT1X)
123456789012345
//ENDBCHJOB
```

次に示すのは、ジョブ入力ストリームを読み取るために形式 1 の ACCEPT ステートメントを使用する COBOL プログラムの例です。

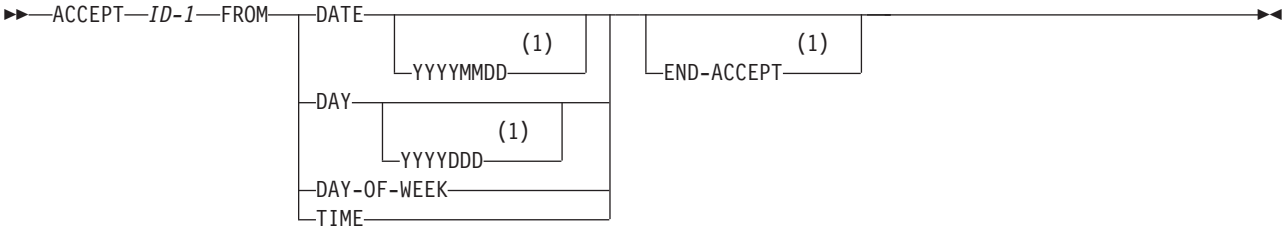
```
IDENTIFICATION DIVISION.
PROGRAM-ID. ACCPT1X.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-ISERIES.
OBJECT-COMPUTER. IBM-ISERIES.
DATA DIVISION.
WORKING-STORAGE SECTION.
77  TRANS-DATA   PIC X(15).
PROCEDURE DIVISION.
BEGIN.
    ACCEPT TRANS-DATA.
    DISPLAY TRANS-DATA.
    STOP RUN.
```

バッチ・ジョブ・ファイル・メンバーを使って ACCPT1X を呼び出す場合、ACCEPT ステートメントは、CALL コマンドのすぐ後にある行からバッチ・ジョブ・ファイル・メンバーを読み取ります。それによって "123456789012345" が TRANS-DATA の中に受け入れられることになります。

形式 2 - システム情報の転送

指定された概念上のデータ項目 DATE、DAY、DAY-OF-WEEK、または TIME に含まれているシステム情報は、ID に転送できます。 DATE、DAY、DAY-OF-WEEK および TIME は概念上のデータ項目なので、COBOL プログラム内では記述されません。この転送を実行する場合は、CORRESPONDING 句のない MOVE ステートメントの規則に従わなければなりません。 7-139 ページの『MOVE ステートメント』を参照してください。

ACCEPT ステートメント - 形式 2 - システム情報転送



注:

1 IBM 拡張

ID-1
受け入れデータ項目。

IBM Extension

ID-1 の説明に TYPE 文節が含まれる場合は、その文節で参照されるタイプ名は基本項目でなければなりません。

End of IBM Extension

形式 2 は、現行のシステム日付およびシステム時刻にアクセスする場合に使用します。オブジェクト・プログラムが特別に実行された日付および時刻を識別する場合にこの形式を使用すると便利です。またこの形式を使用して、ヘッディングおよびフッティングに日付を表示することもできます。

注: 現在日付および時刻も、CURRENT-DATE 組み込み関数を使用してアクセスすることができます (7-291 ページの『CURRENT-DATE』を参照してください)。

DATE、DAY、DAY-OF-WEEK、および TIME: 概念データ項目の DATE、DAY、DAY-OF-WEEK および TIME は暗黙のうちに USAGE DISPLAY をもちます。

DATE (YYYYMMDD 句なし)
暗黙のうちに PICTURE 9(6) をもちます。

データ・エレメントは、左から右へ次のように並んでいます。

2 桁の西暦の年

2 桁のその年の月
2 桁のその月の日

したがって、1963 年 11 月 16 日は次のように表されます。

```
631116
```

IBM Extension

DATE (YYYYMMDD 句あり)

暗黙のうちに PICTURE 9(8) をもちます。

データ・エレメントは、左から右へ次のように並んでいます。

```
4 桁の西暦の年 (グレゴリオ暦)

2 桁のその年の月
2 桁のその月の日
```

したがって、1963 年 11 月 16 日は次のように表されます。

```
19631116
```

End of IBM Extension

DAY (YYYYDDD 句なし)

暗黙のうちに PICTURE 9(5) をもちます。

データ・エレメントは、左から右へ次のように並んでいます。

```
2 桁の西暦の年
3 桁のその年の通算日数
```

したがって、1988 年 12 月 25 日は次のように表されます。

```
88360
```

IBM Extension

DAY (YYYYDDD 句あり)

暗黙のうちに PICTURE 9(7) をもちます。

データ・エレメントは、左から右へ次のように並んでいます。

```
4 桁の西暦の年 (グレゴリオ暦)
3 桁のその年の通算日数
```

したがって、1995 年 12 月 31 日は次のように表されます。

```
1995365
```

End of IBM Extension

DAY-OF-WEEK

暗黙のうちに PICTURE 9(1) をもちます。

次のように単一のデータ・エレメントで表されます。

```
1 は月曜日
2 は火曜日
3 は水曜日
4 は木曜日
5 は金曜日
6 は土曜日
7 は日曜日
```

したがって木曜日は 4と表されます。

TIME

暗黙のうちに PICTURE 9(8) をもちます。

データ・エレメントは、左から右へ次のように並んでいます。

```
2 桁の時 (24 時間表示)
2 桁の分
2 桁の秒
2 桁の 100 分の 1 秒表示
```

したがって、午後 2 時 41 分 12.25 秒は次のように表されます。

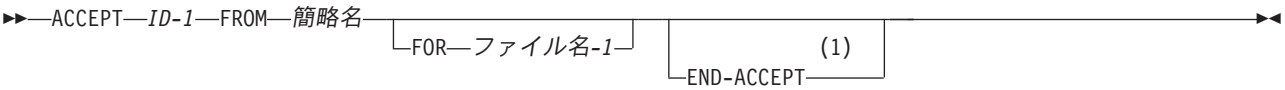
```
14411225
```

形式 3 - フィードバック

IBM Extension

この形式は、活動ファイルから ID へフィードバック情報を転送する場合に使用します。

ACCEPT ステートメント - 形式 3 - フィードバック



注:

1 IBM 拡張

ID-1 は、固定長グループ項目または基本英字項目か、基本英数字項目か、外部 10 進数項目のいずれかが可能です。 ID-1 には日時項目は許されません。 ID-1 の説明に TYPE 文節が含まれる場合は、その文節で参照されるタイプ名は基本項目でなければなりません。

ID-1 は、内部浮動小数点データ項目でも外部浮動小数点データ項目でも可能です。

ファイル名-1 は、FD 記入項目に定義される必要があり、さらに ACCEPT ステートメントの実行より前にオープンされていなければなりません。ファイル名-1 がオープンされていない場合、ID-1 の内容は変更されません。

FROM 句は、SPECIAL-NAMES 段落中の OPEN-FEEDBACK または I-O-FEEDBACK の環境名に関連付けなければならない簡略名を指定します。

FOR 句が指定された場合には、フィードバック情報はこの句で指定されたファイルからのものです。 FOR 句が指定されなかった場合には、フィードバック情報はいちばん最後にオープンされたファイルからのものか、または、現行のプログラムのいちばん最後の入出力操作で使用されたファイルからの情報です。

I-O-FEEDBACK および OPEN-FEEDBACK 領域の説明については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。フィードバック域に含まれているフィールドのレイアウトおよび説明については、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) 内のカテゴリー『データベースおよびファイル・システム』のセクション『*DB2 for i*』を参照してください。

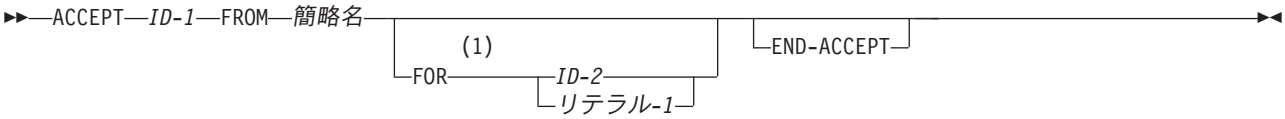
End of IBM Extension

形式 4 - 内部データ域

IBM Extension

この形式は、ジョブごとに作られたシステム定義の内部データ域から ID-1 へデータを転送する場合に使用します。

ACCEPT ステートメント - 形式 4 - 内部データ域



注:

1 構文検査だけ行われます。

この形式が適用可能なのは、SPECIAL-NAMES 段落の中で簡略名が環境名 LOCAL-DATA と関連付けられている場合だけです。

ID-1 への転送は、MOVE ステートメントで、CORRESPONDING 句なしのグループ転送の場合の規則に従って行われます。 ID-1 には日時項目は許されません。 ID-1 の説明に TYPE 文節が含まれる場合は、その文節で参照されるタイプ名は基本項目でなければなりません。

ID-1 は、内部浮動小数点データ項目でも外部浮動小数点データ項目でも可能です。

ID-1 は DBCS または国別データ項目にすることができます。

FOR 句が指定された場合、この句は、コンパイル時に構文上の検査をされますが、プログラムの実行中にはコメントとして扱われます。リテラル-1 または ID-2 の値は、内部データ域に関連付けられている装置のプログラム装置名を示します。各ジョブに設けられる内部データ域は 1 つだけあり、1 つのジョブで使用される装置は、すべて同一の内部データ域をアクセスします。リテラル-1 を指定する場合、リテラル-1 は、非数字で、長さは 10 文字以内でなければなりません。 ID-2 (指定されている場合) は、英数字データ項目で、10 文字以下の長さでなければなりません。内部データ域について詳しくは、「*CL プログラミング・マニュアル*」を参照してください。

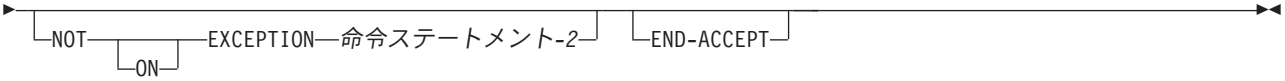
End of IBM Extension

形式 5 - プログラム初期設定パラメーター

IBM Extension

この形式は、PIP (プログラム初期設定パラメーター) データ域から ID へデータを転送するのに使用されます。

ACCEPT ステートメント - 形式 5 - PIP データ域



この形式を適用できるのは、SPECIAL-NAMES 段落内の簡略名を環境名 PIP-DATA と関連付ける場合に限られます。

ID-1 への転送は、MOVE ステートメントで、CORRESPONDING 句なしのグループ転送の場合の規則に従って行われます。 ID-1 には日時項目は許されません。 ID-1 の説明に TYPE 文節が含まれる場合は、その文節で参照されるタイプ名は基本項目でなければなりません。

ID-1 は、内部浮動小数点データ項目でも外部浮動小数点データ項目でも可能です。

PIP データ域が存在する場合には、ジョブは事前開始ジョブであり、NOT ON EXCEPTION 句で指定された命令ステートメントはすべて処理されます。

PIP データ域が存在しない場合には、ジョブは事前開始ジョブではなく、ON EXCEPTION 句で指定された命令ステートメントがすべて処理されます。 PIP データ域が存在しない場合には、ジョブは事前開始ジョブではなく、ON EXCEPTION 句で指定された命令ステートメントがすべて処理されます。 ON EXCEPTION 句が存在せず、しかも PIP データ域も存在していない場合には、実行時メッセージが出されます。

明示範囲終了符号 END-ACCEPT は、ACCEPT ステートメントの有効範囲を区切る働きをします。 END-ACCEPT を使うと、条件 ACCEPT ステートメントを別の条件ステートメント内でネストさせることができます。 END-ACCEPT は命令 ACCEPT ステートメントとともに使用することもできます。詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

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。PIP データ域について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」および「*CL プログラミング*」を参照してください。

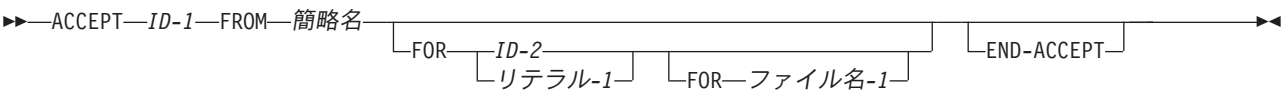
End of IBM Extension

形式 6 - 属性データ域

IBM Extension

ACCEPT ステートメントは、TRANSACTION ファイルに関連付けられている特定のプログラム装置についての情報 (属性データ) を検索します。

ACCEPT ステートメント - 形式 6 - 属性データ



この形式の ACCEPT ステートメントは、TRANSACTION の編成を持つファイルに対してしか使用することができません。 ID-1 には日時項目は許されません。 ID-1 の説明に TYPE 文節が含まれる場合は、その文節で参照されるタイプ名は基本項目でなければなりません。

ID-1 は、内部浮動小数点データ項目でも外部浮動小数点データ項目でも可能です。

ID-1 は DBCS または国別データ項目にすることができます。

ACCEPT の実行時に、ファイル-1 がオープンされていない場合、メッセージ LNR7205 が出されます。

簡略名は、SPECIAL-NAMES 段落の中で環境名 ATTRIBUTE-DATA に関連付けられていなければなりません。

ファイル名-1 が指定されていない場合、ACCEPT ステートメントのデフォルトのファイルは、FILE-CONTROL 段落の中の SELECT 文節で指定された最初の TRANSACTION ファイルです。

リテラル-1、または ID-2 の内容は (指定した場合)、属性データを使用できるプログラム装置名を示します。

ICF ファイルの場合、この装置は、(ICF 装置項目追加 (ADDICFDEVE) コマンド、ICF 装置項目変更 (CHGICFDEVE) コマンド、または ICF プログラム装置項目一時変更 (OVRICFDEVE) コマンドを使って) ファイルによって獲得されるように前もって定義しておかなければなりませんが、実際には、獲得されている必要はありません。ディスプレイ・ファイルの場合、ファイル作成、変更、または一時変更されたとき、そのファイルに対して OPEN が出される前に、DEV パラメーターにプログラム装置が前もって指定されていなければなりません。リテラル-1 を指定する場合、リテラル-1 は、非数字で、長さは 10 文字以内でなければなりません。 ID-2 の内容を指定する場合、その内容は、英数字データ項目で、長さは 10 文字以下でなければなりません。無効なプログラム装置名が指定された場合、メッセージ LNR7205 が出されて、実行が終了します。

FOR 句が両方とも省略されている (デフォルトの TRANSACTION ファイルが使用されている) 場合、ACCEPT ステートメントは、デフォルト・ファイルに対して READ、WRITE、REWRITE、または ACCEPT (属性データ) 操作が最後に実行されたプログラム装置を使用します。ファイルに対する直前の操作が OPEN であった場合、ACCEPT ステートメントは、ファイルがオープンされたときにファイルによって暗黙のうちに獲得されたプログラム装置を使用します。両方の FOR 句が省略されている場合、この形式の ACCEPT ステートメントを使用するためには、プログラム装置は前もって獲得されていなければなりません。装置の獲得について詳しくは、資料「*ICF Programming*」を参照してください。

プログラム装置属性は、CORRESPONDING 句のないグループ MOVE の規則に従って、該当する属性データ・フォーマットから ID-1 に転送されます。

属性データ形式: ACCEPT ステートメントによって検索される属性データは、そのデータおよび関連フィールドがワークステーションに適用可能なものであるか、あるいは通信装置に適用可能なものであるかによって決まります。形式の説明については 9-23 ページの『付録 F. ファイル構造サポートの要約およびファイル状況キーの値』を参照してください。

ATTRIBUTE-DATA 簡略名は、TRANSACTION ファイルによって獲得されたプログラム装置に関する情報を得るためにだけ 使用できます。属性データは、入出力操作が完了したときの状況または試行されたときの状況に関する情報は提供しません。入出力操作に関する情報を得る場合は、I-O-FEEDBACK または OPEN-FEEDBACK 簡略名を持つ形式 3 の ACCEPT ステートメントを使用してください。

End of IBM Extension

ワークステーション入出力

IBM Extension

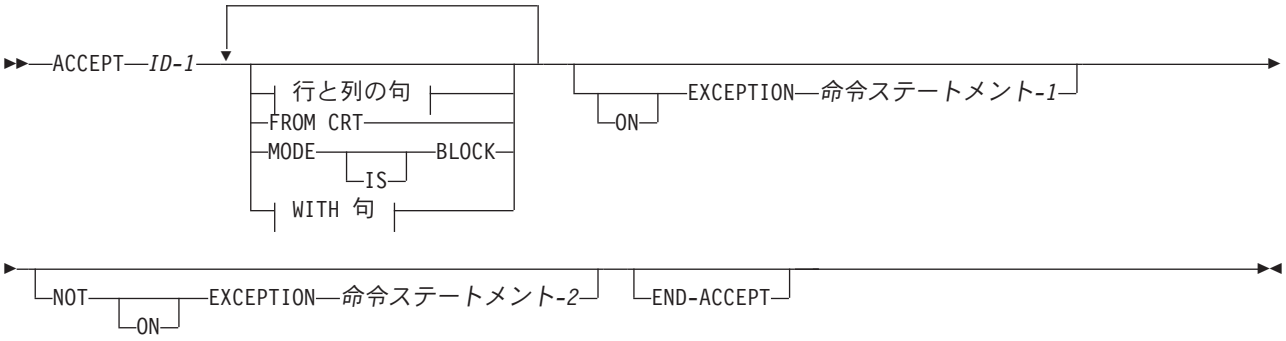
次のいずれかの場合には、ACCEPT ステートメントは**拡張** ACCEPT ステートメントであると見なされます。

- AT 句が付いているとき
- CRT オプションが指定された FROM 句が付いているとき
- MODE IS BLOCK 句が付いているとき
- WITH 句が付いているとき
- ON EXCEPTION 句または NOT ON EXCEPTION 句が付いているとき (しかも簡略名の PIP-DATA を指定していないとき)
- FROM 句は付いていないが、SPECIAL-NAMES 段落の中に CONSOLE IS CRT が指定されているとき

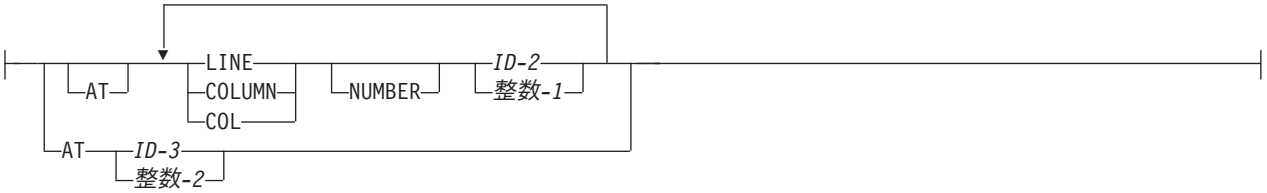
次の場合には、ACCEPT ステートメントは**標準** ACCEPT ステートメントであると見なされます。

- (FROM CRT 以外の) FROM 句が付いていて、しかも SPECIAL-NAMES 段落の中に CONSOLE IS CRT が指定されているとき、または
- FROM 句が付いていなくて、しかも CONSOLE IS CRT が指定されて**いない**とき。

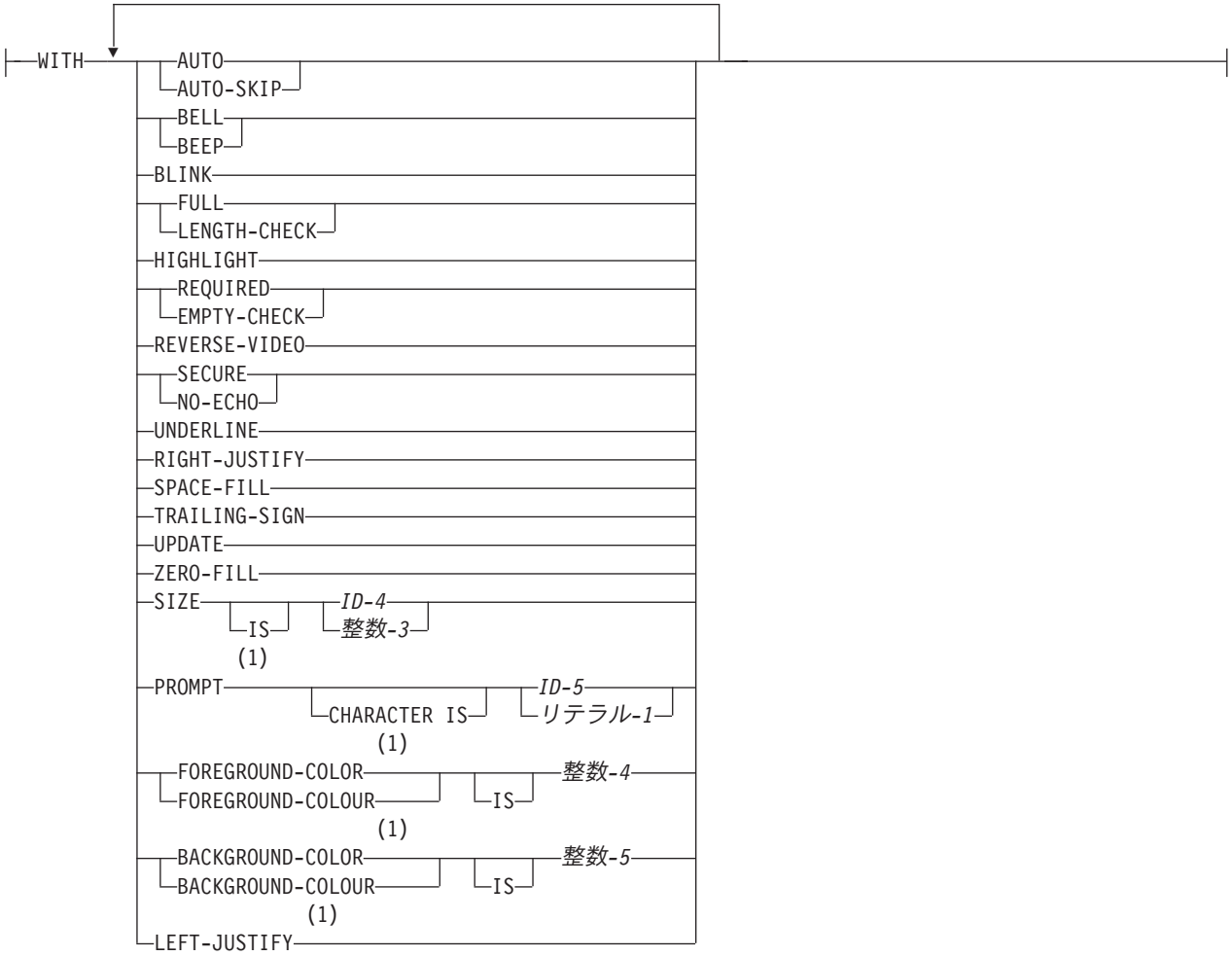
ACCEPT ステートメント - 形式 7 - ワークステーション I/O



行と列の句:



WITH 句:



注:

1 構文検査だけ行われます。

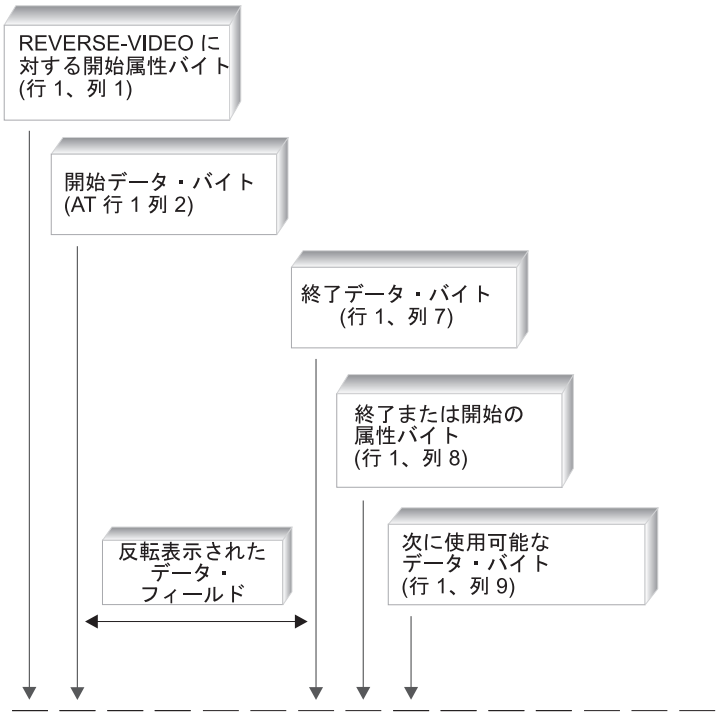
ID-1

更新可能なデータ項目。

ID-1 は、内部浮動小数点データ項目でも外部浮動小数点データ項目でも可能です。

受け入れられた、または表示されたフィールドは、その前後に属性バイトが必要です。これを実現するには、少なくとも初期表示属性を表示するためのスペースが画面上で使用可能になっていなければなりません。したがって、第 1 行、第 1 列は最初の表示属性用に使用されるため、その場所をデータ用に使用することはできません。データ用に使用可能な最初の位置は、第 1 行、第 2 列となります。

例えば、次のとおりです。



AT 句は、受け入れられる、または表示されることになるフィールドの開始行および列を設定します。これは、初期表示属性の位置を示すものではありません。

属性バイトがデータ・バイトと重なり合ったり、データ・バイトが属性バイトと重なり合ったりしないように、各フィールドが画面上に配置されていることを確認する必要があります。さらに、属性バイトの終わりは、特定ワークステーションについて定義した通常の属性になることにも留意してください。したがって、期待どおりの結果を得るには属性が正しい順序で指定されていることを確認する必要があります。

最初に画面をクリアするときは、WITH BLANK SCREEN 句を含んだ DISPLAY ステートメントを使用してください。

ID-1 が画面に合わないときは、英数字データが切り捨てられたり、数字データが画面上に表示されなくなります。

ID-1 がグループ項目で、MODE IS BLOCK 句がなければ、FILLER 以外の名前を持つ基本従属項目が表示されます。基本項目は、その記述が DATA DIVISION に現れる順番で画面に同時に表示されて位置付けされ、グループ中の FILLER 項目の長さによって分離されます。このため、行上の最初の位置は、直前の行の最後の位置にすぐに続くものと見なされます。

項目が FILLER によって分離されている場合、属性バイトは FILLER の長さに含まれます。このようにして、1 または 2 バイトの FILLER に、分離項目の後ろに続く属性および先の属性の両方が含まれていることになります。 1 バイト FILLER の場合には、後ろに続く属性および先の属性が同じバイトを占有します。データ項目は、通常 1 つの属性バイトで分離されているので、1 バイトの FILLER 項目は必要ありません。

REDEFINES、POINTER、PROCEDURE-POINTER、および INDEX-NAME データが MODE IS BLOCK 句のないグループ項目にあるとき、それらのデータは無視されます。

拡張 ACCEPT ステートメントは、実行キーが押された後にワークステーションから変更済みの値だけを受け入れることにより、ID-1 の値を事前表示します。実行キーを押すだけでは、事前表示されたデータ項目は更新されません。

ACCEPT された値は、データ項目に対して指定された画面位置から取られた文字のイメージで、フィールド終了キーが押された時点でのカーソル位置によって区切られます。

FIELD EXIT キーを使用して、英字フィールドおよび英数字フィールド内のすべての後書きスペースを 16 進のゼロに変換します。これはデフォルトのコンパイラー・オプション *UNDSPCHR が有効な場合に使用できます。例えば、次のとおりです。

```
DATA DIVISION.
       01    STRUC1.
             03  F11 PIC AA VALUE 'A'.
             03  F12 PIC 9(4) VALUE 123.
             03  F13 PIC XXX VALUE 'B'.
PROCEDURE DIVISION.
       ACCEPT STRUC1 AT 2102.
```

この例にある ACCEPT ステートメントは、以下の 3 つのフィールドを事前表示します。

- F11 は第 21 行、第 2 列から始まります。この属性バイトは第 21 行、第 1 列を占有します。
- F12 は第 21 行、第 5 列から始まります。この属性バイトは第 21 行、第 4 列を占有します。
- F13 は第 21 行、第 10 列から始まります。その属性バイトは第 21 行、第 9 列を占有します。

ここで、第 21 行には次のような結果が現れます。

```
00000000011111111112
12345678901234567890
A    123 B
```

F13 の開始

F12 の開始

F11 の開始

数字フィールド F12 は右そろえで、その先頭にスペースが入ることに注意してください。

フィールド F13 の最後にカーソルを置き、上記の 3 つのフィールドの値に応じてフィールド終了キーを押すと、それらフィールドの値は変更されずに残ります。ただし、F11 および F13 の後書きスペース (X'40') は確実に 16 進数のゼロになるので、テスト IF F13='Bbb' は ACCEPT 操作の後で失敗します。

16 進数のゼロを後書きしないようにするには、SPACE-FILL 句を使用します。既存プログラムにこの句をもたない拡張 ACCEPT ステートメントがすでに含まれており、プログラム変更が高価であったり望ましい解決方法でない場合には、*NOUNDSPCHR コンパイラー・オプションの使用を考慮できます。このオプションは、拡張 ACCEPT および拡張 DISPLAY ステートメントが表示可能文字だけを処理するプログラムだけに使用してください。

表示だけ可能な文字は、次の場合に処理されます。

- 拡張 DISPLAY ステートメントが、2 進数、内部浮動小数点、またはパック・データ (直接または再定義されたもの) を使用しない場合。
- 拡張 ACCEPT ステートメントが、英数字フィールドとして再定義された 2 進数、内部浮動小数点、またはパック・データを使用しない場合。

注: MODE IS BLOCK 句は英数字フィールドに対する、データ・グループの暗黙的な再定義です。

例えば、次のとおりです。

```
       01     STRUC2.
              03  F21 PIC 99.
              03  F22 PIC 9(10) USAGE COMP-3 VALUE 1111123.
              03  F23 PIC X(5).
```

ACCEPT STRUC2 MODE IS BLOCK AT 0102 は、長さが 13 バイトの英数字フィールドの 1 つとして処理されるために、表示不能文字が含まれることになります。

リモート制御装置と 5250 エミュレーションを使用して、プログラムがワークステーションで実行される場合、ILE COBOL 実行時モジュールは、コンパイラー・オプション *UNDSPCHR を (それが有効な場合) *NOUNDSDPCHR に変更して、通知メッセージをユーザーに送信します。システム構成にさまざまなワークステーション制御機構が含まれているときには、一貫性のある結果が得られるように、*NOUNSDPCHR オプションを使用することをお勧めします。このオプションを有効にするには、COBOL ソース・プログラム内の PROCESS ステートメントに NOUNDSPCHR を指定します。

浮動小数点データ項目の考慮事項: ACCEPT ステートメントで浮動小数点データ項目を使用する場合には、以下について考慮する必要があります。

外部浮動小数点リテラルが ACCEPT される場合は、結果が少し不正確になる可能性があります。このことは、浮動小数点データ項目が ACCEPT された後で転送された場合に、特にそうであると言えます。浮動小数点データ・タイプは近似値で、外部浮動小数点リテラルが転送された場合に、この浮動小数点データ・タイプはまず真の浮動小数点値 (IEEE) に変換されますが、この変換も浮動小数点データ・タイプの正確性に影響を与える可能性があります。

例えば、以下の ACCEPT を考えてみます。

```
77 external-float-1 PIC +9(3).9(13)E+9(3).
    ACCEPT external-float-1 FROM CRT.
    DISPLAY "EXTERNAL-FLOAT-1=" external-float-1.
```

+123455779012.3453E+297 が ACCEPT された後の表示結果は以下のとおりです。

```
EXTERNAL-FLOAT-1=+123.4557790123452E+306
```

データ・カテゴリー: 次の表は、拡張 ACCEPT ステートメントで処理されるデータのカテゴリーを示したものです。ここに示したデータ・カテゴリーは拡張 DISPLAY ステートメントでもサポートされます。(拡張 ACCEPT および DISPLAY ステートメントは PICTURE 文節にスケーリング位置をともなうデータ項目をサポートしていません。)

表 7-11. 拡張 ACCEPT ステートメントで処理されるデータのカテゴリー

カテゴリー	初期表示	入力するデータ	更新されるデータ項目
英字	A、B、C	D	B、C
数字 (内部、2 進、10 進、またはパック 10 進)	B、C、E、F、F1、 F2	D、G、H	O
数字 (ゾーン 10 進)	B、C、F、F1、F2	D、G、H	O
数字編集	A、I	J、H	K、O
英数字	A、B、C	D	B、C
英数字編集項目	A、I	J	L
ブール	A、B、C	D、M、N	B、C
DBCS	A、B、C	D	B、C
DBCS 編集	A、B、C	D	B、C、L
内部浮動小数点	A、I、I1、P	Q	O
外部浮動小数点	A、I、I1	Q	O

左そろえ (デフォルト)。

B RIGHT-JUSTIFY の場合は、後書きスペースおよび 16 進数のゼロが除去されて、データは右端の位置に移動されます。

C ZERO-FILL の場合は、データが左そろえであれば、後書きスペースおよび 16 進数のゼロがゼロに変換されます。また、データが右そろえであれば先行スペースがゼロに変換されます。

SPACE-FILL だけが指定されている場合、後書き 16 進ゼロはスペースに変換されます。

D RIGHT-JUSTIFY および ZERO-FILL (または SPACE-FILL) を指定した場合は、フィールド終了キーを押すと、ワークステーションによってそのフィールドが右端そろえでゼロが埋め込まれるか、またはスペースが埋め込まれます。 ZERO-FILL は DBCS を処理しません。

ZERO-FILL (または SPACE-FILL) だけしか指定しない場合には、ワークステーションは何の変換も行いません。

E 2 進数またはパック 10 進数がゾーン 10 進数に変換されてから表示されます。

F 次の条件が起こります。

- 数字の左側にスペースが埋め込まれてから、その数字が表示されます。
- 最初に、10 進数に小数点が挿入されて、整数部分と小数部分が分けられます。
- 小数部分がある場合は、1 桁が小数点用に予約されます。
- SPECIAL-NAMES 段落の中に DECIMAL-POINT IS COMMA が指定されている場合には、コンマとピリオドの意味が入れ替わります。システム値 QDECFMT とは何の関連もないことに注意してください。
- 数字が符号付きの場合は、符号用に 1 桁が予約されます。
- 数字に負符号が付いている場合は、負符号が表示されます。デフォルトでは、符号は数字の前に付きます。

F1 ZERO-FILL を指定すると、先行ゼロが表示されます。

F2 TRAILING-SIGN を指定すると、符号は右端の桁に入ります。

G 数字、ブランク、および以下に示す記号が受け入れられます。

- - (マイナス)
- + (プラス)
- . (ピリオド)
- , (コンマ)

符号は先行位置または後書き位置に入力しなければなりません。小数点は小数部分の前に入力しなければなりません。桁の位置そろえは行われません。コンマは整数部分を 3 桁ごとに区切ります。

H 数字は以下のような特性をもちます。

1. 符号を示すシンボル値はオプショナルであり、存在する場合は、どの桁の値よりも前にあっても (先行符号)、あるいは、その値の後に付いても (後書き符号) かまいません。使用できる符号は、正符号と負符号です。符号記号が先行符号の場合は、ブランク文字の前に付けることもできます。後書き符号の場合は、常にフィールドの右端の文字位置になければなりません。符号記号は 1 つしか使えません。
2. 最高 31 桁まで指定できます。使用できる数字は 0 から 9 までです。先頭の数字桁の前にはブランク文字を付けることができますが、左端の数字桁よりも右側に位置するブランク文字は無効となります。

10 進数値は整数と小数の 2 つの部分に分けられます。小数点の左側の数字は整数値として解釈されます。右側の数字は小数値として解釈されます。小数点が含まれていない場合、その数字は整数値として解釈されます。小数点記号が左端の数字桁よりも前にある場合、その数字は小数値として解釈されますが、その場合、左端の数字桁は小数点記号と隣り合っていなければなりません。小数点が右端の数字桁のうしろにある場合は、その数字は整数として解釈されますが、右端の数字桁は小数点と隣り合っていなければなりません。

整数部分の数字桁には、3 桁ごとのグループに分割するコンマを付けても付けなくてもかまいません。左端のグループは、1 桁、2 桁、または 3 桁のどれでもかまいませんが、その後の各グループは必ず前にコンマが付いて、しかも 3 桁からなるグループでなければなりません。コンマ記号の両側には必ず数字桁がくるようになっていなければなりません。

小数部分の数字桁をコンマで区切ることはできず、それらの数字桁の隣は必ず数字桁でなければなりません。

I 数字は、暗黙または明示の PIC 記号に従って編集されます。 ZERO-FILL、SPACE-FILL、および RIGHT-JUSTIFY は編集フィールドまたは浮動小数点フィールドには影響を与えません。

I1 TRAILING-SIGN は、編集フィールドまたは浮動小数点フィールドには影響を与えません。

J データの入力には、編集された記号を使う必要があります。

K すべての編集記号が除去された後に、結果として得られる数字が編集によって数字編集フィールドの中に戻されます。非数字が検出されると、実行時メッセージが出されます。

L データがフィールド内に戻されて、編集は**行われません**。ユーザーは、必ずこの後に編集形式を指定するようにしなければなりません。

M 数字、ブランク、および以下に示す記号が受け入れられます。

- - (マイナス)
- + (プラス)
- . (ピリオド)
- , (コンマ)

N ゼロまたは 1 以外のどの文字でもエラー・メッセージが生成されます。

O 想定されている小数点位に数字フィールドが位置合わせされます。データの位置付けに関する規則の詳細は 6-10 ページの『位置合わせの規則』を参照してください。

P 内部浮動小数点数は、外部浮動小数点数に変換されてから表示されます。

- COMP-1 項目は、-.9(8)E-99 という外部浮動小数点 PICTURE 文節を持っているかのように表示されます。
- COMP-2 項目は、-.9(17)E-999 という外部浮動小数点 PICTURE 文節を持っているかのように表示されます。

Q データは、浮動小数点リテラルの形成に関する規則に従って入力しなくてはなりません (2-17 ページの『浮動小数点リテラル』を参照)。指数はオプショナルです。

ID-1 の後に続く句はどのような順序でもかまいません。指定されたすべての句は、それに先行する ID に適用されます。

AT 句: AT 句は、ACCEPT 操作が開始される画面上の絶対アドレスを示します。 AT 句を指定しないと、ACCEPT 操作は第 1 行、第 2 列から開始されます。この句は、先行属性の開始位置を示すものではありません。

LINE 句は、画面上で画面項目が始まる行を指定します。

COLUMN 句は、画面上で画面項目が始まる列を指定します。

COL は、COLUMN の省略形です。

LINE 句および COLUMN 句は、どのような順序で記入してもかまいません。

ID-2、整数-1

ID-2 および整数-1 は、符号の付いていない 0 以上の値を持つ、9 桁未満の整数でなければなりません。 LINE 句または COLUMN 句の値が負の場合は、絶対値がとられます。 ID-2 または整数-1 は、PIC 9(3) 数字の中に移されます。

ID-2 は、内部浮動小数点データ項目であっても外部浮動小数点データ項目であってもいけません。

行および列番号の特定の組み合わせには、次のような特別な意味があります。

- 列が範囲内に入るまでは、その範囲外の列値は行の長さ分だけ減じられ、行値は増分されます。列番号によっては、行番号が数回増分されてしまうこともあります。
- 範囲外の行値は、画面を 1 行分上方に画面移動させます。これは、最下部の行番号が指定された場合と同じです。指定された行にかかわらず、画面が 2 行以上スクロールされることはありません。
- 列番号および行番号の両方ともが範囲外にある場合は、まず範囲外の列のほうが最初に処理されてから、次に範囲外の行が (上記の規則に従って) 処理されます。
- 入力された行および列番号が両方ともゼロの場合は、ACCEPT は、直前の ACCEPT 操作が終了したところの次の位置から開始されます。各行の 1 列目は、直前の行の最終列の後に続いているものと見なされます。
- 行番号がゼロで、しかもゼロ以外の列番号が指定されている場合は、ACCEPT は、直前の表示操作が終了した行の次の行の、指定された列から始まります。
- 列番号がゼロで、しかもゼロ以外の行番号が指定されている場合は、ACCEPT は、指定された行の、直前の表示操作が終了した列の次の列から始まります。

ID-3、整数-2

ID-3 は、PIC 9(4) または PIC 9(6) フィールドでなければなりません。 ID-3 は、内部浮動小数点データ項目であっても、外部浮動小数点データ項目であってもいけません。

整数-2 は、4 バイトまたは 6 バイトの数字フィールドでなければなりません。

ID-3 または整数-2 の長さが 4 バイトの場合は、最初の 2 桁が行を指定し、次の 2 桁が列を指定します。 ID-3 または整数-2 の長さが 6 バイトの場合は、最初の 3 桁が行を指定し、次の 3 桁が列を指定します。

FROM CRT 句: ACCEPT ステートメントが拡張されていることを示します。

MODE IS BLOCK 句: ID は基本項目として扱われます。したがって、ID がグループ項目であっても、1 つの項目として受け入れられます。

ON EXCEPTION 句: ON EXCEPTION を指定すると、ACCEPT 操作が通常の完了以外で終わった場合は、命令ステートメント-1 が実行されます。つまり、CRT 状況キー 1 がゼロ以外の場合です。

ON EXCEPTION 句を使用しても、ワークステーション境界または画面範囲外などの条件に対する実行時メッセージが生成されないようにすることはできません。

NOT ON EXCEPTION を指定している場合、ACCEPT 操作が正常に完了して終了すると、命令ステートメント-2 が実行されます。

END-ACCEPT 句: END-ACCEPT はオプショナルです。 ACCEPT ステートメントをネストする場合は、END-ACCEPT は必要です。

WITH 句: WITH 句を使うことによって、ユーザーは ACCEPT 操作の特定のオプションを指定することができるようになります。そのようなオプションについては、以下に示す各句で説明します。

AUTO (AUTO-SKIP) 句: フィールドがオペレーター入力によって充てんされると、カーソルは、終了文字が入力されるのを待たずに、次の入力フィールドに自動的にスキップします。そのフィールドがグループの最後のフィールドである場合は、AUTO-SKIP は、ENTER キーが押された場合と同じ効力をもちます。

AUTO と AUTO-SKIP はいずれを使ってもかまいません。

BELL (BEEP) 句: この句を含んでいる項目が受け入れられるたびに、警報音が鳴ります。

BELL と BEEP はいずれを使ってもかまいません。

BLINK 句: 画面項目が画面に表示されると明滅します。

FULL (LENGTH-CHECK) 句: オペレーターは、画面項目をまったく空のままにするか、または画面をデータで完全に埋め込むかのいずれかを行わなければなりません。フィールド終了、フィールド +、フィールド - のキーを使用することはできません。また、実行キーを押したあとに、入力フィールド内のデータに対して削除キーを使うこともできません。最初に表示されるデータで FULL 句を満たすことができます。

この句がグループ・レベルに指定されている場合、該当する従属基本項目のすべてに適用されます。

FULL 句は、すべての ACCEPT ステートメントの実行中に効力があります。

FULL および LENGTH-CHECK は、いずれを使ってもかまいません。

HIGHLIGHT 句: 画面項目が高輝度モードで画面に表示されます。

REQUIRED (EMPTY-CHECK) 句: REQUIRED 句は、フィールドが空のまま残らないようにするために使用します。

英数字項目の場合、これは、フィールドにはスペースまたは 16 進ゼロ以外の文字が少なくとも 1 つ含まれていなければならないことを意味します。数字項目の場合は、フィールドには必ずゼロ以外の数字が入っていなければなりません。

この句を指定したときにフィールドが空のままになると、実行時メッセージが出されて、ユーザーはリセット・キーを押してから、データを入力し直すように要求されます。

REQUIRED および EMPTY-CHECK 句は、いずれを使ってもかまいません。

REVERSE-VIDEO 句: 画面項目が反転表示で示されます。

SECURE (NO-ECHO) 句: オペレーターが入力したデータが画面に表示されなくなります。この句はグループ画面項目に指定できます。その場合、この句は、その項目に従属するすべての基本項目に対して適用されます。 SECURE 句が指定されているときは、画面項目内に現れるのはスペースとカーソルだけになります。

SECURE と NO-ECHO 同じ意味で使用されます。

UNDERLINE 句: 画面に表示される画面項目に下線が引かれます。

RIGHT-JUSTIFY 句: オペレーターが入力した文字が画面のフィールドの右端の文字位置に移動されます。後書きスペースおよび後書き 16 進ゼロは除去されます。

このオプションで影響を受けるのは、非編集データ項目だけです。この句が効力を持つのは、データ項目の初期データが表示されたとき、および ACCEPT 操作が終了するときです。これは、数字データを処理する唯一の方法です。

データ項目の定義に DATA DIVISION の JUSTIFIED RIGHT 文節を使った場合は、そのデータ項目は RIGHT-JUSTIFY 句が指定されていた場合と同じように処理されます。

SIZE 句: 画面上にデータ項目のサイズを指定します。この句は基本データ項目にしか使用することができません。

指定されたサイズがゼロであれば、SIZE 句には影響がありません。この場合、データ項目を表示するためにフィールドの長さが使用されます。

関連する PICTURE 文節によって暗黙指定されたサイズよりも小さなサイズが指定されると、ワークステーションの画面にはデータ項目の左端の部分だけが現れます。

数字または数字編集のデータ項目に対して指定したサイズが PICTURE 文節によって暗黙指定されたサイズよりも小さくなる場合には、値の表示や ACCEPT 操作における事前表示の際に、値の右端が切り捨てられた状態で表示されます。データ項目は MOVE 操作の規則に従って更新されます。

フィールドの長さが画面サイズを超える値を持つ SIZE リテラルを指定する場合には、英数字データが切り捨てられたり、数字データが無視されるか、表示されなくなります。

JUSTIFIED が指定された項目に関しては、その項目の長さよりも小さいサイズを指定すると、右端の部分だけが現れます。

ユーザーが指定するサイズが、PICTURE 文節で暗黙指定されたサイズよりも大きい場合には、その項目の表示用部分にはスペースが埋め込まれます。埋め込まれるのは右側です。

SPACE-FILL 句: 非編集データ項目の場合、後書きの 16 進ゼロはスペースに変換されて、項目はすべての文字位置でゼロ消去が行われて画面に表示されます。この効果が現れるのは、データ項目の初期データが表示されたときと、データ項目内への ACCEPT 操作が終了したときです。このオプションは、編集フィールドには影響を与えません。

TRAILING-SIGN 句: 演算符号がフィールドの右端の文字位置に表示されます。この効果が現れるのは、データ項目の初期データが表示されたときと、ACCEPT 操作が終了したときです。このオプションの影響を受けるのは、符号付きの非編集数字データ項目だけです。このオプションを指定していないときは、符号は数字の前に付きます。

UPDATE 句: 新しいデータをキー入力するようオペレーターにプロンプトが出される前に、データ項目の現在の内容が表示されます。次いで、初期データは、オペレーターがキー入力した場合と同じように処理されます。

データ・タイプごとの事前表示: UPDATE 句がない場合には、一部のデータの事前表示を制御できます。数字編集データだけを事前表示するには、EXTDSPOPT パラメーターの *ACCUPDNE オプションを指定します。全データを事前表示するには、デフォルト・オプション *ACCUPDALL を使用します。

ZERO-FILL 句: ゼロ消去が行われずに非編集データ項目が画面に表示されます。左そろえデータの場合、後書きスペースおよび後書き 16 進ゼロはゼロに変換されます。右そろえデータの場合、先行スペースがゼロに変換されます。

この効果が現れるのは、データ項目の初期データが表示されたときと、データ項目内への ACCEPT 操作が終了したときです。編集フィールドには影響を与えません。

構文検査だけを受ける句: 次に示す句は、構文検査を受けますが、以下のとおりです。

構文検査後コンパイラーは文書として扱います。

- **PROMPT CHARACTER**
  - PROMPT CHARACTER 文節により画面上の空の文字位置をマークします。

    ID-5 ID-5 は単一の英字または英数字データ項目でなければなりません。 ID-5 は OCCURS 文節に従属してはなりません。

    リテラル-1
    リテラル-1 は、1 文字の非数字リテラルまたは表意定数でなければなりません。
- **FOREGROUND-COLOR** または **FOREGROUND-COLOUR**
  - FOREGROUND-COLOR/FOREGROUND-COLOUR 文節は画面項目の前景色を指定します。

    整数-4 整数-4 は符号なしの整数でなければなりません。
- **BACKGROUND-COLOR** または **BACKGROUND-COLOUR**
  - BACKGROUND-COLOR/BACKGROUND-COLOUR 文節は画面項目の背景色を指定します。

    整数-5 整数-5 は符号なしの整数でなければなりません。
- **LEFT-JUSTIFY**

形式 7 の考慮事項: ID-1 がグループ項目で、しかも MODE IS BLOCK 句が付いていない場合には、FILLER 以外の名前が付いた基本従属項目が受け入れられます。このような基本項目は、その記述が DATA DIVISION 内に記入されている順序に従って画面上に配置され、それぞれの項目の間はグループの FILLER 項目の長さ分だけ空けられます。

このため、行上の最初の位置は、直前の行の最後の位置にすぐに続くものと見なされます。項目の受け入れもこれと同じ順序で行われます。

CURSOR 文節に特に指定がない限り、はじめはカーソルは最初の項目の先頭を指し示します。各項目への ACCEPT 操作が終了すると、カーソルは次の項目の先頭に移動します。

CURSOR 文節がフィールドの位置に影響を与えることはありません。唯一行えるのは、決められた規則に従って ACCEPT ステートメントのカーソル位置を変更することだけです。

記号 P を含んでいる PICTURE 文節を持つ数字項目は、拡張 ACCEPT ステートメントではサポートされません。

MODE IS BLOCK を指定しない限り、データ項目には固定長テーブルを入れることはできません。また、MODE IS BLOCK を指定するしないにかかわらず、データ項目には可変長のテーブルを入れることはできません。

拡張 ACCEPT および拡張 DISPLAY の考慮事項: 拡張 ACCEPT ステートメントと拡張 DISPLAY ステートメントの両方について、次のことを考慮する必要があります。

画面形式: 拡張 ACCEPT または DISPLAY 操作では、24 行、80 桁の画面形式がサポートされます。

拡張 ACCEPT または DISPLAY 操作が処理されたときは、それ以外のディスプレイ・ファイルをプログラムでオープンすることはできません。拡張 ACCEPT および DISPLAY ステートメントが含まれているプログラムの中に TRANSACTION ファイルがコーディングされている場合、ユーザーは拡張 ACCEPT または DISPLAY ステートメントが TRANSACTION I/O の介入を受けないように気を付ける必要があります。また、逆に、TRANSACTION I/O が拡張 ACCEPT または DISPLAY ステートメントの介入を受けないように気を付ける必要もあります。

添え字および参照変更: 添え字付き項目および参照変更項目は両方ともサポートされています。

パフォーマンス: CRTCBLMOD または CRTBNDCBL コマンドで EXTDSPOPT(*NODFRWRT) パラメーター (非据え置き書き出し) を指定しない場合、ILE COBOL コンパイラーは、次の ACCEPT ステートメントが発生するまで、すべての拡張 DISPLAY ステートメントをバッファーに入れます。 *NODFRWRT オプションがあれば、DISPLAY ステートメントをその発生時に実行することにより、データ・エラーをその原因であるステートメントに関連付けることができます。据え置き書き出し (*DFRWRT) オプションは、連続する DISPLAY ステートメントが生成したデータ・ストリームをバッファリングすることにより、パフォーマンスを改善します。

DBCS 処理: DBCS プログラムは、DBCS システム上でコンパイルされた場合にのみ、DBCS システム上で実行できます。

- DBCS 項目を継続させるには、シフトインおよびシフトアウト文字を正しくコーディングしなければなりません。 DBCS 項目の継続に関する規則については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の 2 バイト文字セットのサポートに関する付録を参照してください。
- DBCS の内容は、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の 2 バイト文字セットのサポートに関する付録で説明されている規則によって決定されます。
- CRTCBLMOD コマンドまたは CRTBNDCBL コマンドの拡張 DISPLAY パラメーターの *NOUNDSPCHAR (表示不能文字なし) オプションか、同等の処理ステートメント・オプションを指定しなければ、データは送信されたとおりに画面にそのまま渡されます。 *NOUNDSPCHAR オプションを指定した場合には、データに制御情報が存在するかどうかワークステーションによって検査されます。この場合、出力データには有効な表示文字だけが含まれることになります。
- 英字フィールドまたは英数字フィールドの長さが 4 バイトより小さい場合には、16 進数の 40 よりも小さい値が検出されてもエラーは生成されません。

句の組み合わせ: 1 つの ACCEPT または DISPLAY ステートメントで、1 つの WITH 句に UNDERLINE、HIGHLIGHT、および REVERSE-VIDEO 句が含まれている場合には、HIGHLIGHT 句が無視されます。このような組み合わせがコーディングされると、コンパイル時に警告メッセージ (LNC0265) が出されます。拡張 DISPLAY ステートメントの中では、UPON CRT-UNDER 句は UNDERLINE 句と同等です。フィールドが画面上に表示されないようにするには、SECURE オプションを使用します。

トランザクション・ファイル: 拡張 ACCEPT/DISPLAY ステートメントと TRANSACTION ファイルを同一のプログラムで使用することはお勧めできません。拡張 ACCEPT/DISPLAY ステートメントが TRANSACTION ファイルと同一のプログラムで使用されている場合には、拡張 ACCEPT/DISPLAY ステートメントの実行時に、TRANSACTION ファイルをクローズする必要があります。 TRANSACTION ファイルのオープン時に、拡張 ACCEPT/DISPLAY ステートメントを実行するならば予測不能な結果が生じます。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。

リモート・ワークステーション: 拡張 ACCEPT および DISPLAY ステートメントは、5251-12 型制御装置に接続しているリモート・ワークステーションでは実行しません。

CRTCBLMOD または CRTBNDCBL コマンドの EXTDSPOPT(*NOUNDSPCHR) パラメーターにより、拡張 ACCEPT ステートメントおよび DISPLAY ステートメントを 3174 および 3274 制御装置に接続しているリモート・ワークステーションで使用できます。ただし、ユーザーのデータに非表示文字が含まれていてはなりません。リモート制御装置の使用中、データの受け入れを行うために CLEAR キーおよび HELP キーを使用することはできません。

COBOL/2* の処理との相違点: ILE COBOL の拡張 ACCEPT および DISPLAY ステートメントは、ACCEPT および DISPLAY ステートメント (形式 2) に類似しています。異なる点については 9-49 ページの『付録 I. ACCEPT/DISPLAY および COBOL/2 に関する考慮事項』で説明します。

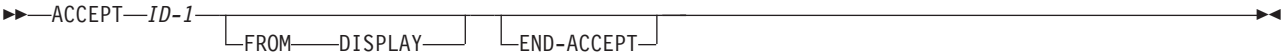
End of IBM Extension

形式 8 - セッション I/O

IBM Extension

ACCEPT ステートメントは ILE 共通セッション管理プログラムから情報を検索します。

ACCEPT ステートメント - 形式 8 - セッション I/O



ID-1 の説明に TYPE 文節が含まれる場合は、その文節で参照されるタイプ名は基本項目でなければなりません。

CONSOLE IS DISPLAY 文節が SPECIAL-NAMES 段落に指定される場合、ACCEPT ステートメントのこの形式では FROM 句の指定はオプショナルです。

形式 8 は、ILE 共通セッション管理プログラムから ID-1 へデータを転送する場合に使用します。入力データは、以下のいずれかの形式で受け取られます。

- USAGE IS DISPLAY 形式。データは変換されません。
- USAGE IS DISPLAY-1 形式。データを囲むシフトアウト文字とシフトイン文字が除去されます。
- USAGE IS NATIONAL 形式。データは、ジョブの現行 CCSID によって指定されるコード・セットから変換されます。

ILE 共通セッション管理プログラムを使用して ACCEPT ステートメントを管理します。スクリーン入出力セッション・サービスについて詳しくは、**IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL* および *API*』のセクション『Dynamic Screen Manager』を参照してください。

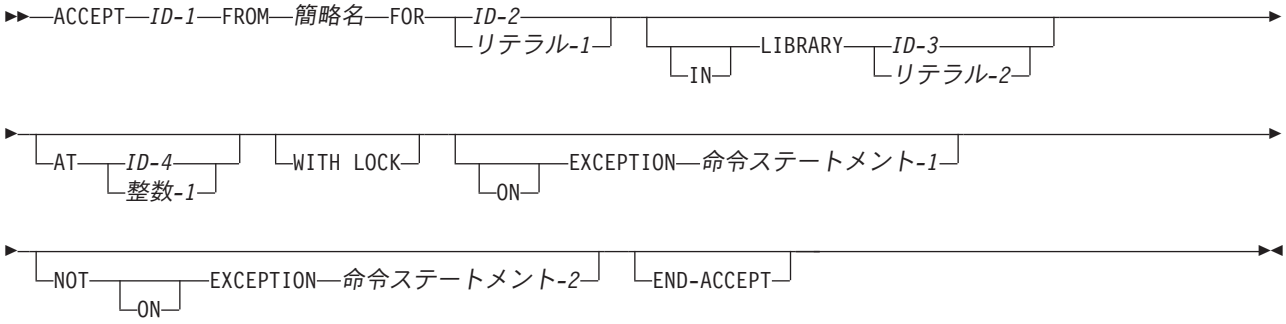
End of IBM Extension

形式 9 - データ域

IBM Extension

ACCEPT ステートメントは、FOR 句で指定されるデータ域から情報を検索します。

ACCEPT ステートメント - 形式 9 - データ域



| データ域の使用方法の詳細および例については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」のセクション『データ域を使用したデータの受け渡し』の『自分で作成したデータ域の使用』の情報を参照してください。

ID-1: ID-1 のクラスは、英数字、数字、DBCS または国別のどれでもかまいません。

ID-1 の説明に TYPE 文節が含まれる場合は、その文節で参照されるタイプ名は基本項目でなければなりません。

FROM 句: FROM 句は、SPECIAL-NAMES 段落中の DATA-AREA の環境名に関連付けなければならない簡略名を指定します。

FOR 句: 情報を検索するためのオペレーティング・システムのデータ域を識別します。 指定されたデータ域が実行時に見つからない場合は、ON EXCEPTION エラーが発生します。

ID-2
英数字データ項目でなければなりません。 ID-2 の内容は、有効なオペレーティング・システムのデータ域名を表していなければなりません。オペレーティング・システムのデータ域名は、最大 10 文字の長さです。したがって ID-2 の最初の 10 文字が、データ域名を形成するために使用されます。

リテラル-1
非数字で、長さが最大 10 文字でなくてはなりません。

LIBRARY 句: データ域を検出するオペレーティング・システムのライブラリーの名前を指定するために使用されます。特殊値 *LIBL (ジョブのライブラリー・リストを使用する検索) または *CURLIB (現行ライブラリーを検索) を指定できます。 LIBRARY 句を省略すると、ジョブのライブラリー・リストを使用して、データ域を検索します。

ID-3
英数字データ項目でなければなりません。 IBM i のライブラリー名の長さは最大 10 文字であるため、ID-3 の最初の 10 文字だけを使用してライブラリー名が形成されます。

リテラル-2
非数字で、長さが最大 10 文字でなくてはなりません。

ID-2、ID-3、リテラル-1、およびリテラル-2 は、*MONOPRC コンパイラー・オプションによって影響を受けません。これらには、オペレーティング・システムの引用符付き名を入れることができます (詳しくは、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL および API*』にある『名前の指定の規則』を参照してください)。

AT 句: AT 句は、テキストを受け取るデータ域内の開始位置を示します。

AT 句が指定されていない場合、開始位置 1 が想定されます。

ID-4、整数-1
ID-4 および整数-1 は正の整数で、その値は 1 から最大データ域サイズ (2000) の範囲になくてはなりません。

WITH LOCK 句: ACCEPT ステートメントが、情報を検索して ID-1 に入れている間、データ域は、変更されないように LSRD ロック (読み取りのための共用ロック) でロックされます。 ID-1 が情報を受け入れた後、WITH LOCK 句は LEAR ロック (排他ロック読み取り可) をデータ域に設置します。ロックをデータ域に設置できない場合、例外条件が発生します。

データの転送後もロックをデータ域に保存するためには、この句を指定してください。このステートメント以前にデータ域にロックが存在し、ステートメントに WITH LOCK 句が含まれていない場合、そのロックは解放されます。

(NOT) ON EXCEPTION 句: データ域にアクセスしている間にエラーが発生すると、ON EXCEPTION 句で指定された任意の命令ステートメントが処理されます。 ON EXCEPTION 句がない場合は、実行時メッセージが発行されます。データ域が正常にアクセスされると、NOT ON EXCEPTION 句で指定された任意の命令ステートメントが処理されます。

END-ACCEPT 句: 明示範囲終了符号 END-ACCEPT は、ACCEPT ステートメントの有効範囲を区切る働きをします。 END-ACCEPT を使うと、条件 ACCEPT ステートメントを別の条件ステートメント内でネストさせることができます。 END-ACCEPT は命令 ACCEPT ステートメントとともに使用することもできます。詳細は 7-29 ページの『範囲区切りステートメント』のセクションを参照してください。

End of IBM Extension

ACQUIRE ステートメント

IBM Extension

ACQUIRE ステートメントは、TRANSACTION ファイル用のプログラム装置を獲得するために使用されます。

ACQUIRE ステートメント - 形式 - TRANSACTION

ID、リテラル
リテラルまたは ID の内容で、指定ファイルによって獲得されるプログラム装置名を指定します。リテラルは、非数字で、長さは 10 文字以下でなければなりません。 ID は英数字データ項目を参照し、長さは 10 文字以下でなければなりません。

ファイル名
ファイル名は TRANSACTION 編成のファイルの名前でなければならず、かつ、ACQUIRE ステートメントの実行時には、このファイルがオープンされていなければなりません。編成が TRANSACTION でない場合には、コンパイル・エラー・メッセージが出されます。

通信装置を獲得するために前もって適合していなければならない条件に関しては、「*ICF プログラミング・マニュアル*」を参照してください。ディスプレイ装置を獲得する前に満たす必要がある条件の説明につ

いては、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) 内のカテゴリー『データベースおよびファイル・システム』のセクション『*DB2 for i*』を参照してください。

ACQUIRE 操作が正常に完了すると、プログラム装置は入出力操作用に使用可能になります。 ACQUIRE が正常に終了しない場合、ファイル状況値は 9H にセットされ、適用可能な USE AFTER EXCEPTION/ERROR プロシージャーが呼び出されます。

ファイルがオープンされているとき、プログラム装置 1 台だけを暗黙のうちに獲得できます。ファイルが ICF ファイルである場合、暗黙的に獲得される 1 個のプログラム装置は、 ICF ファイル作成 (CRTICFF) CL 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・ファイルである場合、暗黙的に獲得される 1 個のプログラム装置は、表示装置ファイル作成 (CRTDSPF) CL コマンドの DEV パラメーターの中の最初の記入項目によって決定されます。追加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。

プログラム装置は、ACQUIRE ステートメントを使用して明示的に獲得されます。 ICF ファイルの場合、ファイルのオープン前に、ICF 装置項目追加 (ADDICFDEVE) コマンドまたは ICF プログラム装置項目一時変更 (OVRICFDEVE) コマンドを使って、プログラム装置がファイルに前もって定義されていなければなりません。ディスプレイ・ファイルの場合、ファイル作成、変更、または一時変更されたとき、そのファイルに対して OPEN が出される前に、DEV パラメーターにプログラム装置が前もって指定されていなければなりません。

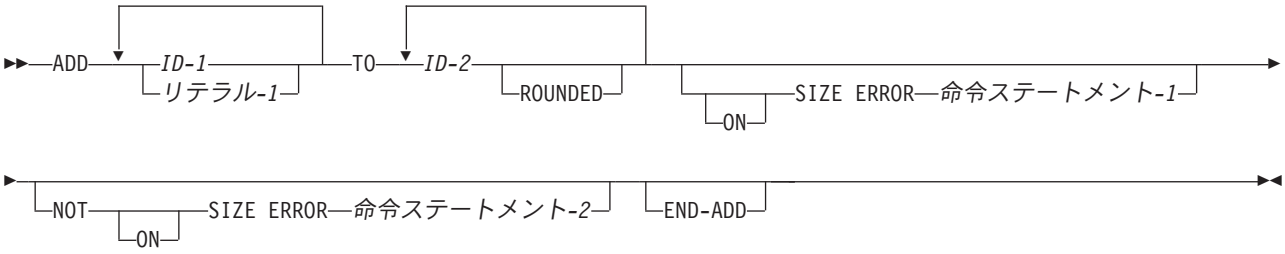
ディスプレイ装置のこれらの CL コマンドについて詳しくは、**IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL および API*』を参照してください。通信装置用のこれらの CL コマンドについては、「*ICF プログラミング・マニュアル*」を参照してください。ACQUIRE ステートメントは、入出力エラーからの回復の補助手段としても使用できます。リカバリー手順について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の通信リカバリーに関するセクションを参照してください。

End of IBM Extension

ADD ステートメント

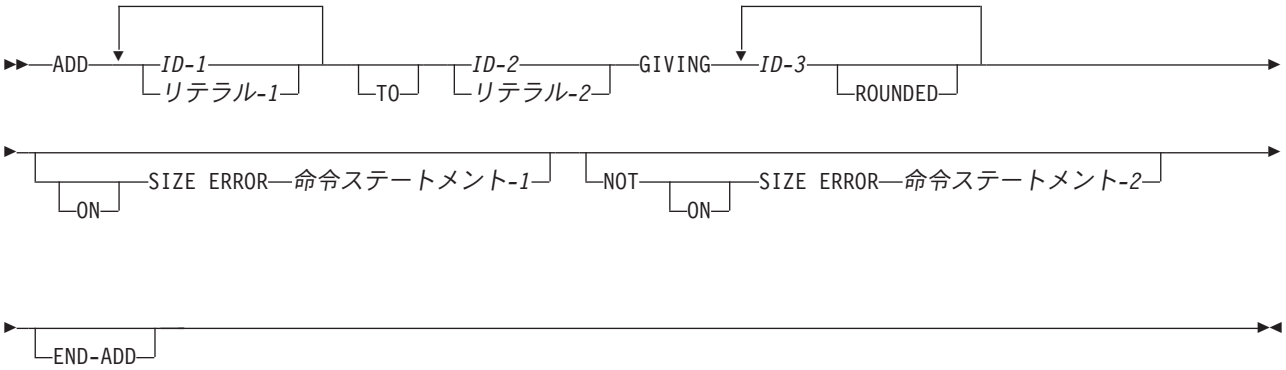
ADD ステートメントは、複数の数字オペランドを加算して、その結果を保管するために使用されます。

ADD ステートメント - 形式 1 - ADD



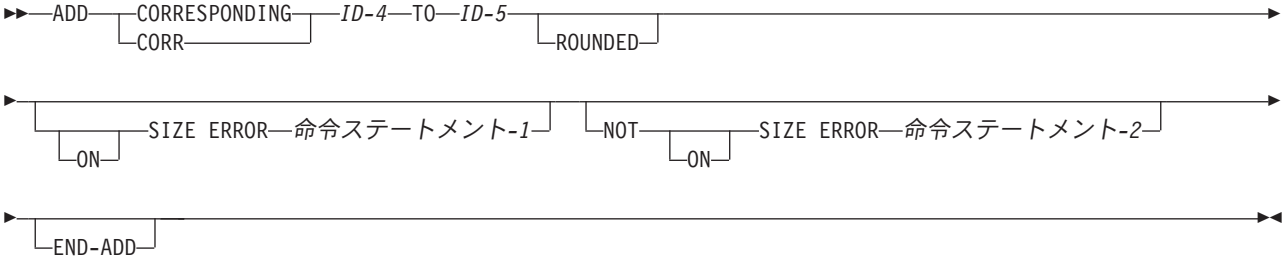
形式 1 では、キーワード TO の前にある ID およびリテラルを加算し、この和を ID-2 に加え、保管します。この最初の和も、連続して出現するそれぞれの ID-2 に対して、ID-2 が指定されている順に、左から右へ加算されます。

ADD ステートメント - 形式 2 - ADD GIVING



形式 2 では、語 GIVING の前にあるオペランドの値をすべて加算し、その和を、ID-3 が参照する各データ項目の新しい値として保管します。

ADD ステートメント - 形式 3 - ADD CORRESPONDING



形式 3 では、ID-4 内の基本データ項目を ID-5 内の対応する基本項目に加算して、保管します。

すべての形式において、以下が適用されます。

ID-1、ID-2
基本数字項目でなければなりません。

ID-3
基本数字項目または数字編集項目でなければなりません。

ID-4、ID-5
グループ項目でなければなりません。

リテラル-1、リテラル-2
数字リテラルでなければなりません。

形式 1 では、該当のステートメント内のすべてのオペランドを使用して、オペランドの合成内容が決定されます。

形式 2 では、語 GIVING に続くデータ項目を除く該当のステートメント内のすべてのオペランドを使用して、オペランドの合成内容が決定されます。

形式 3 では、対応するデータ項目のおのおのの対に対して、個別にオペランドの合成内容が決定されます。

オペランドの合成に関する詳細は 7-34 ページの『オペランドのサイズ』を参照してください。

IBM Extension

浮動小数点データ項目および浮動小数点リテラルは、数字データ項目または数字リテラルが指定できるところではどこでも使用できます。

End of IBM Extension

ROUNDED 句

7-33 ページの『ROUNDED 句』を参照してください。

SIZE ERROR 句

7-33 ページの『SIZE ERROR 句』を参照してください。

CORRESPONDING 句 (形式 3)

7-31 ページの『CORRESPONDING 句』を参照してください。

END-ADD 句

この明示範囲終了符号は、ADD ステートメントの有効範囲を区切ります。 END-ADD は条件 ADD ステートメントを命令ステートメントに変換して、別の条件ステートメント内でネストさせることができます。

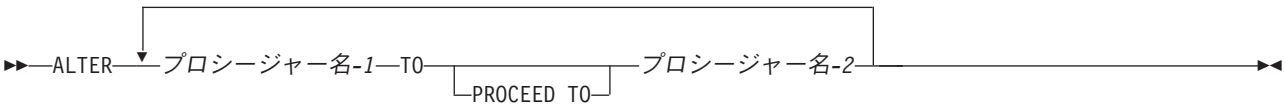
詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

ALTER ステートメント

ALTER ステートメントは、GO TO ステートメントの中で指定されている制御の転送先を変更します。

注: ALTER ステートメントは、構造化されていないプログラミング技法を促すものです。ALTER ステートメントと同じ機能を提供するものに EVALUATE ステートメントがあり、これはプログラムをより良く構成するのに役立ちます。

ALTER ステートメント - 形式



ALTER ステートメントの実行によって、プロシージャー名-1 で名前を指定された段落中の GO TO ステートメントが修正されます。続いてこの修正された GO TO ステートメントが実行されると、プロシージャー名-2 へ制御が移ります。

プロシージャー名-1
手続き部の段落には文が 1 つだけが含まれる必要があります。それは DEPENDING ON 句のない GO TO ステートメントです。

プロシージャー名-2
手続き部のセクションまたは段落でなければなりません。

プロシージャー名-1 およびプロシージャー名-2 が宣言プロシージャー内にある場合、そのどちらも、どの非宣言プロシージャーも参照できません。プログラムの非宣言の中では、EXCEPTION/ERROR 宣言プロシージャーの中に現れるプロシージャー名に対する参照があってはなりません。

ALTER ステートメントの実行前に、プロシージャー名-1 に指定されている段階に制御が移った場合には、GO TO ステートメントは GO TO ステートメントで指定された段落に制御を移します。しかし、ALTER ステートメントの実行後に、制御が再度プロシージャー名-1 によって指定された段落へ渡ると、GO TO ステートメントは制御をプロシージャー名-2 によって指定されている段落へ渡します。

注: RECURSIVE 属性をもつプログラムで、 ALTER ステートメントを使用しないでください。

コーディング例

ALTER ステートメントはプログラム・スイッチの働きをします。例えば、初期設定時にはある順序で実行し、ファイル処理の部分では別の順序で実行できます。変更した GO TO ステートメントはデバッグが困難なため、スイッチをテストしてそのスイッチの値に基づいて特定のコード・シーケンスを実行することをお勧めします。例えば、次のとおりです。

```
PARAGRAPH-1.
  GO TO BYPASS-PARAGRAPH.
PARAGRAPH-1A.
  .
  .
BYPASS-PARAGRAPH.
  .
  .
 ALTER PARAGRAPH-1 TO PROCEED TO
     PARAGRAPH-2.
  .
  .
PARAGRAPH-2.
  .
  .
```

ALTER ステートメントの実行前に、制御が PARAGRAPH-1 へ渡った場合には、GO TO ステートメントは制御を BYPASS-PARAGRAPH へ移します。ただし、ALTER ステートメントの実行後に、制御が PARAGRAPH-1 へ渡ると、GO TO ステートメントは制御を PARAGRAPH-2 へ渡します。

INITIAL 属性のあるプログラム内で修正された GO TO ステートメントは、プログラムに入るたびに最初の状態に戻ります。

CALL ステートメント

CALL ステートメントは実行単位の中で、あるプログラムから別のプログラムに制御を渡します。

CALL ステートメントを含むプログラムは、呼び出し側プログラムです。また、CALL ステートメントに示されるプログラムは、呼び出し先サブプログラムです。呼び出し側プログラムには、他のプログラムを呼び出す時点で CALL ステートメントが入っていなければなりません。

IBM Extension

ILE COBOL では、サブプログラムは COBOL プログラム、他の IBM i 言語で書かれたプログラム、または ILE プロシージャーである可能性があります。

End of IBM Extension

CALL ステートメントの処理によって、呼び出し先サブプログラムの非宣言の最初の命令へ制御が渡されます。呼び出し側プログラムへ制御が戻る場合の戻り先は、CALL ステートメントの次の命令です。呼び出し先サブプログラムに手続き部、または手続き部の非宣言セクションがない場合には、その呼び出し先サブプログラムは暗黙の EXIT PROGRAM を出します。

プログラム制御が CALL ステートメントによって移され、呼び出し先プログラムが、直接または間接的に、呼び出し元のプログラムを実行すると、**再帰**呼び出しが行われたことになります。RECURSIVE 属性が定義されたプログラムは、直接または間接的にそれ自身を実行する CALL ステートメントを実行できます。ILE COBOL では、非再帰的プログラムで再帰を行うことはできません。非再帰的プログラムに対して再帰を試みると、実行時エラー・メッセージが生成されます。呼び出し側プログラムおよび関連した概念と用語について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

RETURN-CODE 特殊レジスターを使用すれば、あるプログラムからその呼び出し側プログラムに戻りコードの情報を渡すことができます。詳しくは、7-227 ページの『RETURN-CODE 特殊レジスター』を参照してください。

CALL ステートメントの処理は、呼び出し先サブプログラムに制御を渡します。 CALL ステートメントにプログラム・オブジェクトのリンケージがあり、その CALL ステートメントが実行時に、指定されたライブラリー内に存在していないプログラム・オブジェクトを指定すると、エラー・メッセージが出されます。ON EXCEPTION 句または OVERFLOW 句を使用して、エラー処理のプロシージャーを指定することが可能です。

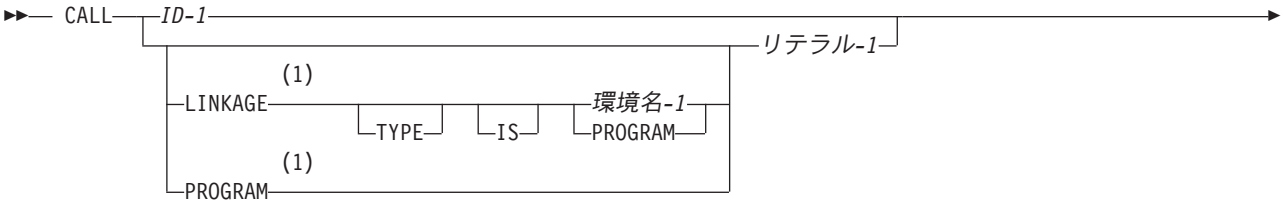
呼び出し先サブプログラムは、実行単位の中で最初に呼び出されたときには初期状態になっています。また、CANCEL ステートメントの後で最初に呼び出されたときにも初期状態になっています。

プログラムが初期プログラムである (その PROGRAM-ID 段落に INITIAL 文節が含まれている) 場合、そのプログラムは呼び出されるごとにその初期状態になります。それ以外の場合はすべて、呼び出し先サブプログラムに入ったとき、そのサブプログラムは最後に使われた状態のままですが、PERFORM ステートメントの制御メカニズムの場合は除きます。 PERFORM ステートメントの制御メカニズムでは、常にそれらの初期状態に設定されます。

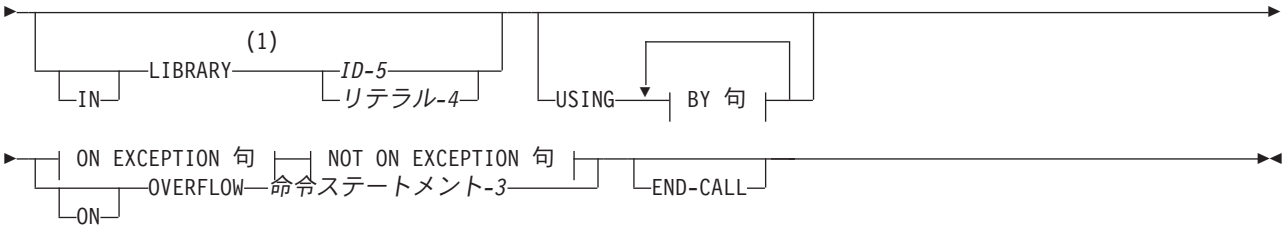
暗黙の、または明示の STOP RUN が生じると、言語とユーティリティーの戻りコードは 0 に設定されます。そうでない場合は、2 に設定されます。 RETURN-CODE レジスターが作業制御ブロックのユーザー部分にコピーされます。戻りコードの詳細については、「*CL プログラミング*」の RTVJOBA コマンドおよび DSPJOB コマンドを参照してください。

ユーザー戻りコードは、COBOL プログラムの実行開始後と他のプログラムに対して呼び出しが行われる前には 0 にセットされます。

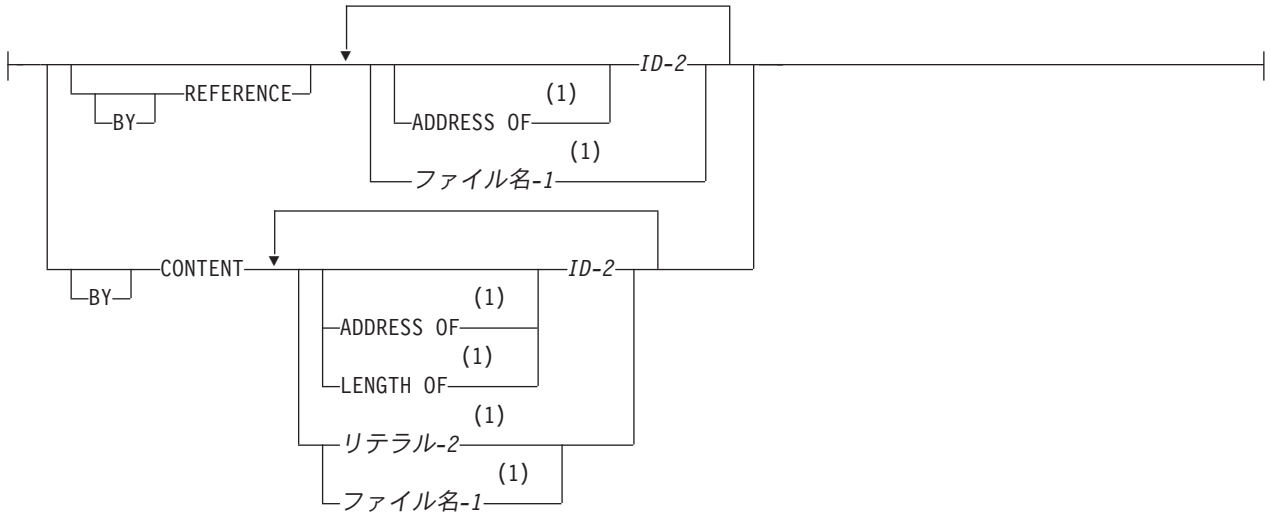
CALL ステートメント - 形式 1



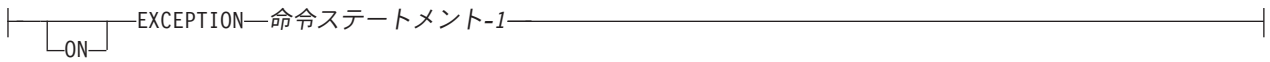
CALL ステートメント



BY 句:



ON EXCEPTION 句:

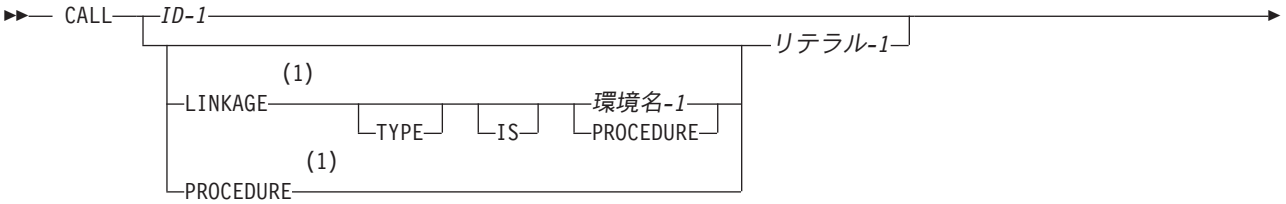


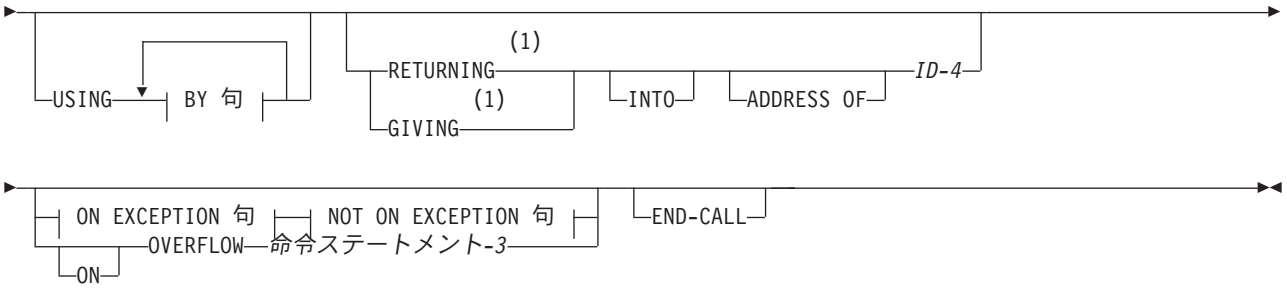
NOT ON EXCEPTION 句:

注:

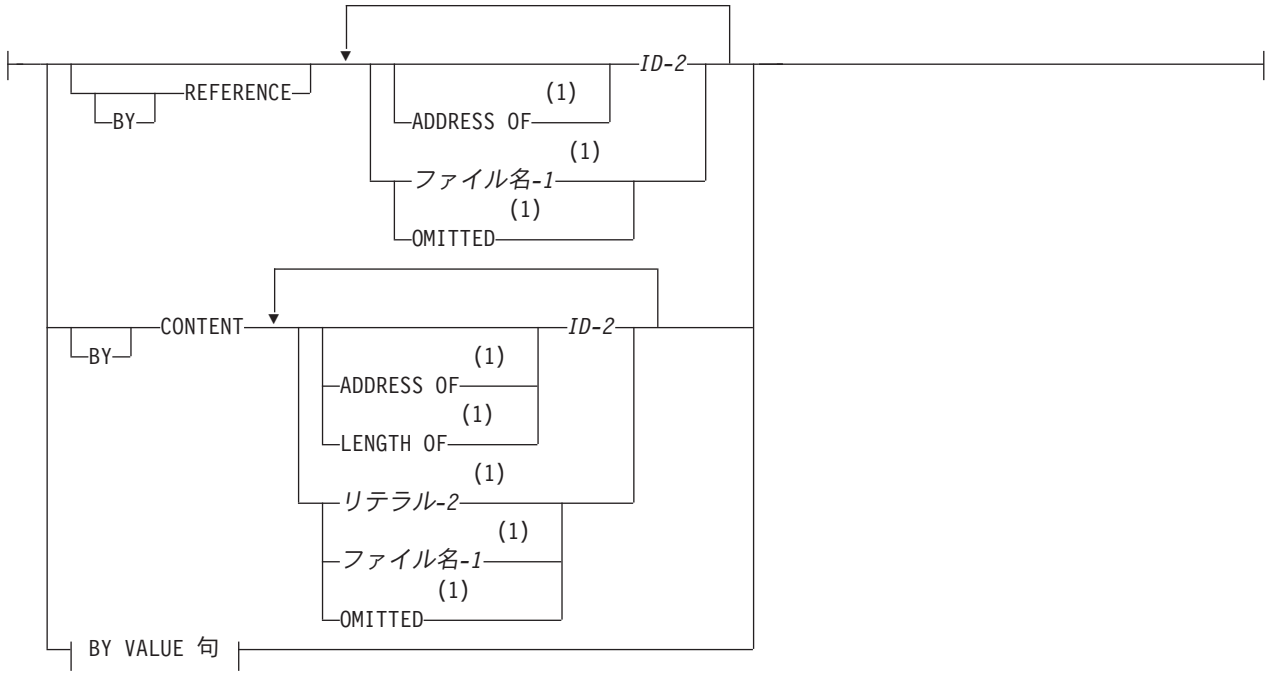
1 IBM 拡張

CALL ステートメント - 形式 2

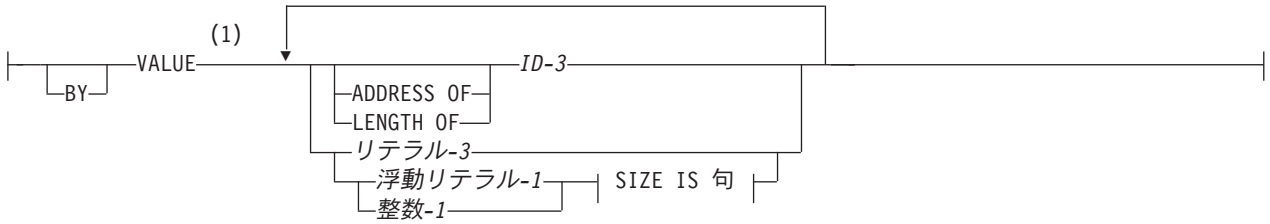




BY 句:

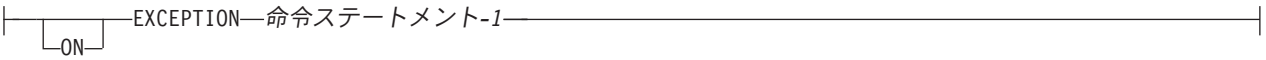


BY VALUE 句:

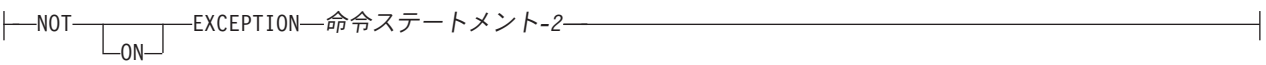


SIZE IS 句:

ON EXCEPTION 句:



NOT ON EXCEPTION 句:



注:

1 IBM 拡張

ID-1

英数字またはプロシージャー・ポインター・データ項目でなければなりません。

英数字データ項目の場合には、以下の規則が適用されます。

- リンケージがプログラム・オブジェクトに対するものであれば、ID-1 の内容はプログラム名の形成規則に従っている必要があります。呼び出し側プログラムと呼び出し先プログラムとの間を対応づけるために、ID-1 の先頭 10 文字が使用されます。
- リンケージがプロシージャーに対するものであれば、ID-1 の最初の 256 文字が使用されます。呼び出し先プロシージャーは、呼び出し側プロシージャーと同じコンパイル単位になければなりません。
- コンパイラー・オプション *MONOPRC によって、ID-1 が英大文字で指定されており、プログラム名の命名規則に従っている必要があります。

プロシージャー・ポインター・データ項目は、CALL の前に SET ステートメントを使用して、プロシージャーまたはプログラムのアドレスに設定する必要があります。プロシージャー・ポインター・データ項目の設定方法について詳しくは 7-209 ページの『形式 6 - プロシージャー・ポインター・データ項目』を参照してください。

リテラル-1

CALL リンケージは呼び出し先プログラムのタイプを判別し、それによりリテラル-1 の内容とサイズを制限することもできます。作成されるリンケージは、プログラム・オブジェクトまたは ILE プロシージャーのいずれかに対するものです。リンケージがプログラム・オブジェクトに対するものであれば、リテラル-1 は非数字で英大文字 (拡張システム名を除く) であり、プログラム名の形成の規則に従っている必要があります。呼び出し側プログラムと呼び出し先サブプログラムとの間に対応づけるために、リテラルの先頭 10 文字が使用されます。リテラル-1 には拡張名を含めることができます。

リンケージが ILE プロシージャーに対するものであれば、リテラル-1 は非数字で長さが最大 256 文字でなければなりません。コンパイラー・オプション *MONOPRC によって、リテラル-1 が英大文字で指定されており、プログラム名の形成規則に従っている必要があります。このリテラルは、呼び出し先サブプログラムのプログラム名を指定していなければなりません。

LINKAGE TYPE 句

IBM Extension

LINKAGE TYPE 句を使用して、CALL のリテラル-1 に対するリンケージのタイプを指定します。

環境名-1
コンパイラーが CALL に対して生成する、リンケージのタイプ。環境名-1 は以下のものとして定義できます。

PGM プログラム・オブジェクト (*PGM) へのリンケージが生成されます。

PRC ILE プロシージャーへのリンケージが生成されます。

PROGRAM
プログラム・オブジェクト (*PGM) へのリンケージ

PROCEDURE
ILE プロシージャーへのリンケージ

CALL ステートメントに LINKAGE TYPE 文節が指定されていない場合、CALL に対して生成されるリンケージは SPECIAL-NAMES 段落の LINKAGE TYPE 句、CRTCBLMOD または CRTBNDCBL コマンドの LINKLIT パラメーターのいずれかを指定することにより、変更できます。

End of IBM Extension

IN LIBRARY 句

IBM Extension

LIBRARY 句を使用して、IBM i のプログラム・オブジェクトを IBM i のライブラリー名で修飾できます。LIBRARY 句が指定されていない場合、プログラム・オブジェクトは、ジョブのライブラリー・リスト (*LIBL) を使用して検索されます。

ID-5
英数字データ項目でなければなりません。 ID-5 の内容は、有効な IBM i ライブラリー名を表していなければなりません。 IBM i ライブラリー名の長さは、最大 10 文字です。 ID-5 の最初の 10 文字が、ライブラリー名を形成するために使用されます。

リテラル-4
非数字でなくてはならず、最大 10 文字まで可能です。

ID-5 およびリテラル-4 は *MONOPRC コンパイラー・オプションの影響を受けず、また、IBM i 拡張名を含むことができます。

End of IBM Extension

USING 句

パラメーターを呼び出し先サブプログラムに渡す必要がある場合に、CALL ステートメントの中に含まれます。呼び出し先サブプログラムが COBOL で書かれた場合は、呼び出し先サブプログラムの手続き部ヘッダーにも USING 句が含まれていなければなりません。両方の USING 句にあるオペランドの数は同一でなければなりません。プログラムの LINKAGE TYPE が指定された CALL ステートメントの場合、オペランドの最大数は 255 であり、またプロシージャーの LINKAGE TYPE の場合はオペランドの最大数が 400 となっています。

CALL ステートメントの USING 句にある ID の順番、および呼び出し先サブプログラムの手続き部ヘッダー中の対応する USING 句にある ID の順序によって、呼び出し側プログラムおよび呼び出し先プログラムで使用される ID の間の対応が決まります。この対応は名前によらず、位置によるものです。 USING 句の詳細は 7-3 ページの『USING 句』を参照してください。

渡されるデータの属性は、呼び出し先サブプログラムの要件によって決定されます。呼び出し先プログラムにいくつかのパラメーターが必要な場合、そのパラメーターを構成するグループ項目でなく、個々のパラメーターの一致するものを指定しなければなりません。

IBM Extension

いくつかのプロシージャー (例えば、ILE CEEDATE および CEEDAYS API) では、1 つまたは複数のパラメーターの**操作記述子**が使用可能である必要があります。この要件は、SPECIAL-NAMES の段落で、適切なパラメーターを指定した USING 句と一緒にプロシージャー用の LINKAGE TYPE 文節を組み込むことにより満たす必要があります。さらに、このようなパラメーターはすべて DISPLAY または DISPLAY-1 の USAGE を使用して基本データ項目として定義される必要があり、参照変更はできません。

End of IBM Extension

CALL ステートメントの USING 句の中で参照されるパラメーター値は、CALL ステートメントが実行されるときに、呼び出し先サブプログラムで利用できるようになります。

USING 句の例

呼び出し側プログラムの記述 (PGMA)

```
WORKING-STORAGE
SECTION.
01 ARG-LIST.
   05 PARTCODE PIC A.
   05 PARTNO PIC X(4).
   05 U-SALES PIC 9(5).
   .
   .
   .
PROCEDURE DIVISION.
   .
   .
   .
      CALL PGMB
       USING ARG-LIST.
```

呼び出し先プログラムの記述 (PGMB)

```
LINKAGE SECTION.
01 PARAM-LIST.
    10 PART-ID PIC X(5).
    10 SALES PIC 9(5).
   .
   .
   .
PROCEDURE DIVISION USING
PARAM-LIST.
```

注: 呼び出し側プログラム内では、パーツ・コード (PARTCODE) とパーツ番号 (PARTNO) が別個に参照されます。呼び出し先サブプログラムにおいては、パーツ・コードとパーツ番号は 1 つのデータ項目 (PART-ID) にまとめられます。そのため、呼び出し先サブプログラムでは PART-ID への参照がそれらに対する有効な、唯一の参照となります。

BY REFERENCE 句

BY REFERENCE 句を通して渡されるパラメーター値は、CALL ステートメントの実行時に評価されます。この値は、呼び出し先プログラムの対応パラメーターに割り当てられます。各パラメーターの文字数は等しくなければなりませんが、データ記述は同一である必要はありません。

ILE COBOL パラメーターに BY REFERENCE が渡されると、元のデータ項目に対するポインターが、呼び出し先プログラムに渡されます。このため、呼び出し先プログラムのパラメーターを変更すると、呼び出し側プログラムのデータ項目が変更されることになります。

ID-2

ファイル・セクション、作業用ストレージ・セクション、ローカル・ストレージ・セクションまたはリンケージ・セクションで、レベル 01、レベル 77、または基本データ項目として定義する必要があります。 関数 ID であってはなりません。

IBM Extension

この句は下記のものになります。

- データ部のあらゆるレベルのデータ項目
- ポインター・データ項目 (USAGE IS POINTER として暗黙に、または明示的に定義された項目)
- プロシージャー・ポインター・データ項目
- DBCS データ項目
- 国別データ項目
- 浮動小数点データ項目
- 日時データ項目

End of IBM Extension

IBM Extension

ADDRESS OF 特殊レジスター

この特殊レジスターについては 6-5 ページの『ADDRESS OF 特殊レジスター』 ページを参照してください。この場合には計算された ADDRESS OF が使用できない点に注意してください。

ファイル名-1

ファイル名-1 は、FD 記入項目内に示さなければなりません。これはヌル・ポインター・データ項目を渡します。

OMITTED

パラメーターに BY REFERENCE が渡されるときの標準のパラメーターの場合、元のデータ項目へのポインターは、呼び出し先プログラムに渡されます。 OMITTED が指定されていると、呼び出し先プログラムに NULL ポインターが渡されます。この場合には、呼び出し先プログラムはそのデフォルト値を使用します。

OMITTED を指定できるのは、プロシージャーの LINKAGE TYPE を指定したプログラムに対する呼び出しに限られます。

End of IBM Extension

BY CONTENT 句

BY CONTENT 句を通して渡されるパラメーター値は、CALL ステートメントの実行時に評価されます。この値は、呼び出し先プログラムの対応パラメーターに割り当てられます。

BY CONTENT が渡される ILE COBOL の各項目に関しては、呼び出し側プログラム中でその項目のコピーが作られ、このコピーのポインターを呼び出し先プログラムに渡します。呼び出し先プログラムの中でそのパラメーターに加えられた変更は、呼び出し側プログラムのデータ項目には影響を与えません。各パラメーターの文字数は等しくなければなりませんが、データ記述は同一である必要はありません。

ID-2

ファイル・セクション、作業用ストレージ・セクション、ローカル・ストレージ・セクションまたはリンケージ・セクションで、レベル 01、レベル 77、または基本データ項目として定義する必要があります。 関数 ID であってはなりません。

IBM Extension

この句は下記のものになります。

- データ部のあらゆるレベルのデータ項目
- ポインター・データ項目 (USAGE IS POINTER として暗黙に、または明示的に定義された項目)
- プロシージャー・ポインター・データ項目
- DBCS データ項目
- 国別データ項目
- 浮動小数点データ項目
- 日時データ項目

End of IBM Extension

IBM Extension

ADDRESS OF 特殊レジスター

この特殊レジスターについては 6-5 ページの『ADDRESS OF 特殊レジスター』 ページを参照してください。

ADDRESS OF データ項目

この項目については 6-4 ページの『ADDRESS OF』 ページを参照してください。

LENGTH OF 特殊レジスター

LENGTH OF 特殊レジスターには、ID で参照されるデータ項目によって使用されるバイト数が含まれます。詳細については 7-78 ページの『LENGTH OF 特殊レジスター』を参照してください。

リテラル-2

リテラル-3 は、下記のものになります。

- 非数字リテラル
- 表意定数
- ブール・リテラル
- DBCS リテラル
- 国別リテラル

ファイル名-1

ファイル名-1 は、FD 記入項目内に示さなければなりません。これはポインター・データ項目を渡します。

OMITTED

パラメーターに BY CONTENT が渡されるときの標準のパラメーターの場合、データ項目のコピーへのポインターは、呼び出し先プログラムに渡されます。 OMITTED が指定されていると、呼び出し先プログラムに NULL ポインターが渡されます。この場合には、呼び出し先プログラムはそのデフォルト値を使用します。

OMITTED を指定できるのは、プロシージャーの LINKAGE TYPE を指定したプログラムに対する呼び出しに限られます。

End of IBM Extension

BY VALUE 句

IBM Extension

BY VALUE 句が指定された場合は、送り出しデータ項目への参照ではなく、パラメーターの値が引き渡されます。呼び出し先プログラムは、BY VALUE パラメーターに対応する仮パラメーターを変更できます。ただし、呼び出し先プログラムは送り出しデータ項目の一時コピーにアクセスできるため、その変更はパラメーターには影響しません。

BY VALUE パラメーターはもともと非 COBOL プログラム (C など) との通信に使用することを意図したものですが、COBOL から COBOL への呼び出しにも使用できます。この場合、BY VALUE は、CALL USING 句におけるパラメーターと PROCEDURE DIVISION USING 句におけるそれに対応する仮パラメーターの両方に対して指定または暗黙に指定しなければなりません。

BY CONTENT 句、BY VALUE 句および BY REFERENCE 句は、他の BY CONTENT 句、BY VALUE 句または BY REFERENCE 句を見つけるまでそれらに続くパラメーターに適用されます。これらの句が最初のパラメーターより前に現れない場合は、BY REFERENCE が想定されます。

BY VALUE 句は、リンケージ・タイプのプログラムを使用して呼び出されたプログラムには使用できません。

ID-3

ファイル・セクション、作業用ストレージ・セクション、ローカル・ストレージ・セクションまたはリンケージ・セクションで、レベル 01、レベル 77、または基本データ項目として定義する必要があります。

この句は、下記のものになります。

- データ部のあらゆるレベルのデータ項目
- ポインター・データ項目 (USAGE IS POINTER として暗黙に、または明示的に定義された項目)
- プロシージャー・ポインター・データ項目
- DBCS データ項目
- 国別データ項目
- 浮動小数点データ項目
- 日時データ項目
- 参照変更。ただし参照変更項目の長さをコンパイル時に知っておく必要があります。

ADDRESS OF 特殊レジスター

この特殊レジスターについては 6-5 ページの『ADDRESS OF 特殊レジスター』 ページを参照してください。

ADDRESS OF データ項目

この項目については 6-4 ページの『ADDRESS OF』 ページを参照してください。

LENGTH OF 特殊レジスター

LENGTH OF 特殊レジスターには、ID で参照されるデータ項目によって使用されるバイト数が含まれます。詳細については 7-78 ページの『LENGTH OF 特殊レジスター』を参照してください。

リテラル-3

リテラル-3 は、下記のものになります。

- 非数字リテラル
- 表意定数

- ブール・リテラル
- DBCS リテラル
- 国別リテラル

浮動リテラル -1
SIZE 句が指定されていない限り、浮動小数点リテラルは、8 バイトの内部浮動小数点 (COMP-2) として渡されます。浮動小数点項目に対して、サイズ句は 4 でも 8 でも可能です。

整数-1
符号付きまたは符号なしの整数です。

整数-1 は 2 進数値として渡されます。整数-2 を指定しないと、整数-1 が 4 バイトの 2 進数値として渡されます。整数-2 は、整数-1 のサイズを指定します。これは、1、2、4 または 8 のいずれかです。

End of IBM Extension

LENGTH OF 特殊レジスター

IBM Extension

LENGTH OF 句は、ID で参照されるデータ項目の現行の長さと等しいバイト数の内容を持つ、特殊レジスターを暗黙に作成します。

LENGTH OF 特殊レジスターには、次のような暗黙の定義があります。

```
USAGE IS BINARY, PICTURE 9(9)
```

LENGTH OF 特殊レジスターの暗黙定義と同じ定義を持つ数字データ項目を使用できる手続き部では、どこでもこの特殊レジスターを使うことができます。

これは、参照修飾子の開始位置または長さの式に現れることがあります。ただし、LENGTH OF 特殊レジスターは、修正された参照であるどのオペランドにも適用できるわけではありません。

LENGTH OF オペランドが関数であってはなりませんが、LENGTH OF 特殊レジスターは、整数のパラメーターが認められている関数では使用できます。

LENGTH OF 特殊レジスターが LENGTH 関数の引数として使用されている場合、その結果は常に 4 になり、LENGTH OF に対して指定された引数に依存しません。

下記のいずれにもすることは**できません**。
- 受け入れデータ項目
- 添え字

LENGTH OF を CALL ステートメントの BY CONTENT 句の中で使用できます。

日時データ項目は、LENGTH OF 特殊レジスターを使用する式の中で使用することができます。 ID は、タイプ名であってもよいし、タイプ名に従属する項目であってもかまいません。

テーブル・エレメントの場合、LENGTH OF 特殊レジスターには 1 回のオカレンスの長さ (バイト単位) が入ります。この場合のテーブル・エレメントを参照する際に、添え字を使用する必要はありません。

可変長エレメントの場合、LENGTH OF 特殊レジスターには、オカレンス依存 (ODO) 変数の現行内容に基づく長さが入ります。

このレジスターは、ID によって参照される区域が現在はプログラムに利用できない場合でも、判別できる長さを持つどの ID に対してもその値を戻します。例えば、ファイル定義において 01 レベル・レコードの一部である ID は該当するファイルがオープンするまで利用できませんが、LENGTH OF などの ID はファイルがオープンする前でも識別可能です。

可変長項目に関して、ODO 変数の内容が利用できない場合、LENGTH OF 特殊レジスターは未定義です。例えば、ODO 変数がオープンしていないファイルの 01 レベル・レコードの中で定義された場合、LENGTH OF 値は存在しないため、エラーが発生します。

LENGTH OF 句を使って参照されるそれぞれの識別名に対して、別個の LENGTH OF 特殊レジスターが存在します。

例えば、次のとおりです。

```
MOVE LENGTH OF A TO B
DISPLAY LENGTH OF A, A
ADD LENGTH OF A TO B
CALL "PROGX" USING BY REFERENCE A BY CONTENT LENGTH OF A
```

注: COBOL 項目によって占有されるバイト数は、組み込み関数 LENGTH を使用してアクセス可能です (7-301 ページの『LENGTH』を参照)。 LENGTH は、データ名の他に非数字リテラルをサポートします。

End of IBM Extension

GIVING/RETURNING 句

IBM Extension

GIVING/RETURNING 句は、リンケージ・タイプのプログラムを使用して呼び出されたプログラムには使用できません。 GIVING と RETURNING は同等です。

ID-4

DATA DIVISION で定義しなければならない RETURNING データ項目。呼び出し先プログラムの戻り値は、ID-4 に暗黙に保管されます。 ID-4 を参照変更することはできません。

ID-4 は、日時データ項目にすることができます。

ADDRESS OF 特殊レジスター

この特殊レジスターについては 6-5 ページの『ADDRESS OF 特殊レジスター』 ページを参照してください。

値を戻す ILE プロシージャーへの呼び出しに RETURNING 句を指定できます。 COBOL サブプログラムへの CALL に RETURNING 句を指定する場合、以下のことを行います。

- 呼び出し先サブプログラムは、その PROCEDURE DIVISION ヘッダーに RETURNING 句を指定しなければなりません。
- ターゲット・プログラムにおける ID-4 および対応する PROCEDURE DIVISION RETURNING ID は、同じ数の文字位置をもっていなければならず、また同じ USAGE と SIGN 文節およびカテゴリーのものでなければなりません。 ID-4 が TYPE 文節を使用して定義された場合は、呼び出し先プログラムの PROCEDURE DIVISION ヘッダーの GIVING/RETURNING 句で参照される項目も、TYPE 文節を使用

して定義しなければなりません。すなわち、両 TYPE 文節において、同一タイプ名を参照する必要があります。制御が呼び出し側プログラムに戻ると、ID-4 または特殊レジスターのその ADDRESS は、戻り値を含みます。

EXCEPTION または OVERFLOW が生じると、ID-4 は変更されません。

RETURNING 句の存在は、RETURN-CODE 特殊レジスターの設定には影響しません。

CALL ステートメントの RETURNING/GIVING 句で参照される項目には、TYPE 句を入れることはできません。

End of IBM Extension

ON EXCEPTION 句

この句は、例外の元の受け取り側が呼び出し側の場合に、プログラムの存在、プログラム活動化、権限、およびストレージから生じる例外を処理するものです。そのときには、以下のいずれかが生じます。

1. CALL ステートメントに ON EXCEPTION 句が現れた場合には、制御が命令ステートメント-1 に移されます。処理はその後、命令ステートメント-1 で指定された各ステートメントの規則に従って続行されます。

   制御を明示的に渡すプロシージャー分岐ステートメントまたは条件ステートメントを実行する場合、制御はそのステートメントの規則に従って渡されます。それ以外の場合は、命令ステートメント-1 がいったん実行されると、制御は CALL ステートメントの最後に移され、また NOT ON EXCEPTION 句が指定されていればそれは無視されます。
2. CALL ステートメントに ON EXCEPTION 句がない場合、NOT ON EXCEPTION 句が指定されていても、無視されます。

NOT ON EXCEPTION 句

例外条件が生じない (言いかえると、呼び出し先サブプログラムが利用可能である) 場合、制御は呼び出し先サブプログラムに移されます。制御が呼び出し先プログラムから戻った後、ON EXCEPTION 句が指定されていれば、それは無視され、制御は CALL ステートメントの最後に (または、NOT ON EXCEPTION が指定されていれば、命令ステートメント-2 に) 移されます。

制御が命令ステートメント-2 に移される場合、処理は命令ステートメント-2 で指定された各ステートメントの規則に従って続行されます。

制御を明示的に渡すプロシージャー分岐ステートメントまたは条件ステートメントを実行する場合、制御はそのステートメントの規則に従って渡されます。それ以外の場合、命令ステートメント-2 がいったん実行されると、制御は CALL ステートメントの最後に移されます。

ON OVERFLOW 句と一緒にこの句を指定すると、エラーが発生します。

ON OVERFLOW 句

ON OVERFLOW 句には、ON EXCEPTION 句と同じ効果があります。

END-CALL 句

この句では CALL ステートメントの有効範囲を定めます。 END-CALL は、条件付き CALL ステートメントが別の条件ステートメントにネストできるようにします。 END-CALL は命令 CALL ステートメントでも使用できます。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

CALL ステートメントの考慮事項

呼び出し ID: CALL *ID* (この場合の *ID* はプロシージャー・ポインターではない) を使用すれば、ネストされた ILE COBOL プログラムまたはプログラム・オブジェクトを呼び出すことができます。 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 をオブジェクトに関連付ける、オープン・ポインターは、その ID を CALL ステートメントで最初に使用する際に設定されます。

あるプログラム・オブジェクトを指す ID を用いて呼び出しを実行し、後でそのプログラムを削除または名前変更する場合は、CANCEL ステートメントを使用して、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、次回、ID を使用してプログラム・オブジェクトを呼び出す際に、関連付けられたオープン・ポインターを設定しなおすことができます。

オープン・ポインターの値は、ID の値を変更してこの新しい値を使用する呼び出しを実行すると、変更されます。

CALL プロシージャー・ポインター: 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*・ステートメントを使用する前に、形式 6 の SET ステートメントを使用して*プロシージャー・ポインター* のデータ項目の値を設定する必要があります。*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 ステートメントに 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、LINKAGE TYPE IS PROGRAM を指定します。

さらに、SPECIAL-NAMES 段落の LINKAGE TYPE 文節、あるいは CRTCBLMOD または CRTBNDCBL 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* のデータ項目がどのタイプのオブジェクトに設定されているかを見分けることができます。 LINKAGE TYPE 文節の使用については 5-20 ページの『LINKAGE TYPE 文節』 を、また LINKLIT パラメーターの使用については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

パラメーターの長さ: 呼び出し側プログラムに定義されるいずれかのパラメーターの長さ (バイト単位) が、呼び出し先プログラムの予期する長さと一致しない場合、呼び出し側プログラムや呼び出し先プログラムで予期しない結果が生じる可能性があります。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」のセクション『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。

プログラム終了ステートメント

STOP RUN、EXIT PROGRAM、GOBACK の各ステートメントを使用して、呼び出し先の ILE COBOL プログラムから制御を戻します。エラー発生時、またはプログラム終了時にプログラム終了ステートメントごとになされる処置は、制御が戻されるのが主プログラムまたはサブプログラムのいずれであるかにより異なります。各種条件における EXIT PROGRAM、STOP RUN、および GOBACK ステートメントの動作について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き* 」の『ILE COBOL プログラムからの戻り』を参照してください。また、個別のプログラム終了ステートメントについての詳細は、以下を参照してください。

- 7-117 ページの『EXIT PROGRAM ステートメント』

- 7-118 ページの『GOBACK ステートメント』
- 7-226 ページの『STOP ステートメント』

IBM i グラフィックス・サポート

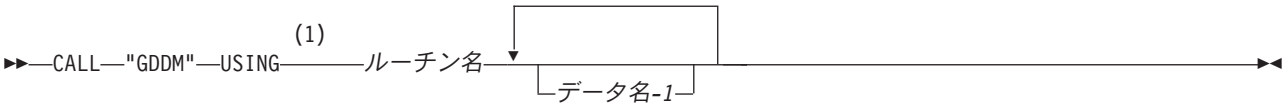
IBM Extension

CALL ステートメントを使用すれば、以下の IBM i グラフィック・ルーチンにアクセスすることができます。

- 図形データ表示管理プログラム (GDDM) - 図形描画のためのグラフィック・プリミティブのセット
- 図形表示ルーチン (PGR) - ビジネス図表ルーチンのセット

上記のすべてのグラフィック・ルーチンにアクセスする場合に、同一形式の CALL ステートメントを使用します。

CALL GDDM ステートメント - 形式



注:

1 IBM 拡張

ルーチン名は、ユーザーが使用しようとしているグラフィック・ルーチンの名前です。

ルーチン名の後に続くデータ名は、特定のグラフィック・ルーチンを使用するのに必要なパラメーターです。指定しなければならないパラメーターの数は、選択するルーチンによって異なります。グラフィック・ルーチンを選択している場合は、おのおののパラメーターが、このルーチンで要求されている大きさとデータ・タイプに合致していることを確認してください。

グラフィック・ルーチン呼び出しの例を下記に示します。 CALL リテラル形式を使用し、かつ、使用するグラフィック・ルーチンが必要とする各パラメーターを定義しなければなりません。

```
MOVE "FSINIT" TO OS-400-GRAPHICS-ROUTINE-NAME.
CALL "GDDM" USING OS-400-GRAPHICS-ROUTINE-NAME.
   .
   .
MOVE "GSFLD" TO OS-400-GRAPHICS-ROUTINE-NAME.
CALL "GDDM" USING OS-400-GRAPHICS-ROUTINE-NAME,
                  PIC-ROW, PIC-COL,
                  PIC-DEPTH, PIC-WIDTH.
```

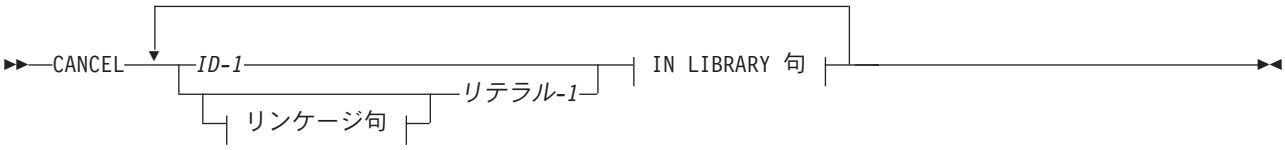
グラフィック・ルーチンおよびそのパラメーターについて詳しくは、資料「*GDDM Programming Guide*」および「*GDDM Reference*」を参照してください。

End of IBM Extension

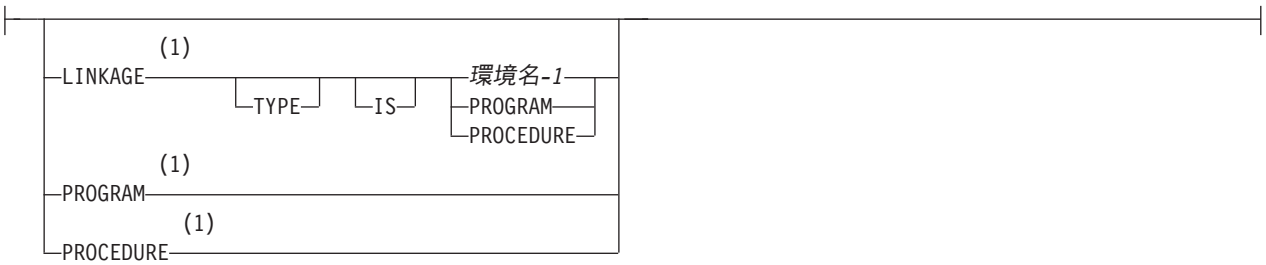
CANCEL ステートメント

CANCEL ステートメントを使用すると、参照されるサブプログラムが次に呼び出されたときに、必ず初期状態から入ります。

CANCEL ステートメント - 形式



リンケージ句:



IN LIBRARY 句:

注:

1 IBM 拡張

リテラル-1

取り消されるサブプログラムの名前を指定します。リテラル-1 には拡張名を含めることができます。CANCEL リンケージは取り消されるプログラムのタイプを判別し、それによりリテラル-1 の内容とサイズを制限することもできます。作成されるリンケージは、プログラム・オブジェクトまたは ILE プロシージャーのいずれかに対するものです。リンケージがプログラム・オブジェクトに対するものであれば、リテラル-1 は非数字で英大文字 (拡張システム名を除く) であり、プログラム名の形成の規則に従っている必要があります。呼び出し側プログラムと呼び出し先サブプログラムとの間に対応づけるために、リテラルの先頭 10 文字が使用されます。

リンケージが ILE プロシージャーに対するものであれば、リテラル-1 は非数字で長さが最大 250 文字でなければなりません。コンパイラー・オプション *MONOPRC によって、リテラル-1 が英大文字で指定されており、プログラム名の形成規則に従っている必要があります。このリテラルは、呼び出し先サブプログラムのプログラム名を指定していなければなりません。

ID-1

以下の規則が適用される場合には、英数字データでなければなりません。

- リンケージがプログラム・オブジェクトに対するものであれば、ID-1 の内容はプログラム名の形成規則に従っている必要があります。呼び出し側プログラムと呼び出し先プログラムとの間を対応づけるために、ID-1 の先頭 10 文字が使用されます。
- リンケージがプロシージャーに対するものであれば、ID-1 の最初の 250 文字が使用されます。
- コンパイラー・オプション *MONOPRC が指定されている場合、ID-1 の内容は英大文字にしてプログラム名の形成規則に従うようにする必要があります。

CANCEL ステートメント

CANCEL ステートメントで指定する各リテラルまたは ID の内容は、関連する CALL ステートメントに指定されているリテラルまたは ID の内容と同じでなければなりません。

IN LIBRARY 句

IBM Extension

この句は、IBM i のプログラム・オブジェクトの取り消しでのみ有効です。つまり、リンケージ・タイプのプログラムを、CANCEL ステートメントで暗黙もしくは明示的に指定しなければなりません。

ID-2
英数字データ項目でなければなりません。 ID-2 の内容は、有効な IBM i ライブラリー名を表していなければなりません。 IBM i ライブラリー名の長さは、最大 10 文字です。ID-2 の最初の 10 文字が、ライブラリー名を形成するために使用されます。

リテラル-2
非数字でなくてはならず、最大 10 文字まで可能です。

ID-2 およびリテラル-2 は *MONOPRC コンパイラー・オプションの影響を受けず、また、IBM i 拡張名を含むことができます。

End of IBM Extension

LINKAGE TYPE 句

IBM Extension

LINKAGE TYPE 句を使用して、CANCEL が対象とするプログラムのタイプを指定します。対象として考えられるのは、プログラム・オブジェクト (*PGM) または ILE プロシージャーです。

環境名-1
CANCEL ステートメントが影響するプログラムのタイプです。環境名-1 は以下のものとして定義できます。

PGM
プログラム・オブジェクト (*PGM)

PRC
ILE プロシージャー

PROGRAM
プログラム・オブジェクト (*PGM) が取り消されます。

PROCEDURE
ILE プロシージャーが取り消されます。

CANCEL ステートメントに LINKAGE TYPE 文節が指定されていない場合、取り消せるプログラムのタイプは SPECIAL-NAMES 段落の LINKAGE TYPE 句、CRTCBLMOD または CRTBNDCBL コマンドの LINKLIT パラメーターのいずれかを指定することにより、変更できます。

End of IBM Extension

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な結びつきは終了します。サブプログラムが記述する外部データ・レコードにあるデ

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。このサブプログラムを指定している実行単位中のプログラムが CALL ステートメントを後に実行する場合には、このサブプログラムには初期状態で入ります。

CANCEL ステートメントは、オープンしているすべての INTERNAL ファイルをクローズします。

呼び出し先サブプログラムは、以下のいずれかの方法で取り消すことができます。

- そのサブプログラムを CANCEL ステートメントのオペランドとして参照する
- そのサブプログラムがメンバーとなっている実行単位を終了する (これは同じ実行単位における STOP RUN、あるいはその実行単位の主プログラムからの GOBACK により実行できます。)
- 呼び出し先サブプログラムに INITIAL 属性が付いている場合、そのサブプログラムで EXIT PROGRAM ステートメントを実行する
- 呼び出し先サブプログラムに INITIAL 属性が付いている場合、そのサブプログラムで GOBACK ステートメントを実行する

CANCEL ステートメントは、指定されたプログラムだけで実行されるものであって、取り消されたプログラムによって呼び出し先プログラムでは実行されません。

呼び出し先サブプログラムの中に CANCEL ステートメントが入っていてもかまいません。また、呼び出し先サブプログラムには、その呼び出し側プログラムや、呼び出し階層でそれ自体より上位にある他のプログラムを、直接または間接的に取り消す CANCEL ステートメントがあってはなりません。呼び出し先サブプログラムがその呼び出し側プログラムを取り消そうとすると、そのサブプログラムにある 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、一度呼び出されたが、まだ制御が呼び出し側プログラムへ戻されていないプログラムを参照することができません。例えば、

```
A が B を呼び出し、B が C を呼び出す        (A が制御を受け取った時点で、
                                  C を取り消すことができる)
A が B を呼び出し、A が C を呼び出す        (C が制御を受け取った時点で、
                                  B を取り消すことができる)
```

実行単位の中で呼び出されていないプログラムの名前を CANCEL ステートメントで指定したり、あるいは呼び出されはしたがすでに取り消されているプログラムの名前を指定した CANCEL ステートメントを実行しても、何も行われません。どちらの場合も、制御は次のステートメントへ移ります。

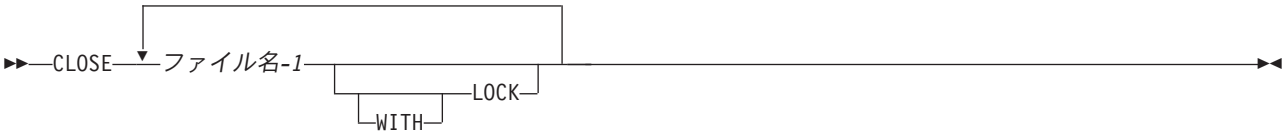
プロシージャーおよびプログラム・オブジェクトの取り消しについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

CLOSE ステートメント

CLOSE ステートメントは、ボリュームおよびファイルの処理を終了させます。ここで指定できるのは、REWIND または LOCK (あるいはその両方)、または REMOVAL です。

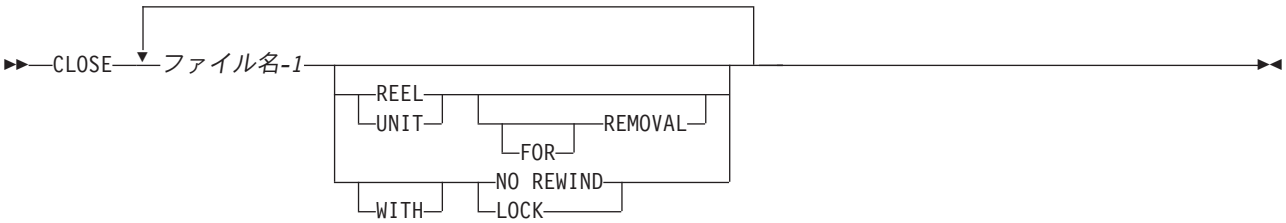
CLOSE ステートメント - 形式 1

CLOSE ステートメント - 形式 1



CLOSE ステートメント - 形式 2 - テープ・ファイル

CLOSE ステートメント - 形式 2 - テープ・ファイル



ファイル名-1

ファイル名-1 は、CLOSE ステートメントが処理するファイルについて指定します。複数のファイル名が指定されている場合、ファイルは同一の編成またはアクセスである必要はありません。また、ファイル名-1 は、ソート・ファイルまたはマージ・ファイルであってはなりません。

CLOSE ステートメントの考慮事項

CLOSE ステートメントは、オープン・モードであるファイルに対してしか実行できません。 REEL 句または UNIT 句のない CLOSE ステートメントが正常に実行されると、次のような結果が生じます。

- ファイル名-1 に関連するレコード域が使用できなくなります。 CLOSE ステートメントの実行が失敗すると、レコード・データの使用可能性が未定義のままにとどまります。
- 該当のファイルに対する OPEN ステートメントは、それ以外の入出力ステートメントに先立って実行しなければなりません。

以下の考慮事項は、CLOSE ステートメントの使用に適用されます。

- ファイルがオープン状況で CLOSE ステートメントの実行が失敗した場合、このファイルの EXCEPTION/ERROR プロシージャーが (指定されていれば) 実行されます。
- REEL 句または UNIT 句のない CLOSE ステートメントが、実行単位が終了する前、あるいはプログラムが取り消される前に処理されなかった場合、ファイルは暗黙的にクローズされます。
- 該当のファイルのファイル制御記入項目で SELECT OPTIONAL 文節が指定されてはいるが、そのファイルが実行時に存在しない場合には、標準のファイル終わり処理は実行されません。
- FILE-CONTROL 記入項目に FILE STATUS 文節が指定されている場合には、CLOSE ステートメントの実行時に、関連した状況キーが更新されます。状況キーの詳細は 7-37 ページの『共通の処理機能』を参照してください。
- **相対ファイルのみ**: 相対ファイルの境界を現時点のレコード数を超えてファイル・サイズの範囲内で拡張するには、削除されたファイルを INZPFM コマンドを使用してそのファイルの処理前に追加します。さ

らに多くのレコードがファイルに追加される必要があり、ファイル状況 0Q が受け取られた場合、これを行う必要があります。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。

WITH LOCK 句: COBOL は、この COBOL プログラムの処理中に該当するファイルが再びオープンされることがないようにします。 WITH LOCK でクローズした外部ファイルは、実行単位内では再びオープンすることができません。これには、外部ファイルで定義されているその他のプログラムも含まれます。

装置タイプ TAPEFILE だけに関する特別の考慮事項

装置タイプ TAPEFILE のファイルは、次の 2 つのカテゴリーに分割できます。

順次単一ボリューム: 1 つのボリューム (リール) に完全に含まれている順次ファイル。このボリュームには複数のファイルが含まれることもあります。

順次マルチボリューム: 複数のボリュームに含まれる順次ファイル。このファイルは、単一ボリューム上で保留できる量以上のデータを含んでいるか、あるいは複数のボリュームにまたがって計画的に分割されたかのいずれかです。

以下の句は、装置タイプ TAPEFILE にのみ適用されます。

- NO REWIND 句
- REEL または UNIT 句
- FOR REMOVAL 句

これらのうちのいずれの句も指定されていない場合には、CLOSE ステートメントによって現行のボリュームが、その先頭に位置付けられます。

順次マルチボリューム・ファイルの場合、REEL 句または UNIT 句を含まない CLOSE ステートメントは、現行のボリューム以外のどのボリュームにも影響を与えません。

NO REWIND 句

現在のボリュームは現在の位置にとどまっています。

REEL または UNIT 句

REEL 句または UNIT 句が出力ファイルに対して指定されている場合、それは、順次マルチボリューム・ファイルが作成されて、ファイルの現行のボリュームに書き込むレコードがこれ以上存在しないことを示しています。次の処理が行われます。

1. 標準ラベルが、現行ボリュームの終わりに書き込まれます。
2. ファイルの続きを受け取るために、新しいボリュームの取り付けを要求するメッセージが発行されます。
3. 標準ラベルが、新しいボリュームの先頭に書き込まれます。
4. 次に処理される WRITE ステートメントによって、新しく取り付けられたボリュームにレコードが書き込まれます。

入力のためにオープンされた順次マルチボリューム・ファイルに対して、REEL 句または UNIT 句が指定された場合、標準ラベルを読み込むために現行のボリュームが位置付けられます。それがファイルの最後のボリュームである場合は、プログラムは継続され、次に処理される READ ステートメントが AT END 条件を発生させます。それがファイルの最後のボリュームではない場合、以下のことが行われます。

1. ファイルの新しいボリュームの取り付けを要求するメッセージが発行されます。

2. 次のボリュームの先頭の標準ラベルが処理されます。
3. 次に処理される READ ステートメントが、新たに取り付けられたボリュームの最初のレコードを要求します。

入力のためにオープンされた順次単一ボリューム・ファイルの場合、REEL 句または UNIT 句の指定はオプショナルです。これらの句は構文チェックされるだけで、実行時にはどのような機能も行いません。

FOR REMOVAL 句

順次マルチボリューム・ファイルの場合、REEL 句または UNIT 句に FOR REMOVAL 句を追加すると、現行のボリュームが巻き戻されてアンロードされます。システムにはその後、ボリュームがこの実行単位から論理的に取り外されたことが通知されます。ただし、このボリュームは、ファイルを REEL 句または UNIT 句なしの CLOSE ステートメントでクローズした後に再オープンすれば、再びアドレッシングできるようになります。

入力のためにオープンされた順次単一ボリューム・ファイルの場合、FOR REMOVAL 句を使用するかどうかはオプショナルです。これらの句は構文チェックされるだけで、実行時にはどのような機能も行いません。

IBM Extension

順次マルチボリューム・ファイルの場合は、CLOSE ステートメントに REEL 句または UNIT 句が指定されていると、たとえ FOR REMOVAL 句が含まれていなくても、システムは常にそのボリュームを巻き戻してアンロードします。

ファイルは、実行単位の終了できたとき、またはプログラムが取り消されたとき、REEL 句または UNIT 句なしの CLOSE ステートメントが処理される前に暗黙的にクローズされます。この状況が発生した場合、現行のボリュームは、ファイルのシステム記述内にある ENDOPT キーワードに定義されているように位置付けされたまま残されます。このキーワード (LEAVE、REWIND、または UNLOAD の値を取ることができる) は、ファイル記述が CRTTAPF コマンドによって作成されるときに設定されます。 ENDOPT キーワードは、CHGTAPF コマンドを使用して変更するか、または OVRTAPF コマンドを使用して指定変更できます。

End of IBM Extension

COMMIT ステートメント

IBM Extension

COMMIT ステートメントを使用することによって、データベース・レコードに対する変更を同期化するとともに、COMMIT が実行されるまでは他のジョブがこれらのレコードを修正しないようにすることができます。 COMMIT ステートメントの形式は次のとおりです。

COMMIT ステートメント - 形式

COMMIT ステートメントが実行されると、前のコミットメント境界以降に現在のコミットメント定義のために、コミットメント制御下のファイルに対して行われたすべての変更が永続的なものになります。コミットメント境界は、ROLLBACK または COMMIT ステートメントが正常に実行されることによって設定され

ます。現行のジョブ中で COMMIT または ROLLBACK がまだ一度も出されていない場合、コミットメント境界は、そのジョブの中でコミットメント制御下のファイルが最初に OPEN されたときに設定されます。変更はコミットメント制御下のすべてのファイルに対して行われるものであり、単に COMMIT ステートメントを出す COBOL プログラム内のコミットメント制御下のファイルに対してのみ行われるものではありません。

COMMIT が実行されると、コミットメント制御下のファイルに対する最後のコミットメント境界以降に、現行のコミットメント定義によって保持されたすべてのレコード・ロックが開放され、レコードが使用可能になります。コミットメント制御は、ジョブ・レベルまたは活動化グループ・レベルで範囲を限定できます。コミットメント制御の範囲は、デフォルトでは活動化グループとなっています。これは、異なる活動化グループ (CL プログラム) でアプリケーションが非 ILE COBOL プログラムに関係している場合に重要です。

COMMIT ステートメントは、コミットメント制御下のファイルに対してのみ影響を与えます。 COMMIT が実行されても、コミットメント制御下のファイルがオープンされていない場合は、COMMIT ステートメントは何ら効力をもたず、コミットメント境界も設定されません。

COMMIT ステートメントは以下のことを*行いません*。

- どのファイルの I-O-FEEDBACK 域も修正しません。
- どのファイルのファイル位置標識も変更しません。
- どのファイルのファイル状況値も設定しません。

コミットメント制御について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

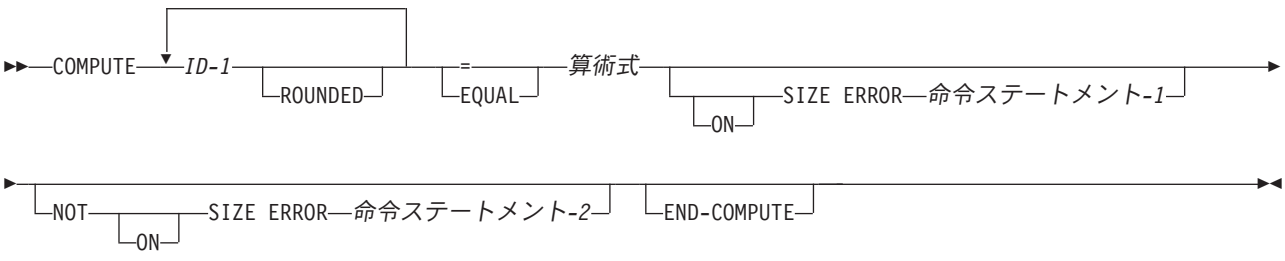
End of IBM Extension

COMPUTE ステートメント

COMPUTE ステートメントは、1 つまたは複数のデータ項目に算術式の値を割り当てます。

COMPUTE ステートメントを使うと、ADD、SUBTRACT、MULTIPLY、および DIVIDE ステートメントの規則による受け入れデータ項目に関する制約に拘束されることなく、算術演算を組み合わせることができます。

COMPUTE ステートメント - 形式



しかし、可搬性が求められるのであれば、COMPUTE を使用する代わりに
ADD、SUBTRACT、MULTIPLY、および DIVIDE を使用してください。これは、システム特有の中間結果が異なる可能性があるためです。

COMPUTE ステートメント

算術演算を組み合わせた場合には、COMPUTE ステートメントは、別個の算術ステートメントを続けてするよりも効率的です。

ID-1
数字基本項目または数字基本編集項目のいずれかでなければなりません。

IBM Extension

基本浮動小数点データ項目が可能です。

End of IBM Extension

算術式
7-8 ページの『算術式』に定義されているように、任意の算術式が可能です。

COMPUTE ステートメントの実行時には、算術式の値が計算され、その結果の値が、ID-1 によって参照される各データ項目の新しい値として保管されます。

単一の ID、数字、関数、または数字リテラルから成る算術式を使用すれば、ID-1 によって参照されるデータ項目の値を、算術式の ID またはリテラルの値に等しく設定できます。

ROUNDED 句
7-33 ページの『ROUNDED 句』を参照してください。

SIZE ERROR 句
7-33 ページの『SIZE ERROR 句』を参照してください。

END-COMPUTE 句
この明示範囲終了符号は、COMPUTE ステートメントの範囲を区切るのに役立ちます。 END-COMPUTE を使用することによって、COMPUTE 条件ステートメントを別の条件ステートメントにネストすることができます。また、END-COMPUTE は COMPUTE 命令ステートメントと一緒に使用することもできます。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

CONTINUE ステートメント
CONTINUE ステートメントにより、ノーオペレーション・ステートメントを指定できます。 CONTINUE は、実行可能な命令が存在しないことを示します。

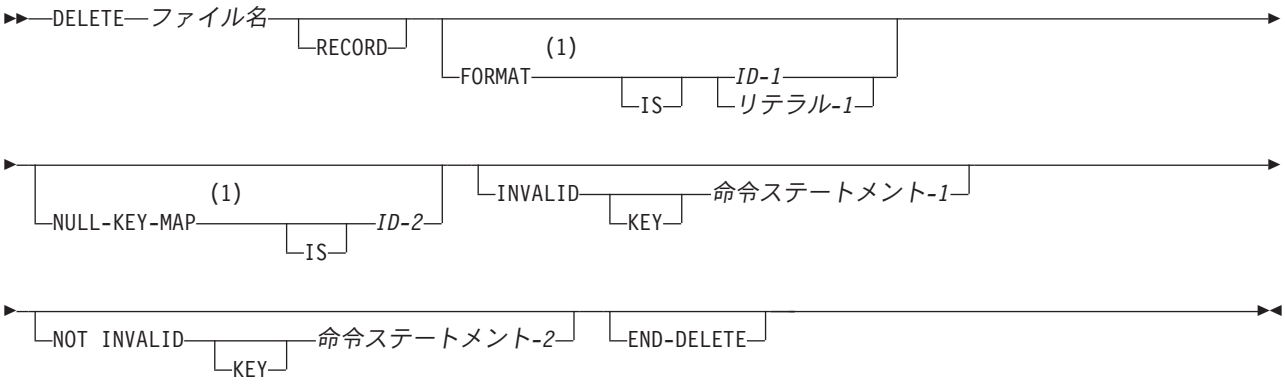
CONTINUE ステートメント - 形式

CONTINUE ステートメントは、条件ステートメントまたは命令ステートメントを使用できるところにはどこでも使用できます。これは、プログラムの実行には影響を与えません。

DELETE ステートメント
DELETE ステートメントは、レコードを索引付きファイルまたは相対ファイルから削除します。索引付きファイルの場合は、レコードを削除した結果生じた未使用キーが、直ちにレコードの追加処理に使用できるようになります。相対ファイルの場合は、この空間が、同じ RELATIVE KEY 値を持つ新しいレコードに対して使用できるようになります。

DELETE ステートメントの実行時には、関連するファイルが I-O モードでオープンされていなければなりません。

DELETE ステートメント - 形式



注:

1 IBM 拡張

ファイル名
ファイル名はデータ部の FD 記入項目で定義されていなければならず、また、索引付きファイルまたは相対ファイルの名前が付けられていなければなりません。

DELETE ステートメントの実行が正常に終了すると、レコードはファイルから論理的に削除され、以後そのレコードにアクセスすることはできません。 DELETE ステートメントの実行は、ファイル名に関連したレコード域の内容 (あるいは、*ファイル名* に関連した RECORD 文節の DEPENDING ON 句に指定されたデータ名が参照するデータ項目の内容) には影響しません。

ファイル制御記入項目の中で FILE STATUS 文節が指定されている場合には、DELETE ステートメントの実行時に、関連する状況キーが更新されます。

ファイル位置標識は、DELETE ステートメントの実行による影響を受けません。

DELETE ステートメントの考慮事項

IBM Extension

データベース・ファイル一時変更 (OVRDBF) CL コマンドの書き込み禁止 (INHWRT) パラメーターによって、このステートメントの処置をプログラム実行時に禁止することができます。このパラメーターが指定された場合、データに依存するエラーに対しては、非ゼロのファイル状況コードが設定されません。データに依存するエラーの例として、重複キーおよびデータ変換エラーがあります。 OVRDBF コマンドについて詳しくは、**IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL および API*』を参照してください。

End of IBM Extension

順次アクセス・モード

順次アクセス・モードにおける索引付きファイルまたは相対ファイルの場合、

- システムは DELETE ステートメントの処理時に、READ ステートメントによって検索されてロックされたレコードを論理的に削除します。

  最終の入出力ステートメントが NO LOCK 句**なしの** READ ステートメントで、正常に処理されていなければなりません。

  最終入出力ステートメントが、NO LOCK 句を**持つ** READ ステートメントで、かつ正常に処理されていれば、次のようになります。
  - ファイル状況キー (定義されている場合) が 9S に設定されます。
  - EXCEPTION/ERROR プロシージャー (指定されている場合) が実行されます。
  - DELETE ステートメントは処理されません。

  IBM Extension

  最後の入出力ステートメントが、正常に終了した READ ステートメントでなかった場合、ファイル状況キー (定義されている場合) は 43 に設定されます。

  End of IBM Extension

  ファイル・ロックおよびレコード・ロックについては、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。
- 順次アクセス・モードのファイルの場合、INVALID KEY 句や NOT INVALID KEY 句を指定することは**できません**。 ただし、EXCEPTION/ERROR プロシージャーは指定できます。

  エラー処理に関する情報については 7-37 ページの『共通の処理機能』を参照してください。

ランダム・アクセス・モードまたは動的アクセス・モード

ランダムおよび動的の各アクセス・モードにおいては、DELETE ステートメントを使用した結果はファイル編成により異なってきます。

これが相対ファイルであれば、システムは RELATIVE KEY データ項目の内容により識別されるレコードを論理的に除去します。同じ RELATIVE KEY の値をもった新規レコード用に空間を入手できます。ファイルに当該レコードがない場合には、INVALID KEY 条件が発生します。

索引付きファイルにおいては、システムは RECORD KEY データ項目の内容により識別されるレコードを論理的に除去します。ファイルに当該レコードがない場合には、INVALID KEY 条件が発生します。

IBM Extension

ファイルに EXTERNALLY-DESCRIBED-KEY が指定されている場合、FORMAT 句によって指定された形式のレコード域 2 の中のキー・フィールドを使用して、削除すべきレコードを見つけます。 FORMAT 句が指定されていない場合、ファイルに対してプログラム定義された最初の形式を使用して、削除すべきレコードを見つけます。

End of IBM Extension

1. レコード域の中のキー・フィールドは、検索引数を作成するためのレコード形式または仕様に従って選択したバッファーのロケーションです。

DUPLICATES 句:

IBM Extension

ファイルに対してこの句が指定されていた場合、DELETE ステートメントの実行に先立ってこのファイルに対して処理された最後の入出力ステートメントは、NO LOCK 句**なしの** READ ステートメントであり、しかも、正常に実行されていなければなりません。 そのステートメントによって読み取られたレコードが、削除されるレコードです。

この場合、FORMAT 句は、削除すべきレコードを見つけるためには使用されません。重複がある場合には、適切なレコードが削除されないようにするために、READ ステートメントが必要になります。

削除操作の前に READ 操作が正常に終了しなかった場合、以下のことが行われます。

- ファイル状況キー (定義されている場合) が 94 に設定されます。
- EXCEPTION/ERROR プロシージャー (指定されている場合) が実行されます。
- DELETE ステートメントは処理されません。

最終入出力ステートメントが、NO LOCK 句を**持つ** READ ステートメントで、かつ正常に処理されていれば、次のようになります。

- ファイル状況キー (定義されている場合) が 9S に設定されます。
- EXCEPTION/ERROR プロシージャー (指定されている場合) が実行されます。
- DELETE ステートメントは処理されません。

レコードが読み取られてから RECORD KEY データ項目の値が変更された場合には、次のようになります。

- 定義されている場合に、ファイル状況キーが 21 に設定されます。
- INVALID KEY 条件になります。
- DELETE ステートメントは処理されません。

End of IBM Extension

FORMAT 句

IBM Extension

FORMAT 句は、装置タイプが DATABASE である索引付きファイルだけに適用されます。これは、複数のレコード形式をもち、固有のキーを持つファイルの処理中に必要です。レコード・キーが複写で定義されていると、FORMAT 句は誤りとなり無視されます。

FORMAT 句で指定する値には、この入出力操作で使用するレコード形式の名前が入ります。システムは、これを使用して、どのレコード形式が操作に際して必要があるかを指定または選択します。

ID を指定する場合には 10 文字以下の文字ストリングでなければならず、さらに以下のいずれかの名前である必要があります。

- 作業用ストレージ・セクションの記入項目
- リンケージ・セクションの記入項目
- 以前にオープンされたファイルに関するレコード記述記入項目

リテラルを指定する場合、10 文字以下の英大文字の文字ストリングでなければなりません。全桁がブランクの値は、FORMAT 句が指定されなかった場合と同じ扱いになります。ブランクの値がそのファイルに関して有効でない場合は、FILE STATUS として 9K が戻され、そのファイルに対応可能であれば USE プロシージャーが呼び出されます。

End of IBM Extension

NULL-KEY-MAP IS 句

IBM Extension

NULL-KEY-MAP IS 句については 7-221 ページの『NULL-KEY-MAP IS 句』の START ステートメントについての説明を参照してください。

End of IBM Extension

INVALID KEY 句

適用可能な USE プロシージャーが指定されていないファイルに対しては、INVALID KEY 句を指定する必要があります。詳細は 7-38 ページの『INVALID KEY 条件』を参照してください。

NOT INVALID KEY 句

NOT INVALID KEY 句のある DELETE ステートメントが正常に処理された後は、その句に関連した命令ステートメントに制御が渡されます。

END-DELETE 句

この明示範囲終了符号は、DELETE ステートメントの範囲を区切ります。これにより DELETE 条件ステートメントを別の条件ステートメントにネストすることができます。また、END-DELETE は DELETE 命令ステートメントでも使用できます。

DISPLAY ステートメント

DISPLAY ステートメントは、各オペランドの内容を出力装置へ転送します。各オペランドの内容は、オペランドがリストされている順序で左から右に出力装置へ表示されます。

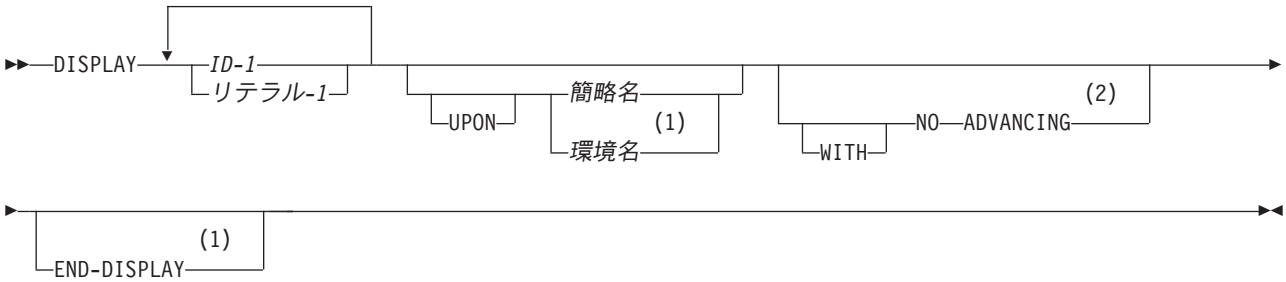
IBM Extension

- 形式 2 - 内部データ域
- 形式 3 - ワークステーション I/O
- 形式 4 - セッション入出力
- 形式 5 - データ域

End of IBM Extension

形式 1 - データの転送

DISPLAY ステートメント - 形式 1 - データ 転送



注:

1 IBM 拡張

2 構文検査だけ行われます。

ID-1

IBM Extension

ID-1 の記述に TYPE 文節が含まれる場合は、その文節で指定されるタイプ名は基本項目でなければなりません。

End of IBM Extension

これが数字であって外部 10 進数としては記述されていない場合、ID は以下の方法で外部形式に自動的に変換されます。

- 2 進数項目または内部 10 進数項目は、外部 10 進数に変換されます。負の符号付きの値を使用すると、低位桁の符号が表示されます。例えば、SEPARATE CHARACTER を設定した SIGN が指定されていなくて、2 つの数値項目が -34 および 34 の値を持つ場合には、これらの値はそれぞれ 3M および 34 と表示されます。 SEPARATE CHARACTER を設定した SIGN が指定されている場合には、その数字がどのように指定されているかに従って、+ または - の符号が数字の先頭または末尾に表示されます。

  注: パック、2 進数、浮動小数点、または日時データ (COMP、COMP-1、COMP-2、COMP-3、PACKED-DECIMAL、BINARY、COMP-4、または COMP-5) の入っているグループ項目をディスプレイ装置に表示してはなりません。このようなデータにはディスプレイ装置制御文字が入っていることがあるので、予期しない結果を引き起こすことになります。

IBM Extension

- 内部浮動小数点データ項目であっても外部浮動小数点データ項目であってもかまいません。内部浮動小数点数は、例えば次のように、表示される前に外部浮動小数点数に変換されます。
  - COMP-1 項目は、-.9(8)E-99 という外部浮動小数点 PICTURE 文節を持っているかのように表示されます。
  - COMP-2 項目は、-.9(17)E-999 という外部浮動小数点 PICTURE 文節を持っているかのように表示されます。

  外部浮動小数点リテラルが表示される場合は、結果が少し不正確になる可能性があります。このことは、DISPLAY が MOVE の後に来る場合に、特にそうであると言えます。浮動小数点データ・タイ

プは近似値で、外部浮動小数点リテラルが転送された場合に、この浮動小数点データ・タイプはまず真の浮動小数点値 (IEEE) に変換されますが、この変換も浮動小数点データ・タイプの正確性に影響を与える可能性があります。

例えば、以下の DISPLAY を考えてみます。

```
77 external-float-1 PIC +9(3).9(13)E+9(3).
    MOVE +123455779012.3453E+297 to external-float-1.
    DISPLAY "EXTERNAL-FLOAT-1=" external-float-1.
```

MOVE の後で表示される結果は以下のとおりです。

```
EXTERNAL-FLOAT-1=+123.4557790123452E+306
```

End of IBM Extension

- 上記以外の ID は、変換する必要がありません。

IBM Extension

- 基本 DBCS データ項目と国別データ項目は出力装置に転送されます。 DBCS、国別、および SBCS オペランドはいずれも verb、DISPLAY を 1 つ使用するだけで指定できます。出力されるデータは、ジョブの現行 CCSID によって指定されるコード・セットに変換されます。

End of IBM Extension

リテラル-1

リテラル-1 は、任意の表意定数を指定できます。 表意定数が指定された場合には、その表意定数の 1 文字だけが表示されます。

おのおののリテラルは、符号なし整数でなければなりません。

IBM Extension

浮動小数点リテラルを使用できます。

End of IBM Extension

IBM Extension

符号付きで非整数の、数字リテラルが使用可能です。

End of IBM Extension

IBM Extension

DBCS リテラルおよび国別リテラルが使用可能です。verb、DISPLAY 内で、表意定数 ALL を DBCS リテラルおよび国別リテラルと一緒に使用できます。

End of IBM Extension

UPON

UPON 句は簡略名を指定しますが、これはワークステーション (REQUESTOR)、またはシステム・オペレーターのメッセージ・キュー (CONSOLE または SYSTEM-CONSOLE) のいずれかに関連した簡略名でなければなりません。

IBM Extension

簡略名の代わりに**環境名**を指定することもできます。有効な環境名は CONSOLE および SYSOUT です。

End of IBM Extension

UPON 句を省略した場合には、DISPLAY ステートメントは REQUESTOR へ出力を送り出します。

WITH NO ADVANCING

この句に対しては構文検査だけが行われ、無視されます。 WITH NO ADVANCING 句の機能的な説明については 7-104 ページの『形式 4 - セッション入出力』を参照してください。

DISPLAY ステートメントの動作

DISPLAY ステートメントが実行されるとき、送り出しフィールドに入っているデータが出力装置へ転送されます。 送り出しフィールドの大きさは、リストされたすべてのオペランドの合計文字数です。ハードウェア装置が、転送中のデータ項目と同じ大きさのデータを受け取ることができる場合は、このデータ項目が転送されます。ハードウェア装置が、転送中のデータ項目と同じ大きさのデータを受け取ることができない場合は、次のいずれかが適用されます。

- 合計文字数が最大の装置論理レコード長より小さい場合には、残りの右端の文字にスペースが埋め込まれます。
- 合計文字数が最大の論理レコード長より大きい場合には、すべてのオペランドを表示するために必要な数のレコードが書き出されます。レコードの終わりに到達したときに印刷中または表示中のオペランドが残っていれば、そのオペランドは次のレコードに継続されます。

IBM Extension

- DBCS オペランドまたは国別オペランドが複数のレコードに分割される場合は、2 バイト境界でのみ分割されます。

End of IBM Extension

最後のオペランドが出力装置に転送された後、この出力装置は、次の行の左端の位置にリセットされます。

論理レコードの長さは装置によって次のとおり異なります。

出力	最大論理レコード・サイズ
ジョブ・ログ	120 文字
ワークステーション	58 文字
システム・オペレーターのメッセージ・キュー	58 文字

IBM Extension

DBCS または国別文字の項目またはリテラルが verb、DISPLAY 内で指定された場合、送り出しフィール

ドの大きさは、リストされたすべてのオペランドの合計文字数です。この合計文字数は、1 つの DBCS 文字を 2 バイトと数え、DBCS に必要なシフト・コードをすべて足した数です。

End of IBM Extension

バッチ・ジョブ中のあるプログラムが、UPON 句のない DISPLAY ステートメントを実行する場合、または REQUESTOR と関連する UPON 句を指定した DISPLAY ステートメントを実行する場合は、重大度 99 の通知メッセージのジョブ・ログに出力が送られます。このメッセージの重大度は、メッセージ記述変更 (CHGMSGD) CL コマンドを使用して変更できます。詳しくは、**IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL および API*』を参照してください。

ディスプレイ・ファイルを使用する対話式ジョブの場合、DISPLAY ステートメントは、通常、使用されません。もしこれらのステートメントを使用する場合は、以下の点を考慮しなければなりません。

対話式ジョブが DISPLAY ステートメントを実行する場合、プログラム・メッセージの表示画面に論理レコードが表示されます。

以下の画面は、プログラム・メッセージの表示の例を示しています。

```
                        プログラム・メッセージの表示

JOB 000745/QPGMR/WS1 started on 02/17/92 at 14:50:22 in subsystem QINTER 1
SAMPLE PROGRAM MESSAGE FROM PREVIOUS CALL OF PROGRAM. 2
SAMPLE PROGRAM MESSAGE FROM CURRENT CALL OF PROGRAM. 2
```

- **1** このセッションに対するシステム・メッセージ。
- **2** このセッションに対するプログラム・メッセージ。

この表示には、このセッションでの他の活動に関連したメッセージの他に、現在実行されているプログラムからのメッセージが含まれています。

DISPLAY ステートメントが処理されるとき、画面に関するディスプレイ・ファイルの特性によって、プログラムの処理を中断すべきかどうかが決められます。

- RSTDSP(*NO)

  このパラメーターをディスプレイ・ファイルの変更・作成中に指定すると、処理中の DISPLAY ステートメントがプログラムの処理を中断し、画面にはプログラム・メッセージ表示が現れます。プログラムの実行を再開するには、実行キーを押し、直前の表示画面にすぐに戻ってください。

- RSTDSP(*YES)

  このパラメーターをディスプレイ・ファイルの変更・作成中に指定するか、コマンド入力画面から DISPLAY ステートメントを実行した場合は、DISPLAY ステートメントの処理はプログラムの処理を中断しません。

  プログラム・メッセージ表示が画面に現れ、以下のいずれかが生じるまでそのまま表示されます。

  – プログラムが、そのファイルに対して非サブファイルの READ または WRITE ステートメントを処理します。プログラム・メッセージ表示は消え、前の表示が画面に戻されます。
  – プログラムは終了します。

注: プログラムの実行を中断したい場合は、DISPLAY ステートメントの後に ACCEPT ステートメントをコーディングします。これによって、実行キーが押されるまでプログラムの実行が中断されます。

プログラム終了後に出力レコードを表示する場合は、コマンド入力画面から F10 キーを押します。

対話式処理の追加情報については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。RSTDSP パラメーターの追加情報については、**IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL および API*』の CHGDSPF および CRTDSPF コマンドを参照してください。

ワークステーションのオペレーターによって開始されたプログラムが DISPLAY をシステム・オペレーターのメッセージ・キュー (ワークステーションとは別のもの) へ送り出す場合には、プログラムの処理は中断されません。

出力データの位置は、次のようにプログラム開始のタイプによって異なります。

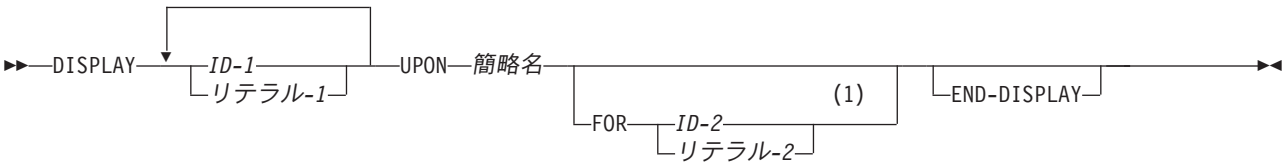
開始の方法	SYSTEM-CONSOLE に関連した簡略名	REQUESTOR に関連した簡略名	UPON 句を省略した場合
BATCH	システム・オペレーターのメッセージ・キュー	ジョブ・ログ	ジョブ・ログ
INTERACTIVE	システム・オペレーターのメッセージ・キュー	ワークステーション	ワークステーション

形式 2 - 内部データ域

IBM Extension

この形式を使用して、ジョブ用に作成されたシステム定義の内部データ域にデータを転送します。

DISPLAY ステートメント - 形式 2 - 内部データ域



注:

1 構文検査だけ行われます。

この形式が適用可能なのは、SPECIAL-NAMES 段落の中で簡略名が環境名 LOCAL-DATA と関連付けられている場合だけです。

ID-1 とリテラル-1 の変換および表示規則については 7-94 ページの『形式 1 - データの転送』に説明があります。 しかし、ID-1 の説明に TYPE 文節が入っており、参照されるタイプ名が基本項目でなければならないという制約は適用されません。

ID-2 とリテラル-2 は、浮動小数点データ項目であってはなりません。

ID-1 は、日時データ項目にすることができます。

ID-1 は DBCS または国別データ項目にすることができます。

DISPLAY ステートメントのリテラル・オペランド、または DISPLAY ステートメントの ID・オペランドの内容は、DISPLAY を出すプログラムを含んでいるジョブのシステム定義内部データ域に書き込まれます。データは、CORRESPONDING 句がなく、しかも右側がスペースが埋めこまれていないグループ転送の場合の MOVE ステートメントの規則に従って内部データ域に書き込まれます。

FOR 句 (指定されている場合) は、コンパイル時に構文上の検査を受けるが、プログラムの実行中にはコメントとして扱われます。リテラル-2 または ID-2 の値は、内部データ域にデータ書き込み中の装置のプログラム装置名を示しています。各ジョブに設けられる内部データ域は 1 つだけあり、1 つのジョブで使用される装置は、すべて同一の内部データ域をアクセスします。リテラル-2 (これが指定されている場合) は、非数字でなければならず、10 文字以下の長さでなければなりません。 ID-2 (これが指定されている場合) は、長さが 10 文字以下の英数字データ項目を参照するものでなければなりません。

ローカル・データ域について詳しくは、資料「*CL プログラミング*」および「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

End of IBM Extension

形式 3 - 拡張 DISPLAY ステートメント

IBM Extension

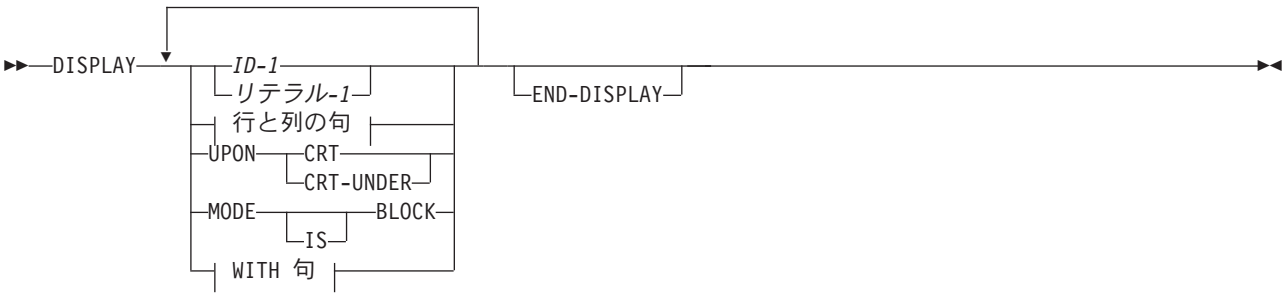
DISPLAY ステートメントは、以下のいずれかがある場合には、**拡張** DISPLAY ステートメントと見なされます。

- AT 句
- UPON CRT 句または UPON CRT-UNDER 句
- MODE IS BLOCK 句
- WITH 句
- SPECIAL-NAMES 段落に UPON 句が指定されず、CONSOLE IS CRT が指定されている

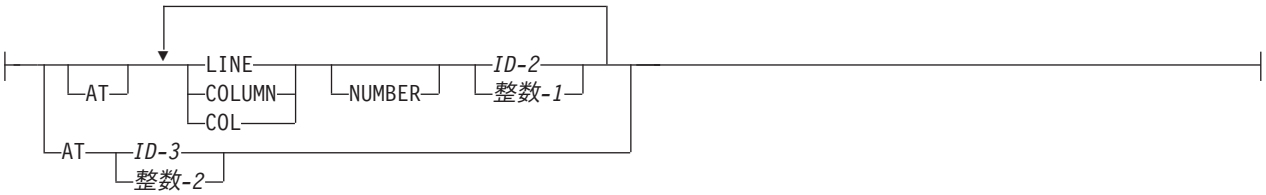
DISPLAY ステートメントに以下のいずれかがある場合には、**標準** DISPLAY ステートメントと見なされます。

- UPON 句 (UPON CRT または UPON CRT-UNDER 以外)
- SPECIAL-NAMES 段落に UPON 句と CONSOLE IS CRT が指定されていない

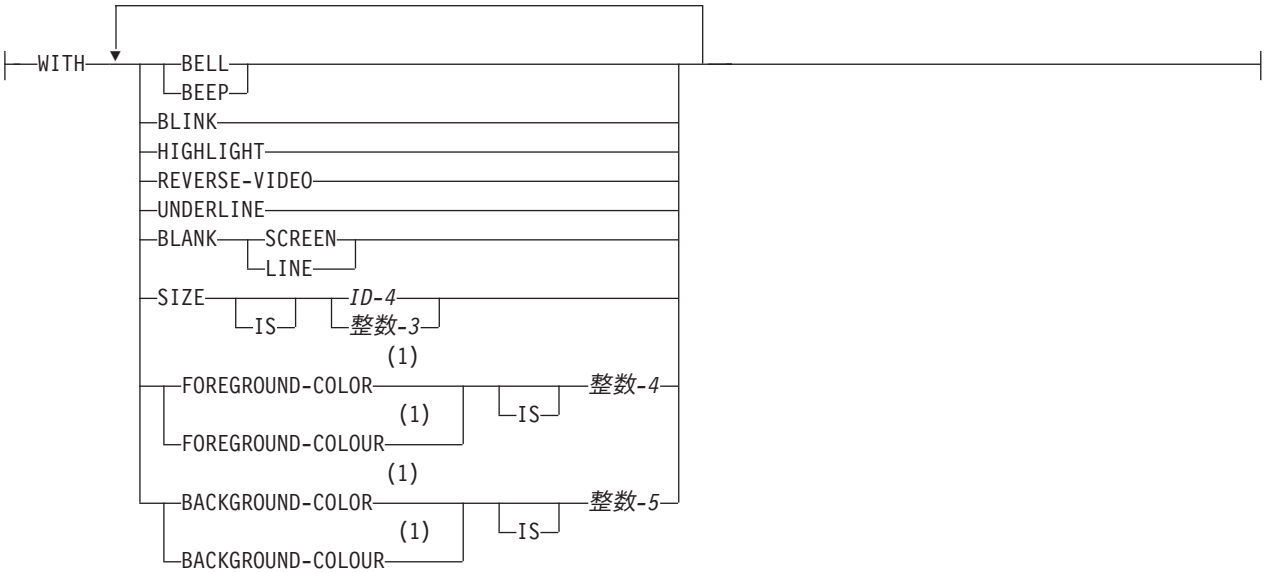
DISPLAY ステートメント - 形式 3 - ワークステーション I/O



行と列の句:



WITH 句:



注:

1 構文検査だけ行われます。

このステートメントの一部を繰り返すことによって、複数のデータ項目を表示させることもできます。最初の ID に AT、LINE、または COLUMN 句が付いていない場合、その ID は第 1 行、第 2 列から始まります。それ以降の各データ項目は、直前のデータ項目の後にある現在使用可能な画面位置から始まります。

ID-1 またはリテラル-1 を指定しないと、MODE IS BLOCK 句または WITH 句のどちらも使用することはできません。

ID-1 は、日時項目にすることはできません。

ID-1 が画面に合わないときは、英数字データが切り捨てられたり、数字データが画面上に表示されなくなります。

ID-1 がグループ項目で、MODE IS BLOCK 句がなければ、FILLER 以外の名前を持つ基本従属項目が表示されます。基本項目は、その記述が DATA DIVISION に現れる順番で画面に同時に表示されて位置付けされ、グループ中の FILLER 項目の長さによって分離されます。このため、行上の最初の位置は、直前の行の最後の位置にすぐに続くものと見なされます。項目が FILLER で区切られるときは、属性バイトは FILLER の長さに含まれます。したがって、FILLER の長さが 1 バイトまたは 2 バイトの場合は、区切られる 2 つの項目の後書き属性と先行属性の両方が含まれることになります。 1 バイト FILLER の場合に

は、後ろに続く属性および先の属性が同じバイトを占有します。データ項目は 1 属性バイトで区切られるのが普通であるため、1 バイトの FILLER は必要ありません。

ID もリテラルもどちらも存在しない場合は、DISPLAY 操作によって、実際にはデータを何も表示せずに画面位置だけが変更されます。

ID またはリテラルの後の句は、どのような順序で記入してもかまいません。指定された句は、すべて直前の ID またはリテラル (指定されている場合) に提供されます。前に ID またはリテラルが指定されていなかった場合は、WITH および MODE 句を指定することはできません。

AT、LINE または COLUMN 句が指定されていない限り、DISPLAY ステートメントの中の複数の ID またはリテラルは、間に属性バイトを 1 バイト挟んで続けられます。このステートメント内に AT、LINE、または COLUMN 句が記入されていない場合には、最初の ID またはリテラルは、第 1 行、第 2 列から始まり、それ以外のすべての ID またはリテラルがその直後に続きます。

AT 句: AT 句は、DISPLAY 操作を開始する画面上の絶対アドレスを示します。この句は、先行属性の開始位置を示すものではありません。

LINE 句は、画面上で画面項目が始まる行を指定します。

COLUMN 句は、画面上で画面項目が始まる列を指定します。

COL は、COLUMN の省略形です。

LINE 句および COLUMN 句は、どのような順序で記入してもかまいません。

ID-2、整数-1

ID-2 および整数-1 は、ゼロまたはそれ以上で 9 桁より小さい、無符号整数でなければなりません。LINE 句または COLUMN 句の値が負の場合には、絶対値がとられます。

ID-2 は、浮動小数点データ項目にすることができません。

行と列の組み合わせ: 行および列番号の特定の組み合わせには、次のような特別な意味があります。

- 列が範囲内に入るまでは、その範囲外の列の値は行の長さ分だけ減じられ、行値は増分されます。したがって、列番号によっては、行番号が数回増分されてしまうこともあります。
- 範囲外の行の値は、画面を 1 行分上方に画面移動させます。これは、最下部の行が指定された場合と同じです。指定された行に関係なく、画面が 2 行以上上方へ移動されることはありません。
- 列番号および行番号の両方ともが範囲外にある場合は、まず範囲外の列のほうが最初に処理されてから、次に範囲外の行が (上記の規則に従って) 処理されます。
- 入力された行および列番号が両方ともゼロの場合は、DISPLAY 操作は、直前の DISPLAY 操作が終了したところの次の位置から開始されます。各行の 1 列目は、直前の行の最終列の後に続いているものと見なされます。
- 行番号がゼロで、しかもゼロ以外の列番号が指定されている場合は、DISPLAY 操作は、直前の DISPLAY 操作が終了した行の次の行の、指定された列から始まります。各行の 1 列目は、直前の行の最終列の後に続いているものと見なされます。
- 行番号がゼロで、しかもゼロ以外の列番号が指定されている場合は、DISPLAY 操作は、直前の DISPLAY 操作が終了した行の次の行の、指定された列から始まります。

ID-3、整数-2

ID-3 は、PIC 9(4) または PIC 9(6) フィールドでなければなりません。整数-2 は、4 バイトまたは 6 バイトの数字フィールドでなければなりません。

ID-3 または整数-2 の長さが 4 バイトの場合は、最初の 2 桁が行を指定し、次の 2 桁が列を指定します。 ID-3 または整数-2 の長さが 6 バイトの場合は、最初の 3 桁が行を指定し、次の 3 桁が列を指定します。

ID-3 は、浮動小数点データ項目にすることができません。

UPON CRT/CRT-UNDER 句: DISPLAY ステートメントが拡張されていることを示します。

CRT-UNDER は、UPON CRT-UNDER 句の前に表示されたデータ項目に下線も付けます。

MODE IS BLOCK 句: ID が基本項目として扱われます。その ID がたとえグループ項目であっても、1 つの項目として表示されます。

WITH 句: WITH 句を使うことによって、ユーザーは DISPLAY 操作の特定のオプションを指定することができるようになります。そのようなオプションについては、以下に示す各句で説明します。

BELL (BEEP) 句: この句を含んでいる項目が表示されるたびに、警報音が鳴ります。

BELL と BEEP はいずれを使ってもかまいません。

BLANK 句: BLANK は、この文節を含んでいる画面項目が表示されるたびに機能します。

BLANK LINE は、現在のカーソル位置から現在行の終わりまでを消去します。 BLANK SCREEN は、画面全体を消去して、カーソルを第 1 行、第 2 列に移動します。

これらの消去は、項目が表示される前に行われます。

BLINK 句: 画面項目が画面に表示されると明滅します。

HIGHLIGHT 句: 画面項目が高輝度モードで画面に表示されます。

REVERSE-VIDEO 句: 画面項目が反転表示で示されます。

SIZE 句: 画面上のデータ項目の現行サイズを指定します。この句は基本データ項目にしか使用することができません。

ID-4、整数-3

整数-4 は符号なし整数でなければならず、OCCURS 文節に従属していてはなりません。整数-3 には符号が付いていてはなりません。

ID-4 または ID-3 に符号がある場合、コンパイラーは絶対値を使用し、警告メッセージを出します。

ID-4 は、浮動小数点データ項目であってはなりません。

指定されたサイズがゼロであれば、SIZE 句には影響がありません。この場合、データ項目を表示するためにフィールドの長さが使用されます。

関連する PICTURE 文節によって暗黙指定されたサイズよりも小さなサイズが指定されると、ワークステーションの画面にはデータ項目の左端の部分だけが現れます。

数字または数字編集のデータ項目に対して指定したサイズが、PICTURE 文節によって暗黙指定されたサイズよりも小さくなる場合があります。その場合、値が表示される際に、右端が切り捨てられます。データ項目は MOVE 操作の規則に従って更新されます。

フィールドの長さが画面サイズを超える値を持つ SIZE リテラルを指定する場合には、英数字データが切り捨てられたり、数字データが無視されるか、表示されなくなります。

JUSTIFIED が指定された項目に関しては、その項目の長さよりも小さいサイズを指定すると、右端の部分だけが現れます。

ユーザーが指定するサイズが、PICTURE 文節で暗黙指定されたサイズよりも大きい場合には、その項目の表示用部分にはスペースが埋め込まれます。埋め込まれるのは右側です。

表意定数 ALL は、ユーザーが指定する長さに達するまで必要に応じて何度でも表示されます。表示が新しい行へ折り返すと、新しい行は定数の先頭から始まります。

SIZE 句の例: 以下に示すのは、指定されたサイズが表意定数よりも大きくて、新しい行に折り返した場合の表意定数の表示例です。

```
     DISPLAY ALL 'ABCD' AT 0270 WITH SIZE 15.
```

この定数は第 2 行、第 70 桁から始まって、画面上に表示されます。

```
                0000000001               677777777778
                1234567890.............901234567890
行 1
行 2                                   ABCDABCDABC
行 3          ABCD
```

以下の例における相違に注目してください。

```
ステートメント 1 DISPLAY 'WORKSTATION' AT 0275 WITH SIZE 10
ステートメント 2 DISPLAY ALL 'WORKSTATION' AT 0275 WITH SIZE 10

                0000000001               677777777778
                1234567890.............901234567890
ステートメント 1                                   WORKST
                ATIO

ステートメント 2                                   WORKST
                WORK
```

UNDERLINE 句: 画面に表示される画面項目に下線が引かれます。

形式 3 の考慮事項: MODE IS BLOCK が指定されているかどうかに関係なく、データ項目にはテーブルを入れることができます。固定長および可変長のテーブルはグループ項目 (MODE IS BLOCK が指定されていない) として扱われ、それらの項目はテーブルの最初のオカレンスから最後のオカレンスまで繰り返されます。

拡張 DISPLAY ステートメントに関する考慮事項のあるものは、拡張 ACCEPT ステートメントにも当てはまります。 (詳しくは 7-60 ページの『拡張 ACCEPT および拡張 DISPLAY の考慮事項』を参照してください。)

ILE COBOL の拡張 DISPLAY ステートメントは、IBM COBOL/2 DISPLAY ステートメント (形式 2) に類似しています。異なる点については 9-49 ページの『付録 I. ACCEPT/DISPLAY および COBOL/2 に関する考慮事項』で説明します。

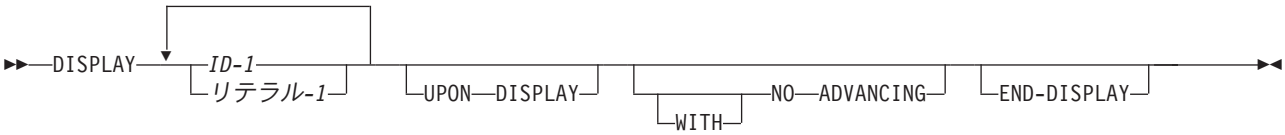
End of IBM Extension

形式 4 - セッション入出力

IBM Extension

この形式は、データを ILE 共通セッション管理プログラムに転送するのに使用します。

DISPLAY ステートメント - 形式 4 - セッション I/O



この形式は、SPECIAL-NAMES 段落に UPON DISPLAY 句または CONSOLE IS DISPLAY 文節が指定されている場合にだけ適用されます。

DISPLAY ステートメントのリテラル・オペランド、または DISPLAY ステートメントの ID オペランドの内容は、ILE 共通セッション管理プログラムに書き込まれます。この際、データは形式 1 - データの転送 (7-94 ページの『形式 1 - データの転送』の下の ID-1 とリテラル-1 の説明を参照) に略述されている規則に従ってセッション管理プログラムに書き込まれます。 ID-1 またはリテラル-1 の内容が 2 行以上にまたがっている場合、データの書き込みは ILE 共通セッション管理プログラムの次の行の最初の位置から継続します。

WITH NO ADVANCING 句が指定されていないと、改行 (new line) 文字がセッション管理プログラムに書き込まれます。しかし、この句が指定されている場合には、表示されている最後のオペランドの最終文字のすぐ後にセッション管理プログラムが位置づけられます。

ID-1 は、日時データ項目にすることができます。

ID-1 は DBCS または国別データ項目にすることができます。 ID-1 が国別項目の場合、出力データは、ジョブの現行 CCSID によって指定されるコード・セットに変換されます。

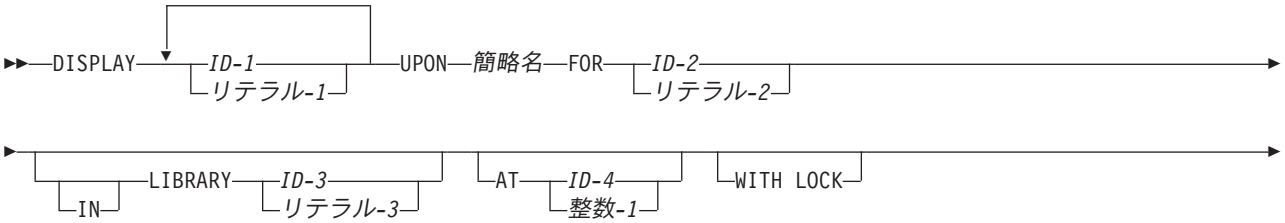
End of IBM Extension

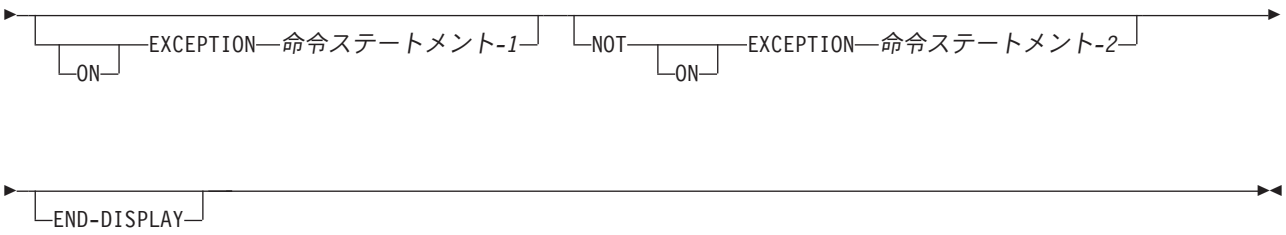
形式 5 - データ域

IBM Extension

このステートメントは、FOR 句で指定されているデータ域にデータを転送するために使用されます。

DISPLAY ステートメント - 形式 5 - データ域





この形式が適用できるのは、SPECIAL-NAMES 段落の中で簡略名が環境名 DATA-AREA と関連付けられている場合だけです。

DISPLAY ステートメントのリテラル・オペランド、または DISPLAY ステートメントの ID オペランドの内容は、CORRESPONDING 句がなく、しかも右側にスペースが埋めこまれていないグループ転送の場合の MOVE ステートメントの規則に従ってデータ域に書き込まれます。

データ域の使用方法の詳細および例については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」のセクション『データ域を使用したデータの受け渡し』の『自分で作成したデータ域の使用』の情報を参照してください。

ID-1/リテラル-1
ID-1 とリテラル-1 の変換および表示規則については 7-94 ページの『形式 1 - データの転送』に説明があります。

ID-1 は、日時データ項目にすることができます。

UPON: SPECIAL-NAMES 段落中の**簡略名**は、環境名 DATA-AREA と関連付けられていなくてはなりません。

UPON 句を省略した場合には、DISPLAY ステートメントは REQUESTOR へ出力を送り出します。

FOR 句: 情報を書き込むための、オペレーティング・システムのデータ域を識別します。指定されたデータ域の位置が実行時に確認できないか、またはアクセスできない場合、ON EXCEPTION 条件が存在します。

ID-2
英数字データ項目でなければなりません。 ID-2 の内容は、有効なオペレーティング・システムのデータ域名を表していなければなりません。オペレーティング・システムのデータ域名は、最大 10 文字の長さです。したがって ID-2 の最初の 10 文字が、データ域名を形成するために使用されます。

リテラル-2
非数字で、長さが最大 10 文字でなくてはなりません。

IN LIBRARY 句: データ域を検出するオペレーティング・システムのライブラリーの名前を指定するために使用されます。特殊値 *LIBL (ジョブのライブラリー・リストを使用する検索) または *CURLIB (現行ライブラリーを検索) を指定できます。 LIBRARY 句を省略すると、ジョブのライブラリー・リストを使用して、データ域を検索します。

ID-3
英数字データ項目でなければなりません。オペレーティング・システムのライブラリー名の長さは最大 10 文字であるため、ID-3 の最初の 10 文字だけを使用してライブラリー名が形成されます。

リテラル-3
非数字で、長さが最大 10 文字でなくてはなりません。

ID-2、ID-3、リテラル-1、およびリテラル-2 は、*MONOPRC コンパイラー・オプションによって影響を受けません。これらには、オペレーティング・システムの引用符付き名を入れることができます (詳しくは、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL* および *API*』にある『名前の指定の規則』を参照してください)。

AT 句: AT 句は、テキストを書き込むデータ域中の開始位置を示します。

AT 句が指定されていない場合、開始位置 1 が想定されます。

ID-4、整数-1
ID-4 および整数-1 は正の整数で、その値は 1 から最大データ域サイズ (2000) の範囲になくてはなりません。

WITH LOCK 句: FOR 句中の指定されたデータ域にデータが転送される前に、ロックを獲得しなければなりません。ロックが得られない場合には、データは転送されず、ON EXCEPTION 条件が存在します。

データの転送後もロックをデータ域に保存するためには、この句を指定してください。このステートメント以前にデータ域にロックが存在し、ステートメントに WITH LOCK 句が含まれていない場合、そのロックは解放されます。

(NOT) ON EXCEPTION: データ域にアクセスしている間にエラーが発生すると、ON EXCEPTION 句で指定された任意の命令ステートメントが処理されます。 ON EXCEPTION 句がない場合は、実行時メッセージが発行されます。データ域が正常にアクセスされると、NOT ON EXCEPTION 句で指定された任意の命令ステートメントが処理されます。

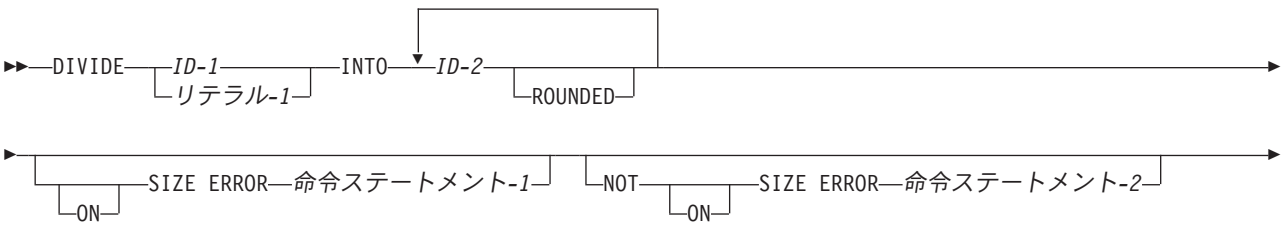
END-DISPLAY 句: END-DISPLAY 明示範囲終了符号は、DISPLAY ステートメントの有効範囲を区切るのに役立ちます。 END-DISPLAY を使うと、条件 DISPLAY ステートメントを別の条件ステートメント内でネストさせることができます。また、END-DISPLAY は命令 DISPLAY ステートメントでも使用できます。詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

End of IBM Extension

DIVIDE ステートメント

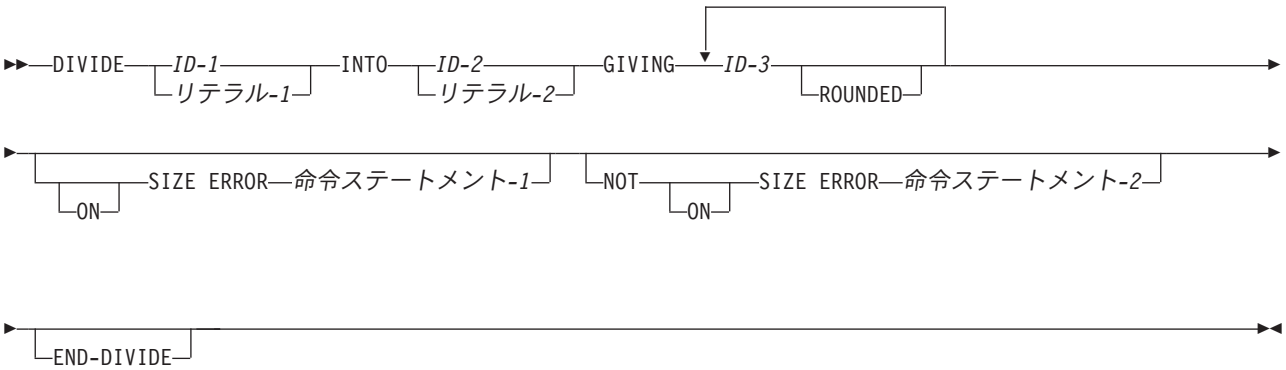
DIVIDE ステートメントは 1 つの数字データ項目を、1 つまたは複数の他の数字データ項目で除算して、その結果を商と余りとして保管します。

DIVIDE ステートメント - 形式 1 - INTO



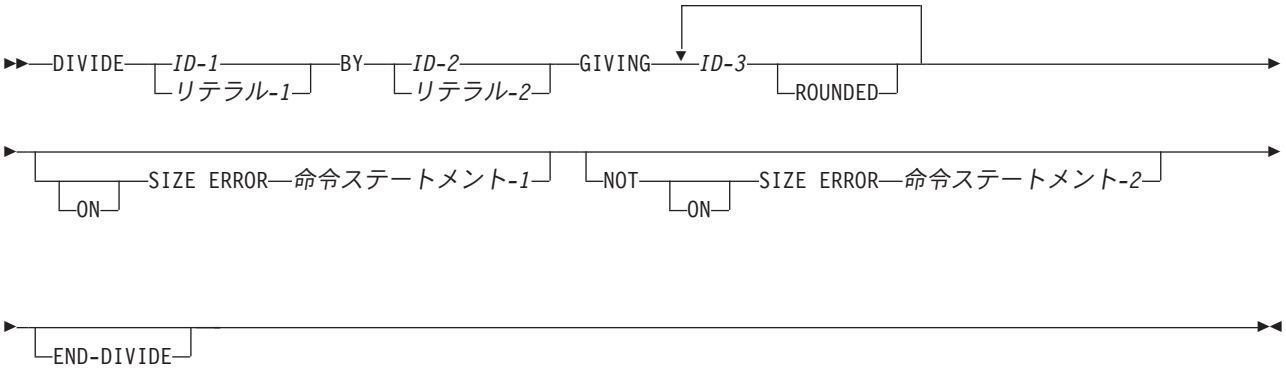
形式 1 では、ID-1 またはリテラル-1 の値は ID-2 の値で除算が行われます。その後、商が ID-2 に入れられます。このプロセスは ID-2 のそれ以後のオカレンスごとに繰り返されます。

DIVIDE ステートメント - 形式 2 - INTO GIVING



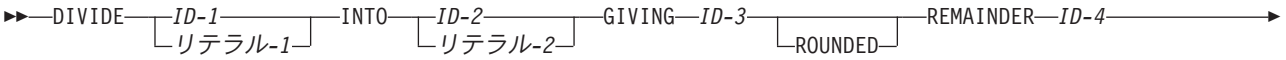
形式 2 では、ID-1 またはリテラル-1 の値は、ID-2 またはリテラル-2 の値で除算されます。商の値は、ID-3 によって参照される各データ項目に保管されます。

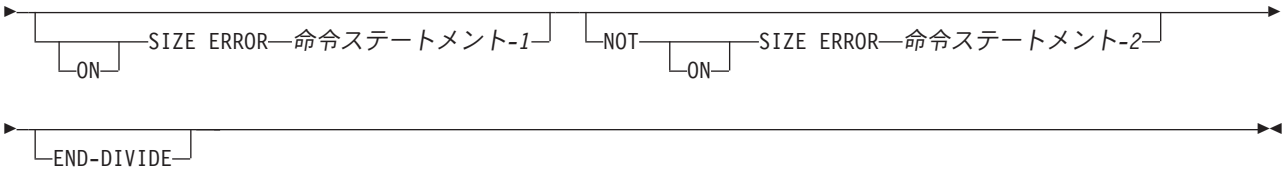
DIVIDE ステートメント - 形式 3 - BY GIVING



形式 3 では、ID-1 またはリテラル-1 の値は、ID-2 またはリテラル-2 の値で除算されます。この商は、ID-3 で参照される各データ項目に保管されます。

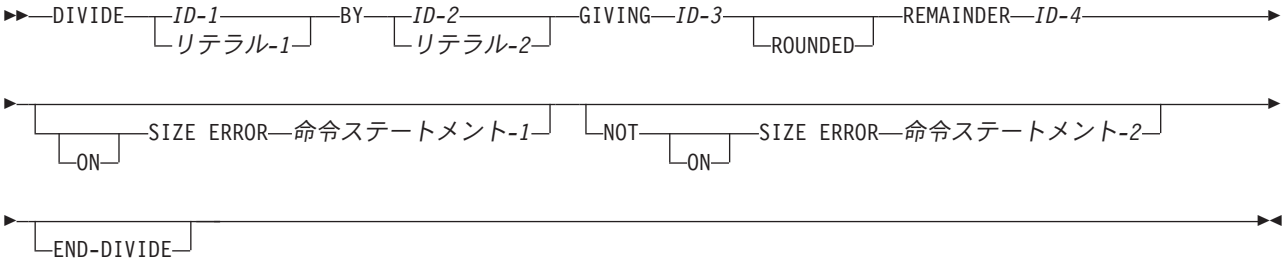
DIVIDE ステートメント - 形式 4 - INTO GIVING REMAINDER





形式 4 では、ID-1 またはリテラル-1 の値は、ID-2 またはリテラル-2 の値で除算されます。この商は ID-3 に保管され、余りの値は ID-4 に保管されます。

DIVIDE ステートメント - 形式 5 - BY GIVING REMAINDER



形式 5 では、ID-1 またはリテラル-1 の値は、ID-2 またはリテラル-2 の値で除算されます。この商は ID-3 に保管され、余りの値は ID-4 に保管されます。

すべての形式において、以下が適用されます。

ID-1、ID-2
基本数字項目でなければなりません。

ID-3、ID-4
基本数字項目または基本数字編集項目でなければなりません。

リテラル-1、リテラル-2
数字リテラルでなければなりません。

オペランドの合成は、REMAINDER データ項目を除いた、すべての受け入れデータ項目を重ね合わせることによって決定されます。オペランドの合成に関する詳細は 7-34 ページの『オペランドのサイズ』を参照してください。

IBM Extension

形式 1 ～ 3 では、浮動小数点データ項目および浮動小数点リテラルは、数字データ項目または数字リテラルが指定できるところであればどこでも使用できます。

形式 4 および形式 5 では、浮動小数点データ項目または浮動小数点リテラルは使用できません。

End of IBM Extension

ROUNDED 句

形式 1、形式 2、および形式 3 については 7-33 ページの『ROUNDED 句』を参照してください。

形式 4 および形式 5 では、剰余を計算するために使用される商は、中間フィールドに置かれます。中間フィールドの値に対して、丸めではなく切り捨てが行われます。

REMAINDER 句

商と除数の積を被除数から減算した結果が、ID-4 に保管されます。 ID-3 (商) が数字編集項目である場合には、剰余を計算するために使用される商は、編集されていない商を含む中間フィールドです。

IBM Extension

REMAINDER 句は、受け取り側または任意のオペランドが浮動小数点項目である場合、無効になります。

End of IBM Extension

REMAINDER 句内の ID-4 に対する添え字は、除算の結果が GIVING 句の ID-3 に保管された後に評価されます。

SIZE ERROR 句

ON SIZE ERROR がコーディングされている場合、ゼロ除算により、ON SIZE ERROR がトリガーされます。それ以外の場合、ゼロ除算の結果は未定義になります。

形式 1、形式 2、および形式 3 については 7-33 ページの『SIZE ERROR 句』を参照してください。

形式 4 および形式 5 で、商もサイズ・エラーが起こった場合には、剰余の計算は意味をもちません。したがって、商フィールド (ID-3) および剰余フィールド (ID-4) の値は変わりません。

剰余にサイズ・エラーが起こった場合には、剰余フィールド (ID-4) の内容は変わりません。

前述の 2 つの場合には、ユーザーはどのような条件が起こったのかを判断するために結果を分析しなければなりません。

NOT ON SIZE ERROR 句については、 7-34 ページの『NOT ON SIZE ERROR』のページを参照してください。

END-DIVIDE 句

この明示範囲終了符号は、DIVIDE ステートメントの範囲を区切ります。 END-DIVIDE は DIVIDE 条件ステートメントを命令ステートメントに変換することにより、それが他の条件ステートメントにネストできるようにします。

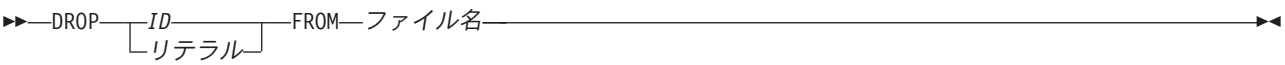
詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

DROP ステートメント

IBM Extension

DROP ステートメントは、TRANSACTION ファイルによって獲得したプログラム装置を解放します。

DROP ステートメント - 形式



リテラル、ID

リテラルまたは ID の内容は、ドロップされる装置のプログラム装置名を示しています。リテラルを指

定する場合、非数字で、かつ、その長さが 10 文字以内でなければなりません。 ID を指定する場合は、長さが 10 文字以内の英数字データ項目でなければなりません。

ファイル名
ファイル名には、TRANSACTION 編成のファイルを指定しなければなりません。これを DROP ステートメントの中で使用するためには、そのファイルがオープンされていなければなりません。 DROP ステートメントが出されなかった場合、TRANSACTION ファイルに接続されたプログラム装置は、そのファイルが最終的にクローズされるときに、暗黙のうちに解放されます。

DROP ステートメントで指定されたプログラム装置は、OPEN 時に明示的な ACQUIRE、または OPEN 時の暗黙的な ACQUIRE によって、TRANSACTION ファイルに前もって獲得されていなければなりません。

DROP ステートメントが正常に実行されると、そのプログラム装置は、TRANSACTION ファイルを介する入力操作または出力操作用に、使用できなくなります。そのプログラム装置は、必要ならば再び獲得できます。解放されたプログラム装置に関連したレコード域の内容は、その装置が再び獲得された場合でも、もはや使用できません。

DROP ステートメントが正常に終了しなかった場合、適用可能な USE AFTER EXCEPTION/ERROR プロシージャーが実行されます。

DROP ステートメントは、入出力エラーからの回復の補助手段としても使用できます。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の『TRANSACTION ファイルのリカバリー』の手順を参照してください。

End of IBM Extension

ENTER ステートメント

ENTER ステートメントにより、1 つのソース・プログラム内で複数のソース言語を使用できます。これは構文検査だけを受けます。

ENTER ステートメント - 形式

注:

1 構文検査だけ行われます。

言語名
意味が定義されていないシステム名です。ユーザー定義語の形成に関する規則に従わなければなりません。少なくとも 1 文字は英字でなければなりません。

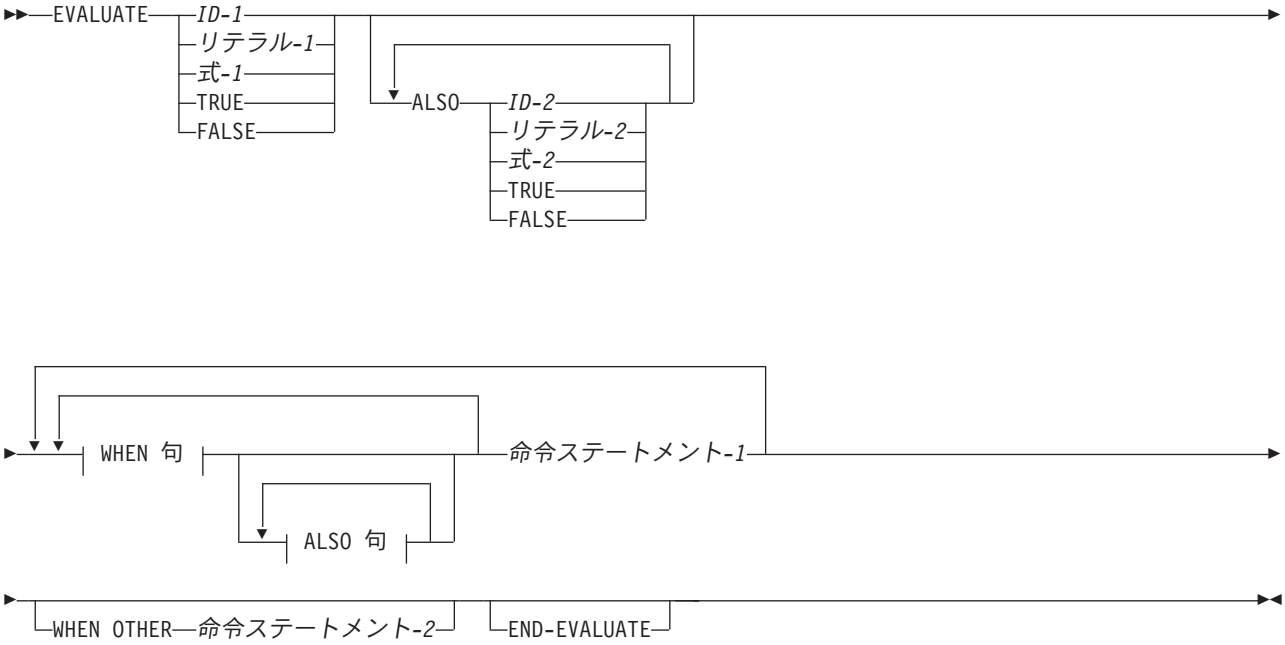
ルーチン名
ユーザー定義語の形成に関する規則に従わなければなりません。少なくとも 1 文字は英字でなければなりません。

EVALUATE ステートメント

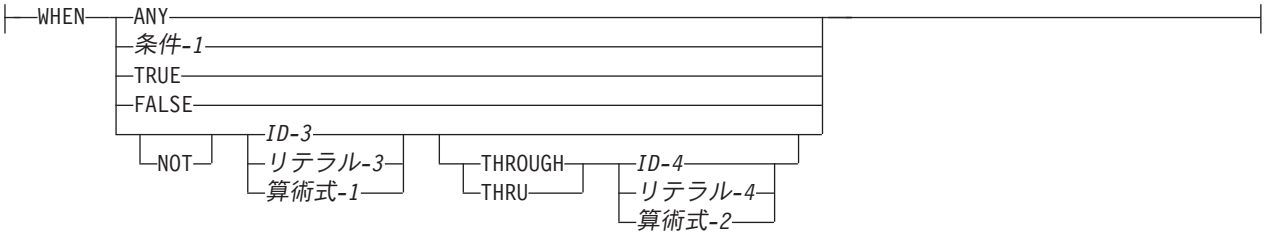
EVALUATE ステートメントは、ネストされた一連の IF ステートメントに対する省略表現を提供します。これは、複数の条件を評価できます。すなわち、IF ステートメントでは複合条件での構成になります。オブジェクト・プログラムのその後の処置は、これらの評価の結果によって決まります。

EVALUATE ステートメント

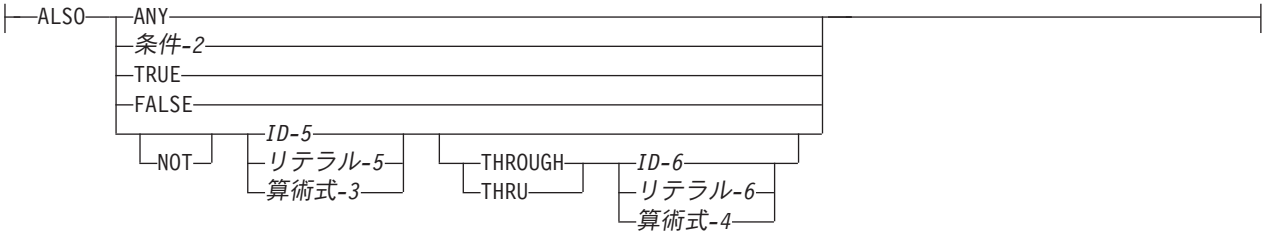
EVALUATE ステートメント - 形式



WHEN 句:



ALSO 句:



次の例は、EVALUATE ステートメントのコーディングと、それに対応する IF ステートメントのコーディングを示したものです。

コーディング例

EVALUATE ステートメントの簡単な例

```
EVALUATE  MENU-INPUT
    WHEN "0"
        PERFORM INIT-PROC
    WHEN "1" THRU "9"
        PERFORM PROCESS-PROC
    WHEN "R"
        PERFORM READ-PARMS
    WHEN "X"
        PERFORM CLEANUP-PROC
    WHEN OTHER
        PERFORM ERROR-PROC
END-EVALUATE.
```

対応する IF ステートメント

```
IF (MENU-INPUT = "0") THEN
   PERFORM INIT-PROC
ELSE
   IF (MENU-INPUT ≥ "1") AND (MENU-INPUT ≤ "9") THEN
      PERFORM PROCESS-PROC
   ELSE
      IF (MENU-INPUT = "R") THEN
         PERFORM READ-PARMS
      ELSE
         IF (MENU-INPUT = "X") THEN
            PERFORM CLEANUP-PROC
         ELSE
            PERFORM ERROR-PROC
         END-IF
     END-IF
   END-IF
END-IF.
```

次の例は、EVALUATE ステートメントおよびそれに対応する IF ステートメントの、より複雑な例です。

EVALUATE ステートメントの複雑な例

```
EVALUATE  A = B  ALSO  C > D  ALSO  TRUE
    WHEN  TRUE   ALSO  TRUE   ALSO  E = F + 15
        imp-stat-1
    WHEN  TRUE   ALSO  TRUE   ALSO  E > 12
        imp-stat-2
    WHEN  TRUE   ALSO  FALSE  ALSO  ANY
        imp-stat-3
    WHEN  FALSE  ALSO  TRUE   ALSO  ANY
        imp-stat-4
    WHEN  FALSE  ALSO  FALSE  ALSO  ANY
        imp-stat-5
END-EVALUATE.
```

対応する IF ステートメント

```
IF A = B THEN
    IF C > D THEN
        IF E = F + 15 THEN
            imp-stat-1
        ELSE
            IF E > 12 THEN
                imp-stat-2
            END-IF
        END-IF
    ELSE
        imp-stat-3
    END-IF
ELSE
    IF C > D THEN
        imp-stat-4
```

```
    ELSE
        imp-stat-5
    END-IF
END-IF.
```

選択サブジェクトと選択オブジェクトの解釈

WHEN 句の前のオペランド

個別に指定されている場合には、これらは選択**サブジェクト**と呼ばれます。
集合で指定されている場合には、選択サブジェクトの**セット**と呼ばれます。

WHEN 句の中のオペランド

個別に指定されている場合には、選択**オブジェクト**と呼ばれます。
集合で指定されている場合には、選択オブジェクトの**セット**と呼ばれます。

ALSO
選択サブジェクトのセット内のサブジェクトどうしを区切ります。また、選択オブジェクトのセット内のオブジェクトどうしを区切ります。

THROUGH および THRU
これらは、同じ意味です。 THRU 句で結ばれた 2 つのオペランドは、同じクラスのものでなければなりません。このように結ばれた 2 つのオペランドは、1 つの選択オブジェクトを形成します。

選択オブジェクトの各セット内の選択オブジェクトの数は、選択サブジェクトの数と等しくなければなりません。選択オブジェクトのセット内の各選択オブジェクトは、次の規則に従って、選択サブジェクトのセット内の同じ順位を持つ選択サブジェクトに対応していなければなりません。

- 選択オブジェクトの中に現れる ID、リテラル、または算術式は、選択サブジェクトのセット内の対応するオペランドと比較した場合に、有効なオペランドとなっていなければなりません。
- 選択オブジェクトとして現れる条件-1、条件-2、または TRUE や FALSE の語は、選択サブジェクトのセットの中の条件式または TRUE や FALSE の語と対応していなければなりません。
- 条件-1 および条件-2 は、任意の形式の条件式とすることができます。
- ANY という語は、任意のタイプの選択サブジェクトに対応することがあります。
- 条件式は、単純条件または複合条件のどちらにもすることができます。

IBM Extension

- 数字リテラルが許可されるところでは、浮動小数点リテラルが許可されます。
- ID が参照できる項目は、その使用法が POINTER または PROCEDURE-POINTER として暗黙に、または明示的に定義されている項目です。
- ID は、DBCS、国別、または浮動小数点の各データ項目を参照できます。
- ID は日時データ項目を参照できます。
- 非数字リテラルが許可されると、DBCS リテラルおよび国別リテラルもまた許可されます。

End of IBM Extension

END-EVALUATE 句

この明示範囲終了符号は、EVALUATE ステートメントの有効範囲を区切るのに役立ちます。END-EVALUATE を使用することによって、EVALUATE 条件ステートメントを別の条件ステートメントにネストすることができます。詳細は、7-29 ページの『範囲区切りステートメント』を参照してください。

値の決定

EVALUATE ステートメントの実行は、あたかも、各選択サブジェクトおよび選択オブジェクトが演算され、 数字または非数字の値、数字または非数字の値の範囲、あるいは真理値が割り当てられているかのように行われます。これらの値は、次のようにして決定されます。

- ID-1、ID-2、... で指定された選択サブジェクト、および NOT 句や THRU 句のない ID-3 または ID-5 (あるいはその両方) で指定された選択オブジェクトはすべて、それらが参照するデータ項目の値とクラスが割り当てられます。
- リテラル-1、リテラル-2、... で指定された選択サブジェクト、および NOT 句や THRU 句のないリテラル-3 またはリテラル-5 (あるいはその両方) で指定された選択オブジェクトはすべて、指定されたリテラルの値とクラスが割り当てられます。リテラル-3 またはリテラル-5 (あるいはその両方) が表意定数 ZERO である場合には、対応する選択サブジェクトのクラスがこれに割り当てられます。
- 式-1、式-2、... が**算術**式として指定されている選択サブジェクト、そして算術式-1 または算術式-3 (あるいはその両方) が指定されている、NOT 句や THRU 句のない選択オブジェクトはすべて、算術式の評価の規則に従って数字を割り当てられます。 (7-8 ページの『算術式』を参照。)

  注: ある算術式を別の算術式と比較する方法はシステム特有のものです。比較の真または偽の状況は、該当するシステムで作成される中間結果に左右されることがあります。
- 式-1、式-2、... が**条件**式として指定されている選択サブジェクト、そして条件-1 または条件-2 (あるいはその両方) が指定されている選択オブジェクトはすべて、条件式の評価の規則に従って真理値を割り当てられます。 (7-10 ページの『条件式』を参照。)
- TRUE または FALSE の語が指定されている選択サブジェクトまたは選択オブジェクトには、真理値が割り当てられます。 TRUE の語が指定されている項目には、真理値「真」が割り当てられ、FALSE の語が指定されている項目には真理値「偽」が割り当てられます。
- ANY の語が指定されている選択オブジェクトは、これ以上評価されません。
- 選択オブジェクトに THRU 句が指定されていて、NOT 句が指定されていない場合には、値の範囲は比較の規則により、選択サブジェクトと比較したときに、最初のオペランド以上で、2 番目のオペランド以下のすべての値となります。最初のオペランドが 2 番目のオペランドより大きい場合には、範囲内には値がないことになります。

  注: 比較がシステムの固有照合順序によるものであれば、非数字オペランドの比較の結果はシステム間で一貫していないことがあります。
- 選択オブジェクトに NOT 句が指定されている場合には、その項目に割り当てられる値は、NOT 句が省略されている項目に割り当てられる値または値の範囲とは等しくないすべての値となります。

選択サブジェクトと選択オブジェクトの比較

EVALUATE ステートメントの実行は、あたかも、 選択サブジェクトのセットを満足させる WHEN 句があるかどうかを判別するために、選択サブジェクトおよび選択オブジェクトに割り当てられた値を比較するかのように進行します。この比較は、次のように進められます。

1. 最初の WHEN 句に対する選択オブジェクトのセット内の各選択オブジェクトが、選択サブジェクトのセット内の同じ順位を持つ選択サブジェクトと比較されます。比較を実行する場合には、次の条件の 1 つを満たしていなければなりません。
   a. 比較する項目に数字または非数字の値、あるいは数字または非数字の値の範囲が割り当てられている場合には、比較の規則により、選択オブジェクトに割り当てられた値、または値の範囲の中の 1 つの値が、選択サブジェクトに割り当てられた値と等しい場合に比較が実行されます。
   b. 比較する項目に真理値が割り当てられている場合には、その項目に同じ真理値が割り当てられた場合に比較が実行されます。
   c. 比較する選択オブジェクトに ANY が指定されている場合には、選択サブジェクトの値に関係なく、常に比較が実行されます。
2. 比較する選択オブジェクトのセット内のすべての選択オブジェクトに対して上記の比較が実行された場合には、その選択オブジェクトのセットが入っている WHEN 句が、選択サブジェクトのセットの条件を満たす句として選択されます。
3. 上記の比較が、比較する選択オブジェクトのセット内のすべての選択オブジェクトに対して実行されたわけではない場合には、その選択オブジェクトのセットは選択サブジェクトの条件を満たしません。
4. このプロシージャーは、その後の選択オブジェクトのセットで、それらがソース・プログラムに現れる順序で繰り返されます。これは、選択サブジェクトのセットの条件を満たす WHEN 句が選択されるか、あるいは選択されたすべてのセットがなくなるまで続けられます。

EVALUATE ステートメントの実行

比較操作が完了すると、EVALUATE ステートメントの実行が、次のように進められます。

- WHEN 句が選択された場合には、選択された WHEN 句の後の最初の命令ステートメント-1 から実行されます。 1 つの命令ステートメント-1 に対して、複数の WHEN ステートメントを使用できることに注意してください。
- WHEN 句が選択されず、WHEN OTHER 句が指定されている場合には、実行は、命令ステートメント-2 から続行されます。
- WHEN 句が選択されず、WHEN OTHER 句も指定されていない場合には、実行は、範囲分離文字の後の次の実行可能ステートメントから続行されます。
- EVALUATE ステートメントの実行範囲は、実行が、選択された WHEN 句または WHEN OTHER 句の有効範囲の終わりに達したとき、あるいは WHEN 句が選択されず、WHEN OTHER 句も指定されないときに終了します。

EXIT ステートメント

EXIT ステートメントは、一連の段落に対する共通の終了点を与えます。

EXIT ステートメント - 形式

EXIT ステートメントを使って、プログラム中の特定の点に名前を割り当てることができます。 EXIT ステートメントは、プログラムのコンパイルおよび実行にはその他の影響を与えません。 EXIT ステートメントは段落名の後に続くものでなければならず、また、このステートメント自身が 1 つの文になっていなければなりません。さらに、この文はその段落の中でただ 1 つの文でなければなりません。

EXIT ステートメントは一連の段落の終了点を文書化する場合に役に立ちます。 EXIT 段落が宣言プロシージャーの最後の段落、あるいは実行される一連のプロシージャーの最後の段落として書かれた場合には、制御権が渡される点が以下のいずれかであるかを明らかにします。

- 活動状態の PERFORM または USE ステートメントによって左右されるプロシージャーの範囲の終わりにある EXIT 段落に制御が達すると、制御は該当する PERFORM または USE ステートメントの規則に従って渡されます。
- 制御が、実行中の PERFORM ステートメントまたは USE ステートメントが制御する一連のプロシージャーの終了点でない EXIT 段落に到達した場合は、EXIT ステートメントの次の段落にある最初のステートメントへ制御が移ります。

EXIT ステートメントが書かれていない場合には、ユーザーがプログラムの論理を理解していない限り、一連のステートメントの終了点を判別することは困難です。

EXIT PROGRAM ステートメント

EXIT PROGRAM ステートメントは、呼び出し先プログラムの終わりを指定して、呼び出し側プログラムに制御を戻します。また、グローバル宣言の範囲内にあるステートメントが呼び出す異なるプログラムにあるのでない限り、これをグローバル宣言の範囲内で使用してはなりません。

EXIT PROGRAM ステートメント

注:

1 IBM 拡張

AND CONTINUE RUN UNIT
実行単位を停止することなく、呼び出し側プログラムを終了します。

あるプログラムの EXIT PROGRAM ステートメントに制御が達し、そのプログラムには呼び出し側プログラムの制御下での操作中に INITIAL 属性がない場合には、呼び出し側プログラムの CALL ステートメントに制御が戻ります。

呼び出し側プログラムのプログラム状態は、それが CALL ステートメントを実行した時に存在したプログラムと同一になります。ただし、その 2 つのプログラムで共有するデータ項目の内容とデータ・ファイルの内容は変更されている可能性があります。呼び出し先プログラムのプログラム状態は、この呼び出し先プログラムの実行するすべての PERFORM ステートメントの終了点に到達したと見なされる場合を除き、変更されません。

INITIAL 属性を持つ、呼び出し先プログラムの EXIT PROGRAM ステートメントを実行すると、参照されるプログラムの暗黙の CANCEL が実行されます。

主プログラム中で継続句なし の EXIT PROGRAM ステートメントに制御が達した場合、制御は出口点を通過して、次の実行可能なステートメントに渡されます。

EXIT PROGRAM ステートメントは、ある文の一連の命令ステートメントの中で最後のステートメントとして現れなければなりません。

EXIT PROGRAM ステートメント

呼び出し先プログラム内に次の実行可能ステートメントがない場合には、暗黙の EXIT PROGRAM ステートメントが想定され、実行されます。

RETURN-CODE 特殊レジスターを使用すれば、あるプログラムからその呼び出し側プログラムに戻りコードの情報を渡すことができます。詳しくは、 7-227 ページの『RETURN-CODE 特殊レジスター』を参照してください。

AND CONTINUE RUN UNIT 句

IBM Extension

主プログラム中で継続句*あり* の EXIT PROGRAM ステートメントに制御が達した場合、制御は呼び出し側プログラムの CALL ステートメントに移ります。名前付きの活動化グループにおいては、

- 活動化グループは活動状態のまま残ります。
- 主プログラムは最後に使用された状態のまま残りますが、呼び出し先プログラムによって実行されるすべての PERFORM ステートメントの範囲の終わりまで達したと考えられる場合は除きます。

ただし、*NEW 活動化グループにおいて主プログラムが呼び出し側へ制御を戻したとき、その活動化グループは終了します。その活動化グループは、活動化グループの有効範囲にある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有効範囲にあるすべての保留のコミット操作は、暗黙にコミットされます。その活動化グループに割り振られているすべてのリソースは、システムに戻されます。その活動化グループが終了した結果、活動化グループ中で活動状態にあったすべてのプログラムがそれらの初期状態に置かれます。

End of IBM Extension

GOBACK ステートメント

IBM Extension

GOBACK ステートメントは、COBOL 実行単位中のサブプログラムであるプログラムの一部としてコーディングされた場合は EXIT PROGRAM ステートメントと同じように働き、COBOL 実行単位中の主プログラムであるプログラム内にコーディングされた場合は STOP RUN ステートメントと同様に機能します。

GOBACK ステートメントは、呼び出し側プログラムの論理終了点を指定するものです。

GOBACK ステートメント - 形式

GOBACK ステートメントの後に続くステートメントは実行されないため、GOBACK ステートメントは文中における唯一のステートメントとして、または一連の命令ステートメントの最後のステートメントとして指定する必要があります。

CALL ステートメントが活動状態であるときに制御が GOBACK ステートメントに達すると、EXIT PROGRAM ステートメントの場合と同様、CALL ステートメントのすぐ後にある呼び出し側プログラムの地点に制御が戻ります。

RETURN-CODE 特殊レジスターを使用すれば、GOBACK ステートメントを実行する前に戻りコード情報を渡すことができます。 7-227 ページの『RETURN-CODE 特殊レジスター』を参照してください。

マルチスレッド環境では (例えば、THREAD(SERIALIZE) PROCESS オプションが指定されている場合)、
GOBACK ステートメントは、スレッドおよび実行単位を終了させずに、プログラムの呼び出し元に戻りま
す。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の章『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。

COBOL 実行単位について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の章『ILE COBOL プログラム相互間での呼び出しとデータ共用』を参照してください。

End of IBM Extension

GO TO ステートメント

GO TO ステートメントによって、手続き部のある点から別の点へ制御を渡すことができます。 GO TO ステートメントには、次の 3 種類の形式があります。

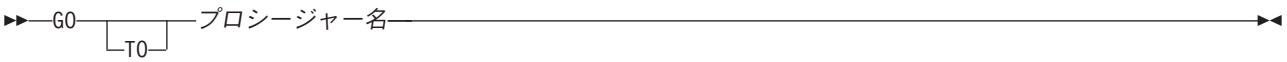
- 無条件
- 条件
- 変更

プロシージャー名またはプロシージャー名-1 が宣言プロシージャーの中にある場合、他の宣言プロシージャーを参照することも、非宣言プロシージャーを参照することもできません。プログラムの非宣言の中では、EXCEPTION/ERROR 宣言プロシージャーの中に現れるプロシージャー名に対する参照があってはなりません。

無条件 GO TO

無条件 GO TO ステートメントは ALTER ステートメントによって変更されない限り、プロシージャー名で指定された名前の段落やセクション内の最初のステートメントへ制御を渡します。 (7-67 ページの『ALTER ステートメント』を参照。)

GO TO ステートメント - 形式 1 - 無条件



プロシージャー名

GO TO ステートメントと同じ手続き部にあるセクションまたは段落でなければなりません。

無条件 GO TO ステートメントが一連の命令ステートメントの中に現れる場合には、その一連のステートメントの最後のステートメントでなければなりません。

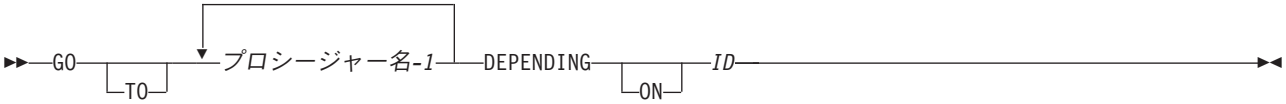
ある段落が ALTER ステートメントによって参照される場合には、その段落は段落名とそれに続く無条件または変更 GO TO ステートメントだけで構成しなければなりません。

条件付き GO TO

条件付き GO TO ステートメントは ID により参照されるデータ項目の値によって、連続したプロシージャーの 1 つに制御を移します。

GO TO ステートメント - 形式 2 - 条件付き

GO TO ステートメント



プロシージャー名-1
手続き部にあるセクションまたは段落でなければなりません。

ID ID は基本データ項目の整数値でなければなりません。

IBM Extension

浮動小数点データ項目であってはなりません。

End of IBM Extension

1 であれば、最初のプロシージャー名-1 で名前を設定されたプロシージャー中の先頭ステートメントへ制御が移ります。

2 の場合は、2 番目のプロシージャー名-1 で名前を指定されたプロシージャー中の先頭ステートメントへ制御が移ります。

ID の値が 1 ～ n (n はこの GO TO ステートメントで指定されたプロシージャー名の数です) の範囲内にない場合には、制御の移動は起こりません。そのかわり、制御は正規の実行順序でステートメントへ移ります。

変更 GO TO ステートメント

変更 GO TO ステートメントは、ALTER ステートメントに指定された名前の段落の最初のステートメントに制御を渡します。

この GO TO ステートメントを含む段落を参照する ALTER ステートメントは、この GO TO ステートメントが実行される前に実行されていなければなりません。 RECURSIVE 属性をもつプログラムで、変更 GO TO ステートメントを指定することはできません。

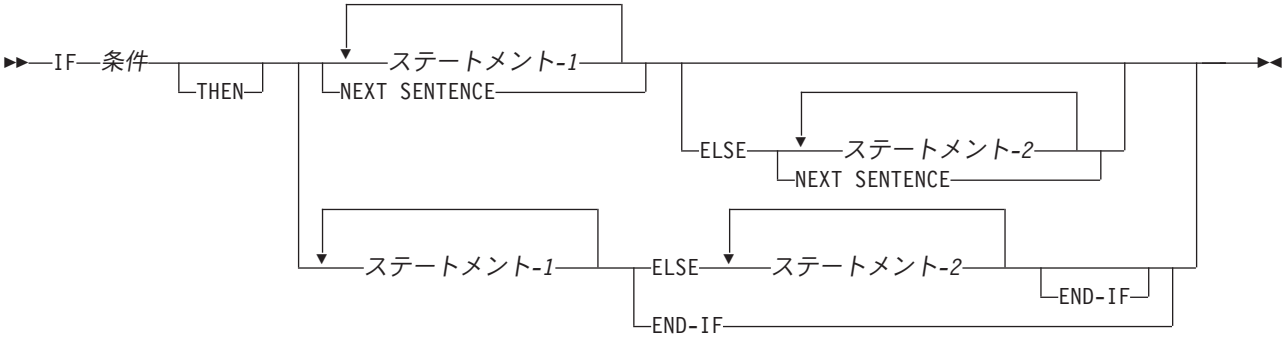
GO TO ステートメント - 形式 3 - 変更

RECURSIVE 属性をもつプログラムで、変更 GO TO ステートメントを指定することはできません。

IF ステートメント

IF ステートメントにより、条件が評価され、その評価に従ってオブジェクト・プログラムの中で代替処置がとられます。

IF ステートメント — 形式



IF ステートメントの範囲は以下のいずれかによって区切ることができます。

- 同じネスト・レベルにある END-IF 句
- 分離文字ピリオド
- ネストされている場合、さらに上位のネスト・レベルにある IF ステートメントに関連した ELSE 句

条件

7-10 ページの『条件式』で説明されているように、単純条件または複合条件が指定されます 。

ステートメント-1、ステートメント-2

ステートメント-1 またはステートメント-2 として、次のいずれかを指定できます。

- 命令ステートメント
- 条件ステートメント
- 条件ステートメントの後にある命令ステートメント

NEXT SENTENCE

END-IF 句を指定する場合には、NEXT SENTENCE 句を指定することはできません。

IBM Extension

NEXT SENTENCE は、END-IF があっても指定できます。

End of IBM Extension

ELSE NEXT SENTENCE

IF ステートメントを終了させる分離文字ピリオドの直前にある場合には、省略できます。

END-IF 句

この明示範囲終了符号は、IF ステートメントの範囲を区切る際に役立ちます。 END-IF 句を使用することによって、IF 条件ステートメントを別の条件ステートメントにネストすることができます。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

制御の転送

テストされた条件が**真**である場合には、次の処理のうちの 1 つが選択されます。

- ステートメント-1 が指定されている場合には、ステートメント-1 が実行されます。ステートメント-1 がプロシージャー分岐ステートメントを含む場合には、そのステートメントの規則に従って制御が移動します。ステートメント-1 がプロシージャー分岐ステートメントを含まない場合には、ELSE 句は指定されていても無視され、制御は対応する (暗黙または明示の) END-IF または分離文字としてのピリオドの後にある、次の実行可能なステートメントに渡されます。

- NEXT SENTENCE が指定されている場合は、これが実行されます。つまり、ELSE 句は指定されていても無視され、最も近い位置にある分離文字のピリオドの後にあるステートメントに制御が渡されます。

テストされた条件が**偽**である場合には、次の処理のうちの 1 つが選択されます。

- ELSE ステートメント-2 が指定されている場合には、ステートメント-2 が実行されます。ステートメント-2 にプロシージャー分岐ステートメントが含まれている場合には、そのステートメントの規則に従って制御が移動します。ステートメント-2 にプロシージャー分岐ステートメントが含まれていない場合には、制御は対応する END-IF または分離文字ピリオドの後にある次の実行可能ステートメントへ移されます。
- ELSE NEXT SENTENCE が指定されている場合には、次の文が実行され、最も近い位置にある分離文字ピリオドの後のステートメントに制御が渡されます。
- ELSE NEXT SENTENCE が省略されている場合には、最も近い位置にある END-IF または分離文字ピリオドの後のステートメントに制御が渡されます。

注: ELSE または ELSE NEXT SENTENCE が省略されている場合には、条件の後で、対応する END-IF またはその文の分離文字ピリオドより前にあるすべてのステートメントが、ステートメント-1 の一部であると見なされます。

ネストされた IF ステートメント

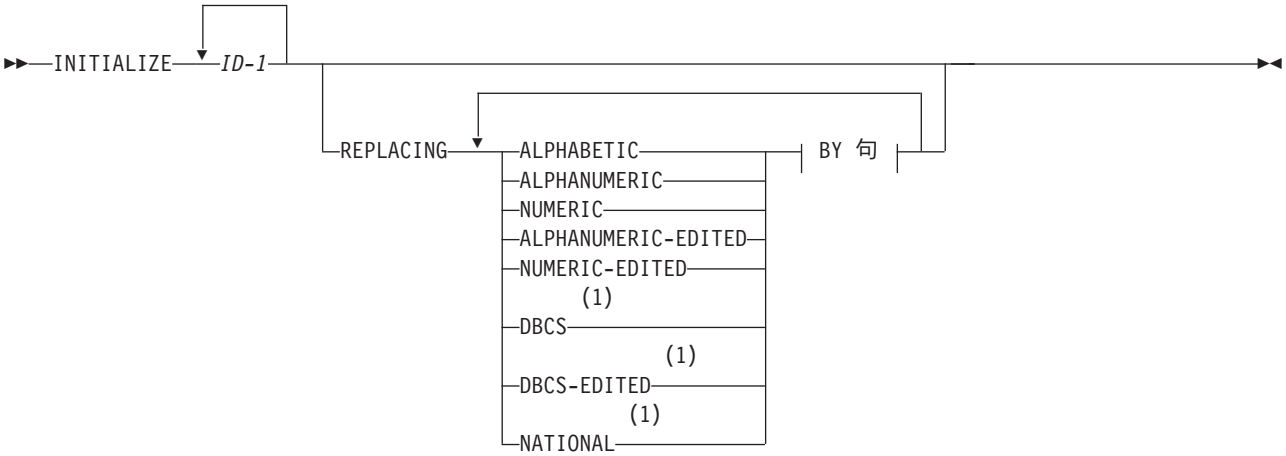
最初の IF ステートメントの中に、1 つまたは複数の IF ステートメントがあると、「ネストされた IF ステートメント」になります。ステートメントのネストは、より大きな算術式の中に従属算術式を括弧で囲んで組み込む指定をするのとよく似ています。

IF ステートメントに含まれる IF ステートメントは、左から右へと進む一対の IF、ELSE、および END-IF の組み合わせとして考えなければなりません。したがって、ELSE または END-IF が存在する場合、これと対になる IF は、直前にある IF で、対になる ELSE または END-IF がないものであると見なされます。

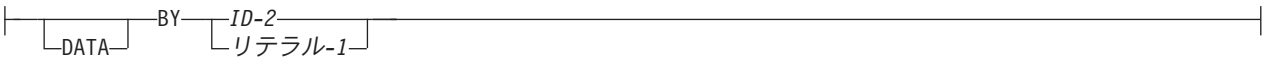
INITIALIZE ステートメント

INITIALIZE ステートメントは、選択されたデータ・フィールドのカテゴリーをあらかじめ決められた値に設定します。これは、機能的には、1 つまたは複数の MOVE ステートメントと同じです。

INITIALIZE ステートメント - 形式



BY 句:



注:

1 IBM 拡張

ID-1
受け入れ区域 (複数可)

ID-2、リテラル-1
送り出し区域 (複数可)

ID-1 には添え字が付いていたり、参照が変更された項目である場合があります。完全なテーブルを初期設定できるのは、ID-1 が完全なテーブルを含むグループ項目である場合です。

ID-1 も、それに従属する項目も OCCURS 文節の DEPENDING ON 句を含みません。 ID-1 のデータ記述記入項目には、RENAMES 文節を入れてはなりません。 指標データ項目は、INITIALIZE のオペランドとすることはできません。

注: 同じ 01 レベル内 OCCURS 文節の DEPENDING ON 句に続く、可変位置項目またはグループを初期設定するために、INITIALIZE ステートメントを使用することはできません。

IBM Extension

浮動小数点データ項目または浮動小数点リテラルは、数字 ID または数字リテラルが指定できるところではどこでも使用できます。

DBCS または国別文字のデータ項目またはリテラルは、ID またはリテラルが指定されているところならば、どこでも使用できます。

End of IBM Extension

REPLACING 句

REPLACING 句を使用する場合には、

- ID-2 またはリテラル-1 のカテゴリーは、MOVE の規則により、対応する REPLACING 句で指示されたカテゴリーとの互換性がなければなりません。

  IBM Extension

  浮動小数点データ項目または浮動小数点リテラルは、NUMERIC カテゴリーであるかのように取り扱われます。

  End of IBM Extension

- REPLACING 句の中で、同じカテゴリーのものを反復することはできません。
- REPLACING という語の後にあるキーワードは 6-9 ページの『データのクラスおよびカテゴリー』に示されるデータのカテゴリーに対応しています。

REPLACING 句を使用しない場合には、

IBM Extension

- SPACE が、英字、英数字、および英数字編集項目用の暗黙の送り出しフィールドとなります。

End of IBM Extension

- SPACE が DBCS 項目および国別項目用の暗黙の送り出しフィールドとなります。
- ZERO が、数字および数字編集項目用の暗黙の送り出しフィールドとなります。

INITIALIZE ステートメントの規則

1. ID-1 が基本項目またはグループ項目のいずれかを参照しても、すべての操作は、あたかも一連の MOVE ステートメントが書かれていて、それぞれに受け入れフィールドとして基本項目が入っているかのように実行されます。

   REPLACING 句が指定されている場合には、次のようになります。

   - ID-1 がグループ項目を参照する場合には、 ID-1 によって参照されるデータ項目内のすべての基本項目は、それが REPLACING 句に指定されたカテゴリーに属している場合にだけ初期設定されます。
   - ID-1 が基本項目を参照する場合には、その項目は、それが REPLACING 句に指定されたカテゴリーに属している場合にだけ初期設定されます。

   この初期設定は、ID-2 またはリテラル-1 によって参照されるデータ項目が、あたかも、識別された項目に対する暗黙の MOVE ステートメント中の送り出しオペランドとして働くかのように行われます。

   このような基本的な受け入れフィールド (グループ内でのテーブル項目のすべてのオカレンスを含めて) は、次の例外を除いて、すべて影響を受けます。

   - 指標、ポインター、およびプロシージャー・ポインター・データ項目
   - 基本 FILLER データ項目
   - ID-1 に従属していて、REDEFINES 文節が入っている項目、あるいはそのような項目に従属している項目。 (ただし、ID-1 は、REDEFINES 文節が入っていてもよく、あるいは再定義項目に従属していてもかまいません。)
   - BOOLEAN データ項目
   - 日付、時刻、またはタイム・スタンプの FORMAT 文節で記述されるデータ項目。
2. ID-1 によって参照される区域は、ステートメント中で ID-1 が現れる順序 (左から右) で初期設定されます。受け入れフィールドのグループ内では、影響を受ける基本項目が、グループ内でそれらが定義された順序で初期設定されます。
3. ID-1 が ID-2 と同じ記憶域を使用している場合には、たとえ、これらのオペランドが同じデータ記述記入項目で定義されていても、このステートメントの実行の結果は未定義となります。
4. ID-1 がグループ項目である場合には、そのグループ項目内のすべての項目がプログラム中で参照されているものと見なされます。

INSPECT ステートメント

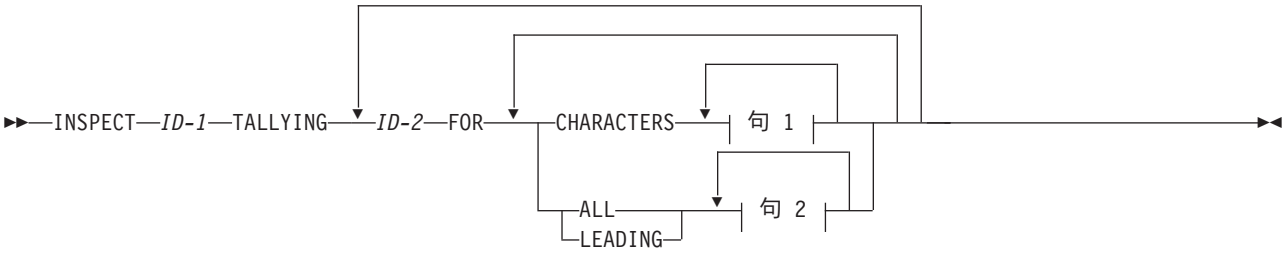
INSPECT ステートメントは、データ項目中の文字をカウントする (数える)、または置き換える (あるいはその両方) ように指定します。

- データ項目中に特定の文字 (英字、数字、または特殊文字) が現れるたびにカウントします。
- あるデータ項目の一部または全部をスペースまたはゼロで埋めます。

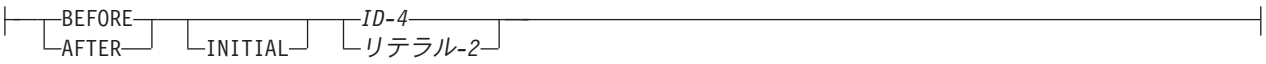
- 一連の文字の照合順序を別の照合順序に変換します。

INSPECT ステートメント - 形式 1

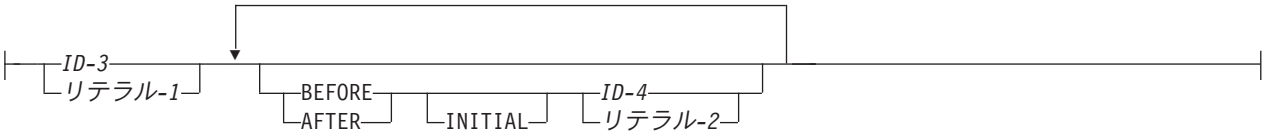
INSPECT ステートメント - 形式 1



句 1:

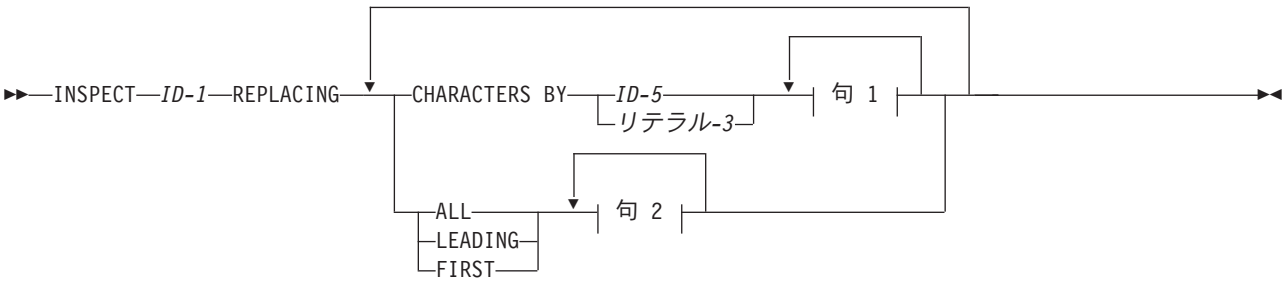


句 2:



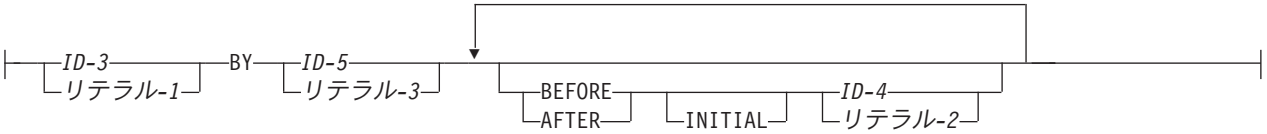
INSPECT ステートメント - 形式 2

INSPECT ステートメント - 形式 2



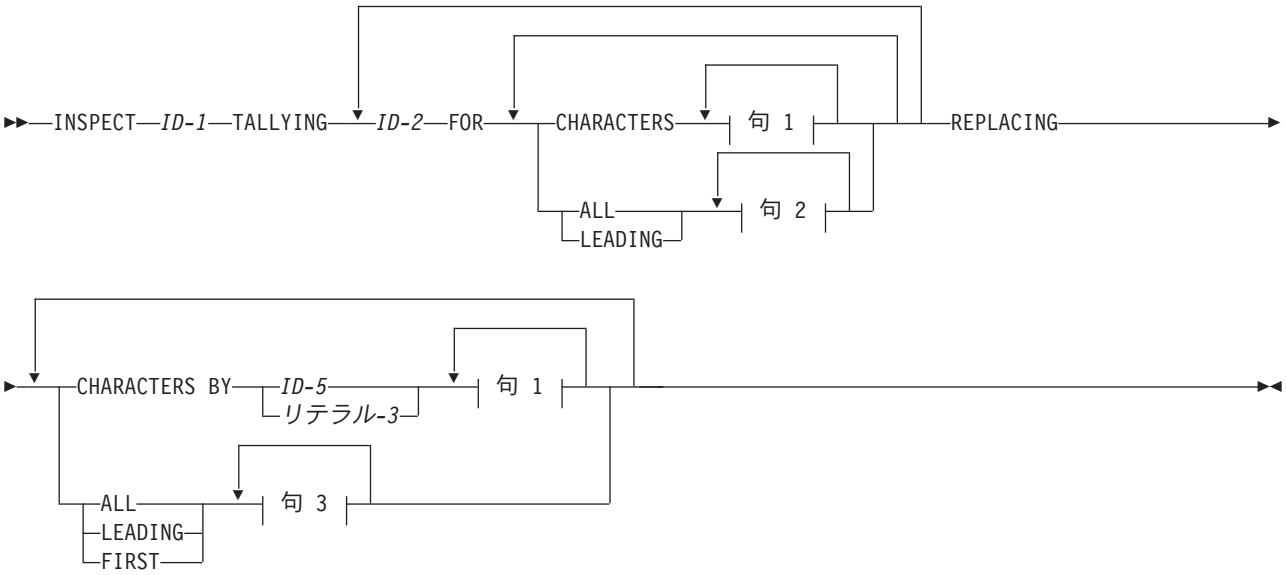
句 1:

句 2:



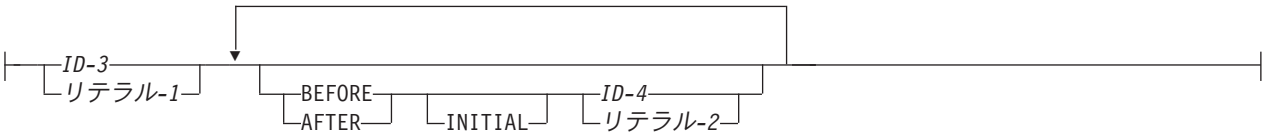
INSPECT ステートメント - 形式 3

INSPECT ステートメント - 形式 3

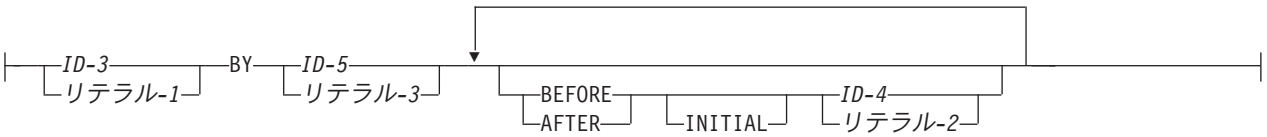


句 1:

句 2:

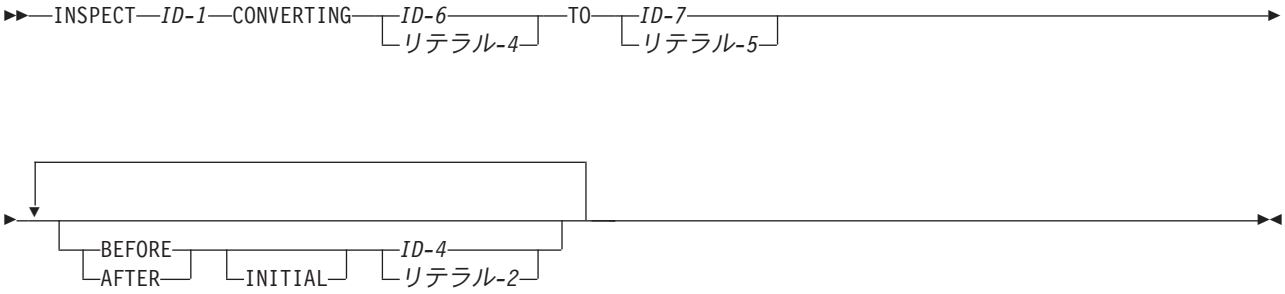


句 3:



INSPECT ステートメント - 形式 4

INSPECT ステートメント - 形式 4



ID-1
検査される項目、すなわち USAGE DISPLAY が指定された基本項目またはグループ項目です。

形式 1 の場合、ID-1 は送り出し項目です。それ以外の形式の場合、これは長さを判別する目的で送り出しデータ項目として扱われます。

ID-2
基本数字データ項目でなければなりません。

ID-3 . . . ID-7
USAGE DISPLAY が指定された基本データ項目でなければなりません。

INSPECT ステートメントの中で使用する場合は、 ID-2 (カウント・フィールド) 以外のすべての ID で参照される各データ項目の内容は、次のように取り扱われます。

英字項目または英数字項目
文字ストリングとして取り扱われます。

英数字編集項目、数字編集項目、または符号のない数字 (外部 10 進) 項目
英数字項目として定義されたものとして取り扱われ、INSPECT ステートメントはその英数字項目を参照します。

符号付きの (外部 10 進) 項目
長さが同じで符号のない外部 10 進項目へ転送されたものとして取り扱われ、その後で英数字項目を参照する INSPECT ステートメントを指定し、英数字項目として再定義されます。符号が分離文字の場合には、この符号を含むバイトが検査されず、したがって、置き換えられません。

リテラル-1 . . . リテラル-5
リテラルは非数字でなければならず、ALL で始まらない任意の表意定数とすることができます。 リテラル-1、リテラル-2、またはリテラル-4 が表意定数の場合は、1 文字で構成される暗黙のデータ項目を参照します。

INSPECT ステートメントの考慮事項

IBM Extension

ID-2 (カウント・フィールド) 以外の任意の ID またはリテラルが DBCS 項目である場合は、すべての ID またはリテラルを DBCS 項目にしなければなりません。

ID-2 を DBCS 項目にすることはできません。 ID-2 では、データのバイト数ではなく DBCS 文字が計数されます。

ID-2 (カウント・フィールド) 以外の ID はすべて、外部浮動小数点項目であってもかまいません。外部浮動小数点項目は、英数字項目を参照する INSPECT ステートメントで英数字として再定義されたかのように取り扱われます。

End of IBM Extension

BEFORE または AFTER 句が指定されている場合を除き、 検査される項目 (ID-1) の左端の文字位置から検査が開始され、右端の位置に向かって文字単位に検査が続けられます。

次の句のオペランドは、INSPECT ステートメントにおいて指定された左から右の順序で比較されます。

- TALLYING (リテラル-1 または ID-3, . . .)
- REPLACING (リテラル-3 または ID-5, . . .)

ID が、添え字付き、参照変更されている、または関数 ID のいずれかである場合、添え字、参照修飾子、または関数は、INSPECT ステートメントの最初の実行時に一度だけ評価されます。

比較の規則

1. TALLYING 句および REPLACING 句の両方が指定されている場合には、INSPECT ステートメントは、INSPECT TALLYING ステートメントが指定されていて、その直後に INSPECT REPLACING ステートメントが指定されているかのように実行されます。
2. 最初の被比較項目は、検査される項目の左端から連続した文字について同じ桁数分だけ比較されます。両方の各文字が等しい場合に限り、被比較項目は検査文字と一致します。
3. 最初の項目が一致しない場合には、比較は後続の被比較数項目について次々と行われ、一致するものが見つかるか、すべての被比較数項目に対する比較が完了するまで繰り返されます。
4. 一致するものが見つかった場合には、TALLYING/REPLACING 句の記述に従ってカウントまたは置き換えが行われます。検査される項目において、右端の一致する文字の後に続く最初の文字は、左端の文字桁であるものと見なされます。その後で比較規則の 2 および 3 で説明した処理が繰り返されます。
5. 一致するものが見つからない場合には、検査される左端の文字に続く項目中の最初の文字が、ここでは左端の文字であると見なされます。その後で比較規則の 2 および 3 で説明した処理が繰り返されます。
6. CHARACTERS 句が指定されている場合には、暗黙の 1 文字の項目が規則 2 および 3 で説明した処理で使用されます。暗黙のうちに指定された文字は、検査される項目の文字と常に一致するものと見なされます。
7. 比較規則 1 ～ 6 で取られる処理 (**比較サイクル**として定義されます) は、検査される項目中の右端の文字が一致するか、または左端の文字桁と見なされるまで繰り返されます。その後で検査が終了します。

7-131 ページの『BEFORE および AFTER 句 (すべての形式)』で説明されているように、BEFORE 句または AFTER 句が指定されていると前述の規則は変更されます。

INSPECT ステートメントの実行結果の例を 7-129 ページの図 7-3 に示します。

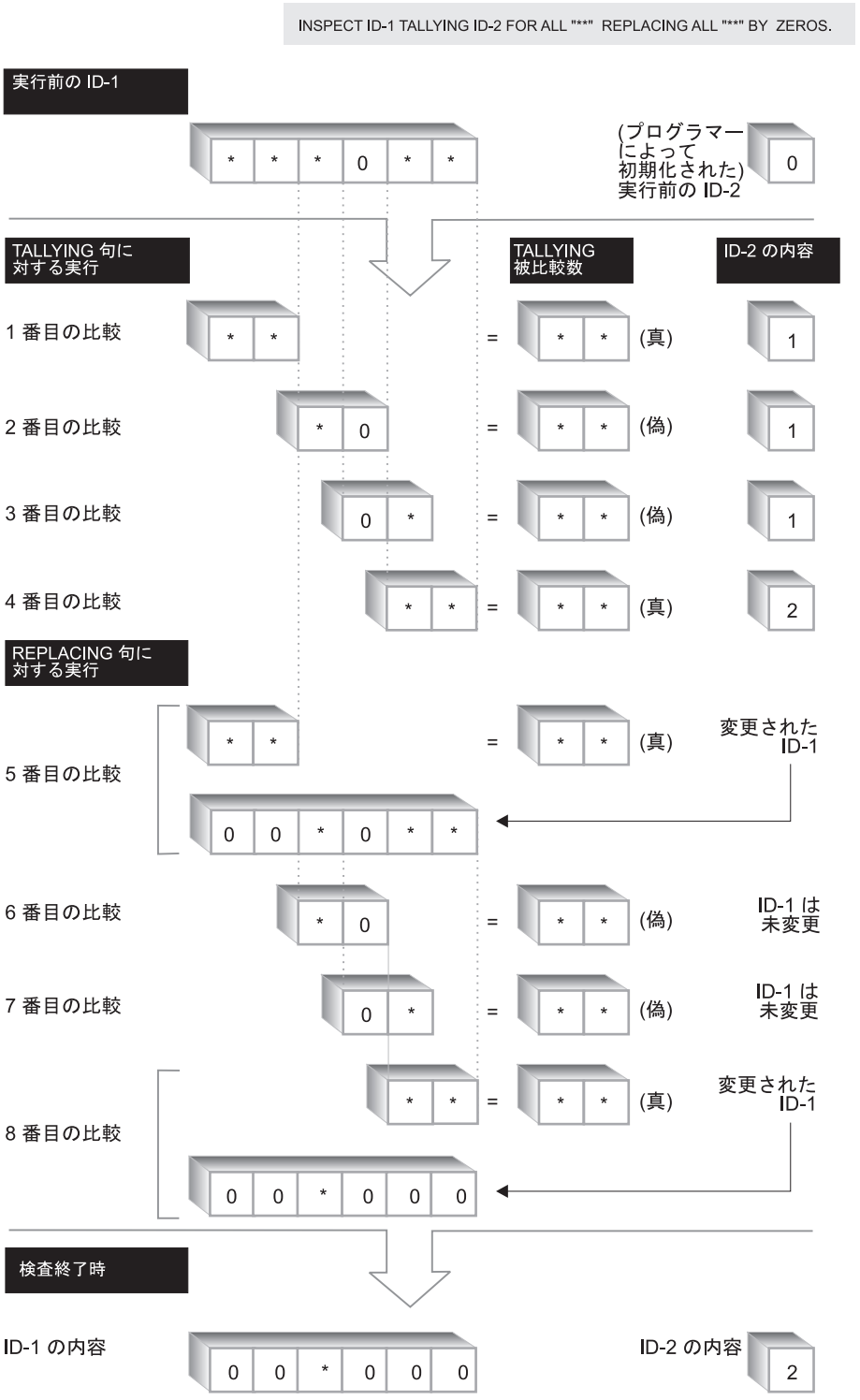


図 7-3. INSPECT ステートメントの実行結果例

INSPECT の例

INSPECT ステートメントの例を示します。

```
.. 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  ID-1          PIC X(10)   VALUE "ACADEMIANS".
 01  CONTR-1       PIC 99      VALUE 00.
 01  CONTR-2       PIC 99      VALUE ZEROS.
 PROCEDURE DIVISION.
*    THIS ILLUSTRATES AN INSPECT STATEMENT WITH 2 VARIABLES.
 100-BEGIN-PROCESSING.
     DISPLAY CONTR-1 SPACE CONTR-2.
 101-MAINLINE-PROCESSING.
     PERFORM COUNT-IT THRU COUNT-EXIT.
     STOP RUN.
 COUNT-IT.
     INSPECT ID-1
     TALLYING CONTR-1
      FOR CHARACTERS BEFORE INITIAL "AD"
             CONTR-2
      FOR ALL "MIANS".
 DISPLAY-COUNTS.
     DISPLAY "CONTR-1 = " CONTR-1.
     DISPLAY "CONTR-2 = " CONTR-2.
     DISPLAY "*********EOJ*********"
COUNT-EXIT.
     EXIT.
```

出力結果

```
     00 00
     CONTR-1 = 02
     CONTR-2 = 01
     *********EOJ*********
```

TALLYING 句 (形式 1 と 3)

ID-2

カウント・フィールド。 PICTURE 文字ストリングに記号 P を指定せずに定義した数字基本項目でなければなりません。 INSPECT ステートメントの実行前に ID-2 を初期設定しなければなりません。

ID-3 またはリテラル-1

計数オペランド (オカレンスが数えられる項目)。計数オペランドが表意定数である場合、これは 1 文字の非数字リテラルと見なされます。

BEFORE 句も AFTER 句も指定されていない場合には、INSPECT TALLYING ステートメントの実行時に次の処置が取られます。

- ALL 句が指定されている場合には、左端の文字位置から右端まで続いて検査される項目中に重複しない計数オペランドが現れるたびに、カウント・フィールドに 1 が加えられます。
- LEADING 句が指定されている場合には、検査される項目中に連続する重複しない計数オペランドが現れるたびにカウント・フィールドに 1 が加えられます。ただしこれは、そのような左端のオカレンスが比較の開始点であり、該当する計数オペランドが関与できる最初の比較サイクルにこの開始点があることを前提にしています。
- CHARACTERS 句が指定されている場合には、検査される項目中の各文字 (スペース文字を含む) ごとにカウント・フィールドに 1 が加えられます。したがって、INSPECT TALLYING ステートメントの実行によって、検査される項目中の文字数の値がカウント・フィールドの値に加えられます。

REPLACING 句 (形式 2 と 3)

ID-3 またはリテラル-1
サブジェクト・フィールド。

ID-5 またはリテラル-3
置き換えフィールド。

サブジェクト・フィールドおよび置き換えフィールドは、同じ長さでなければなりません。置き換えには次の規則が適用されます。

- サブジェクト・フィールドが表意定数である場合には、サブジェクト・フィールドは 1 文字の非数字リテラルと見なされます。検査される項目中の表意定数と等しい文字が、置き換えフィールドの 1 つの文字に置き換えられます。置き換えフィールドは長さが 1 文字でなければなりません。
- 置き換えフィールドが表意定数である場合には、そのフィールドはサブジェクト・フィールドと同じ長さであると見なされます。検査される項目にあるサブジェクト・フィールドの重複しない各オカレンスは、置き換えフィールドに置き換えられます。
- サブジェクト・フィールドおよび置き換えフィールドが文字ストリングである場合には、検査される項目中に重複しないサブジェクト・フィールドが現れるたびに、それが置き換えフィールドで指定された文字ストリングに置き換えられます。
- 検査される項目の所定の文字位置で置き換えが一度行われた後は、この INSPECT ステートメント実行中には、この文字位置に対する置き換えは行われません。

CHARACTERS 句を使用する場合、リテラル-3 または ID-5 の長さは 1 文字で、リテラル-2 または ID-4 の長さは 1 文字でなければなりません。

BEFORE 句も AFTER 句も指定されていない場合には、INSPECT REPLACING ステートメントの実行時に次の処置が取られます。

- CHARACTERS 句が指定されている場合、置き換えフィールドの長さは 1 文字でなければなりません。検査されるフィールドの各文字は、この置き換えフィールドによって置き換えられます。このプロセスは左端の文字から開始され、右端の文字まで続けられます。
- ALL 句が指定されている場合、検査される項目中のサブジェクト・フィールドの重複しない各オカレンスは、置き換えフィールドによって置き換えられます。これは左端の文字から開始され、右端の文字まで続けられます。
- LEADING 句が指定されている場合は、検査される項目中のサブジェクト・フィールドの重複しない一連の各オカレンスは置き換えフィールドによって置き換えられます。ただしこれは、そのような左端のオカレンスが比較の開始点であり、該当する計数オペランドが関与できる最初の比較サイクルにこの開始点があることを前提にしています。
- FIRST 句が指定されている場合には、検査される項目の中で左端にサブジェクト・フィールドの内容が現れたときに、それが置き換えフィールドによって置き換えられます。

BEFORE および AFTER 句 (すべての形式)

1 つの ALL 句、LEADING 句、CHARACTERS 句、FIRST 句、または CONVERTING 句に対して、1 つの BEFORE 句および 1 つの AFTER 句しか指定できません。これらの句が指定されている場合には、カウントおよび置き換えに関する前の規則が変更されます。

ID-4、リテラル-2
ID-4 およびリテラル-2 は、カウントまたは置き換えの対象になりません。ただし、検査される項目のカウントまたは置き換え、あるいはその両方は、ID およびリテラルによって区切られます。分離文字 (ID-4 およびリテラル-2) が表意定数である場合には、その長さは 1 文字であるものと見なされます。

BEFORE 句が指定されている場合には、検査される項目のカウントまたは置き換え、あるいはその両方が左端の文字から開始され、最初の分離文字が出てくるまで続けられます。検査される項目中に分離文字が存在しない場合には、カウントまたは置き換え、あるいはその両方は、右端の文字まで続けられます。

AFTER 句が指定されている場合は、検査される項目のカウントまたは置き換え (あるいはその両方) は、分離文字の右側にある最初の文字から開始され、検査される項目の右端の文字まで続けられます。検査される項目中に分離文字が存在しない場合には、カウントまたは置き換えは行われません。

CONVERTING 句 (形式 4)

置換値のストリングは、この句で表現できます。受け入れ側 (ID-7 またはリテラル-5) のサイズは、送り出し側 (ID-6 またはリテラル-4) のサイズと同じでなければなりません。表意定数をリテラル-5 として使用するときには、表意定数のサイズは、リテラル-4 または ID-6 のサイズと等しくなります。リテラル-4 または ID-6 のどちらの場合も、同じ文字が複数回現れてはなりません。

形式 4 の INSPECT ステートメントは、あたかも形式 2 の INSPECT ステートメントが同じ ID-1 を指定する一連の ALL 句 (リテラル-4 の各文字ごとに 1 つ) を使用して書かれているかのように解釈され、実行されます。その結果、リテラル-4 の各単一文字は、あたかもリテラル-1 として参照されているかのようになり、また、対応するリテラル-5 の各単一文字はリテラル-3 として参照されているかのようになります。リテラル-4 の文字とリテラル-5 の文字との間の対応は、データ項目中の順位によります。

ID-4、ID-6、または ID-7 が、ID-1 と同じ記憶域を使用している場合には、たとえ記憶域が同じデータ記述記入項目によって定義されていても、このステートメントの実行結果は未定義となります。リテラル-2、リテラル-4、またはリテラル-5 には、国別リテラルを使用することはできません。

INSPECT ステートメントの例

次の例は INSPECT ステートメントのいくつかの用法を示したものです。どの例の場合にも、INSPECT ステートメントの実行前にプログラマーが COUNTR フィールドをゼロに初期設定しています。

```
INSPECT ID-1
      REPLACING CHARACTERS BY ZERO.
```

実行前の ID-1	実行後の COUNTR	実行後の ID-1
1234567	0	0000000
HIJKLMN	0	0000000

```
INSPECT ID-1
TALLYING COUNTR FOR CHARACTERS
REPLACING CHARACTERS BY SPACES.
```

実行前の ID-1	実行後の COUNTR	実行後の ID-1
1234567	7	
HIJKLMN	7	

```
INSPECT ID-1
      REPLACING CHARACTERS BY ZEROS
           BEFORE INITIAL QUOTE.
```

実行前の ID-1	実行後の COUNTR	実行後の ID-1
456"ABEL	0	000"ABEL
ANDES"12	0	00000"12
"TWAS BR	0	"TWAS BR

```
INSPECT ID-1
     TALLYING COUNTR FOR CHARACTERS AFTER INITIAL "S"
     REPLACING ALL "A" BY "O".
```

実行前の ID-1	実行後の COUNTR	実行後の ID-1
ANSELM	3	ONSELM
SACKET	5	SOCKET
PASSED	3	POSSED

```
INSPECT ID-1
     TALLYING COUNTR FOR LEADING "0"
     REPLACING FIRST "A" BY "2"
          AFTER INITIAL "C".
```

実行前の ID-1	実行後の COUNTR	実行後の ID-1
00ACADEMY00	2	00AC2DEMY00
0000ALABAMA	4	0000ALABAMA
CHATAM0000	0	CH2THAM0000

```
INSPECT ID-1
     CONVERTING "ABCD" TO "XYZX"
          AFTER QUOTE
          BEFORE "#".
```

実行前の ID-1	実行後の ID-1
AC"AEBDFBCD#AB"D	AC"XEYXFYZX#AB"D

MERGE ステートメント

MERGE ステートメントは、同じ方法で順序付けられた複数のファイル (すなわち、前もって同じ昇順 / 降順キーのセットに従ってソートされているファイル) を 1 つまたは複数のキーに基づいて結合します。レコードは、マージされた順序で、出力プロシージャーまたは出力ファイルに対して使用可能になります。

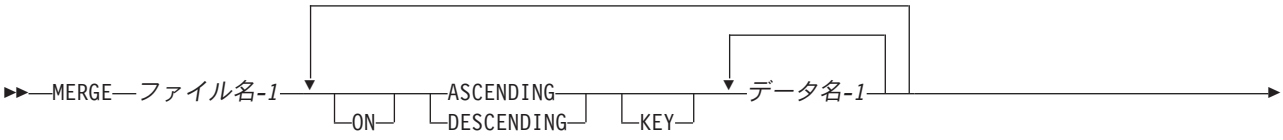
MERGE ステートメントは、宣言セクション内でなければ手続き部のどこに記入してもかまいません。USING または GIVING ファイルの最大数は 32 です。

IBM Extension

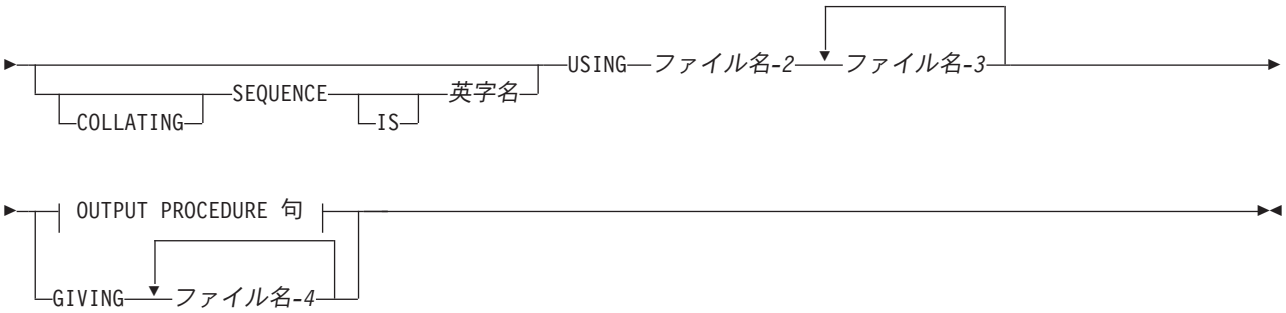
マージ操作の前に入力ファイルを順番に並べる必要はありません。

End of IBM Extension

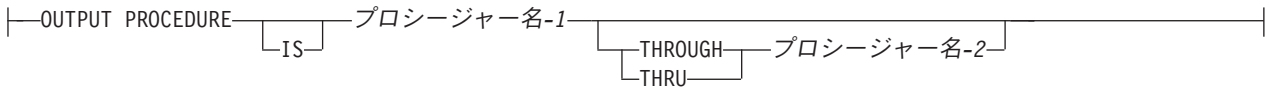
MERGE ステートメント - 形式



MERGE ステートメント



OUTPUT PROCEDURE 句:



ファイル名-1
SD 記入項目で指定された、レコードを記述する名前。

MERGE ステートメントの中でファイル名を繰り返すことはできません。

MERGE ステートメントでは、SAME AREA、SAME SORT AREA、または SAME SORT-MERGE AREA 文節を介してすでにストレージを共用しているファイル名のペアを指定しないでください。ただし、MERGE ステートメントでは、SAME RECORD AREA 文節を共用しているファイルが GIVING 文節 (ファイル名-4) にも関連付けられる場合には、それらを指定できます。

MERGE ステートメントの実行時には、ファイル名-2、ファイル名-3、... に含まれているすべてのレコードがマージ・プログラムによって受け入れられ、その後、指定されているキーに従ってマージされます。

ヌル可能フィールドはサポートされますが、 ヌル値は ASSIGN 文節で ALWNULL が指定された DATABASE ファイルにだけサポートされます。

ASCENDING/DESCENDING KEY 句

この句は、指定されたマージ・キーに基づいてレコードが昇順または降順 (指定された句に従います) に処理されるよう指定します。

データ名-1
キー・データ名です。レコードは、このキーに基づいて昇順または降順に処理されます。

データ名-1 は、マージの基礎となる KEY データ項目を指定します。各データ名は、ファイル名-1 に関連付けられたレコード内のデータ項目を識別します。 MERGE ステートメントでは、KEY の後に続くデータ名は、それらの項目がどのように KEY 句に分割されているかにかかわりなく、重要度の高いものから順に左から右に列挙されます。左端のデータ名が主要なキーであり、次のデータ名がその次に重要度の高いキーというようになります。

次の規則が適用されます。

- 特定の KEY データ項目は、各入力ファイル内で物理的に同じ位置に存在し、同じデータ・フォーマットをもっていなければなりません。ただし、同じデータ名を持つ必要はありません。

- ファイル名-1 が複数のレコード記述をもっている場合には、KEY データ項目は 1 つだけのレコード記述中で記述されている必要があります。
- ファイル名-1 に可変長レコードが含まれている場合には、すべての KEY データ項目はレコードの最初の *n* 文字の位置に入っていなければなりません。ここで *n* は、ファイル名-1 について指定された最小レコード・サイズです。
- KEY データ項目は、OCCURS 文節を含んではならず、また、OCCURS 文節を含む項目に従属してもなりません。
- KEY データ項目を修飾することはできますが、添え字付けまたは指標付けすることはできません。
- KEY データ項目の位置は可変であってはなりません。

IBM Extension

- KEY データ項目は、浮動小数点項目または日時項目にすることができます。
- KEY データ項目を参照変更することはできますが、添え字付けまたは指標付けすることはできません。

End of IBM Extension

- KEY データ項目の全長 (バイト単位) が 2 000 を超えてはなりません。
- MERGE キーの中で可変長フィールドを可変長フィールドとして使用することはできません。可変長フィールドは、ILE COBOLによってグループ項目に変換されます。可変長フィールドはグループ項目に変換されるため、MERGE キーの中で使用されると英数字項目として比較されます。

マージ操作の方向は、キーワード ASCENDING または DESCENDING の指定によって、次のように異なります。

- ASCENDING を指定すると、最下位キー値から最上位キー値への順序となります。
- DESCENDING を指定すると、最上位キー値から最下位キー値への順序となります。
- KEY データ項目が英字、英数字、英数字編集、または数字編集項目である場合には、キー値の順序は使用される照合順序によって決まります (以下の 『COLLATING SEQUENCE 句』を参照してください)。KEY データ項目が DBCS または DBCS 編集である場合には、キー値の順序は DBCS 文字の 16 進数値の 2 進数照合順序に基づきます。 COLLATING SEQUENCE 句は無視されます。

IBM Extension

- KEY が外部浮動小数点項目である場合には、キーは英数字として取り扱われます。レコードがマージされる順序は、使用される照合順序によって異なります。
- KEY が内部浮動小数点数項目である場合には、キー値の順序は数字の順序になります。
- KEY が日時項目である場合は、いくつかの形式だけが日付項目または時刻項目としてソートされます。ILE COBOL は、IBM i DDS より多くの日時形式をサポートします。一般的に、IBM i DDS 形式に一致する ILE COBOL の日時形式は、日付項目または時刻項目としてソートされます。その他のすべての形式は英数字項目として扱われ、その 16 進値に基づいてソートされます。

End of IBM Extension

キーの比較は、比較条件内のオペランドの比較に関する規則に従って実行されます (7-14 ページの『比較条件』を参照してください)。

COLLATING SEQUENCE 句

この句は、このマージ操作において KEY データ項目の非数字比較に使用される照合順序を指定します。

英字名
SPECIAL-NAMES 段落の ALPHABET 文節の中で指定されていなければなりません。英字名、文節、句の任意の 1 つを指定することができ、結果は次のとおりになります。

- NATIVE を指定した場合には、すべての非数字比較に対して EBCDIC 照合順序が使用されます。
- NLSSORT を指定した場合には、照合順序は CRTCBLMOD および CRTBNDCBL コマンドの LANGID および SRTSEQ パラメーターによって判別されます。
- リテラル句を指定した場合には、すべての非数字比較に対して、英字名文節中のリテラルの指定によって確立された照合順序が使用されます。
- STANDARD-1 を指定した場合には、すべての非数字比較に対して ASCII 照合順序が使用されます。
- STANDARD-2 を指定した場合には、International Standard 646, 7-bit Coded Character Set for Information Processing Interchange に定義されている ISO 7 ビット・コードの International Reference Version が使用されます。

COLLATING SEQUENCE 句を省略した場合には、OBJECT-COMPUTER 段落内の PROGRAM COLLATING SEQUENCE 文節 (指定されている場合) によって、使用される照合順序が決まります。

COLLATING SEQUENCE 句および PROGRAM COLLATING SEQUENCE 文節の両方を省略した場合には、EBCDIC 照合順序が使用されます。

USING 句

ファイル名-2、ファイル名-3、...
入力ファイルを指定します。

USING 句を指定した場合には、ファイル名-2、ファイル名-3、... (つまり、入力ファイル) のすべてのレコードが自動的にファイル名-1 に転送されます。 MERGE ステートメントの実行時には、これらのファイルがオープンされていてはなりません。コンパイラーによって、入力ファイルをオープン、読み取り、およびクローズするコードが自動的に生成されます。これらのファイルについて EXCEPTION/ERROR プロシージャーが指定されている場合には、COBOL コンパイラーはこれらのプロシージャーに対して必要なリンケージを行います。

すべての入力ファイルは、データ部の FD 記入項目で記述されていなければなりません。さらに、それらのレコード記述は、マージ・ファイルについて記述されたレコードと同じサイズのレコードを記述するものでなければなりません。これらのレコードを構成する基本項目が同じでない場合には、入力レコードは、マージ・レコードと同じ文字桁数をもっていなければなりません。

入力ファイルは、順次、相対、または索引編成になっていなければなりません。

ファイル名-1 に可変長レコードが含まれている場合には、入力ファイル内のレコードのサイズは、ファイル名-1 について記述されている最小レコードよりも小さくてはならず、最大レコードよりも大きくてはなりません。ファイル名-1 に固定長レコードが含まれている場合には、入力ファイル内のレコードのサイズは、ファイル名-1 について記述されている最小レコードよりも小さくてはならず、最大レコードよりも大きくてはなりません。

GIVING 句

ファイル名-4、...
入力ファイルを指定します。

GIVING 句を指定した場合には、ファイル名-1 の中のマージ済みの全レコードは自動的に出力ファイル (ファイル名-4) へ転送されます。 MERGE ステートメントの実行開始時には、ファイル名-4 によって参照されているファイルがオープンしていてはなりません。ファイル名-4 によって参照されるファイルのおのおのに対して、MERGE ステートメントの実行時に次のような処置が取られます。

1. ファイルの処理が開始されます。この開始処理は、OUTPUT 句を指定した OPEN ステートメントが実行されたかのように行われます。
2. マージ済みの論理レコードが戻され、ファイルに書き込まれます。おのおののレコードは、オプショナルの句を指定しない WRITE ステートメントが実行されたかのように書き込まれます。レコードは、ファイルの以前の内容 (もしあれば) を上書きします。

IBM Extension

ファイル名-1 が論理データベース・ファイルである場合には、レコードはファイルの終わりに追加されます。

End of IBM Extension

ファイル名-4 によって参照されるファイルが INDEXED ファイルである場合には、このファイルに関連するキー・データ名が、マージ・ステートメント中に ASCENDING KEY 句をもっていなければなりません。これと同じデータ名は、それのレコード内で、ファイルの基本レコード・キーに関連するデータ項目と同じ文字位置を占めていなければなりません。

相対ファイルの場合には、戻される最初のレコードについての相対キー・データ項目に値 '1' が入り、戻される 2 番目のレコードについての相対キー・データ項目に値 '2' が入り、... というようになります。 MERGE ステートメントの実行後には、相対キー・データ項目の内容は、ファイルに戻された最後のレコードを示します。

3. ファイルの処理が、オプショナルの句を指定しない CLOSE ステートメントが実行されたかのように終了されます。

注: 索引付きファイルへの書き込み中に重複キーが見つかると、MERGE ステートメントの実行が終了し、すべての GIVING ファイル内のマージ済みデータは未完成のままとなります。

これらの暗黙の機能は、関連する USE AFTER EXCEPTION/ERROR プロシージャーが実行されるように実行されます。ただし、そのような USE プロシージャーの実行により、ファイル名-4 によって参照されているファイルを操作するか、あるいはファイル名-4 に関連付けられているレコード域にアクセスするステートメントの実行が生じてはなりません。このファイルの外部定義境界を超えてデータを書き込もうとする最初の試みが行われると、このファイルについて指定されている USE AFTER STANDARD EXCEPTION/ERROR プロシージャーが実行されます。この USE プロシージャーから制御が返された場合、あるいはこのような USE プロシージャーが指定されていない場合は、このファイルの処理が終了します。

出力ファイルは、データ部の FD 記入項目で記述されていなければなりません。さらに、そのレコード記述は、マージ・ファイルについて記述されたレコードと同じサイズのレコードを記述するものでなければなりません。これらのレコードを構成する基本項目が同じでない場合には、出力レコードは、マージ・レコードと同じ文字桁数をもっていなければなりません。

出力ファイルは、順次、相対、または索引編成でなければなりません。

出力ファイルは、キー順アクセス・パスなしで作成されたものでなければなりません。そうでなければ、MERGE ステートメントは、データ記述仕様 (DDS) で指定された照合順序を指定変更することができません。

出力ファイル (ファイル名-4) に可変長レコードが含まれている場合には、ファイル名-1 内のレコードのサイズは、出力ファイルについて記述されている最大レコードよりも小さくてはなりません。出力ファイルに固定長レコードが含まれている場合には、ファイル名-1 内のレコードのサイズは、出力ファイルについて記述されている最大レコードよりも大きくてはなりません。

OUTPUT PROCEDURE 句

この句は、マージ操作からの出力レコードを選択または変更するためのプロシージャーの名前を指定します。

プロシージャー名-1
出力プロシージャー内の最初の (または唯一の) セクションまたは段落を指定します。

プロシージャー名-2
OUTPUT PROCEDURE の最後のセクションまたは段落を識別します。

OUTPUT PROCEDURE は、ファイル名-1 によって参照されたファイルからのマージされた順序で、RETURN ステートメントによって一度に 1 つずつ使用可能にされるレコードを選択、変更、またはコピーするのに必要な任意のプロシージャーから構成できます。この中には、出力プロシージャー内の CALL、EXIT、GO TO、および PERFORM ステートメントによる制御権の移動の結果として実行されるすべてのステートメントが含まれます。また、出力プロシージャー内のステートメントの実行の結果として実行される宣言プロシージャー内のすべてのステートメントも含まれます。出力プロシージャー内では、MERGE、RELEASE、または SORT ステートメントの実行を引き起こしてはなりません。

出力プロシージャーを指定すると、ファイル名-1 によって参照されたファイルが MERGE ステートメントによって順序付けられた後で、制御権がこの出力プロシージャーに渡されます。

注: OUTPUT PROCEDURE 句は、基本 PERFORM ステートメントと類似しています。例えば、OUTPUT PROCEDURE でプロシージャーを指定すると、そのプロシージャーは、PERFORM ステートメントで指定された場合と同じように、マージ操作時に実行されます。 PERFORM ステートメントの場合と同様に、プロシージャーの実行は最後のステートメントの実行が完了した後で終了されます。 OUTPUT PROCEDURE の最後のステートメントは、EXIT ステートメントにすることができます (7-116 ページの『EXIT ステートメント』を参照してください)。

SORT-RETURN 特殊レジスター

IBM Extension

SORT-RETURN 特殊レジスターは、2 進データ項目の名前であり、ソートとマージの両方のプログラムで使用可能です。

SORT-RETURN 特殊レジスターには、次の暗黙定義があります。

```
01     SORT-RETURN GLOBAL PICTURE S9(4) USAGE BINARY VALUE ZERO.
```

ネストされたプログラムの中で使用されるときには、SORT-RETURN 特殊レジスターは、最外部の COBOL プログラム内で GLOBAL として暗黙に定義されます。 SORT-RETURN 特殊レジスターには、ソート・マージ操作の完了時に 0 (成功) または 16 (失敗) の戻りコードが入ります。

エラー宣言または入出力プロシージャーの中で SORT-RETURN 特殊レジスターを 16 に設定して、すべてのレコードが処理される前にソート・マージ操作を終了させることができます。操作は、レコードが RETURN または RELEASE される前に終了されます。整数の引数が認められているところではどこでも、SORT-RETURN 特殊レジスターを関数内で指定できます。

End of IBM Extension

MOVE ステートメント

MOVE ステートメントは、ストレージの区域間でデータを転送します。

MOVE ステートメント - 形式 1

MOVE ステートメント - 形式 1

MOVE ステートメント - 形式 2

MOVE ステートメント - 形式 2

ID-1、リテラル-1
送り出し項目。

ID-2
受け入れ項目 (複数可)。

形式 1 では、すべての ID をグループまたは基本項目にすることができます。送り出し項目内のデータは、ID-2 が指定されている順序で、それぞれの ID-2 によって参照されているデータ項目に移動されます。 7-140 ページの『基本移動』および 7-146 ページの『グループ移動』を参照してください。

形式 2 では、ID-1 および ID-2 はグループ項目でなければなりません。 ID-1 内の選択された項目は、7-31 ページの『CORRESPONDING 句』 ページに説明されている CORRESPONDING 句に関連する規則に従って、ID-2 に移動されます。

MOVE ステートメントの規則

IBM Extension

送り出し項目または受け入れ項目が DBCS データ項目の場合は、両方とも DBCS 項目でなければなりません。 DBCS 送り出し項目は DBCS リテラルまたは表意定数 SPACE にすることができます。データ変換が行われない場合、データは切り捨てられるかまたは DBCS スペースで右側から埋められます。

End of IBM Extension

MOVE ステートメントでは、指標データ項目を指定することはできません。

IBM Extension

MOVE ステートメントでは、ポインター・データ項目 (USAGE POINTER) またはプロシージャー・ポインター・データ項目 (USAGE PROCEDURE-POINTER) を指定することはできません。アドレスをポインターまたはプロシージャー・ポインター・データ項目に移動するには、SET ステートメントを使用してください。

End of IBM Extension

送り出し項目または受け入れ項目の長さの評価は、OCCURS 文節の DEPENDING ON 句によって影響を受けることがあります (6-47 ページの『OCCURS 文節』を参照してください)。

送り出し項目 (ID-1 またはリテラル-1) に関連する長さ評価、添え字付け、参照変更、または関数は 、データが最初の受け入れ項目に移動される直前に一度だけ評価されます。受け入れ項目 (ID-2) に関連する長さ評価、添え字付け、または参照変更は、データが受け入れ項目に移動される直前に評価されます。

例えば、次のステートメント

```
MOVE A(B) TO B, C(B).
```

の結果は、次のステートメントの結果と同じです。

```
MOVE A(B) TO TEMP.
MOVE TEMP TO B.
MOVE TEMP TO C(B).
```

ここで TEMP は、中間結果項目です。添え字 B は、最初の転送が行われる時点と最後の C(B) への転送が行われる時点では値が異なります。

MOVE ステートメントの実行後、送り出し項目は、実行前と同じデータを含んでいます (受け入れ項目が、ストレージで送り出し項目と重複する場合を除く。重複する場合、その内容は予測不可能です)。

基本移動

基本移動とは、受け入れ項目が基本項目であり、送り出し項目が基本項目またはリテラルである場合の移動のことです。移動時には、データの内部表現をある形式から別の形式に変換する処理が行われます (必要であれば)。また、このときに、受け入れ項目内で指定された編集、または受け入れ項目によって暗黙指定された編集解除が実行されます。

編集解除とは、すべての編集文字を数字編集データ項目から論理的に除去することです。これは、この項目の未編集数字値を決定するためです。

IBM Extension

編集解除は、日時クラスにも起こります。このケースでは、すべての分離文字と数字でない変換指定子はすべて日時項目から取り除かれ、結果として数字になります。

End of IBM Extension

各基本項目は次のカテゴリーの 1 つに属します。

- **英字** - 英字データ項目および表意定数 SPACE が含まれます。
- **英数字** - 英数字データ項目、非数字リテラル、および SPACE を除くすべての表意定数が含まれます。(表意定数 ZERO は、英数字項目または英数字編集項目へ移動される場合に限って、英数字として扱われます。)

- **英数字編集** - 英数字編集データ項目が含まれます。
- **数字** - 数字データ項目、数字リテラル、および表意定数 ZERO が含まれます。 (表意定数 ZERO は、数字項目または数字編集項目へ移動される場合に限って、数字として扱われます。)
- **数字編集** - 数字編集データ項目が含まれます。

IBM Extension

- **浮動小数点** - 内部浮動小数点項目 (USAGE COMP-1 または USAGE COMP-2 として定義)、外部浮動小数点項目 (USAGE DISPLAY として定義)、および浮動小数点リテラルが含まれます。
- **ブール** - ブール・データ項目およびブール・リテラルが含まれます。
- **DBCS** - DBCS データ項目および DBCS リテラルが含まれます。
- **国別** - 国別データ項目および国別リテラルが含まれます。
- **日時** — 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が含まれます。日時データ項目は、USAGE DISPLAY または PACKED-DECIMAL として定義されます。

End of IBM Extension

次の規則は、有効な基本移動を実行する場合の概略です。受け入れ項目別に示します。

英字:

- 位置合わせおよび必要なスペース埋め込みが 6-10 ページの『位置合わせの規則』で説明したとおりに行われます。
- 送り出し項目の長さが受け入れ項目の長さより大きい場合には、右側のあふれた文字は受け入れ項目が満たされた後で切り捨てられます。

IBM Extension

- 送り出し項目が国別データ項目である場合には、受け入れフィールドに渡される前に変換されます。変換は 7-144 ページの『国別』で説明したデータ変換規則に基づいて実行されます。

End of IBM Extension

英数字または英数字編集:

- 位置合わせおよび必要なスペース埋め込みが、 6-10 ページの『位置合わせの規則』で説明されているように行われます。
- 送り出し項目が国別 10 進数の整数項目の場合、その送り出しデータ項目は USAGE DISPLAY に変換され、送り出し項目と同じ文字位置数の英数字カテゴリーの一時データ項目へ移動されるように処理されます。生成された英数字データ項目は、送り出し項目として扱われます。
- 送り出し項目の長さが受け入れ項目の長さより大きい場合には、右側のあふれた文字は受け入れ項目が満たされた後で切り捨てられます。
- 送り出し項目が演算符号をもっている場合には、絶対値が使用されます。演算符号が 1 文字を占めている場合には、その文字は転送されず、送り出し項目のサイズは、実際のサイズより 1 文字小さいものと見なされます。

IBM Extension

- 送り出し項目がブールである場合には、送り出し項目が長さ 1 の英数字項目として記述されているかのようにデータが移動されます。

- 送り出し項目が国別データ項目である場合には、受け入れフィールドに渡される前に変換されます。変換は 7-144 ページの『国別』で説明したデータ変換規則に基づいて実行されます。
- 送り出し項目が日時項目である場合には、日時項目は英数字項目のように扱われ、英数字から英数字への移動の規則に従って受け入れ項目へ移動されます。日時の送り出し項目が PACKED-DECIMAL の USAGE を持っている場合は、これは最初に DISPLAY の USAGE に変換されます。
- 受け入れ項目が英数字編集または数字編集であり、送り出し項目が位取りされた整数である (すなわち、PICTURE 文字ストリングの右端の文字として P を持つ) 場合、MOVE ステートメントの実行時に位取りの桁は後続ゼロとして扱われます。
- 受け入れ項目が数字であり、送り出し項目が英数字リテラル、国別リテラル、または ALL リテラルである場合は、リテラルのすべての文字が数字でなければなりません。

End of IBM Extension

数字または数字編集:

- 受け入れ項目が数字である場合は、10 進小数点による位置合わせおよび必要なゼロの埋め込みが 6-10 ページの『位置合わせの規則』で説明されているとおりに行われます。
- 受け入れ項目が符号付きである場合には、送り出し項目の符号は必要に応じて変換が行われて受け入れ項目の中へ入れられます。送り出し項目が符号なしである場合には、受け入れ項目に対して正の演算符号が生成されます。
- 受け入れ項目が符号なしである場合には、送り出し項目の絶対値が移動され、受け入れ項目に対して演算符号が生成されません。
- 送り出し項目が英数字である場合には、送り出し項目が符号なしの整数として記述されているかのようにデータが移動されます。

IBM Extension

- 送り出し項目が浮動小数点である場合、データは、まず 2 進表記または内部 10 進表記のいずれかに変換されてから移動されます。

End of IBM Extension

- 編集解除により、 数字編集データ項目を数字または数字編集受け入れ項目に移動することが可能になります。コンパイラーはまず、数字編集項目の未編集値 (この値には符号を付けられる) を受け取り数字または数字編集データ項目に転送することによって、転送を完了します。

IBM Extension

- 送り出し項目が日時項目である場合には、この項目は最初に編集解除されます。日時項目の編集されなかった値は、数字または数字編集データ受け入れ項目に移動されます。
- 受け入れ項目が数字編集の場合は、これは LOCALE 句を使用するか、あるいは使用せずに指定できます。 PICTURE 文節の LOCALE 句が、そのデータ記述記入項目で指定されなかった 場合は、編集されたデータ項目に移されたデータは、必要に応じてデータ項目の受け入れ文字位置内の最初か最後かいずれかの端で、ゼロを埋めるかまたは切り捨てるかして 10 進小数点による位置合わせが行われます。ただし、編集によって先頭のゼロを置き換える必要がある場合は除きます。 LOCALE 句が指定されている 場合は 6-57 ページの『LOCALE 句』の説明に従って位置合わせおよびゼロで埋める切り捨てが行われます。

- 受け入れ項目が英数字編集または数字編集であり、送り出し項目が位取りされた整数である (すなわち、PICTURE 文字ストリングの右端の文字として P を持つ) 場合、MOVE ステートメントの実行時に位取りの桁は後続ゼロとして扱われます。
- 受け入れ項目が数字であり、送り出し項目が英数字リテラル、国別リテラル、または ALL リテラルである場合は、リテラルのすべての文字が数字でなければなりません。

End of IBM Extension

浮動小数点:

IBM Extension

- 送り出し項目は、まず内部浮動小数点項目に変換されてから移動されます。
- データが、外部浮動小数点項目に、または外部浮動小数点項目から移動される場合、データは、同等の内部浮動小数点値に、または同等の内部浮動小数点値から変換されます。
- 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る場合には、外部浮動小数点データ項目が不正確な値を受け取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が移動される場合には、これはまず真の浮動小数点値 (IEEE) に変換されますが、この変換も浮動小数点データ・タイプの正確性に影響を与える可能性があります。

  例えば、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 +123455779012.3453E+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
```

End of IBM Extension

日時:

IBM Extension

- 送り出し項目が日時である場合は、日時送り出し項目はまず受け入れ項目の形式に変換された後で移動されます。送り出し項目がタイム・スタンプで、受け入れ項目が日付または時刻の項目である場合は、タイム・スタンプ項目の日付または時刻の部分だけが受け入れ項目へ移動されます。送り出し項目が日付または時刻の項目で、受け入れ項目がタイム・スタンプである場合は、タイム・スタンプの日付または時刻の部分だけが置き換えられます。
- 送り出し項目が数字の場合、各受け入れ項目の数字変換指定子は送り出し項目からの数字で置き換えられます。この置き換えは、右端の変換指定子、およびその変換指定子の右端数字で開始されます。すべての英数字変換指定子はデフォルト値をもちます。
- 送り出し項目が数字編集である場合は、数字編集項目が編集解除されます。結果の数字が日時項目へ移動されます。
- 送り出し項目が英数字または英数字編集である場合は、日時受け入れ項目は英数字項目として扱われ、英数字から英数字への移動の規則に従って移動が行われます。

End of IBM Extension

ブール:

IBM Extension

- ブール受け入れ項目の場合、送り出し項目の最初のバイトだけが転送されます。
- 送り出し項目が英数字であれば、送り出し項目の先頭の文字が転送されます。文字 "0" および "1" はそれぞれ、ブール値の B"0" および B"1" に相当します。
- 送り出し項目が ZERO である場合には、ブール・リテラル B"0" として扱われます。

End of IBM Extension

DBCS または DBCS 編集:

IBM Extension

- 送り出し項目が国別データ項目である場合には、受け入れフィールドに渡される前に変換されます。変換は 『国別』で説明したデータ変換規則に基づいて実行されます。
- それ以外の場合、変換は行われません。
- 送り出し項目および受け入れ項目のサイズが異なる場合、データ項目は切り捨てられたり、DBCS スペースが必要に応じて (右側に) 埋められたりします。

End of IBM Extension

国別:

IBM Extension

| 国別データ項目は、英字、英数字、DBCS、または国別データ項目から、さらに、非数値、DBCS、または
| 国別リテラル、あるいは表意定数 SPACE/SPACES からもデータを受け入れます。

このような項目に移動されたデータは、左端の文字位置に位置合わせされ、必要に応じて、右側が、切り捨てられるか、あるいは埋め込み文字コンパイル・オプションまたは PROCESS ステートメントの NTLPADCHAR オプションに指定されている埋め込み文字によって埋められます。 PROCESS ステートメントについては、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

転送中のデータが国別データでない場合には、受け入れフィールドに置かれる前に、データ変換規則に従って、送り出しフィールドの表記から変換されます。

国別 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 は、国別データ CCSID を定義するために使用されます。

他のデータ項目に関連する CCSID は、以下の規則によって決まります。

- 非数値リテラルは、プログラム・ソース・ファイルに指定された CCSID を使用します。
- DBCS リテラルは、プログラム・ソース・ファイルに指定された CCSID に対応する DBCS CCSID を使用します。
- 英字または英数字などの単一バイトのデータ項目は、PROCESS ステートメントの CCSID オプションの 2 番目の項目によって指定された CCSID を使用します。
- DBCS データ項目は、PROCESS ステートメントの CCSID オプションの 3 番目の項目によって指定された CCSID を使用します。

End of IBM Extension

有効な基本移動: 表 7-12 に、各カテゴリーの有効および無効な基本移動を示します。表の中では、以下のように表記します。

- 可 = 移動は有効である。
- 不可 = 移動は無効である。

表 7-12. 有効な基本移動

送り出し項目のカテゴリー	受け入れ項目のカテゴリー										
	英字	英数字、英数字編集項目	数字、数字編集	BOOLEAN (6)	DBCS (8)	外部浮動小数点 (6)	内部浮動小数点 (6)	日付 (6)	時刻 (6)	タイム・スタンプ (6)	国別 (6)
英字および SPACE	可	可	不可	不可	不可	不可	不可	不可	不可	不可	可
英数字 (1)	可	可	可	可 (5)	不可	可 (9)	可 (9)	可	可	可	可
英数字編集項目	可	可	不可	不可	不可	不可	不可	可	可	可	不可
数字の整数 (2)	不可	可	可	不可	不可	可	可	可	可	可	可
数字非整数 (3)	不可	不可	可	不可	不可	可	可	不可	不可	不可	不可
数字編集	不可	可	可	不可	不可	可	可	可	可	可	不可
LOW/HIGH-VALUE、QUOTES	不可	可	不可	不可	不可	不可	不可	不可	不可	不可	不可
ZERO	不可	可	可	可	不可	可	可	不可	不可	不可	可
ブール (4) (6)	不可	可	不可	可	不可	不可	不可	不可	不可	不可	不可
DBCS (6) (7) (8) DBCS 編集	不可	不可	不可	不可	可	不可	不可	不可	不可	不可	可
浮動小数点 (10)	不可	不可	可	不可	不可	可	可	不可	不可	不可	不可
日付 (6)	不可	可	可	不可	不可	不可	不可	可	不可	可	不可
時刻 (6)	不可	可	可	不可	不可	不可	不可	不可	可	可	不可
タイム・スタンプ (6)	不可	可	可	不可	不可	不可	不可	可	可	可	不可
国別 (6)	可	可	YES (11)	不可	可	不可	不可	不可	不可	不可	可

表 7-12 に関する注:

(1) 非数字リテラルを含みます。

(2) 整数の数字リテラルを含みます。

(3) 非整数の数字リテラルを含みます。

(4) ブール・リテラルを含みます。

(5) 送り出し項目の先頭の文字は、その値に関係なく移動されます。

(6) ブール項目、DBCS 項目、DBCS 編集項目、国別項目、内部浮動小数点項目、外部浮動小数点項目、および日時項目は、IBM 拡張です。

(7) DBCS リテラルおよび SPACE を含みます。

(8) DBCS データ項目を含みます。

(9) 表意定数および非数字リテラルは、数字のみから構成されていなくてはならず、数字整数フィールドとして取り扱われます。 ALL リテラルは、送り出し項目として使用することはできません。

(10) 浮動小数点リテラル、外部浮動小数点データ項目 (USAGE DISPLAY)、および内部浮動小数点データ項目 (USAGE COMP-1 または USAGE COMP-2) を含みます。

(11) 国別データ項目は、数字編集項目に移動できませんが、数字項目に移動できます。

グループ移動

グループ移動とは、送り出し項目と受け入れ項目の片方または両方がグループ項目である場合の移動のことです。グループ移動は、データの内部表現がある形式から別の形式に変換されない点を除き、英数字基本項目の移動とまったく同じように取り扱われます。グループ移動では、送り出し項目や受け入れ項目に含まれる個々の基本項目については考慮せずに、受け入れ項目が埋め込まれます。すべてのグループ移動が有効です。

IBM Extension

MOVE ステートメントおよびポインターに関する以下の説明では、*ポインター* はポインター・データ項目 (USAGE POINTER) とプロシージャー・ポインター・データ項目 (USAGE PROCEDURE-POINTER) の両方を指します。

ポインターは、MOVE ステートメント内で参照されるグループの一部にすることができます。

ポインターの移動が起こるのは、以下のすべての条件が満たされる場合です。

- MOVE ステートメントの送り出しまたは受け入れ項目にポインターが含まれている。
- 両方の項目が少なくとも 16 バイトの長さであり、正しく位置合わせされている。
- 両方が英数字またはグループ項目である。

移動される項目が 01 レベルの項目、または 01 レベル構造の一部であれば、それらの項目は 16 バイトの境界に対して同じオフセットに位置していなければなりません。作業用ストレージ内の 01 レベルの項目はすべて、16 バイトの境界で位置合わせされています。

ポインターの位置合わせの詳細は 6-100 ページの『ポインターの位置合わせ』を参照してください。

ポインターは、MOVE CORRESPONDING ステートメント内で参照されるグループの一部にすることはできますが、ポインターの移動は起こりません。

End of IBM Extension

WHEN-COMPILED 特殊レジスター

IBM Extension

この特殊レジスターには、コンパイルの開始日付が入ります。このレジスターは、次の暗黙定義および形式を持つ英数字データから構成されます。

```
01     WHEN-COMPILED GLOBAL PICTURE X(16) USAGE DISPLAY
```

形式

```
MM/DD/YYhh.mm.ss (MONTH/DAY/YEARhour.minute.second)
```

例えば、コンパイルが 1994 年 12 月 15 日の午後 2 時 4 分に開始された場合には、WHEN-COMPILED に値 12/15/9414.04.00 が入ります。

ジョブ関連コマンド (CHGJOB など) の DATSEP または TIMSEP パラメーターは、WHEN-COMPILED 特殊レジスター内で使用される日付区切りまたは時刻分離文字を指定します。 DATFMT パラメーターは、WHEN-COMPILED 特殊レジスター内で使用される日付形式を指定します。

これは MOVE ステートメントの送信項目についてのみ有効です。

特殊レジスターのデータは、送り出しデータ項目として使用されているときにのみ参照変更できます。

ネストされたプログラムでは、この特殊レジスターは最外部のプログラム内で暗黙に定義されます。

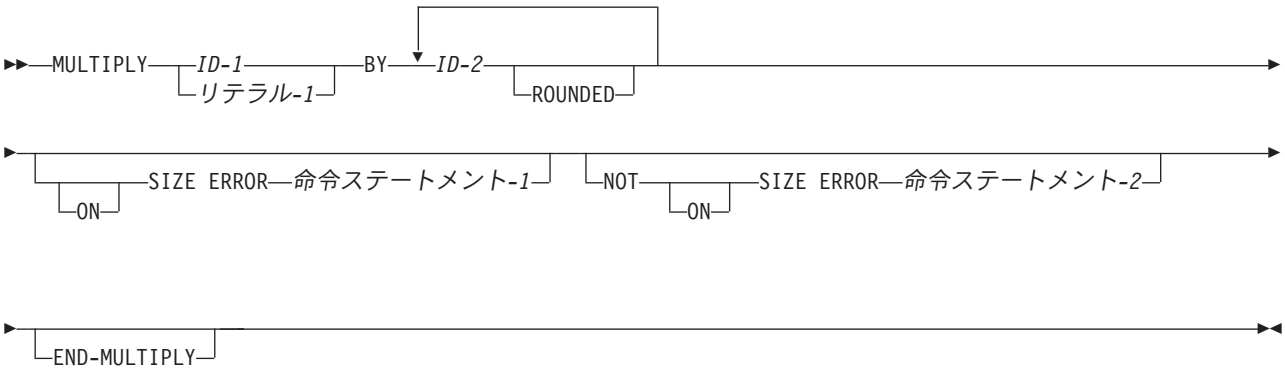
注: 日付および時刻の組み込み関数 WHEN-COMPILED を使用して、コンパイル日付および時刻をアクセスすることもできます (7-325 ページの『WHEN-COMPILED』を参照してください)。この関数は 4 桁の年の値をサポートし、追加情報を提供します。

End of IBM Extension

MULTIPLY ステートメント

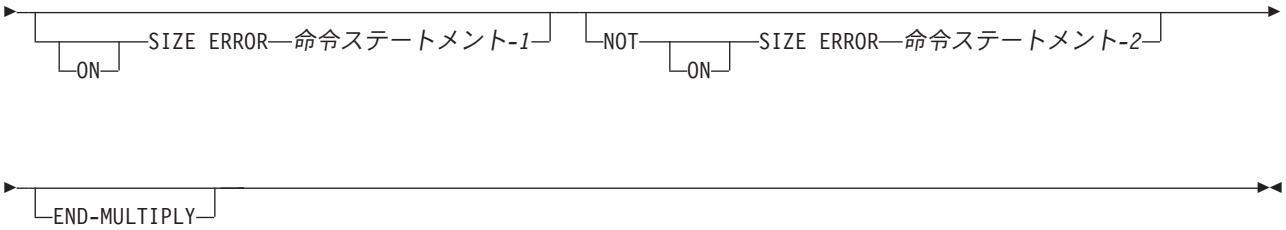
MULTIPLY ステートメントは、数字項目を乗算し、結果を保管します。

MULTIPLY ステートメント - 形式 1



形式 1 では、ID-1 またはリテラル-1 の値が保管されます。この値は、ID-2 が指定されている順序で左から右に、それぞれの ID-2 によって乗算され、保管されます。

MULTIPLY ステートメント - 形式 2 - GIVING



形式 2 では、ID-1 またはリテラル-1 の値が ID-2 またはリテラル-2 の値によって乗算されます。その後、積が、ID-3 によって参照されているそれぞれのデータ項目に保管されます。

すべての形式において、以下が適用されます。

ID-1、ID-2
基本数字項目でなければなりません。

リテラル
数字リテラルでなければなりません。

ID-3
基本数字項目または基本数字編集項目でなければなりません。

オペランドの合成は、すべての受信データ項目を重ね合わせることによって決定されます。オペランドの合成に関する詳細は 7-34 ページの『オペランドのサイズ』を参照してください。

IBM Extension

浮動小数点データ項目および浮動小数点リテラルは、数字データ項目または数字リテラルが指定できるところではどこでも使用できます。

End of IBM Extension

注: MULTIPLY の実行中に生成される中間結果は、システム固有なものであり、プログラムの可搬性に影響する可能性があります。

ROUNDED 句

形式 1 および形式 2 については 7-33 ページの『ROUNDED 句』を参照してください。

SIZE ERROR 句

形式 1 および形式 2 については 7-33 ページの『SIZE ERROR 句』を参照してください。

END-MULTIPLY 句

この明示範囲終了符号は、MULTIPLY ステートメントの範囲を区切る働きをします。 END-MULTIPLY は、MULTIPLY 条件ステートメントを命令ステートメントに変換します。この句を使用することによって、MULTIPLY 条件ステートメントを別の条件ステートメントにネストすることができます。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

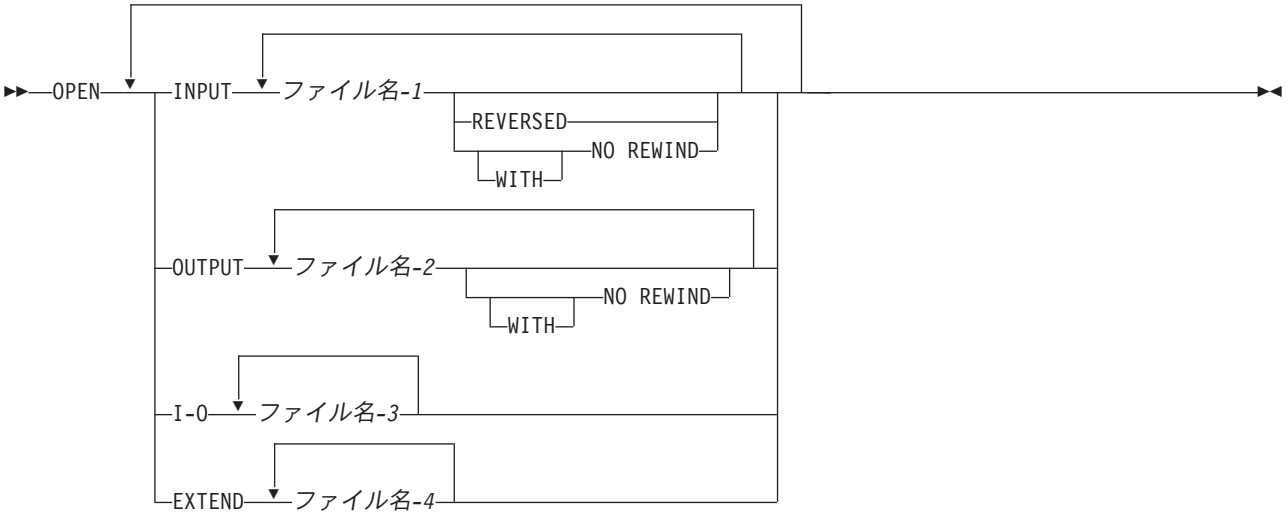
OPEN ステートメント

OPEN ステートメントは、ファイルの処理を開始し、 ラベルの検査または書き込みを行います。

OPEN ステートメントの形式は、ファイルのタイプによって異なります。

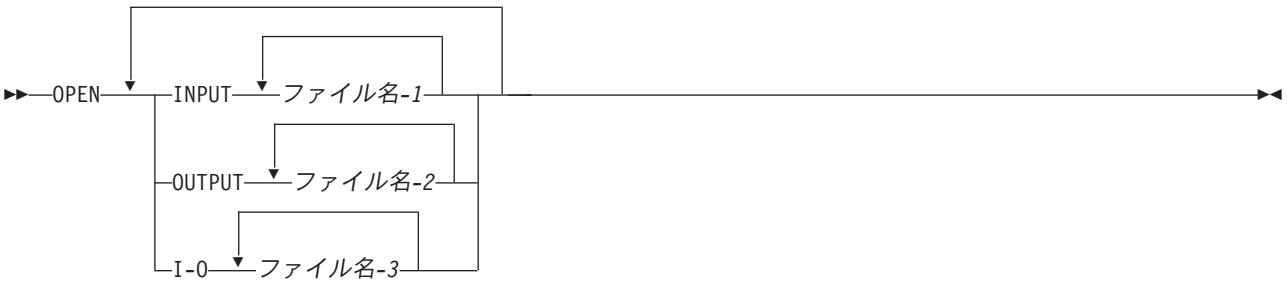
OPEN ステートメント - 形式 1 - 順次

OPEN ステートメント - 形式 1 - 順次



OPEN ステートメント - 形式 2 - 索引付きおよび相対

OPEN ステートメント - 形式 2 - 索引付きおよび相対



相対ファイルの場合だけ、次の論理ファイル・メンバーに対しては OPEN ステートメントを使用できません。

- 複数の物理ファイルに基づくもの
- 選択または省略の論理を含むもの

OPEN ステートメント - 形式 3 - TRANSACTION

IBM Extension

OPEN ステートメント - 形式 3 - TRANSACTION

OPEN ステートメントを使用することにより、プログラム装置を TRANSACTION ファイル用に暗黙に獲得できます。プログラム装置の獲得に関する詳細は 7-64 ページの『ACQUIRE ステートメント』を参照してください。

End of IBM Extension

INPUT
ファイルを入力操作用にオープンするために使用します。

FORMATFILE またはプリンター・ファイルには使用できません。

OUTPUT
ファイルを出力操作用にオープンするために使用します。この句を使用すると、順次および相対 DISK ファイルが存在せず、CRTF オプションが指定されている場合には、それらのファイルが動的に作成されます。ファイルが OUTPUT 用にオープンされるときには、レコードを含みません。

既存のレコードは、物理ファイルの場合にだけ除去 (消去) されます。論理ファイルの場合には、EXTEND が指定されているかのようにファイルが扱われます。

I-O
ファイルを入力と出力の両方の操作用にオープンするために使用します。 I-O 句は、DISK、DATABASE、およびワークステーション・ファイルなど、直接アクセス装置に割り当てられているファイルについてだけ指定できます。

EXTEND
ファイルを出力操作用にオープンするために使用します。

EXTEND 句は、複数ファイル・リールに対して指定してはなりません。

EXTEND 句は、次のものについては使用できません。

- FORMATFILE ファイル
- プリンター・ファイル
- DISKETTE ファイル

ファイル名-1、ファイル名-2、ファイル名-3、ファイル名-4
OPEN ステートメントが作用する対象のファイルを指定します。複数のファイルを指定する場合、これらのファイルの編成およびアクセスが同じである必要はありません。各ファイル名はデータ部の FD 記入項目の中で定義されていなければならず、ファイル名としてソート・ファイルまたはマージ・ファイルの名前を指定することはできません。 FD 記入項目は、ファイルの定義時に指定された情報と同じでなければなりません。

REVERSED
順次の単一リール・テープ・ファイルについてだけ有効です。

NO REWIND
順次の単一リール・テープ・ファイルについてだけ有効です。

OPEN ステートメントに関する考慮事項

OPEN ステートメントの実行が正常に行われると、ファイルの可用性が決まり、そのファイルがオープン・モードになります。 OPEN 操作が失敗すると、そのファイルは使用できなくなります。ファイルが物理的に存在していて、入出力制御システムによって認識されている場合には、そのファイルは使用可能です。 7-156 ページの『OPEN ステートメントのプログラミング上の注意事項』に、使用可能および使用不可なファイルのオープンの結果を示します。

動的ファイル作成: OPEN ステートメントで作成されない限りは使用できないファイルが、OPEN ステートメントで作成される場合があります。この機能は、*動的ファイル作成* と呼ばれます。

IBM Extension

ILE COBOL では、動的ファイル作成は DISK に割り当てられているファイルに対してのみ、発生します。さらに、OPTION(*CRTF) が CRTCBLMOD コマンドまたは CRTBNDCBL コマンドで指定されているか、あるいは CRTF オプションが PROCESS ステートメントに含まれていなくてはなりません。OPTION(*NOCRTF) または PROCESS NOCRTF が指定されているか、あるいはオプションが定義されていない場合は、プログラムで定義されているいずれのファイルも動的に作成されません。

動的ファイル作成が指定されていた場合、次のタイプのファイルが OPEN ステートメントの実行時に存在しなければ、それらが作成されます。

- OUTPUT 用にオープンされた順次ファイルおよび相対ファイル。
- I-O 用にオープンされた順次ファイルおよび相対ファイル (オプショナル)。
- EXTEND でオープンされた順次ファイル (オプショナル)。

オプショナル・ファイルは、SELECT OPTIONAL 文節を使用して定義されているファイルです。コンパイル時エラー・メッセージが、オプショナル・ファイルに対する OPEN I-O ステートメントまたは OPEN EXTEND ステートメント用に発行されますが、ただし、動的ファイル作成がそのファイルに対して有効でない場合です。

動的に作成されたファイルのデフォルトの属性は、ライブラリー QSYS 中に保留されているファイル QAXXDBF のデフォルトの属性に基づきます。 CHGPF コマンドは、これらの属性を変更するために使用できます。例えば、レコードの最大数を増やしたり、レコードの待機時間を減らしたりすることができます。

ファイル指定変更によってライブラリー名が提供されている場合、そのライブラリーにファイルが作成されます。いずれのファイル指定変更も有効でない場合は、ファイルは現行ライブラリーに作成され、また現行のライブラリーが定義されていない場合は、ライブラリー QTEMP 中に作成されます。

動的に作成できるファイルの最大レコード長は、32,766 文字です。

End of IBM Extension

装置タイプ DATABASE に関する特別の考慮事項:

IBM Extension

ファイルをコミットメント制御下に置くことができます。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の『コミットメント制御』を参照してください。

ファイルにヌル可能フィールドが含まれ、 ALWNULL がそのファイルの ASSIGN 文節上で指定されなかった場合は、ファイル状況が 0P に設定され、ヌル・フィールドの入っていないレコードだけが処理されます。一方 ALWNULL がヌル可能ファイルに指定された場合は、ヌル・フィールドは処理可能で、ファイル状況は 0P に設定されません。

End of IBM Extension

INPUT 句 (順次ファイル)

ファイルは、入力操作用にオープンされます。 ファイル位置標識は、ファイル内の最初のレコードに設定されます。ファイル内にレコードが存在しない場合、ファイル位置標識は、最初の順次 READ ステートメントの実行により AT END 条件が起こるように設定されます。

ファイル制御記入項目の中に SELECT OPTIONAL を指定しておくと、OPEN ステートメントの処理によって、プログラムは実行時にこのファイルが存在するかどうかを検査します。ファイルが存在しない場合、ファイルに対する最初の READ ステートメントによって、AT END 条件が起こります。

OPTION(*NOBLK) オプションが指定されている場合には、コンパイラーは、以下の条件が満たされる場合に出力レコードをブロック化し、入力レコードを非ブロック化するためのコードを生成します。

- ファイル・アクセスが順次である。
- ファイルの編成が順次であり、ファイルが入力または出力専用にオープンされる。
- ファイルが DISK、DATABASE、DISKETTE、または TAPEFILE に割り当てられている。

BLOCK CONTAINS 文節は、テープ・ファイル以外のファイルについてのブロック化因数を制御しません。 BLOCK CONTAINS 文節は、すべてのファイルについてのブロック化因数を制御します。

装置タイプ DATABASE、TAPEFILE、および DISKETTE に関する特別の考慮事項: ファイル制御記入項目に SELECT OPTIONAL が指定されており、CRTCBLMOD または CRTBNDCBL コマンドに OPTION(*CRTF) が指定されている場合には、この組み合わせは無効です。

装置タイプ DISK および DATABASE に関する特別の考慮事項:

IBM Extension

プログラムにとって最初に使用可能になるレコードは、プログラムの実行時に、OVRDBF CL コマンドで POSITION パラメーターを使用することによって指定できます。このコマンドについて詳しくは、**IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL および API*』を参照してください。

End of IBM Extension

OUTPUT 句 (順次ファイル)

ファイルは、出力操作だけを認めるようにオープンされます。ファイルが正常にオープンされたときには、レコードは入っていません。

OPTION(*NOBLK) が指定されている場合には、コンパイラーは、以下の条件が満たされる場合に出力レコードをブロック化し、入力レコードを非ブロック化するためのコードを生成します。

- ファイル・アクセスが順次である。
- ファイルの編成が順次であり、ファイルが入力または出力専用にオープンされる。
- ファイルが DISK、DATABASE、DISKETTE、または TAPEFILE に割り当てられている。

BLOCK CONTAINS 文節は、テープ・ファイル以外のファイルについてのブロック化因数を制御しません。

装置タイプ FORMATFILE および PRINTER は、出力用にだけオープンできます。

装置タイプ DISK に関する特別の考慮事項: 動的ファイル作成が指定されていた場合、ファイルがすでに使用可能でなければ、OPEN ステートメントはそのファイルを作成します。

装置タイプ DISK、DATABASE、および FORMATFILE に関する特別の考慮事項:

IBM Extension

OUTPUT 用にオープンされる場合には、物理ファイルだけが消去されます。ファイルが正常にオープンされたときには、レコードは入っていません。論理ファイルを OUTPUT 用にオープンしようとすると、ファイルはオープンされますが、レコードは削除されません。ファイルは、EXTEND 句が指定された場合と同じ扱いになります。論理ファイルを消去するには、論理ファイルの基礎となっているすべてのメンバーを消去しなければなりません。

End of IBM Extension

I-O 句 (順次ファイル)

装置タイプ DISK および DATABASE だけが、入出力用にオープンできます。

ファイルは、入力と出力の両方の操作用にオープンされます。ファイル位置標識は、ファイル内の最初のレコードに設定されます。ファイル内にレコードが存在しない場合、ファイル位置標識は、最初の順次 READ ステートメントの実行により AT END 条件が起こるように設定されます。

IBM Extension

プログラムにとって最初に使用可能になるレコードは、プログラムの実行時に、OVRDBF CL コマンドで POSITION パラメーターを使用することによって指定できます。このコマンドについて詳しくは、**IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL* および *API*』を参照してください。

End of IBM Extension

装置タイプ DISK に関する特別の考慮事項: 動的ファイル作成が指定され、そのファイルがオプショナル・ファイル (ファイル制御記入項目中で SELECT OPTIONAL) である場合、ファイルがすでに使用可能でなければ、OPEN ステートメントはそのファイルを作成します。

NO REWIND 句 (順次ファイル)

この句は装置タイプ TAPEFILE にのみ適用されます。

OPEN ステートメントは、 ファイルの位置変更を行いません。テープは、OPEN ステートメントの処理前に、所要ファイルの先頭に位置決めされなければなりません。

リールという概念がストレージ・メディアにとって意味をもたない場合 (例えば、直接アクセス装置の場合) は、REVERSED 句および NO REWIND 句が適用されません。この状況でこれらの句が使用されると、ファイル状況が 07 に設定されます。

IBM Extension

システムは、テープ上の現在位置を記録し、自動的にテープを適切な場所に位置決めします。マルチファイル・テープ・ボリュームを処理する際には、すべての CLOSE ステートメントで NO REWIND 句を指定しなければなりません。そのテープ・ボリューム上で次のファイルがオープンされるときに、システムは、どの方向にテープを移動すれば最も効率的に所要ファイルに到達するかを判別します。

End of IBM Extension

REVERSED 句 (順次ファイル)

この句は装置タイプ TAPEFILE にのみ適用されます。

OPEN ステートメントの処理によって、ファイルがその末尾に位置決めされます。後続の READ ステートメントでは、データ・レコードが、最後のレコードから始まる逆順で使用可能になります。 REVERSED は、入力ファイルに対してのみ指定できます。

リールという概念がストレージ・メディアにとって意味をもたない場合 (例えば、直接アクセス装置の場合) は、REVERSED 句および NO REWIND 句が適用されません。この状況でこれらの句が使用されると、ファイル状況が 07 に設定されます。

EXTEND 句 (順次ファイル)

装置タイプ TAPEFILE、DISK、および DATABASE は、EXTEND としてオープンすることができます。

EXTEND 句は、 ファイルを出力の操作用にオープンするために使用します。 OPEN EXTEND ステートメントの処理により、ファイルがレコードの追加に向けて準備されます。追加レコードは、ファイル内の最後のレコードのすぐ後に続きます。後続の WRITE ステートメントでは、ファイルが OUTPUT 用にオープンされた場合と同様にレコードが追加されます。 EXTEND 句は、ファイルの作成中に指定できます。

装置タイプ DISK に関する特別の考慮事項: 動的ファイル作成が指定され、そのファイルがオプショナル・ファイル (ファイル制御記入項目中で SELECT OPTIONAL) である場合、ファイルがすでに使用可能でなければ、OPEN ステートメントはそのファイルを作成します。

INPUT 句 (索引付きおよび相対ファイル)

ファイルは、入力操作用にオープンされます。 ファイル位置標識は、ファイル内の最初のレコードに設定されます。ファイル内にレコードが存在しない場合、ファイル位置標識は、最初の順次 READ ステートメントの実行により AT END 条件が起こるように設定されます。

順次アクセス・モードに関する特別の考慮事項:

IBM Extension

プログラムにとって最初に使用可能になるレコードは、プログラムの実行時に、OVRDBF CL コマンドで POSITION パラメーターを使用することによって指定できます。このコマンドについて詳しくは、**IBM i**

Information Center (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL および API*』を参照してください。

End of IBM Extension

OPTION(*NOBLK) が指定されている場合には、コンパイラーは、以下の条件が満たされる場合に出力レコードをブロック化し、入力レコードを非ブロック化するためのコードを生成します。

- ファイル・アクセスが順次である。
- ファイルの編成が索引付きであり、ファイルが入力または出力専用にオープンされるか、ファイルの編成が相対であり、ファイルが入力専用にオープンされる。
- ファイルが DISK または DATABASE に割り当てられている。
- ファイルについて START ステートメントが指定されていない。

BLOCK CONTAINS 文節は、ブロック化因数を制御しません。

START ステートメントは、OPTION(*BLK) と BLOCK CONTAINS 文節の両方を指定する場合に使用できます。BLOCK CONTAINS 文節は、すべてのファイルについてのブロック化因数を制御します。

動的アクセス・モードに関する特別の考慮事項:

IBM Extension

プログラムにとって最初に使用可能になるレコードは、プログラムの実行時に、OVRDBF CL コマンドで POSITION パラメーターを使用することによって指定できます。このコマンドについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

End of IBM Extension

また、OPTION(*BLK) が指定されており、BLOCK CONTAINS 文節が指定されていると、コンパイラーは、以下の条件が満たされる場合に出力レコードをブロック化し、入力レコードを非ブロック化するためのコードを生成します。

- ファイル・アクセスが動的である。
- ファイルの編成が索引付きであり、ファイルが入力または出力専用にオープンされるか、ファイルの編成が相対であり、ファイルが入力専用にオープンされる。
- ファイルが DISK または DATABASE に割り当てられている。

BLOCK CONTAINS 文節でゼロのレコード・サイズを指定すると、システムのデフォルトのブロック化因数が適用されます。

OUTPUT 句 (索引付きおよび相対ファイル)

IBM Extension

OUTPUT 用にオープンされる場合には、物理ファイルだけが消去されます。ファイルが正常にオープンされたときには、レコードは入っていません。論理ファイルを OUTPUT 用にオープンしようとすると、ファイルはオープンされますが、レコードは削除されません。論理ファイルを消去するには、論理ファイルの基礎となっているすべてのメンバーを消去しなければなりません。

End of IBM Extension

相対ファイルに関する特別の考慮事項 - 装置タイプ DISK: 動的ファイル作成が指定されていた場合、ファイルがすでに使用可能でなければ、OPEN ステートメントはそのファイルを作成します。

索引付きファイル関する特別の考慮事項 - 順次アクセス: OPTION(*NOBLK) が指定されている場合には、コンパイラーは、以下の条件が満たされる場合に出力レコードをブロック化し、入力レコードを非ブロック化するためのコードを生成します。

- ファイル・アクセスが順次である。
- ファイルの編成が索引付きであり、ファイルが入力または出力専用にオープンされる。
- ファイルが DISK または DATABASE に割り当てられている。
- ファイルについて START ステートメントが指定されていない。

BLOCK CONTAINS 文節は、ブロック化因数を制御しません。

OPTION(*BLK) と BLOCK CONTAINS 文節の両方を指定すると、ブロック化因数が適用されます。

索引付きファイルに関する特別の考慮事項 - 動的アクセス: OPTION(*BLK) が指定されており、BLOCK CONTAINS 文節が指定されていると、コンパイラーは、以下の条件が満たされる場合に出力レコードをブロック化し、入力レコードを非ブロック化するためのコードを生成します。

- ファイル・アクセスが動的である。
- ファイルの編成が索引付きであり、ファイルが入力または出力専用にオープンされる。
- ファイルが DISK または DATABASE に割り当てられている。

BLOCK CONTAINS 文節でゼロのレコード・サイズを指定すると、システムのデフォルトのブロック化因数が適用されます。

I-O 句 (索引付きおよび相対ファイル)

ファイルは、入力と出力の両方の操作用にオープンされます。ファイル位置標識は、ファイル内の最初のレコードに設定されます。ファイル内にレコードが存在しない場合、ファイル位置標識は、最初の順次 READ ステートメントの実行により AT END 条件が起こるように設定されます。

相対ファイルに関する特別の考慮事項 - 装置タイプ DISK: 動的ファイル作成が指定され、そのファイルがオプショナル・ファイル (ファイル制御記入項目中で SELECT OPTIONAL) である場合、ファイルがすでに使用可能でなければ、OPEN ステートメントはそのファイルを作成します。

順次または動的アクセス・モードに関する特別の考慮事項:

IBM Extension

プログラムにとって最初に使用可能になるレコードは、プログラムの実行時に、OVRDBF CL コマンドで POSITION パラメーターを使用することによって指定できます。このコマンドについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

End of IBM Extension

OPEN ステートメントのプログラミング上の注意事項

OPEN ステートメントの実行が正常に行われると、ファイルの使用可能性が決まり、そのファイルがオープン・モードになります。 7-157 ページの表 7-13 に、ファイルが使用可能な場合と使用不可の場合のオープンの結果を要約しています。

表 7-13. ファイルの使用可能性

	ファイルが使用可能である場合	ファイルが使用不可である場合
INPUT	正常なオープン	オープンが失敗する
INPUT (オプショナル・ファイル)	正常なオープン	正常なオープン。最初の読み取りによって AT END 条件が発生する。
I-O	正常なオープン	オープンが失敗する
I-O (オプショナル・ファイル)	正常なオープン	オープンによってファイルが作成される (1)
OUTPUT	正常なオープン。ファイルにはレコードが入っていない	オープンによってファイルが作成される (1)
EXTEND	正常なオープン	オープンが失敗する
EXTEND (オプショナル・ファイル)	正常なオープン	オープンによってファイルが作成される (1)
注: (1) 動的ファイル作成が指定され、ファイルが適切に編成されている場合、そのファイルが作成されます。7-151 ページの『動的ファイル作成』を参照してください。		

1. OPEN ステートメントの正常な実行により、関連するレコード域がプログラムで使用可能になります。最初のデータ・レコードは取得または解放されません。
2. OPEN ステートメントは、実行可能なあらゆる入出力ステートメント (USING または GIVING 句を指定した SORT または MERGE ステートメントを除く) の実行に先立って、正常に実行されなければなりません。
3. READ ステートメントは、INPUT または I-O 用にオープンされているファイルに対して実行されます。
4. WRITE ステートメントは、OUTPUT または EXTEND 用にオープンされているファイル (順次ファイルだけ) に対して実行されます。 WRITE ステートメントは、I-O 用にオープンされているランダムまたは動的アクセス・モードの索引付きまたは相対ファイル、および I-O 用にオープンされている TRANSACTION ファイルに対しても実行されます。
5. REWRITE ステートメントは、I-O 用にオープンされているファイルに対して実行されます。
6. START ステートメントは、INPUT または I-O 用にオープンされている索引付きまたは相対ファイルに対して実行されます。
7. DELETE ステートメントは、I-O 用にオープンされている索引付きまたは相対ファイルに対して実行されます。
8. 1 つのファイルを同じプログラムの中で INPUT、OUTPUT、I-O、または EXTEND (順次ファイルだけ) 用にオープンすることができます。あるファイルについて OPEN ステートメントを最初に実行した後では、後続のそれぞれの OPEN ステートメントを実行する前に、REEL または UNIT 句 (順次ファイルの場合だけ) または LOCK 句を指定しない CLOSE ファイル・ステートメントが正常に実行されなければなりません。
9. FILE-CONTROL 記入項目に FILE STATUS 文節が指定されている場合は、OPEN ステートメントの実行時に関連する状況キーが更新されます。状況キーの詳細は、 7-37 ページの『共通の処理機能』を参照してください。
10. すでにオープン状況にあるファイルに対して OPEN ステートメントが出されると、このファイルに対する EXCEPTION/ERROR プロシージャー (指定されている場合) が実行され、ファイル状況 41 が戻されます。

PERFORM ステートメント

PERFORM ステートメントは、明示的に 1 つまたは複数のプロシージャーに制御を転送し、指定されたプロシージャーまたは命令ステートメントの実行完了後に、制御を暗黙に次の実行可能ステートメントに戻します。

次の 2 種類の PERFORM ステートメントがあります。

行外 PERFORM ステートメント
プロシージャー名-1 が指定されます。

行内 PERFORM ステートメント
プロシージャー名-1 が省略されます。

行内 PERFORM ステートメントは、END-PERFORM 句で区切らなければなりません。

行内形式と行外形式を組み合わせることはできません。例えば、プロシージャー名-1 を指定する場合には、命令ステートメントおよび END-PERFORM 句を指定することはできません。

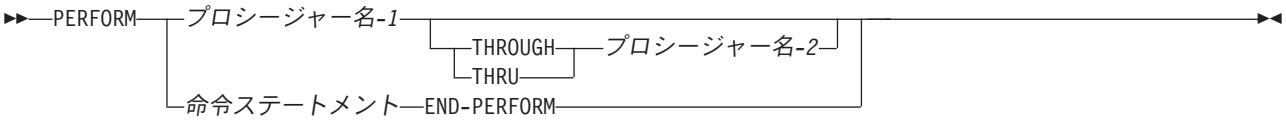
PERFORM ステートメントには次の 4 つの形式があります。

- 形式 1 - 基本 PERFORM
- 形式 2 - TIMES 句を指定する PERFORM
- 形式 3 - UNTIL 句を指定する PERFORM
- 形式 4 - VARYING 句を指定する PERFORM

基本 PERFORM ステートメント

基本 PERFORM ステートメント内で参照されるプロシージャーは一度実行され、その後、PERFORM ステートメントに続く次の実行可能ステートメントに制御が渡されます。

PERFORM ステートメント - 形式 1



プロシージャー名
手続き部にあるセクションまたは段落でなければなりません。

プロシージャー名-1 とプロシージャー名-2 の両方を指定する場合、いずれか一方が宣言プロシージャー内のプロシージャー名であれば、両方が同じ宣言プロシージャー内のプロシージャー名でなければなりません。

PERFORM ステートメントを宣言セクションに入れる場合には、プロシージャー名-1 とプロシージャー名-2 も宣言セクションに入れなければなりません。

プロシージャー名-1 を指定する場合には、命令ステートメントおよび END-PERFORM 句を指定してはなりません。

プロシージャー名-1 を省略する場合には、命令ステートメントおよび END-PERFORM 句を指定しなければなりません。

命令ステートメント
行内 PERFORM について実行されるステートメント。

END-PERFORM
行内 PERFORM ステートメントの有効範囲を区切ります。行内 PERFORM の実行は、それに含まれている最後のステートメントが実行された後に完了します。

行内 PERFORM ステートメント: 行内 PERFORM ステートメントは、 プロシージャー名-1 (さらに、指定されている場合はプロシージャー名-2) の範囲内に含まれているステートメントの代わりに行内 PERFORM に含まれているステートメントが実行される点を除き、行外 PERFORM ステートメントと同じ一般規則に従って機能します。特に**行内**または**行外**という語によって限定されていない限り、行外 PERFORM ステートメントに適用される規則は、すべて行内 PERFORM にも適用されます。

行外 PERFORM ステートメント: 行外 PERFORM ステートメントが実行されるたびに、プロシージャー名-1 に指定されたプロシージャーの最初のステートメントに制御が転送されます。制御は、PERFORM ステートメントに続くステートメントに必ず戻されます。この制御がどの点から戻されるかは、次のように判別されます。

- プロシージャー名-1 が段落名であり、プロシージャー名-2 が指定されていない場合には、プロシージャー名-1 段落の最後のステートメントの実行後に制御が戻されます。
- プロシージャー名-1 がセクション名であり、プロシージャー名-2 が指定されていない場合には、プロシージャー名-1 セクション内の最後の段落の最後のステートメントの実行後に制御が戻されます。
- プロシージャー名-2 が指定されていて、それが段落名である場合には、プロシージャー名-2 段落の最後のステートメントの実行後に制御が戻されます。
- プロシージャー名-2 が指定されていて、それがセクション名である場合には、プロシージャー名-2 セクション内の最後の段落の最後のステートメントの実行後に制御が戻されます。

プロシージャー名-1 とプロシージャー名-2 の間に必要なのは、連続した一連の操作が、プロシージャー名-1 に指定されたプロシージャーから開始され、プロシージャー名-2 に指定されたプロシージャーで終了されるという関係だけです。

ネストされた PERFORM ステートメント: PERFORM ステートメントを、実行されるプロシージャー内に指定できます。戻り点への論理パスが複数ある場合には、プロシージャー名-2 に EXIT ステートメントだけで構成される段落を指定できます。そのようにする場合には、戻り点へのすべてのパスをこの段落に導かなくてはなりません。

プロシージャー名-1 とプロシージャー名-2 の両方を指定する場合には、GO TO および PERFORM ステートメントをこれらの段落またはセクションに含まれる一連のステートメント内に指定できます。 GO TO ステートメントでは、プロシージャー名-1 とプロシージャー名-2 の範囲外のプロシージャー名を参照してはなりません。それを行うと、結果は予測できないものとなり、診断もできなくなります。

プロシージャー名-1 だけを指定する場合には、PERFORM および GO TO ステートメントをそのプロシージャー内に指定できます。 GO TO ステートメントでは、プロシージャー名-1 の範囲外のプロシージャー名を参照してはなりません。それを行うと、結果は予測できないものとなり、診断もできなくなります。

実行されるプロシージャーに別の PERFORM ステートメントが含まれている場合、組み込まれた PERFORM ステートメントと関連する一連のプロシージャーは、最初の PERFORM ステートメントの実行されるプロシージャーに全体が含まれているか、さもなければ全体が範囲外になければなりません。すなわち、他の活動状態の PERFORM ステートメントの実行されるプロシージャーの範囲内から実行点が始まる PERFORM ステートメントは、他の活動状態の PERFORM ステートメントの出口点を超えて制御を渡してはなりません。さらに、そのような複数の活動状態の PERFORM ステートメントが、共通の出口をもってはなりません。

IBM Extension

複数の活動状態の PERFORM ステートメントが共通の出口点を持つことができます。

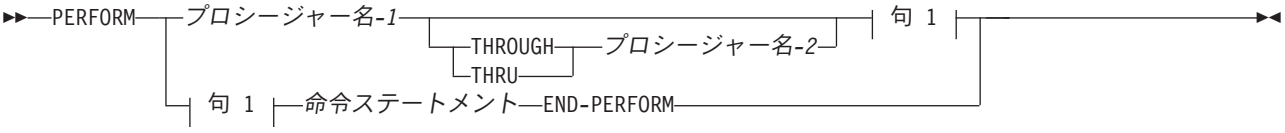
End of IBM Extension

PERFORM ステートメント以外の方法で制御が一連のプロシージャーへ渡された場合には、これらのプロシージャーを参照する PERFORM ステートメントがない場合と同じように、制御が出口点を超えて次の実行可能ステートメントへ渡されます。

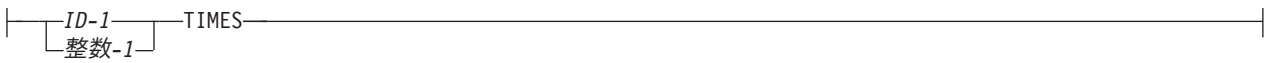
TIMES 句を指定する PERFORM

TIMES 句を指定した PERFORM ステートメント内で参照されるプロシージャーは、 ID-1 または整数-1 の値によって指定された**回数**だけ実行されます。その後、制御は、PERFORM ステートメントに続く次の実行可能ステートメントに渡されます。

PERFORM ステートメント - 形式 2



句-1:



ID-1

整数項目でなければなりません。

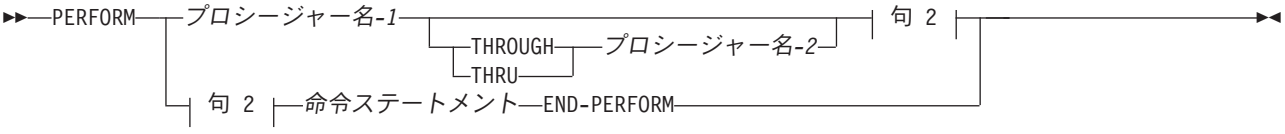
PERFORM ステートメントの開始時に ID-1 がゼロまたは負数である場合には、制御は PERFORM ステートメントに続くステートメントに渡されます。

PERFORM ステートメントの開始後には、ID-1 の値を変更しても、プロシージャーの実行回数を変える上での効果はありません。

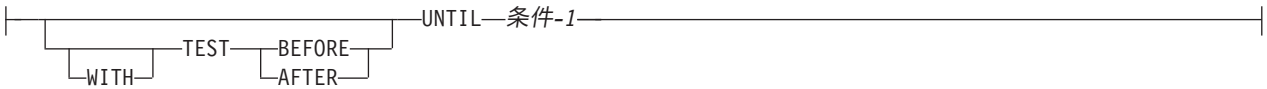
UNTIL 句を指定する PERFORM

UNTIL 句を指定する形式では、参照されるプロシージャーは、UNTIL 句で指定された条件が真になる**まで**実行されます。その後、制御は、PERFORM ステートメントに続く次の実行可能ステートメントに渡されます。

PERFORM ステートメント - 形式 3



句 2:



条件-1

7-10 ページの『条件式』で記述されている条件を指定できます。 PERFORM ステートメントの開始時に条件が真である場合には、指定されたプロシージャーは実行されません。

条件-1 に指定されたオペランドに関連する、添え字付け、参照修飾子、または関数は、その条件がテストされるたびに評価されます。

TEST BEFORE 句が指定または想定されると、ステートメントが実行される前に条件がテストされます (DO WHILE 句に相当します)。

TEST AFTER 句を指定した場合には、実行すべきステートメントは、条件がテストされる前に少なくとも 1 回実行されます (DO UNTIL 句に相当します)。

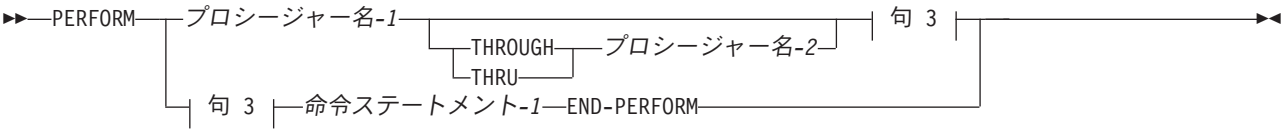
どちらの場合でも、条件が真の場合には、PERFORM ステートメントの終わりに続く次の実行可能ステートメントに制御権が移動されます。 TEST BEFORE 句も TEST AFTER 句も指定しない場合には、TEST BEFORE 句が想定されます。

VARYING 句を指定する PERFORM

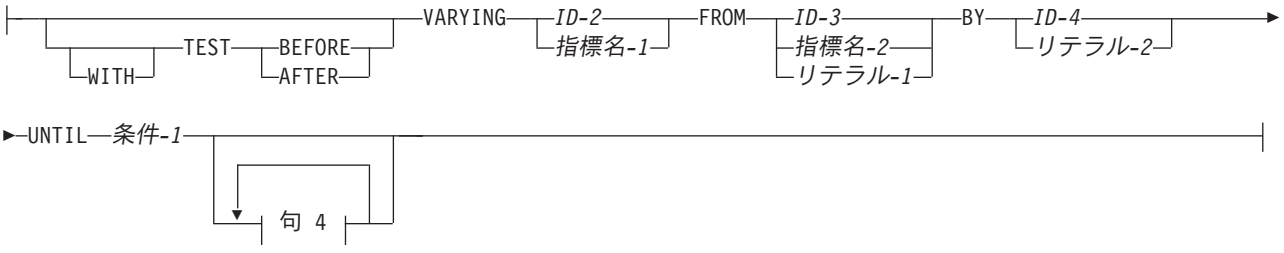
VARYING 句は、1 つまたは複数の ID または指標名の値を所定の規則に従って増大または減少させます。 (7-168 ページの『Varying 句の規則』を参照。)

VARYING 句を指定した PERFORM ステートメント (形式 4) では、7 次元テーブル全体を逐次検索できます。

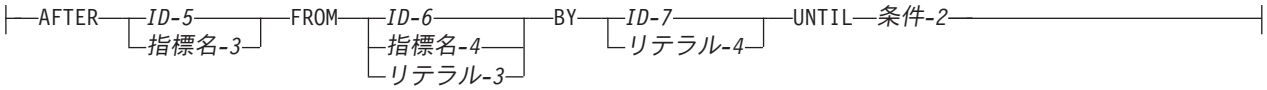
PERFORM ステートメント - 形式 4



句 3:



句 4:



条件-1、条件-2

7-10 ページの『条件式』で記述されている条件を指定できます。 PERFORM ステートメントの開始時に条件が真である場合には、指定されたプロシージャーは実行されません。

UNTIL 句内に指定された条件が満たされると、PERFORM ステートメントに続く次の実行可能ステートメントに制御権が渡されます。

条件-1 または条件-2 に指定されている任意のオペランドが、添え字付きであるか、参照変更されているか、または関数 ID である場合、その添え字、参照修飾子、または関数は、条件がテストされるたびに評価されます。

IBM Extension

浮動小数点データ項目および浮動小数点リテラルは、数字データ項目または数字リテラルが指定できるところではどこでも使用できます。

End of IBM Extension

TEST BEFORE が指定されると、指定されたすべての条件が最初の実行前にテストされます。実行すべきステートメントは、指定されたすべてのテストが失敗した場合にだけ実行されます。 TEST AFTER が指定されると、実行すべきステートメントは、条件がテストされる前に少なくとも 1 回実行されます。条件-1 に指定されたオペランドに関連する添え字付けは、条件がテストされるたびに評価されます。

TEST BEFORE 句も TEST AFTER 句も指定しない場合には、TEST BEFORE 句が想定されます。

ID の変更: オペランドを増大または減少させる方法は、指定される変数の数によって異なります。以下の説明では、ID-n という表現を使用する場合、指標名-n にも等しく当てはまります (ただし、ID-n が BY 句の対象である場合を除きます)。

ID-2 あるいは ID-5 が添え字付けされている場合、添え字は、ID によって参照されるデータ項目の内容が設定または増補されるたびに評価されます。 ID-3、ID-4、ID-6、または ID-7 が添え字付けされている場合、添え字は、ID によって参照されるデータ項目の内容が設定または増補操作に使用されるたびに評価されます。

1 つの ID の変更: このセクションでは、PERFORM ステートメントを使用して 1 つの ID を変更する方法を記述する例を示します。

例:

```
PERFORM procedure-name-1 THROUGH procedure-name-2
       VARYING identifier-2 FROM identifier-3
             BY identifier-4 UNTIL condition-1
```

1. **ID-2** が、それの開始値である ID-3 (またはリテラル-1) と等しく設定されます。
2. **条件-1** が次のように評価されます。
   a. 条件-1 が偽である場合には、ステップ 3 ～ 5 が実行されます。
   b. 条件-1 が真である場合には、制御権が PERFORM ステートメントに続くステートメントに直接渡されます。
3. **プロシージャー名-1** から**プロシージャー名-2** (指定されている場合) までのすべてが 1 回実行されます。
4. **ID-2** が ID-4 (またはリテラル-2) だけ増やされ、条件-1 が再び評価されます。
5. 条件-1 が真になるまで、ステップ 2 からステップ 4 が繰り返されます。

PERFORM ステートメントの実行終了時には、 ID-2 は、最後に使用された設定値を増加 / 減少値分超える値をもちます (ただし、PERFORM ステートメントの実行開始時に条件-1 が真であった場合には、 ID-2 に ID-3 の現行値が入ります)。

図 7-4 は、TEST BEFORE 句を使用して ID を変更する場合の PERFORM ステートメントの論理を示したものです。 7-164 ページの図 7-5 は、TEST AFTER 句を使用して ID を変更する場合の PERFORM ステートメントの論理を示したものです。

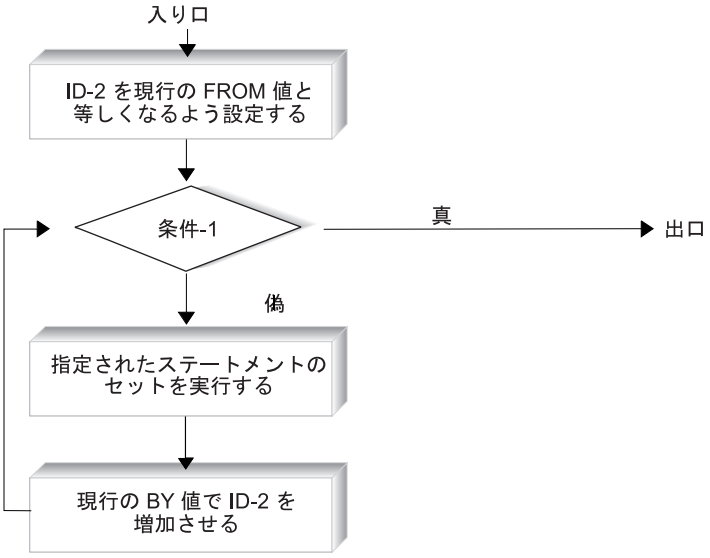


図 7-4. TEST BEFORE を使用して 1 つの ID を変更する場合

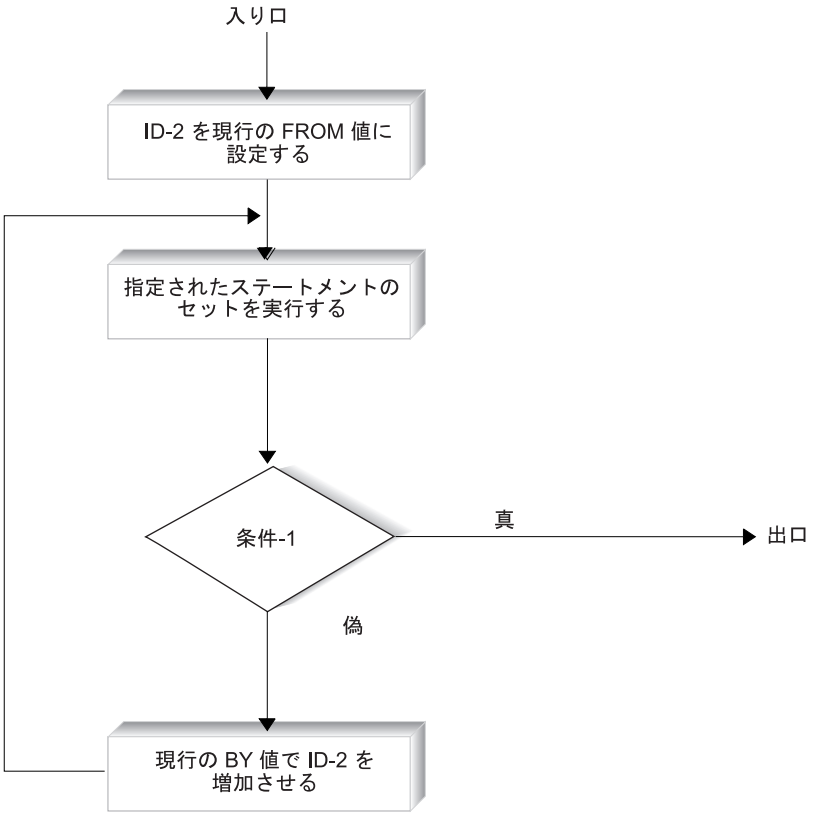


図 7-5. TEST AFTER を使用して 1 つの ID を変更する場合

2 つの ID の変更: このセクションでは、PERFORM ステートメントを使用して 2 つの ID を変更する方法を記述する例を示します。

例:

```
PERFORM procedure-name-1 THROUGH procedure-name-2
      VARYING identifier-2 FROM identifier-3
            BY identifier-4 UNTIL condition-1
         AFTER identifier-5 FROM identifier-6
            BY identifier-7 UNTIL condition-2
```

1. **ID-2** と **ID-5** が、それらの初期値である ID-3 と ID-6 にそれぞれ設定されます。
2. **条件-1** が次のように評価されます。
   a. 条件-1 が偽である場合には、ステップ 3 ～ 7 が実行されます。
   b. 条件-1 が真である場合には、制御権が PERFORM ステートメントに続くステートメントに直接渡されます。
3. **条件-2** が次のように評価されます。
   a. 条件-2 が偽である場合には、ステップ 4 ～ 6 が実行されます。
   b. 条件-2 が真である場合には、ID-2 が ID-4 だけ増やされ、ID-5 が ID-6 の現行値に設定され、ステップ 2 が繰り返されます。
4. **プロシージャー名-1** から**プロシージャー名-2** (指定されている場合) までのすべてのものが 1 回実行されます。
5. **ID-5** が ID-7 だけ増やされます。
6. 条件-2 が真になるまで、ステップ 3 からステップ 5 が繰り返されます。

7. 条件-1 が真になるまで、ステップ 2 (7-164 ページ) からステップ 6 (7-164 ページ) が繰り返されます。

PERFORM ステートメントの実行終了時には、次のようになります。

- **ID-5** には ID-6 の現行値が入ります。
- **ID-2** は、最後に使用された設定値を増大 / 減少値分超える値をもちます (ただし、PERFORM ステートメントの実行開始時に条件-1 が真であった場合には、 ID-2 に ID-3 の現行値が入ります)。

図 7-6 は、TEST BEFORE 句を使用して 2 つの ID を変更する場合の PERFORM ステートメントの論理を示したものです。 7-166 ページの図 7-7 は、TEST AFTER 句を使用して 2 つの ID を変更する場合の PERFORM ステートメントの論理を示したものです。

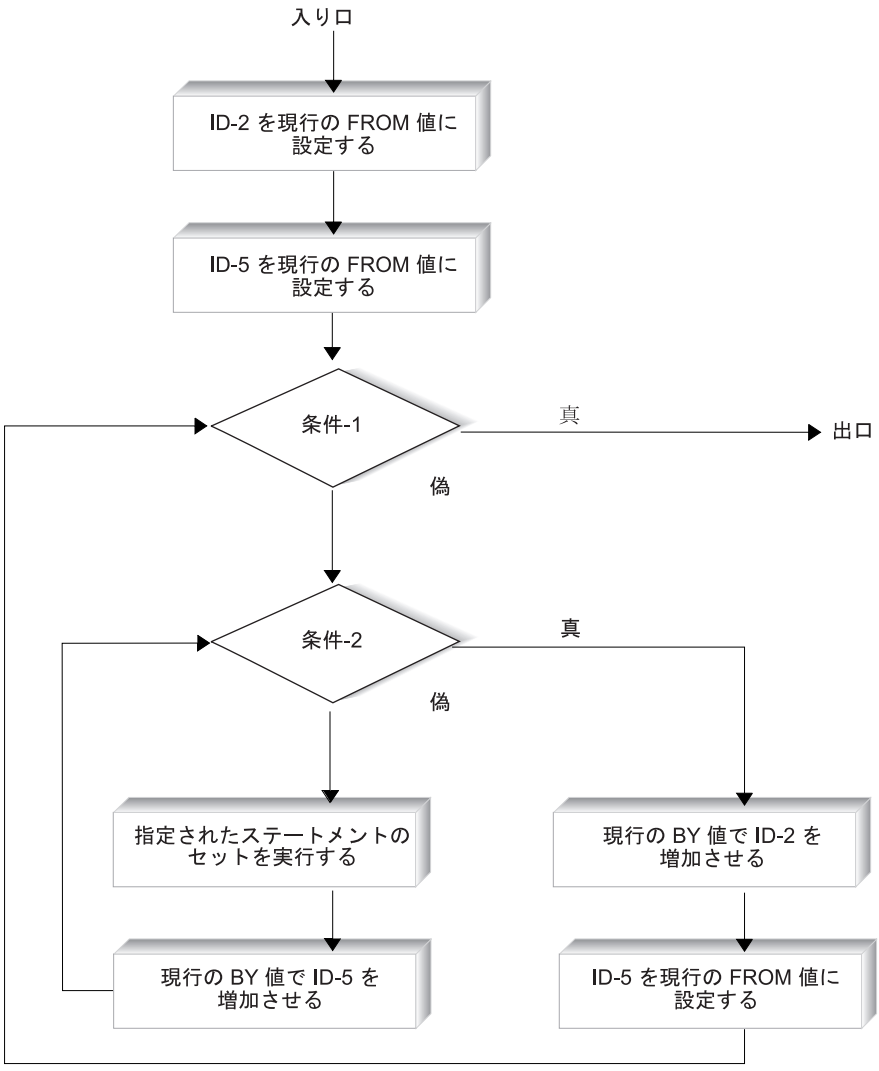


図 7-6. TEST BEFORE を使用して 2 つの ID を変更する場合

上の図では、ID-5 と ID-2 が関係をもたないことを想定してします。一方が他方に依存している場合 (例えば添え字を介して)、結果は予測できますが、一般的に望ましいものではありません。

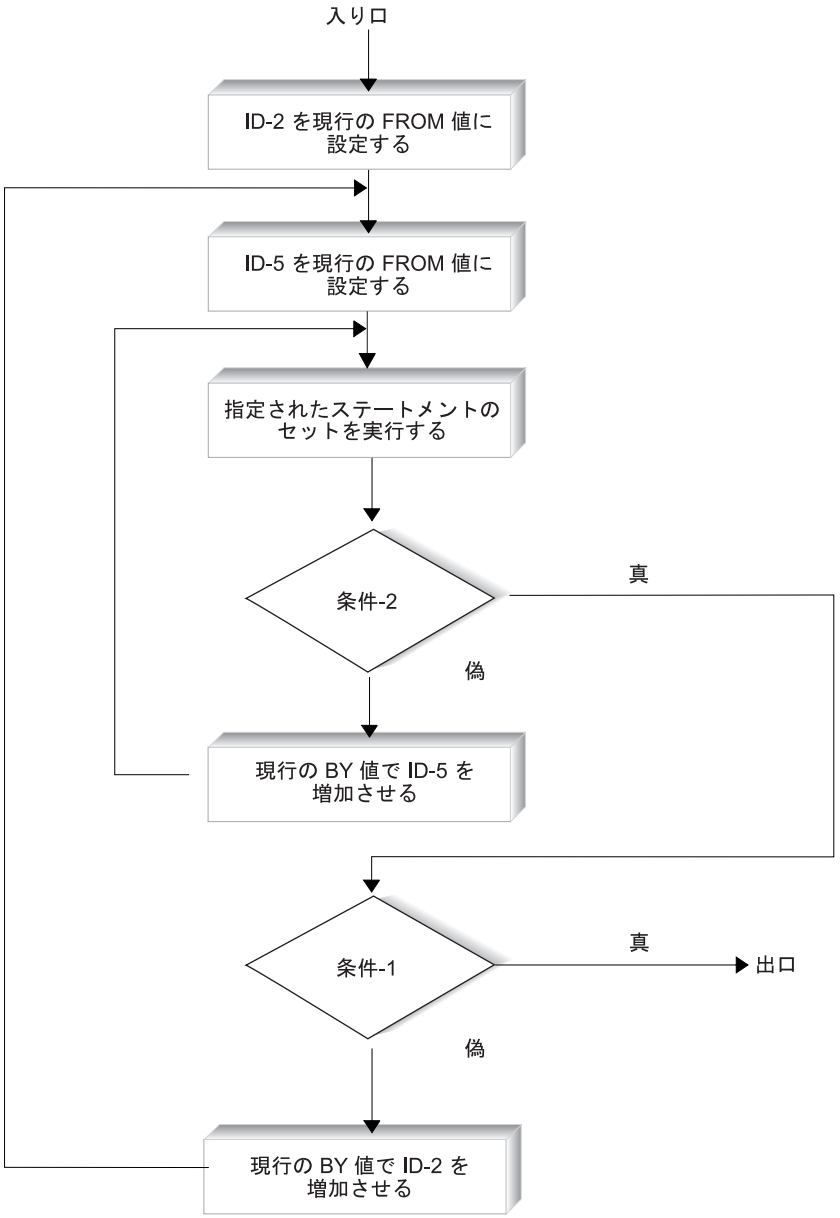


図 7-7. TEST AFTER を使用して 2 つの ID を変更する場合

3 つの ID の変更: このセクションでは、PERFORM ステートメントを使用して 3 つの ID を変更する方法を記述する例を示します。

例:

```
PERFORM procedure-name-1 THROUGH procedure-name-2
      VARYING identifier-2 FROM identifier-3
            BY identifier-4 UNTIL condition-1
          AFTER identifier-5 FROM identifier-6
            BY identifier-7 UNTIL condition-2
          AFTER identifier-8 FROM identifier-9
            BY identifier-10 UNTIL condition-3
```

処理は、次の点を除いては 2 つの ID を変更する場合と同じです。つまり、ID-5 が ID-7 だけ増やされるたびに ID-8 が完全なサイクルを通過し、次に ID-2 が変更されるたびに ID-5 が完全なサイクルを通過します。

PERFORM ステートメントの実行終了時には、次のようになります。

- **ID-5** と **ID-8** には、それぞれ ID-6 と ID-9 の現行値が入ります。
- **ID-2** は、最後に使用された設定値を増大 / 減少値分超える値をもちます (ただし、PERFORM ステートメントの実行開始時に条件-1 が真であった場合には、 ID-2 に ID-3 の現行値が入ります)。

図 7-8 は、3 つの ID を変更する場合の PERFORM ステートメントの論理を示したものです。

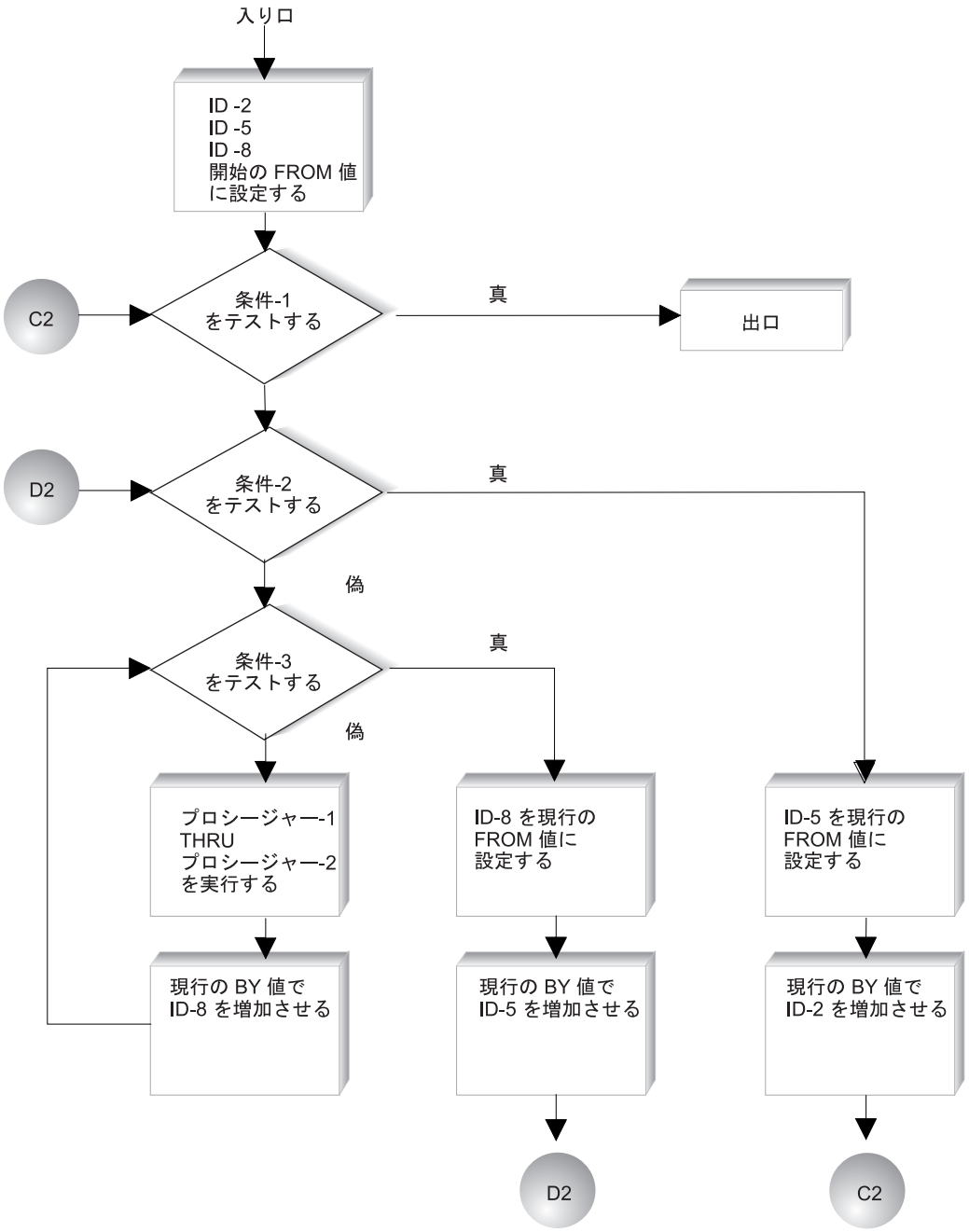


図 7-8. 形式 4 の PERFORM ステートメントの論理 - 3 つの ID を変更する場合

上の図では、ID-5 と ID-2 が関係をもたないことを想定してします。一方が他方に依存している場合 (例えば添え字を介して)、結果は予測できますが、一般的に望ましいものではありません。

上の図では、さらに、ID-8 と ID-5 が関係をもたないことも想定しています。一方が他方に依存している場合 (例えば添え字を介して)、結果は予測できますが、一般的に望ましいものではありません。

4 つ以上の ID の変更: VARYING 句を使用するときは、最高 4 つまでの AFTER 句を追加することによって、上記の例を合計 6 個の AFTER 句を持つように拡張できます。

Varying 句の規則: 指定される変数の数に関係なく、次の規則が適用されます。

1. VARYING/AFTER 句の中で指標名を指定する場合には、次のようになります。
   a. 指標名は、 6-52 ページの『INDEXED BY 句』の規則に従って初期設定され、増加または減少されます。 (7-204 ページの『SET ステートメント』も参照してください。)
   b. 関連する FROM 句では、ID は整数として記述されているものでなければならず、正の値をもたなければなりません。リテラルは正の値をもたなければなりません。
   c. 関連する BY 句では、ID は整数として記述されているものでなければなりません。リテラルはゼロ以外の整数でなければなりません。
2. FROM 句の中で指標名を指定する場合には、次のようになります。
   a. 関連する VARYING/AFTER 句では、ID は整数として記述されているものでなければなりません。これは、SET ステートメントのところで説明されているように初期設定されます。
   b. 関連する BY 句では、ID は整数として記述されているものでなければならず、ゼロ以外の値をもたなければなりません。リテラルはゼロ以外の整数でなければなりません。
3. BY 句では、ID およびリテラルはゼロ以外の値をもたなければなりません。
4. 実行時に VARYING、FROM、および BY 句における ID または指標名、あるいはその両方の値を変更すると、プロシージャーの実行回数が変わります。
5. オペランドを増加または減少させる方法は、指定する変数の数によって異なります。

READ ステートメント

READ ステートメントは、レコードをプログラムが使用できるようにします。

- 順次アクセスの場合には、READ ステートメントは、ファイル内の次のレコードをオブジェクト・プログラムが使用できるようにします。
- ランダム・アクセスの場合には、READ ステートメントは、直接アクセス・ファイル内の指定されたレコードをオブジェクト・プログラムが使用できるようにします。

READ ステートメントの実行時には、関連するファイルが INPUT モードまたは I-O モードでオープンされていなければなりません。 READ ステートメントの実行方法は、ファイル編成によって異なります。ファイル編成には次のものがあります。

- 順次
- 相対
- 索引付き

ファイル制御記入項目に FILE STATUS 文節が指定されている場合は、READ ステートメントの処理時に関連する状況キーが更新されます。

READ ステートメントの処理が失敗すると、関連するレコード域の内容およびファイル位置標識の位置は未定義になります。

装置タイプ DISK および DATABASE に関する特別の考慮事項

IBM Extension

READ ステートメントが DISK または DATABASE 装置上にあるファイルに対して実行される場合には、ヌル可能フィールドがサポートされます。 しかし、ヌル値は ASSIGN 文節で ALWNULL が指定された DATABASE ファイルにだけサポートされます。 ALWNULL が指定されず、フィールドにヌル値が含まれている場合は、READ 操作は失敗し、ファイル状況の 90 が戻されます。 READ ステートメントに NULL-MAP/NULL-KEY-MAP を指定することによっても、どのフィールドにヌル値が含まれているかを知ることができます。

End of IBM Extension

順次アクセス・モード

形式 1 は、順次アクセス・モードのすべてのファイルに対して使用しなければなりません。

形式 1 の READ ステートメントを実行すると、ファイル内の次のレコードが検索されます。次にアクセスされるレコードは、ファイル編成によって決まります。

動的アクセス・モード

索引付き編成または相対編成のファイルの場合には、FILE-CONTROL 記入項目に動的アクセス・モードを指定することができます。動的アクセス・モードでは、使用する形式によって、レコードを順次またはランダムに検索できます。

順次検索を実行する場合は、NEXT 句を含む形式 2 を指定しなければなりません。順次アクセスに関するこれ以外の規則がすべて適用されます。

ランダム検索を実行する場合は、形式 3 を指定しなければなりません。ランダム・アクセスに関するこれ以外の規則がすべて適用されます。

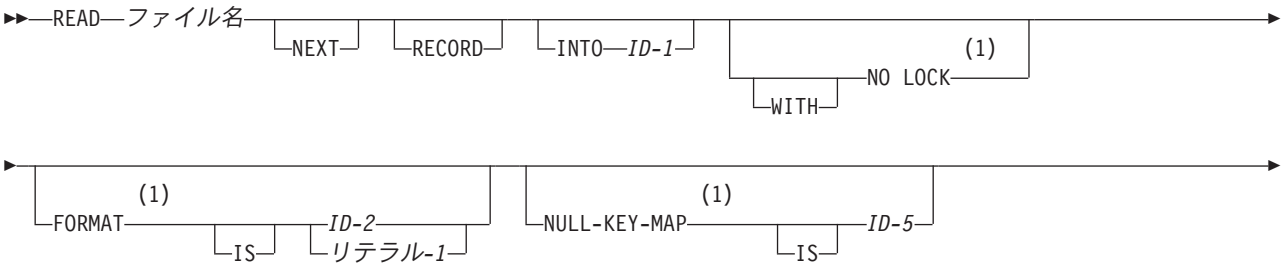
ランダム・アクセス・モード

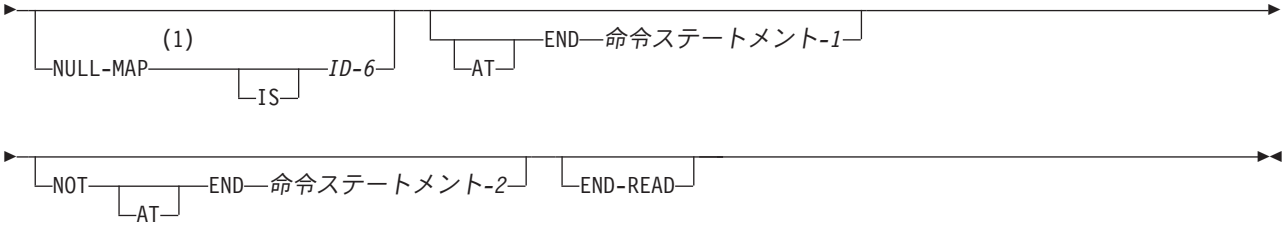
形式 3 は、ランダム・アクセス・モードの索引付きおよび相対ファイルに対して指定しなければなりません。また、レコードの検索がランダムである場合の動的アクセス・モードのファイルに対しても指定しなければなりません。

以下の節で説明するように、READ ステートメントの実行方法はファイル編成によって異なります。

READ ステートメント - 形式 1 - 順次検索 / 順次アクセス

READ - 形式 1 - 順次検索 / アクセス



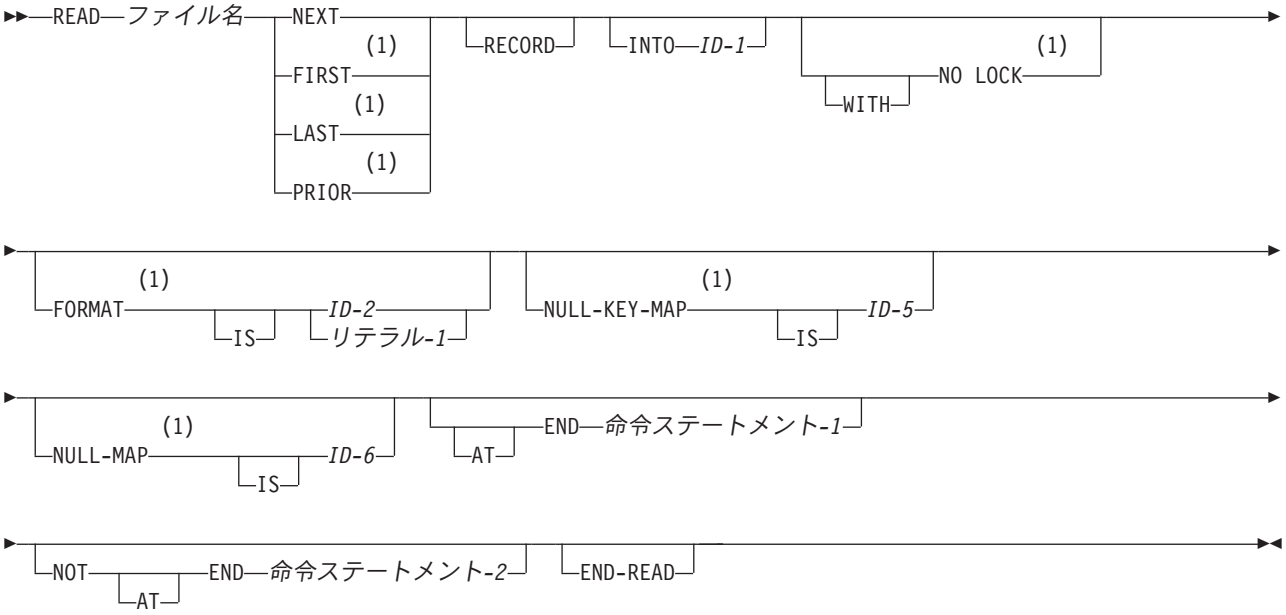


注:

1 IBM 拡張

READ ステートメント - 形式 2 - 順次検索 / 動的アクセス

READ - 形式 2 - 順次検索 / 動的アクセス

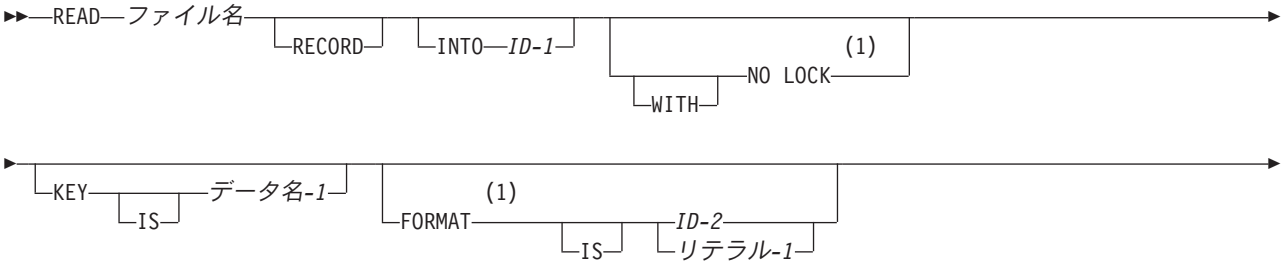


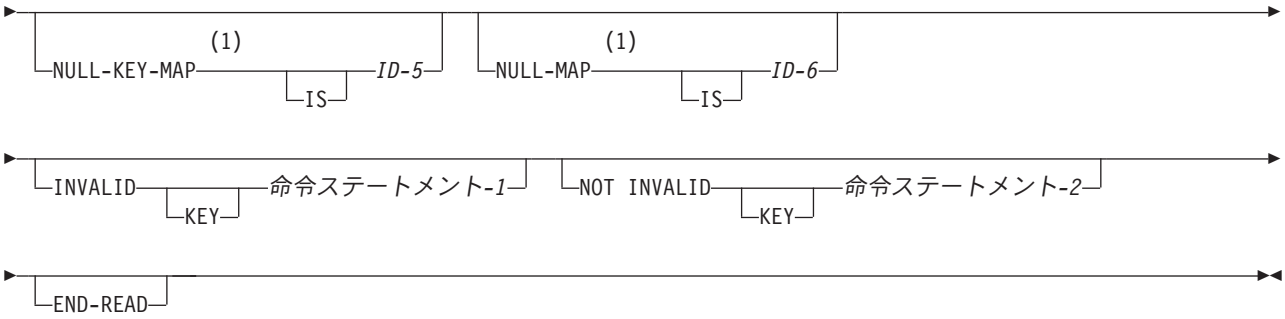
注:

1 IBM 拡張

READ ステートメント - 形式 3 - ランダム検索

READ ステートメント - 形式 3 - ランダム検索





注:

1 IBM 拡張

ファイル名

ファイル名は、データ部の FD 記入項目の中で定義されていなければならず、ソート・ファイルまたはマージ・ファイルの名前であってはなりません。

RECORD

一連のレコードの中の次のレコードです。

KEY IS データ名-1

この句は、索引付きファイルの場合にだけ指定します。データ名-1 は修飾できますが、添え字付けすることはできません。データ名-1 では、ファイル名に関連するレコード・キーを識別しなければなりません。

IBM Extension

データ名-1 は DBCS データ項目として定義できます。 RECORD KEY 文節が DBCS データ項目を指定する場合、READ ステートメントに指定された KEY は DBCS データ項目でなければなりません。

End of IBM Extension

INTO 句

INTO ID 句によって、READ ステートメントは次のステートメントと同等になります。

- READ ファイル名 RECORD
- MOVE レコード名 TO ID

READ ステートメントの処理が正常に終了すると、現行レコードは、レコード名と ID の両方で使用可能になります。

INTO ID 句が指定されている場合、現行レコードは、CORRESPONDING 句のない MOVE ステートメントの規則に従って、入力域から ID 域に移動されます。 ID に関連する添え字付け、指標付け、または参照変更は、レコードが読み取られた後、 ID に転送される直前に評価されます。 (7-38 ページの『INTO/FROM ID 句』も参照してください。)

INTO 句を READ ステートメントに指定できるのは、次の場合です。

- ファイル記述記入項目に従属しているレコード記述が 1 つだけである。または
- ファイル名に関連付けられているすべてのレコード名、および ID-1 によって参照されているデータ項目が、グループ項目または基本英数字項目を記述している。

可変長レコードについて INTO ID 句を指定すると、受け入れフィールドに移動されるデータの量が、読み取られる可変長レコードの長さと等しくなります。

ID-1

ID-1 は受け入れフィールドです。現行レコードは、CORRESPONDING 句のない MOVE ステートメントの規則に従って、レコード域から ID-1 によって指定されている区域に移動されます。次の使用上の注意が適用されます。

- 現行レコードのサイズは、RECORD 文節内について指定されている規則によって異なります。
- ファイル記述記入項目に RECORD IS VARYING 文節が含まれている場合には、移動はグループ移動です。
- 暗黙の MOVE ステートメントが起こるのは、READ ステートメントの実行が成功する場合だけです。
- ID-1 に関連する添え字付けまたは参照変更は、レコードが読み取られた後、データ項目に移動される直前に適用されます。
- レコードは、レコード域と、ID-1 によって参照されるデータ項目の両方で使用可能になります。
- READ ステートメントで INTO 句を使用できるのは、次の場合だけです。
  - ファイル記述記入項目に従属しているレコード記述が 1 つだけである。または
  - ファイル名-1 に関連付けられているすべてのレコード名、および ID-1 によって参照されている項目が、グループ項目または基本英数字項目を記述している。
- ファイル名-1 および ID-1 に対応するレコード域は、同じ記憶域であってはなりません。

IBM Extension

- ID-1 は、浮動小数点データ項目にすることができます。
- ID-1 は DBCS データ項目にすることができます。
- ID-1 は、日時データ項目にすることができます。

End of IBM Extension

IBM Extension

NO LOCK 句

NO LOCK 句は、READ 操作で、I-O (更新) モードでオープンするファイルに対するレコード・ロックを獲得できないようにします。さらに、NO LOCK 句を持つ READ 操作は、使用可能となるレコードが他のジョブによってロックされた場合でも成功します。さらに、この句を持つ READ ステートメントは、前の READ 操作によってロックされたレコードを解放します。

DUPLICATES 句がそのファイルに対して指定されると、NO LOCK 句を指定したステートメントによって読み取られるレコードは、DELETE または REWRITE ステートメントによっては処理できません。

I-O モードでオープンしていないファイルに対して NO LOCK 句を使用すると、コンパイル時にエラー・メッセージを受け取ります。

ファイル・ロックおよびレコード・ロックについては、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

FORMAT 句

FORMAT 句が適用されるのは、READ ステートメントが、ASSIGN のファイル装置タイプとして DATABASE が指定された索引付きファイルに対して実行される場合だけです。

FORMAT 句で指定する値には、この入出力操作で使用するレコード形式の名前が入ります。システムはこれを使用して、どのレコード形式に対して操作を行うかを指定または選択します。

ID-2 を指定する場合には、10 文字以下の英数字データ項目にしなければなりません。

リテラル-1 を指定する場合は、10 文字以下の大文字の文字ストリングにしなければなりません。

FORMAT 句を指定せずに、ランダム・アクセス・モードで索引付きファイルにアクセスする場合には、定義された最初の形式が使用されます。

全桁がブランクの値は、FORMAT 句が指定されなかった場合と同じ扱いになります。その値がファイルに対して有効でない場合は、9K の FILE STATUS が戻され、(そのファイルに適用可能であれば) USE プロシージャーが呼び出されます。

ファイルが**順次アクセス・モード**で読み取られるときには、キー順アクセス・パス内の、要求された形式を持つ次のレコードが使用可能になります。形式が省略された場合には、キー順アクセス・パスの中の次のレコードが使用可能になります。

ファイルが**ランダム・アクセス・モード**で読み取られるときには、指定された形式について定義されたキーが、その形式のレコードを獲得するために使用されます。その形式のレコードが見つからない場合には、INVALID KEY 条件が発生します。この条件は、定義されたキーを持つが、レコード形式が異なるレコードがある場合でも発生します。

形式が省略された場合には、ファイルの共通キーが使用されて、その共通キー値を持つ任意の形式の最初のレコードが獲得されます。ファイルの共通キーは、データベース上に存在するレコードに関するファイルのすべての形式に共通なキー・フィールドから成り立っています。ファイルの共通キーは、ファイル中のすべてのレコード形式で共通な左端のキー・フィールドです。共通キーは、ファイルについてプログラム内で定義された最初のレコード形式を使用して、レコード記述域内のデータから作成されます。

ファイルが**動的アクセス・モード**で読み取られるときには、次に使用可能になるレコードは以下のように判別されます。

レコード	FORMAT 句	
	指定された場合	省略
NEXT	指定された形式を持つ、キー順アクセス・パス内の次のレコードが使用可能になります。	形式に関係なく、キー順アクセス・パス内の次のレコードが使用可能になります。
PRIOR	キー順アクセス・パス内のレコードで、ファイル位置標識によって識別されるレコードに先行しており、指定された形式を持つものが使用可能になります。	キー順アクセス・パス内のレコードで、ファイル位置標識によって識別されるレコードに先行しているものが、形式に関係なく使用可能になります。
FIRST	指定された形式を持つ、キー順アクセス・パス内の最初のレコードが使用可能になります。	形式に関係なく、キー順アクセス・パス内の最初のレコードが使用可能になります。
LAST	指定された形式を持つ、キー順アクセス・パス内の最後のレコードが使用可能になります。	形式に関係なく、キー順アクセス・パス内の最後のレコードが使用可能になります。

レコード	FORMAT 句	
	指定された場合	省略
上記以外	指定された形式について定義されたキーが、その形式のレコードを獲得するために使用されます。その形式のレコードが見つからない場合には、INVALID KEY 条件が発生します。この条件は、定義されたキーを持つが、レコード形式が異なるレコードがある場合でも発生します。	ファイルの共通キーが使用されて、その共通キー値を持つ任意の形式の最初のレコードが獲得されます。ファイルの共通キーは、データベース上に存在するレコードに関するファイルのすべての形式に共通なキー・フィールドから成り立っています。ファイルの共通キーは、ファイル中のすべてのレコード形式で共通な左端のキー・フィールドです。共通キーは、ファイルについてプログラム内で定義された最初のレコード形式を使用して、レコード記述域内のデータから作成されます。

NULL-KEY-MAP IS 句

NULL-KEY-MAP IS 句は、処理されるレコードのキーに対してデータ管理機能が提供するヌル・バイト・マップ値に対応する ID の値を示します。 ID は、添え字を付けたり、参照変更できます。

句は、ASSIGN 文節が装置タイプとして DATABASE を指定し、ALWNULL 属性を指定した索引付きファイルに対してだけ指定できます。

ファイルに代替キーがある場合、ID-5 は、参照の現行キーのヌル・キー・マップに関連しています。

ヌル可能フィールドの使用について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

NULL-MAP IS 句

NULL-MAP IS 句は、処理されるレコードのデータ管理が提供するヌル・バイト・マップ値に対応する ID の値を示します。 ID は、添え字を付けたり、参照変更できます。

この句は、ASSIGN 文節が装置タイプとして DATABASE を指定し、ALWNULL 属性を指定したすべてのファイルに対して指定できます。

ヌル可能フィールドの使用について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

End of IBM Extension

AT END 句

AT END 句は、ファイルが順次アクセス・モードまたは動的アクセス・モードで読み取られる場合にだけ適用されます。

順次読み取りの処理時に次のレコードが存在しない場合には、AT END 条件が発生し (ファイル状況の高位桁は 1)、READ ステートメントの処理が正常に行われません。次の処置が取られます。

1. FILE STATUS 文節が指定されている場合には、状況キーが AT END 条件を示すように更新されます。
2. AT END 句が指定されている場合には、制御が AT END 命令ステートメントに渡されます。このファイルの EXCEPTION/ERROR プロシージャーは実行されません。
3. AT END 句が指定されていない場合には、このファイルについての EXCEPTION/ERROR プロシージャーが実行されます。制御は、このプロシージャーから、READ ステートメントの終わりに続く次の実行可能ステートメントに戻されます。

AT END 条件が起こると、READ ステートメントが正常に実行されません。関連するレコード域の内容は未定義となり、ファイル位置標識は、次の有効なレコードが確立されていないことを示すように設定されます。

READ ステートメントの実行時に AT END 条件が起こらなかった場合には、AT END 句 (指定されている場合) が無視され、次の処置が取られます。

1. ファイル位置標識が設定され、ファイル名-1 と関連する I-O 状況が更新されます。
2. AT END 条件ではない例外条件が存在する場合は、ファイル名-1 について適用可能な USE AFTER EXCEPTION プロシージャーの実行に続いて、制御権が USE ステートメントの規則に従って移動されます。
3. 例外条件が存在しない場合には、レコード域でレコードが使用可能になり、INTO 句の存在のために生じる暗黙の移動が実行されます。制御権は、READ ステートメントの終わりまたは命令ステートメント-2 (指定されている場合) に移動されます。後者の場合には、命令ステートメント-2 に指定されている各ステートメントについての規則に従って実行が続けられます。制御権の明示的な移動をもたらすプロシージャー分岐ステートメントまたは条件ステートメントが実行される場合には、制御権はそのステートメントについての規則に従って移動されます。それ以外の場合には、制御権は命令ステートメント-2 の実行完了時に READ ステートメントの終わりに移動されます。

READ ステートメントが正常に実行されなかった場合には、関連するレコード域の内容は未定義となり、ファイル位置標識は、次の有効なレコードが確立されていないことを示すように設定されます。

ファイルに対して明示または暗黙の EXCEPTION/ERROR プロシージャーが指定されていない場合には、AT END 句を指定しなければなりません。

注: 順次読み取りとは、順次アクセスのファイルに対する任意の READ ステートメント、または動的アクセスのファイルに対する READ NEXT、READ PRIOR、READ FIRST、または READ LAST ステートメントのことです。

READ FIRST または READ LAST ステートメントでは、ファイルが空の場合、または FORMAT 句が使用されていて、ファイルのどのレコードも指定された形式を持たない場合、AT END 条件になります。

ファイルを順次アクセス・モードで読み込んでいるときに AT END 条件が検出された場合には、さらに READ ステートメントを処理する前に、正常な CLOSE ステートメントとそれに続く正常な OPEN ステートメントを、このファイルに対して処理する必要があります。

ファイルを動的アクセス・モードで読み込んでいるときに AT END 条件が検出された場合には、さらに READ NEXT または READ PRIOR ステートメントを処理する前に、そのファイルに対して次のいずれかを処理する必要があります。

- 正常な CLOSE ステートメントとそれに続く正常な OPEN ステートメント。
- 正常な START ステートメント。
- 正常なランダム・アクセス READ ステートメント。
- 正常な READ FIRST または READ LAST ステートメント。

順次アクセスのファイルに対する READ ステートメント、または動的アクセスのファイルに対する READ NEXT または READ PRIOR ステートメントを、AT END 条件が発生した後で試み、また指定されたメソッドのいずれかでファイル位置標識がリセットされていなかった場合、ファイル状況 46 が戻されます。 AT END 句も NOT AT END 句も実行されません。

NOT AT END 句

NOT AT END 句を指定した READ ステートメントが正常に完了するたびに (ファイル状況の高位桁は 0)、この句に関連付けられている命令ステートメントに制御権が移動します。

INVALID KEY 句

INVALID KEY 句は、相対または索引付きファイルがランダム・アクセス・モードまたは動的アクセス・モードで読み取られる場合にだけ適用されます。

該当する EXCEPTION/ERROR プロシージャーが存在していないファイルについては、INVALID KEY 句を指定しなければなりません。

INVALID KEY 句の処理については 7-38 ページの『INVALID KEY 条件』を参照してください。

NOT INVALID KEY 句

NOT INVALID KEY 句は、相対または索引付きファイルがランダム・アクセス・モードまたは動的アクセス・モードで読み取られる場合にだけ適用されます。

NOT INVALID KEY 句を指定した READ ステートメントが正常に完了すると、この句に関連付けられている命令ステートメントに制御権が移動します。

NEXT 句

NEXT 句は、動的アクセス・モードの場合にだけ適用されます。

相対ファイルが動的に読み取られる場合に、NEXT 句が指定されていると、順次読み取りが行われます。この句が省略されると、ランダム読み取りが行われます。

索引付きファイルが動的に読み取られる場合に、NEXT 句が指定されていると、順次読み取りが行われます。 NEXT、FIRST、LAST および PRIOR がすべて省略されると、ランダム・アクセス読み取りが行われます。

レコードのブロックに対して READ NEXT 操作が実行される場合には、ブロックが空になるまで READ PRIOR 操作は実行できません。 READ PRIOR 操作が最初に実行されると、ブロックが空になるまで READ NEXT 操作は実行できません。これが試みられると、ファイル状況 9U が生じます。ファイル状況 9U から回復するには、ファイルをクローズし、その後再びオープンしてください。

IBM Extension

FIRST 句

FIRST 句は、索引付きファイルが動的にアクセスされる場合にだけ適用されます。
NEXT、FIRST、LAST および PRIOR がすべて省略されると、ランダム・アクセス読み取りが行われます。

LAST 句

LAST 句は、索引付きファイルが動的にアクセスされる場合にだけ適用されます。
NEXT、FIRST、LAST および PRIOR がすべて省略されると、ランダム・アクセス読み取りが行われます。

PRIOR 句

PRIOR 句は、索引付きファイルが動的にアクセスされる場合にだけ適用されます。これが指定されると、順次読み取りが行われます。 NEXT、FIRST、LAST および PRIOR がすべて省略されると、ランダム・アクセス読み取りが行われます。

レコードのブロックに対して READ PRIOR 操作が実行される場合には、ブロックが空になるまで READ NEXT 操作は実行できません。 READ NEXT 操作が最初に実行されると、ブロックが空になるまで READ PRIOR 操作は実行できません。これが試みられると、ファイル状況 9U が生じます。ファイル状況 9U から回復するには、ファイルをクローズし、その後再びオープンしてください。

End of IBM Extension

END-READ 句

この明示範囲終了符号は、READ ステートメントの範囲を区切る働きをします。END-READ 句を使用することによって、READ 条件ステートメントを別の条件ステートメントにネストすることができます。また、END-READ 句は、READ 命令ステートメントで使用することもできます。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

順次ファイル

順次ファイルは、次の装置タイプから読み取ることができます。

- TAPEFILE
- DISKETTE
- DISK
- DATABASE

順次ファイルは、**順次アクセス・モード**でだけ読み取ることができます。

READ ステートメントによって使用可能となるレコードは、次のように判別されます。

- ファイル位置標識が OPEN ステートメントの処理によって設定された場合には、ポインターによって指し示されているレコードが使用可能になります。
- ファイル位置標識が前の READ ステートメントの処理によって設定された場合には、ポインターがファイル内に存在する次のレコードを指し示すように更新されます。次に、そのレコードが使用可能となります。

このファイルに対してのファイル制御記入項目に SELECT OPTIONAL が指定されており、このプログラムの実行時にファイルが使用可能でない場合には、最初の READ ステートメントの実行により AT END 条件が起こります。ファイルが使用できないので、ファイルのクローズ時にシステムの標準的なファイル終了処理は行われません。

- 装置タイプ TAPEFILE および DISKETTE に関する特別の考慮事項

装置タイプ TAPEFILE および DISKETTE に関する特別の考慮事項: READ ステートメントの実行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、以下の処置が示されている順番で行われます。

1. 標準の終了ボリューム・ラベル・プロシージャーが実行されます。
2. ボリュームの切り替えが行われます。
3. 標準の開始ボリューム・ラベル・プロシージャーが実行されます。
4. 次のボリュームの最初のデータ・レコードが使用可能になります。

プログラムでは、読み取り操作中に上記の処置が取られたという標識を受け取りません。

相対ファイル

相対ファイルは、次の装置タイプから読み取ることができます。

- DISK
- DATABASE

相対ファイルは、**順次**、**ランダム**、または**動的アクセス・モード**で読み取ることができます。

相対ファイルが**順次アクセス・モード**で読み取られるときには、READ ステートメントによって使用可能となるレコードは、次のように判別されます。

- ファイル位置標識が START または OPEN ステートメントの処理によって設定された場合には、ポインターによって指し示されているレコードが使用可能になります (それが、ファイル位置標識によって示されているパスを介してアクセスできる場合)。そのレコードがアクセス可能でなくなっている (例えば、レコードの削除が原因で) 場合には、現行レコード・ポインターがファイル内に存在する次のレコードを指し示すように更新されます。次に、そのレコードが使用可能となります。
- ファイル位置標識が前の READ ステートメントの処理によって設定された場合には、ファイル位置標識がファイル内に存在する次のレコードを指し示すように更新されます。次に、そのレコードが使用可能となります。

このファイルに対して RELATIVE KEY 句が指定されている場合には、READ ステートメントの処理によって、RELATIVE KEY データ項目が、使用可能にされるレコードの相対レコード番号を示すように更新されます。

相対ファイルが**ランダム・アクセス・モード**で読み取られるときには、RELATIVE KEY データ項目に入っている相対レコード番号を持つレコードが使用可能になります。ファイル内にそのようなレコードがない場合には、INVALID KEY 条件が発生し、READ ステートメントの処理は正常に行われません。

索引付きファイル

索引付きファイルは、次の装置タイプから読み取ることができます。

- DISK
- DATABASE

索引付きファイルは、**順次**、**ランダム**、または**動的アクセス・モード**で読み取ることができます。

索引付きファイルが**順次アクセス・モード**で読み取られるときには、READ ステートメントによって使用可能となるレコードは、次のように判別されます。

- ファイル位置標識が START または OPEN ステートメントの処理によって設定された場合には、ポインターによって指し示されているレコードが使用可能になります (それが、現行レコード・ポインターによって示されているパスを介してアクセスできる場合)。そのレコードがアクセス可能でなくなっている (例えば、レコードの削除が原因で) 場合には、ファイル位置標識がファイル内に存在する次のレコードを指し示すように更新されます。次に、そのレコードが使用可能となります。
- ファイル位置標識が前の READ ステートメントの処理によって設定された場合には、ファイル位置標識がファイル内に存在する次のレコードを指し示すように更新されます。次に、そのレコードが使用可能となります。

IBM Extension

重複キーが認められている (ファイル制御記入項目に DUPLICATES 句が指定されている) ファイルの場合、重複キー値を持つレコードは、ファイルの作成時に指定された順序で使用可能になります。システム・オプションとしては、先入れ先出し (FIFO)、後入れ先出し (LIFO)、および指定順序なし (FIFO も LIFO も指定されない場合) があります。

End of IBM Extension

索引付きファイルが**ランダム・アクセス・モード**で読み取られるときには、参照の現行キーのキー値と等しいキー値を持つファイル内のレコードが使用可能になります。ファイル内にそのようなレコードがない場合には、INVALID KEY 条件が発生し、READ ステートメントの処理は正常に行われません。索引付きファイルがランダム・アクセス・モードで読み取られるときに、入出力ステートメントで FORMAT 句が指定されていない場合には、ファイル内で定義された最初の形式が使用されます。外部記述キーが使用され、形

式が指定されていない場合には、プログラム内に含まれている最初の形式がキーの作成に使用されることに注意してください。この形式は、ファイル内の最初の形式であるとは限りません。

KEY 句が指定されていない場合には、基本 RECORD KEY が、この要求に対する参照キーとなります。動的アクセスが指定されている場合、異なる参照キーが確立されるまで、基本 RECORD KEY も、順次 READ ステートメントのそれ以降の実行に対して、参照キーとして使用されます。

KEY 句が指定されると、データ名-1 は、この要求に対する参照キーとなります。 動的アクセスが指定されると、異なる参照キーが確立されるまで、この参照キーが順次 READ ステートメントのそれ以降の実行に対して使用されます。

IBM Extension

重複キーが認められている (ファイル制御記入項目に DUPLICATES 句が指定されている) ファイルの場合、指定されたキー値を持つ最初のレコードが使用可能になります。最初のレコードは、ファイルの作成時に指定された順序で判別されます。システム・オプションとしては、先入れ先出し (FIFO)、後入れ先出し (LIFO)、および指定順序なし (FIFO も LIFO も指定されない場合) があります。

DUPLICATE KEY 検査用のファイル状況 02 を使用可能にするためには、次のものが必要です。

- SELECT 文節内の WITH DUPLICATES 句
- OPEN I-O または OPEN INPUT
- OPTION パラメーターの *DUPKEYCHK オプション、または PROCESS ステートメントの DUPKEYCHK オプション

End of IBM Extension

複数レコードの処理

ファイル名-1 に複数のレコード記述記入項目が関連付けられている場合には、これらのレコードは自動的に同じ記憶域を共用します。つまり、それらは暗黙に再定義されます。 READ ステートメントが実行された後、現行レコードの範囲内のデータ項目だけが置き換えられます。その範囲を超えて保管されているデータ項目は未定義となります。 7-180 ページの図 7-9 はこの概念を表したものです。 現行レコードの範囲がファイル名についてのレコード記述記入項目を超えている場合には、レコードが最大サイズの右側で切り捨てられます。いずれの場合にも、READ ステートメントは正常に実行され、入出力状況はレコード長の矛盾が生じたことを示すように設定されます。

```
FD 記入項目:
FD  INPUT-FILE LABEL RECORDS OMITTED.

01  RECORD-1  PICTURE X(30).

01  RECORD-2  PICTURE X(20).


READ ステートメント実行時の入力域の内容:

    ABCDEFGHIJKLMNOPQRSTUVWXYZ1234

(RECORD-2) で読み取られているレコードの内容:

    01234567890123456789

READ 実行後の入力域の内容:

    0123456789012345678?????????
                        └──┘
                (入力域のこれらの文字は未定義)
```

図 7-9. 複数のレコード記述を持つ READ ステートメント

マルチボリューム・ファイル

READ ステートメントの実行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、以下の処置が行われます。

- システム定義の終了ボリューム・ラベル・プロシージャー
- ボリュームの切り替え
- システム定義の開始ボリューム・ラベル・プロシージャー
- 次のボリュームの最初のデータ・レコードが使用可能になります。

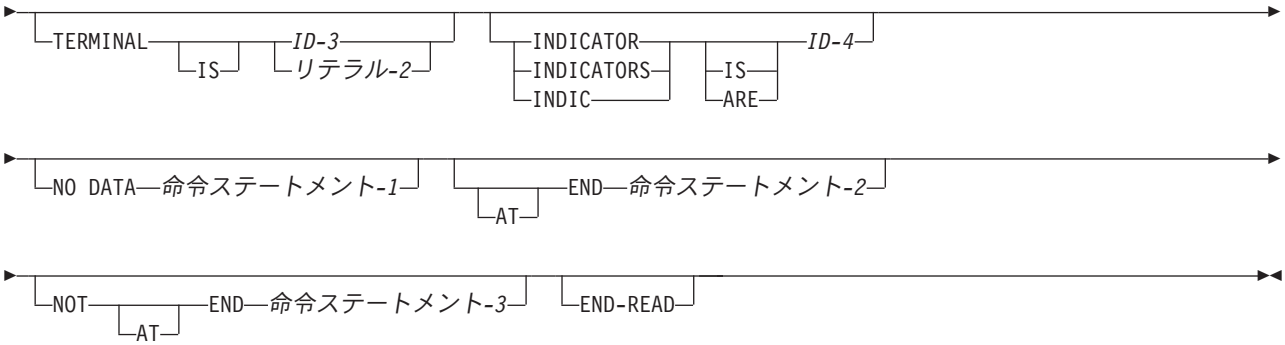
トランザクション・ファイル

IBM Extension

READ ステートメントは、指定された形式を使用して、ある装置のレコードを使用可能にします。形式がサブファイルである場合には、READ ステートメントは、そのサブファイルの指定されたレコードを使用可能にします。

READ ステートメント - 形式 4 - トランザクション (非サブファイル):

READ ステートメント - 形式 4 - トランザクション (非サブファイル)



形式 4 を使用するのは、サブファイル・レコードではない形式を読み取る場合だけです。
RELATIVE KEY データ項目が FILE-CONTROL 記入項目に指定されていても使用されません。形式 4 の READ ステートメントは、サブファイル・レコードについては無効です。ただし、表示画面上で更新されたサブファイル・レコードをサブファイルに入れるためには、サブファイル制御レコード形式について形式 4 の READ ステートメントを使用しなければなりません。

データが使用可能である場合、それはレコード域に戻されます。レコード形式とプログラム装置の名前は、I-O-FEEDBACK 域と CONTROL-AREA に戻されます。

READ ステートメントは、ファイル用に獲得された装置がある場合にだけ有効です。 READ が実行され、獲得された装置がなかった場合は、ファイル状況が 92 (論理エラー) に設定されます。

形式 4 の READ ステートメントが機能する方法は、次の条件によって異なります。

- READ が単一装置ファイルと複数装置ファイルのどちらに対するものであるか
- TERMINAL 句を介して特定のプログラム装置が要求されているかどうか
- FORMAT 句を介して特定のレコード形式が要求されているかどうか
- NO DATA 句が指定されているかどうか

以下の説明で、「データが使用可能 / 戻される」という表現は、応答標識のみが設定される状態も含んでいます。これは、別個の標識域が使用され、標識がファイル用のレコード域に戻されない場合にも当てはまります。

次の表は、句の可能な組み合わせと、単一装置ファイルまたは複数装置ファイルに対して実行される機能を示しています。例えば、TERMINAL が N、FORMAT が N、NO DATA が N であれば、単一装置は D、複数装置は A です。

	句	**Y= 可 N= 不可**
コンパイル時に検査される	TERMINAL[3] FORMAT[3] NO DATA	N N N N Y Y Y Y N N Y Y N N Y Y N Y N Y N Y N Y
実行時に判別される	単一装置 複数装置	D C D B D C D B A A D B D C D B

コード A ～ D について以下に説明します。

2. 句が指定され、データ項目またはリテラルがブランクの場合には、実行時に句はそれが指定されなかったかのように扱われます。

コード A - 送信勧誘されたプログラム装置からの読み取り (複数装置ファイルだけ)

このタイプの READ は、使用可能なデータを持つ、最初に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。送信勧誘されたプログラム装置は、入力を送信するよう勧誘されているワークステーションまたは通信装置 (APPC、SNUF、 BSCEL、非同期通信など) です。送信勧誘は、DDS キーワード INVITE を指定した形式でプログラム装置に書き込むことによって行われます。送信勧誘されたプログラム装置からの読み取りが実際に行われると、その装置は送信勧誘されなくなります。そのプログラム装置は、再送信勧誘されるか、あるいは TERMINAL 句または FORMAT 句を指定してそのプログラム装置に READ が出されない限り、別の READ ステートメントによる入力用には使用されません。

プログラム装置から戻されるレコード形式は、システムによって判別されます。ワークステーションでこれがどのように決定されるのかについては、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) 内のカテゴリー『データベースおよびファイル・システム』のセクション『ファイル・システムと管理』を参照してください。通信装置の場合、ICF ファイルの形式選定処理に関する詳細は、「*ICF プログラミング・マニュアル*」を参照してください。

この READ は、次の場合にはデータを戻さずに完了することがあります。

1. 送信勧誘された装置がなく、タイマー機能が有効でない。 (AT END 条件)
2. ジョブの制御付き取り消しが発生する。この結果、ファイル状況値は 0A になり、メジャー / マイナー戻りコード値は 0309 になります。
3. NO DATA 句が省略されており、指定された待機時間が満了した。この結果、ファイル状況値は 00 になり、メジャー / マイナー戻りコード値は 0310 になります。指定された待機時間は、ファイルに対しての WAITRCD パラメーターで入力された値、またはタイマー機能に関して指定された時間間隔です。
4. NO DATA 句が指定されており、READ の実行時、すぐに使用できるデータがない。

データが使用可能である場合、それはレコード域に戻されます。レコード形式は、I-O-FEEDBACK 域と CONTROL-AREA に戻されます。「プログラム装置送信勧誘からの読み取り」についての詳細は、ディスプレイ装置については、資料「*アプリケーション表示プログラミング*」を、通信装置については、資料「*ICF Programming*」を参照してください。

コード B - 1 台のプログラム装置からの読み取り (無効な組み合わせ)

コンパイル時メッセージが出され、NO DATA 句は無視されます。 NO DATA 句が省略される場合の同じ句の組み合わせについて、表の項目を参照してください。

コード C - 1 台のプログラム装置からの読み取り (NO DATA 句を指定している場合)

READ ステートメントのこの機能を使用すると、プログラムの実行を停止して、データが使用可能になるまで待つことはありません。データが即時に使用可能となるか、そうでなければ NO DATA 命令ステートメントが実行されます。

この READ 機能を使用して、特定のプログラム装置 (デフォルトのプログラム装置または TERMINAL 句によって指定されたプログラム装置) のデータが使用可能であるかどうかを定期的に検査できます。このデータの検査は、次のように行われます。

1. プログラム装置が次のように判別されます。
   a. TERMINAL 句が省略されたかまたはブランクである場合には、デフォルトのプログラム装置が使用されます。デフォルトのプログラム装置は、最後に試行された

READ、WRITE、REWRITE、ACQUIRE、または DROP ステートメントによって使用された装置です。上記の入出力操作が以前に実行されていない場合は、デフォルトのプログラム装置は最初に獲得されたプログラム装置です。

b. TERMINAL 句が指定された場合は、指示されたプログラム装置が使用されます。

2. データが使用可能であるかどうか、およびプログラム装置が送信勧誘されているかどうかを判別するために検査が行われます。
3. データが使用可能である場合、そのデータがレコード域に戻され、プログラム装置は送信勧誘されなくなります。すぐに使用できるデータがない場合、NO DATA 命令ステートメントが実行され、プログラム装置は送信勧誘されたままになります。
4. プログラム装置が送信勧誘されていない場合、AT END 条件が起こり、ファイル状況は 10 に設定されます。

コード D - 1 台のプログラム装置からの読み取り (NO DATA 句が指定されていない場合)

この READ は、常に、データが使用可能になるのを待ちます。ジョブが制御付き取り消しを受け取った場合や、ファイルに対して WAITRCD 時間が指定されている場合でも、プログラムが READ ステートメントから制御を取り戻すことはありません。この READ 操作は次のように実行されます。

1. プログラム装置が次のように判別されます。
   a. TERMINAL 句が省略されているかまたはブランクの値である場合には、デフォルトのプログラム装置が使用されます。デフォルトのプログラム装置は、最後に試行された READ、WRITE、REWRITE、ACQUIRE、DROP、または ACCEPT (属性データ) ステートメントによって使用されたプログラム装置です。これらの操作がどれも行われていない場合は、ファイルのオープン時に暗黙に獲得されたプログラム装置が使用されます。獲得された装置がない場合、AT END 状態が起こります。
   b. TERMINAL 句が指定されている場合は、指定されたプログラム装置が使用されます。
2. レコード形式が次のように判別されます。
   a. FORMAT 句が省略されているかまたはブランクである場合、戻されるレコード形式はシステムによって判別されます。ワークステーション装置の場合のレコード形式が判別される方法については、資料「*ICF Programming*」を参照してください。通信装置の場合のレコード形式が判別される方法については、資料「*ICF Programming*」の ADDICFDEVE および OVRICFDEVE コマンドの FMTSLT パラメーターに関する章を参照してください。
   b. FORMAT 句が指定されている場合は、指示されたレコード形式が戻されます。使用可能なデータが、要求されたレコード形式と一致しない場合は、ファイル状況値 9K が設定されます。
3. データが使用可能になるまで、プログラムの実行は停止します。 READ ステートメントの実行後、データはレコード域に戻されます。プログラム装置が以前に送信勧誘されていた場合には、この READ ステートメントの後には送信勧誘されなくなります。

INTO 句

次のいずれかの場合でなければ、INTO 句を指定することはできません。

- ファイルに関連付けられているすべてのレコードと、INTO 句で指定されているデータ項目が、グループ項目または基本英数字項目である。

  OR

- ファイル記述記入項目に従属しているレコード記述が 1 つだけである。

KEY IS 句

KEY IS 句は、索引付きファイルに対してだけ指定できます。データ名では、ファイル名-1 に関連するレコード・キーを識別しなければなりません。データ名-1 は修飾できますが、添え字付けすることはできません。

注: KEY IS 句は、構文検査だけが行われ、READ ステートメントの操作には影響を与えません。

FORMAT 句

リテラル-1 または ID-2 では、読み取られるレコード形式の名前を指定します。リテラル-1 を指定する場合は、非数字で、10 文字以下の大文字にしなければなりません。 ID-2 (指定されている場合) は、英数字データ項目で、10 文字以下の長さでなければなりません。 ID-2 がブランクである場合、READ ステートメントは、FORMAT 句が省略された場合と同じように実行されます。

NO DATA 句

NO DATA 句が指定されている場合、READ ステートメントは、データがすぐに使用できるかどうかを判別します。データが使用可能である場合は、そのデータがレコード域に戻されます。データがすぐには使用できない場合は、命令ステートメント-1 が実行されます。 NO DATA 句を指定すると、READ ステートメントはデータが使用可能になるのを待つことがなくなります。

TERMINAL 句

リテラル-2 または ID-3 では、プログラム装置名を指定します。リテラル-2 を指定する場合は、10 文字以下の非数字にしなければなりません。 ID-3 を指定する場合は、10 文字以下の英数字データ項目を参照しなければなりません。プログラム装置は、READ ステートメントの実行前に獲得されていなければなりません。 ID-3 がブランクである場合、READ ステートメントは、TERMINAL 句が省略された場合と同じように実行されます。単一装置ファイルの場合は、TERMINAL 句を省略できます。プログラム装置がその単一装置であると見なされます。

複数のプログラム装置を獲得しているトランザクション・ファイルの READ について TERMINAL 句を省略すると、デフォルトのプログラム装置が使用されます。

INDICATOR 句、INDICATORS 句、INDIC 句

データ・レコードが読み取られるときに使用される標識を指定します。標識は、データ・レコードについての情報と、それがプログラムに入力された方法を渡すために使用できます。

ID-4 は、OCCURS 文節なしで指定された基本ブール・データ項目か、または基本データ・ブールの項目が従属しているグループ項目でなければなりません。

INDICATORS 句について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」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。

AT END 句

AT END 句は、ステートメントの有効範囲を明示的に区切る働きをします。 AT END 条件が検出されると、命令ステートメント-2 が実行されます。 AT END 条件は、送信勧誘されたプログラム装置がなく、タイマー機能が有効でない場合に発生します。

ファイル名について適用可能な USE プロシージャーが指定されていない場合には、AT END 句を指定しなければなりません。ファイルに対して AT END 句と USE プロシージャーの両方が指定されていて、AT END 条件が生じた場合、制御権は AT END 命令ステートメントに移動され、USE プロシージャーは実行されません。

NOT AT END 句

この句では、使用されるステートメントについて AT END 条件が存在しない場合に実行されるプロシージャーを指定できます。

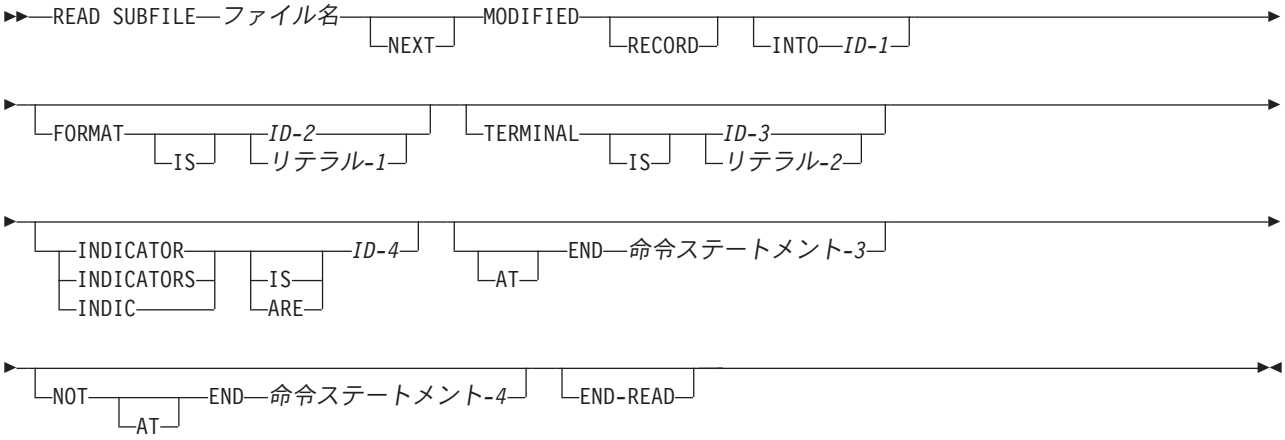
END-READ 句

この明示範囲終了符号は、READ ステートメントの範囲を区切る働きをします。END-READ 句を使用することによって、READ 条件ステートメントを別の条件ステートメントにネストすることができます。また、END-READ 句は、READ 命令ステートメントで使用することもできます。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

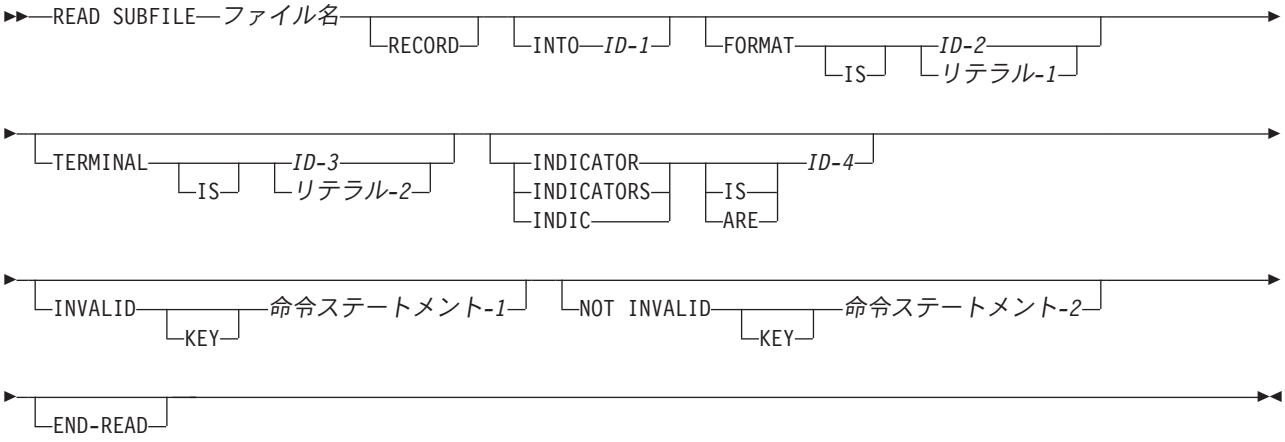
READ ステートメント - 形式 5 - トランザクション (サブファイル):

READ - 形式 5a - トランザクション (サブファイル順次)



形式 5a は、サブファイル・レコードである形式を順次アクセス・モードで読み取る場合に使用します。サブファイル・レコードを順次にアクセスするためには、NEXT MODIFIED 句を指定しなければなりません。 AT END 句は、NEXT MODIFIED 句と一緒でなければ指定できません。

READ - 形式 5b - トランザクション (サブファイル・ランダム)



形式 5b は、サブファイル・レコードである形式をランダム・アクセス・モードで読み取る場合に使用します。 INVALID KEY 句は、サブファイル・レコードにランダムにアクセスする場合にしか使用することができません。 NEXT MODIFIED 句は、サブファイル・レコードにランダムにアクセスする場合には、使用してはなりません。

形式 5a または 5b は、通信装置については使用できません。通信装置についてサブファイル形式の READ ステートメントを使用すると、READ が失敗し、ファイル状況が 90 に設定されます。

NEXT MODIFIED 句

NEXT MODIFIED 句が指定されていない場合、使用可能になるデータ・レコードは、RELATIVE KEY データ項目の値と一致する相対レコード番号を持つ、サブファイル内のレコードです。

NEXT MODIFIED 句が指定されておらず、RELATIVE KEY データ項目にサブファイル内のレコードの相対レコード番号以外の値が入っている場合には、INVALID KEY 条件が発生し、READ ステートメントの実行は失敗します。

NEXT MODIFIED 句を指定した場合、使用可能になるレコードは、サブファイル内の変更された (変更データ・タグがオンになっている) 最初のレコードです。

次に変更されたレコードの検索は、次のように開始されます。

- 次の場合、サブファイルの始まりから開始されます。
  - サブファイル制御レコードに対して入出力操作が実行された場合。
  - 入出力操作によってサブファイルの消去、初期設定、または表示が行われた場合。
- その他のすべてのケースでは、前の読み取り操作によって読み取られたレコードの次のレコードから開始されます。

RELATIVE KEY データ項目の値は、プログラムにとって使用可能となったレコードの相対レコード番号を反映するために更新されます。

NEXT MODIFIED が指定されており、サブファイル内にユーザーによって変更されたレコード (RELATIVE KEY データ項目に入っている相対レコード番号よりも大きい相対レコード番号を持つ) がない場合には、AT END 条件が生じ、ファイル状況が 12 に設定され、RELATIVE KEY データ項目の値は更新されません。その後、命令ステートメント-2 または適用可能な USE AFTER ERROR/EXCEPTION プロシージャーが実行されます。

FORMAT 句

形式名が指定されていない場合、使用される形式は、入力フィールド、入出力フィールド、または潜在フィールドを含むディスプレイ装置に書き込まれた最後のレコード形式です。ディスプレイ・ファイルにそのような形式がない場合には、ディスプレイ装置に対する最後の WRITE 操作のレコード形式が使用されます。

FORMAT 句を指定する場合、リテラル-1、または ID-2 の内容は、該当するプログラム装置に対して活動状態である形式にしなければなりません。 READ ステートメントは、指定された形式のデータ・レコードを読み取ります。

複数形式ファイルの場合、正しい結果を得るために、FORMAT 句を必ず指定しなければなりません。

TERMINAL 句

TERMINAL 句に関する一般的な考慮事項については、上記の形式 4 を参照してください。

形式 5a または 5b の READ の場合、複数の装置が獲得されているファイルに対して TERMINAL 句が省略されると、デフォルトのプログラム装置に関連付けられているサブファイルからレコードが読み取られます。

INDICATOR 句、INDICATORS 句、INDIC 句

データ・レコードが読み取られるときに使用される標識を指定します。標識は、データ・レコードについての情報と、それがプログラムに入力された方法を渡すために使用できます。

INDICATORS 句について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」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。

ID-4 は、OCCURS 文節なしで指定された基本ブール・データ項目か、または基本データ・ブールの項目が従属しているグループ項目でなければなりません。

INVALID KEY 句

READ ステートメントの実行時に、サブファイルの相対レコード番号と一致しない値が RELATIVE KEY データ項目に入っている場合には、INVALID KEY 条件が生じ、READ ステートメントは正常に実行されません。

NEXT MODIFIED 句が指定されておらず、ファイル名について適用可能な USE プロシージャーがない場合には、INVALID KEY 句を指定しなければなりません。

無効キー条件が発生した場合に生じる事柄の詳細は 7-38 ページの『INVALID KEY 条件』を参照してください。

NOT INVALID KEY 句

この句では、使用されるステートメントについて無効キー条件が存在しない場合に実行されるプロシージャーを指定します。

AT END 句

NEXT MODIFIED が指定されており、サブファイル内にユーザーによって変更されたレコードがない場合には、AT END 条件が生じ、READ ステートメントの実行は失敗します。

NEXT MODIFIED 句が使用されており、ファイル名について適用可能な USE プロシージャーが指定されていない場合には、AT END 句を指定しなければなりません。ファイルに対して AT END 句と USE プロシージャーの両方が指定されていて、AT END 条件が生じた場合、制御権は AT END 命令ステートメントに移動され、USE プロシージャーは実行されません。

NOT AT END 句

この句では、使用されるステートメントについて AT END 条件が存在しない場合に実行されるプロシージャーを指定できます。

END-READ 句

この明示範囲終了符号は、READ ステートメントの範囲を区切る働きをします。END-READ 句を使用することによって、READ 条件ステートメントを別の条件ステートメントにネストすることができます。また、END-READ 句は、READ 命令ステートメントで使用することもできます。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

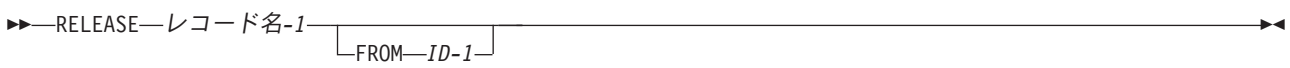
End of IBM Extension

RELEASE ステートメント

RELEASE ステートメントは、入出力域からソート操作の最初のフェーズへレコードを転送します。

RELEASE ステートメントは、SORT ステートメントと関連する入力プロシージャーの範囲内でしか使用できません。

RELEASE ステートメント - 形式



INPUT PROCEDURE の中には、少なくとも 1 つの RELEASE ステートメントを指定しなければなりません。

RELEASE ステートメント

RELEASE ステートメントが実行されると、レコード名-1 の現行の内容がソート・ファイル内に置かれます。つまり、ソート操作の最初のフェーズで使用できるようになります。

レコード名-1
ソート・マージ・ファイル記述記入項目 (SD) 内のレコードの名前を指定しなければなりません。レコード名-1 は修飾できます。

IBM Extension

浮動小数点にすることも、日時データ項目にすることもできます。

End of IBM Extension

FROM ID-1
この句によって、RELEASE ステートメントは次のステートメントと同等になります。

```
MOVE identifier-1 to record-name-1
RELEASE record-name-1
```

移動は、CORRESPONDING 句のない MOVE ステートメントの規則に従って実行されます。

ID-1 は、英数字または DBCS 関数 ID の名前にすることができます。

IBM Extension

ID-1 は、レコード名-1 が DBCS データ項目の場合、DBCS データ項目でなければなりません。

End of IBM Extension

IBM Extension

ID-1 は、浮動小数点にすることも、日時データ項目にすることもできます。

End of IBM Extension

レコード名-1 と ID-1 で同じ記憶域を参照してはなりません。

SAME RECORD AREA 文節でファイル名-1 についての SD 記入項目を指定せずに RELEASE ステートメントを実行した場合には、レコード名-1 内の情報は使用可能でなくなります。

SAME RECORD AREA 文節で SD 記入項目が指定**されている**場合には、レコード名-1 は、その文節内に指定されている他のファイルのレコードとして、まだ使用できます。

FROM ID-1 を指定した場合には、情報は ID-1 でまだ使用可能です。

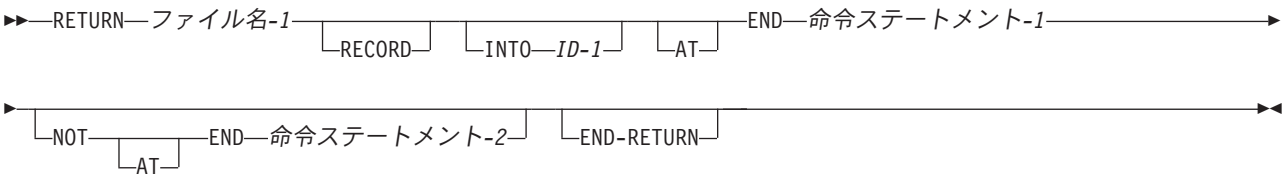
制御が INPUT PROCEDURE から渡されるときには、ソート・ファイルは、RELEASE ステートメントの実行により入れられたすべてのレコードから構成されています。

RETURN ステートメント

RETURN ステートメントは、ソートまたはマージの操作の最終フェーズから OUTPUT PROCEDURE へレコードを転送します。

RETURN ステートメントは、SORT または MERGE ステートメントと関連する出力プロシージャーの範囲内でしか使用できません。

RETURN ステートメント - 形式



OUTPUT PROCEDURE の中には、少なくとも 1 つの RETURN ステートメントを指定しなければなりません。

RETURN ステートメントが実行されると、ファイル名-1 内の次のレコードが OUTPUT PROCEDURE による処理用に使用できるようになります。

ファイル名-1 および ID-1 に対応するレコード域は、同じ記憶域であってはなりません。

レコードは、レコード域と、ID-1 によって参照されるデータ項目の両方で使用可能になります。

ファイル名-1

ファイル名-1 は、データ部の SD 記入項目の中で記述されていなければなりません。

ファイル名-1 に複数のレコード記述が関連付けられている場合には、これらのレコードは自動的に同じ記憶域を共用します。つまり、それらは暗黙に再定義されます。 RETURN ステートメントが実行された後、現行レコードの内容だけが使用可能になります。現行レコードの長さを超えて存在するデータ項目がある場合には、それらの内容は未定義となります。

INTO ID-1

この句によって、RETURN INTO ステートメントは次のステートメントと同等になります。

```
RETURN file-name-1
MOVE record-name TO identifier-1
```

IBM Extension

ID-1 は、DBCS、浮動小数点、または日時データ項目とすることが できます。

End of IBM Extension

移動は、CORRESPONDING 句のない MOVE ステートメントの規則に従って実行されます。

現行レコードのサイズは、RECORD 文節について指定されている規則によって判別されます。ファイル記述記入項目に RECORD IS VARYING 文節が含まれている場合には、暗黙の MOVE はグループの移動となります。ただし、RETURN が成功しなければ、暗黙の MOVE は行われません。

ID-1 に関連する添え字付け、指標付け、または参照変更は、レコードが戻された後、 ID-1 へ移動される直前に評価されます。

次の 1 つまたは両方が真の場合には、RETURN ステートメントに INTO 句を指定できます。

- ソート・マージ・ファイル記述記入項目に従属しているレコード記述が 1 つだけである。
- ファイル名-1 に関連付けられているすべてのレコード名、および ID-1 によって参照されているデータ項目が、グループ項目または基本英数字項目を記述している。

AT END 句

AT END 句に指定された命令ステートメント-1 は、すべてのレコードがファイル名-1 から戻された後で実行されます。これ以降は、現行の出力プロシージャーの一部として RETURN ステートメントを実行することはできません。

RETURN ステートメントの実行時に AT END 条件が生じない場合には、レコードが使用可能になり、INTO 句の存在のために生じる暗黙の移動が実行された後で、制御権は NOT AT END 句によって指定されている命令ステートメントに移動されます。それ以外の場合には、制御権は RETURN ステートメントの終わりに渡されます

END-RETURN 句

この明示範囲終了符号は、RETURN ステートメントの範囲を区切る働きをします。 END-RETURN 句を使用することによって、RETURN 条件ステートメントを別の条件ステートメントにネストすることができます。また、END-RETURN 句は、RETURN 命令ステートメントで使用することもできます。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

REWRITE ステートメント

REWRITE ステートメントは、直接アクセス・ファイル内に存在するレコードを論理的に置き換えます。

REWRITE ステートメントの実行時には、関連する直接アクセス・ファイルが I-0 モードでオープンされていなければなりません。

IBM Extension

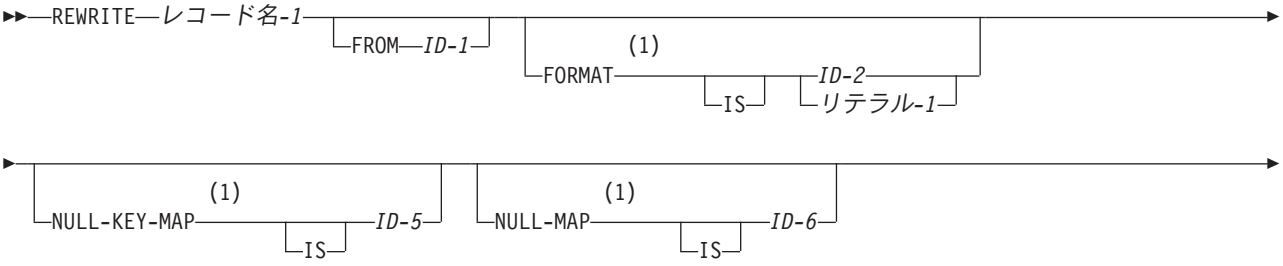
データベース・ファイル一時変更 (OVRDBF) CL コマンドの書き込み禁止 (INHWRT) パラメーターによって、このステートメントの処置をプログラム実行時に禁止することができます。このパラメーターが指定されている場合には、データに依存するエラーがあっても、ゼロでないファイル状況コードは設定されません。データに依存するエラーの例として、重複キーおよびデータ変換エラーがあります。

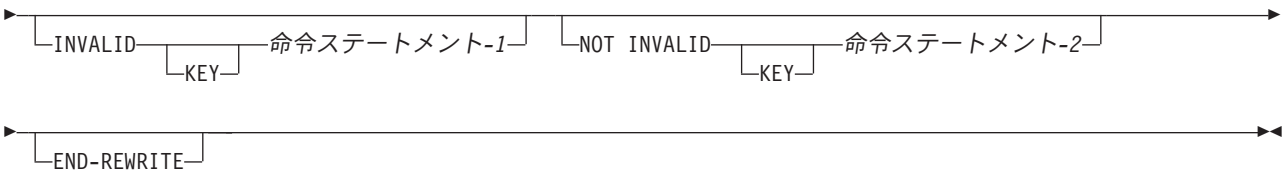
このコマンドについて詳しくは、**IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL および API*』を参照してください。

End of IBM Extension

REWRITE ステートメント - 形式 1

REWRITE ステートメント - 形式 1





注:

1 IBM 拡張

レコード名-1
置き換えられるレコードと同じ数の文字桁を持つ、ファイル・セクション内のレコードの名前です。レコードの名前は参照変更できません。

ID-1
これは送り出し項目です。

FROM 句
この句は、次の効果をもちます。

```
MOVE identifier-1 TO record-name-1.
REWRITE record-name-1.
```

REWRITE ステートメントの処理が成功すると、現行レコードはレコード名-1 では使用可能でなくなりますが、ID-1 ではまだ使用可能です。レコード名-1 と ID-1 の両方で同じ記憶域を参照してはなりません。

IBM Extension

FORMAT 句
この句が適用されるのは、REWRITE ステートメントが、ASSIGN ファイルの装置タイプとして DATABASE が指定された索引付きファイルに対して実行される場合です。レコード形式が 1 つであるファイルを処理する場合には、この句の指定はオプショナルです。

FORMAT 句で指定する値には、この入出力操作で使用するレコード形式の名前が入ります。システムはこれを使用して、どのレコード形式に対して操作を行うかを指定または選択します。

ID-2 を指定する場合には、10 文字以下の英数字データ項目にしなければなりません。

リテラル-1 を指定する場合は、10 文字以下の大文字の文字ストリングにしなければなりません。

FORMAT 句を指定せずに、ランダム・アクセス・モードで索引付きファイルにアクセスする場合には、定義された最初の形式が使用されます。

全桁がブランクの値は、FORMAT 句が指定されなかった場合と同じ扱いになります。その値がファイルに対して有効でない場合は、9K の FILE STATUS が戻され、(そのファイルに適用可能であれば) USE プロシージャーが呼び出されます。

NULL-KEY-MAP IS 句
ページ『NULL-KEY-MAP IS 句』のこの句についての説明を参照してください。

NULL-MAP IS 句
ページ『NULL-MAP IS 句』のこの句についての説明を参照してください。

End of IBM Extension

INVALID KEY 句

この句は、索引編成ファイルと、ランダムまたは動的アクセスを持つ相対編成ファイルにおいて有効です。これは、レコード域内のキー・フィールドによって指定されたレコードが見つからない場合に処理されます。

INVALID KEY 条件が発生した場合、更新操作は行われません。レコード名の中のデータは影響を受けません。この句は、必要に応じて、対応する命令ステートメントに制御権を移動します。

レコード名-1 について適用可能な EXCEPTION/ERROR プロシージャーが指定されていない場合は、INVALID KEY 句を指定しなければなりません。

次のいずれかの場合に、INVALID KEY 条件が発生します。

- アクセス・モードが順次であり、置き換えられるレコードの基本 RECORD KEY に含まれる値が、ファイルの最後に検索されたレコードの基本 RECORD KEY データ項目の値と等しくない場合。
- 基本 RECORD KEY に含まれている値が、ファイル内のレコードのものと等しくない場合。
- DUPLICATES が指定されていない ALTERNATE RECORD KEY データ項目の値は、ファイル内に既にあるレコードの値と同じです。

適用可能な USE プロシージャーが指定されていないファイルに対しては、INVALID KEY 句を指定しなければなりません。

詳細は 7-37 ページの『共通の処理機能』の『INVALID KEY 条件』を参照してください。

装置タイプ DISK 上の順次にアクセスされる索引付きファイルの場合、この句は、置き換えられるレコードの RECORD KEY に含まれる値が、ファイルの最後に検出されたレコードの RECORD KEY データ項目と等しくない場合に処理されます。

NOT INVALID KEY 句

この句は、索引編成ファイルと、ランダムまたは動的アクセスを持つ相対編成ファイルに対して有効です。 NOT INVALID KEY 句が指定されている REWRITE ステートメントの実行が正常に完了すると、制御権は、この句に関連する命令ステートメントに渡されます。

REWRITE ステートメントに関する考慮事項

REWRITE ステートメントの実行が正常に終了すると、レコードは、レコード名-1 では使用可能でなくなります。ただし、関連するファイルが SAME RECORD AREA 文節内で指定されている場合を除きます (この場合には、レコードは、SAME RECORD AREA 文節の中で指定されている他のファイルのレコードとしても使用可能です)。

ファイル位置標識は、REWRITE ステートメントの実行による影響を受けません。

FILE-CONTROL 記入項目に FILE STATUS 文節が指定されている場合は、REWRITE ステートメントの実行時に、関連する状況キーが更新されます。

順次ファイル: ファイルに対して実行された最後の入出力ステートメントが、正常に実行された READ ステートメントでなければなりません。置き換えられるレコードは、そのステートメントによって検索されたレコードです。

INVALID KEY および NOT INVALID KEY 句を指定してはなりません。 EXCEPTION/ERROR プロシージャーは指定できます。

順次編成のファイルについては、レコード名-1 の文字桁数は、置き換えられるレコードの文字桁数と等しくなければなりません。

索引付きファイル: 順次アクセス・モードの場合には、ファイルに対して実行された最後の入出力ステートメントが、正常に実行された READ ステートメントでなければなりません。置き換えられるレコードは、そのステートメントによって検索されたレコードです。RECORD KEY データ項目の値は、レコードが読み取られてから変更されていてはなりません。値が変更されていると、INVALID KEY 条件が発生します。

ランダムまたは動的アクセス・モードの場合には、置き換えられるレコードは、RECORD KEY データ項目内の値によって指定されます。ファイルに当該レコードがない場合には、INVALID KEY 条件が発生します。

EXCEPTION/ERROR プロシージャーがファイルに定義されていない場合は、INVALID KEY 句を指定しなければなりません。

索引付き編成のファイルについては、レコード名-1 の文字桁数は、置き換えられるレコードの文字桁数と異なっていても構いません。

IBM Extension

ファイルに対して EXTERNALLY-DESCRIBED-KEY が指定されているときには、FORMAT 句によって指定されている形式 (あるいは、FORMAT 句が使用されていない場合には、最初の形式) に対応するレコード域内にあるキー・データが、RECORD KEY データ項目の現行値を決定するために使用されます。

WITH DUPLICATES 句がファイルに指定されている場合には、すべてのアクセス・モード (順次、ランダム、動的) において、ファイルに対して実行された最後の入出力ステートメントが、正常に実行された READ ステートメントでなければなりません。置き換えられるレコードは、そのステートメントによって検索されたレコードです。RECORD KEY データ項目の値は、レコードが読み取られてから変更されていてはなりません。値が変更されていると、INVALID KEY 条件が発生します。

注: 重複キーがファイル内にあるような場合に正しいレコードが置き換えられるようにするには、READ ステートメントが必要です。重複キーを持つ一連のレコードのうちの特定の 1 つのレコードを再書き込みする唯一の方法は、各レコードを順次に読み取り、識別したあとで、該当するレコードを再書き込みすることです。

End of IBM Extension

相対ファイル: 順次アクセス・モードの場合には、ファイルに対して実行された最後の入出力ステートメントが、正常に実行された READ ステートメントでなければなりません。置き換えられるレコードは、そのステートメントによって検索されたレコードです。INVALID KEY および NOT INVALID KEY 句を指定してはなりません。 EXCEPTION/ERROR プロシージャーは指定できます。

ランダムまたは動的アクセス・モードの場合には、置き換えられるレコードは、RELATIVE KEY データ項目内の値によって指定されます。ファイルに当該レコードがない場合には、INVALID KEY 条件が発生します。EXCEPTION/ERROR プロシージャーがファイルに定義されていない場合は、INVALID KEY 句を指定しなければなりません。

相対編成のファイルについては、レコード名-1 の文字桁数は、置き換えられるレコードの文字桁数と異なっていても構いません。

レコード・ロック:

IBM Extension

以下のファイル・タイプの REWRITE ステートメントの前に、READ ステートメントを正常に実行しておく必要があります。

- 順次編成のファイル
- 順次アクセス・モードを使用したファイル
- 索引付き編成で、複写キーを持つファイル

このような READ ステートメントには NO LOCK 句を入れてはいけません。 READ ステートメントによって選択されているレコードを置き換えようとしており、そのレコードが読み取られたときにロックされなかった場合には、REWRITE ステートメントは正常に実行されません。

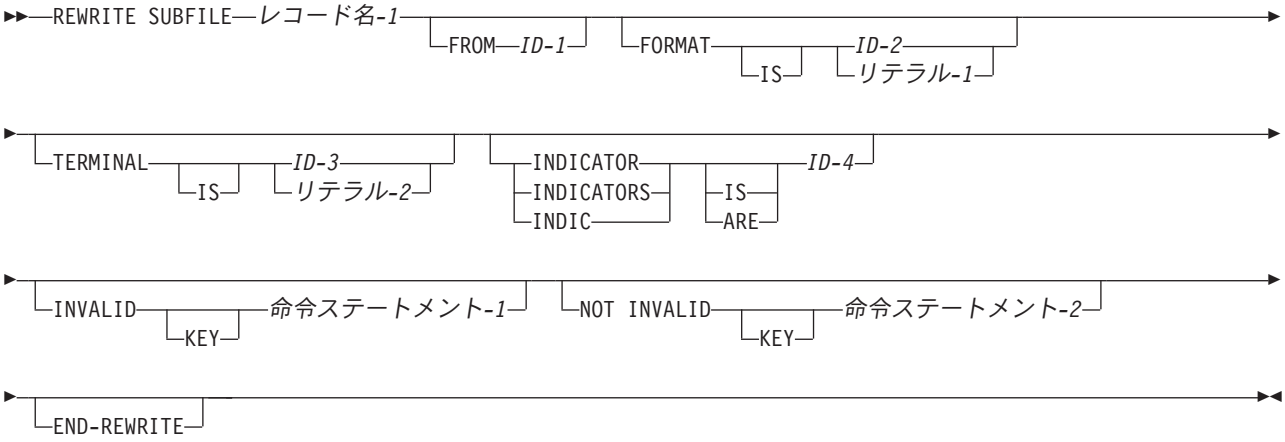
End of IBM Extension

トランザクション (サブファイル) 形式

IBM Extension

REWRITE ステートメントは、すでにサブファイル内に存在するサブファイル・レコードを置き換える場合に使用されます。

REWRITE ステートメント - 形式 2 - トランザクション (サブファイル)



レコード名-1 によって参照されるレコードの文字桁数は、置き換えられるレコードの文字桁数と等しくなければなりません。再書き込み操作に先立って、レコードの読み取り操作を正常に実行しなければなりません。サブファイル内の置き換えられるレコードは、直前の読み取り操作でアクセスされたレコードです。

FORMAT 句

トランザクション・ファイルの場合には、それぞれ異なる形式を持つ複数のデータ・レコードを同時に活動状態にすることができます。 FORMAT 句を指定する場合には、システムに定義されている有効な形式名を指定しなければなりません。また、同じ形式のデータ・レコードに対して入出力操作を実行しなければなりません。形式が無効な名前であるかまたは存在しない場合には、FILE STATUS データ項目 (指定されている場合) の値が 9K に設定され、レコード域の内容が未定義になります。

注:

1. FORMAT 句に指定するレコード形式は、直前の読み取り操作でアクセスされたレコード形式でなければなりません。
2. ID-2 の内容またはリテラル-1 は、直前の READ でアクセスされたサブファイル形式の名前でなければなりません。

TERMINAL 句

TERMINAL 句は、どのプログラム装置のサブファイルにレコードを再書き込みするのかを指示します。TERMINAL 句を指定する場合、リテラル-2 および ID-3 では、トランザクション・ファイルによって獲得されたワークステーションを参照しなければなりません。リテラル-2 または ID-3 がブランク桁である場合は、TERMINAL 句は無効です。 TERMINAL 句によって指定されるプログラム装置は、明示的または暗黙に獲得されていなければならず、関連付けられたサブファイルをもっていなければなりません。

リテラル-2 または ID-3 は、有効なプログラム装置名でなければなりません。リテラル-2 を指定する場合には、10 文字以下の非数字にしなければなりません。 ID-3 を指定する場合は、10 文字以下の英数字データ項目を参照しなければなりません。

複数のプログラム装置を獲得しているトランザクション・ファイルから TERMINAL 句が省略された場合、使用されるサブファイルは、そのトランザクション・ファイルの READ が試みられた最後のプログラム装置に関連付けられているサブファイルです。

REWRITE ステートメントを通信装置に対して使用することはできません。 REWRITE ステートメントを通信装置に対して使用すると、操作が失敗し、ファイル状況が 90 に設定されます。

INDICATOR 句、INDICATORS 句、INDIC 句

データ・レコードが再書き込みされるときに使用される標識を指定します。標識は、データ・レコードについての情報と、それがプログラムに入力された方法を渡すために使用できます。

INDICATORS 句について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」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。

ID-4 は、OCCURS 文節なしで指定された基本ブール・データ項目か、または基本データ・ブールの項目が従属しているグループ項目でなければなりません。

INVALID KEY 句

再書き込み操作時に、RELATIVE KEY データ項目に含まれる値が、直前の読み取り操作からのレコードの相対レコード番号と一致しない場合は、INVALID KEY 条件が生じます。

該当する USE プロシージャーが指定されていないファイルについては、INVALID KEY 句を指定しなければなりません。 INVALID KEY 句が指定されておらず、USE プロシージャーも指定されていない場合は、望ましくない結果が生じることがあります。

NOT INVALID KEY 句

NOT INVALID KEY 句が指定されている REWRITE ステートメントの実行が正常に完了すると、制御権は、この句に関連する命令ステートメントに渡されます。

END-REWRITE 句

この明示範囲終了符号は、REWRITE ステートメントの範囲を区切る働きをします。 END-REWRITE 句を使用することによって、REWRITE 条件ステートメントを別の条件ステートメントにネストすることができます。また、END-REWRITE 句は、REWRITE 命令ステートメントで使用することもできます。詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

End of IBM Extension

ROLLBACK ステートメント

IBM Extension

ROLLBACK ステートメントは、データベース・レコードに対する 1 つまたは複数の変更を永続化すべきでないときに、それらを取り消す手段となります。

ROLLBACK ステートメント - 形式

ROLLBACK ステートメントが実行されると、最後のコミットメント境界以降にコミットメント制御下のファイルに対して行われたすべての変更が、データベースから除去されます。ファイルが OUTPUT 用にオープンされているときに消去されると、ROLLBACK ステートメントを実行しても、消去されたレコードがファイルに復元されないことに注意してください。

コミットメント境界とは、ROLLBACK または COMMIT ステートメントの前のオカレンスを指します。COMMIT または ROLLBACK が出されていなかった場合、コミットメント境界は、コミットメント制御下のファイルの最初の OPEN です。変更の除去は、ROLLBACK を発行する COBOL プログラム内のコミットメント制御下のファイルに対してだけではなく、コミットメント制御下にあるすべてのファイルに対して行われます。

ROLLBACK が正常に実行されると、コミットメント制御下のファイルに対して保持されていたすべてのレコード・ロックが解放され、レコードが他のジョブにとって使用可能になります。コミットメント制御は、ジョブ・レベルまたは活動化グループ・レベルで範囲を限定できます。 (コミットメント制御は、デフォルトによって、活動化グループ・レベルで範囲が限定されます。)

ROLLBACK は、コミットメント制御下にないファイルには影響しません。 ROLLBACK が実行され、コミットメント制御下にあるファイルがない場合は、ROLLBACK は無視されます。

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、オープンまたはクローズすることができます。なお、変更を永続的なものにするには、COMMIT を出さなければなりません。 ROLLBACK が実行されると、ファイルは、実行前と同じオープンまたはクローズ状態に置かれます。

ROLLBACK ステートメントは、次のことは行いません。

- どのファイルの I-O-FEEDBACK 域も修正しません。
- どのファイルのファイル状況値も設定しません。

ROLLBACK ステートメントに対して、以下の考慮事項が適用されます。

- ROLLBACK ステートメントは、ファイル位置標識を前のコミットメント境界での位置に設定します。順次処理を行っている場合には、このことを覚えておくことが大切です。
- ファイルがオープンされてから COMMIT ステートメントが発行されていない場合には、ROLLBACK ステートメントは、ファイル位置標識を OPEN 時の位置に設定します。
- 変更がコミットされないままファイルがクローズされている場合には、ROLLBACK 後にファイル位置標識は未定義となります。

コミットメント制御の範囲がジョブ・レベルで限定される場合には、それぞれのジョブの終わりでコミットメント制御下のすべてのファイルに対して、コミットされていないレコードの暗黙の ROLLBACK が自動的に行われます。データベースに対してコミットされていない変更が取り消されます。

コミットメント制御の範囲が活動化グループ・レベルで限定される場合には、活動化グループが正常に終了するときに、暗黙のコミットが起こります。活動化グループが異常に終了すると、暗黙の ROLLBACK が起こります。

コミットメント制御について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

End of IBM Extension

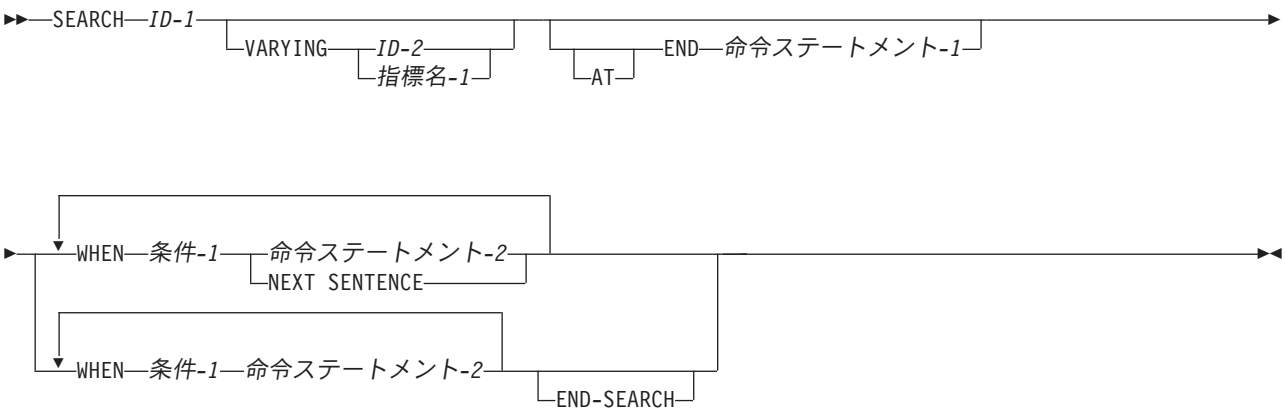
SEARCH ステートメント

SEARCH ステートメントは、指定された条件を満たすエレメントを見つけるためにテーブルを検索し、関連する指標がそのエレメントを指し示すように調整します。

- SEARCH ステートメントに関する考慮事項
- SEARCH の例

SEARCH ステートメント - 形式 1 - 逐次検索

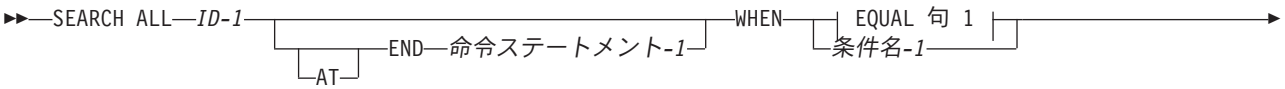
SEARCH ステートメント - 形式 1 - 逐次検索

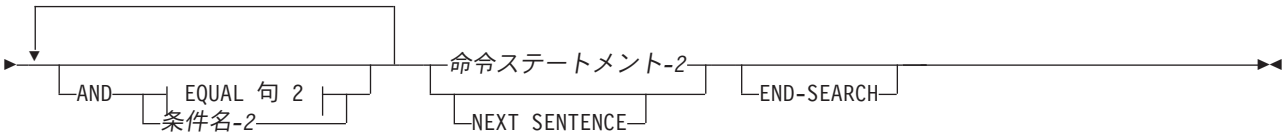


- 逐次検索の実行

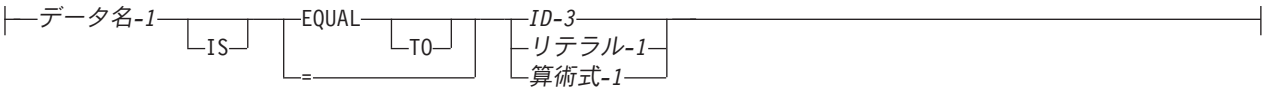
SEARCH ステートメント - 形式 2 - 二分検索

SEARCH ステートメント - 形式 2 - 二分検索

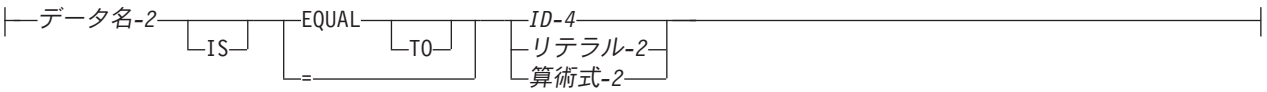




EQUAL 句 1:



EQUAL 句 2:



- 二分検索の実行

ID-1

OCCURS 文節を含むデータ項目に従属するデータ項目を指定できます。つまり、多次元テーブルの一部を指定できます。この場合には、データ記述記入項目で、テーブルの各次元について INDEXED BY 句が指定されていなければなりません。

IBM Extension

ID-1 には、浮動小数点データ項目を含むテーブル、DBCS 項目を含むテーブル、または、日時項目を含むテーブルを指定できます。

End of IBM Extension

ID-1 は、テーブル・エレメント内のすべてのオカレンスを指している必要があります。つまり、添え字が付けられたり、参照変更されてはなりません。

ID-1 のデータ部の記述には、INDEXED BY 句を伴う OCCURS 文節が入っていなければなりません。

SEARCH ステートメントの実行によって変更されるのは、ID-1 に関連する指標名の値と、もし存在する場合は、指標名-1 または ID-2 の値だけです (7-199 ページの『VARYING 句』を参照してください)。したがって、2 次元から 7 次元のテーブル全体を検索する場合には、各次元に対して SEARCH ステートメントを実行しなければなりません。それぞれの実行の前に、SET ステートメントを実行して、関連する指標名を初期設定し直さなければなりません。

AT END/WHEN 句

命令ステートメント-1 または命令ステートメント-2 が実行された後、 命令ステートメント-1 または命令ステートメント-2 が GO TO ステートメントで終わらない限り、制御は SEARCH ステートメントの終わりに渡されます。

条件-1

条件-1 は 7-10 ページの『条件式』で記述されている条件にすることができます。

IBM Extension

条件-1 には、DBCS 比較または DBCS 条件名条件を含めることができます。

End of IBM Extension

NEXT SENTENCE 句

この句を指定すると、制御権が、次の分離文字ピリオドの直前の暗黙の CONTINUE ステートメントに渡されます。 NEXT SENTENCE 句を指定する場合には、END-SEARCH 句を指定してはなりません。

END-SEARCH 句

この明示範囲終了符号は、SEARCH ステートメントの範囲を区切る働きをします。 END-SEARCH 句を使用することによって、SEARCH 条件ステートメントを別の条件ステートメントにネストすることができます。 END-SEARCH 句を指定する場合には、NEXT SENTENCE 句を指定することはできません。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

逐次検索

形式 1 の SEARCH ステートメントでは、現在設定されている指標から開始して、逐次検索が実行されます。検索開始時に、ID-1 と関連する指標名の値が可能な最大オカレンス番号以下である場合には、次の処理が行われます。

- WHEN 句の中の条件が、それらが書かれている順序で評価されます。
- 条件がどれも満たされない場合には、ID-1 についての指標名が次のテーブル・エレメントに対応するように増やされてから、ステップ 1 が繰り返されます。
- 評価時に WHEN 条件の 1 つが満たされた場合には、検索はただちに終了し、その条件に関連した命令ステートメントが実行されます。指標名は、条件を満たしたテーブル・エレメントを指します。 NEXT SENTENCE が指定されている場合には、最も近いピリオドの後のステートメントに制御が渡されます。
- WHEN 条件が満たされないままテーブルの終わりに達した (すなわち、増やされた指標名の値が可能な最大オカレンス番号より大きくなった) 場合には、以下に説明されているように検索が終了されます。

検索開始時に、ID-1 と関連する指標名の値が可能な最大オカレンス番号よりも大きい場合には、検索は直ちに終了し、AT END 命令ステートメントが指定されていれば実行されます。 AT END 句が省略されている場合には、制御は SEARCH ステートメントの後の次のステートメントに移されます。

VARYING 句

指標名-1

次のいずれかの処理が適用されます。

- 指標名-1 が ID-1 についての指標である場合には、この指標が検索に使用されます。そうでない場合には、最初の (または唯一の) 指標名が使用されます。
- 指標名-1 が別のテーブル・エレメントについての指標である場合には、ID-1 についての最初の (または唯一の) 指標名が検索に使用されます。指標名-1 によって表されるオカレンス番号は、検索指標名と同じだけ、同じときに増やされます。

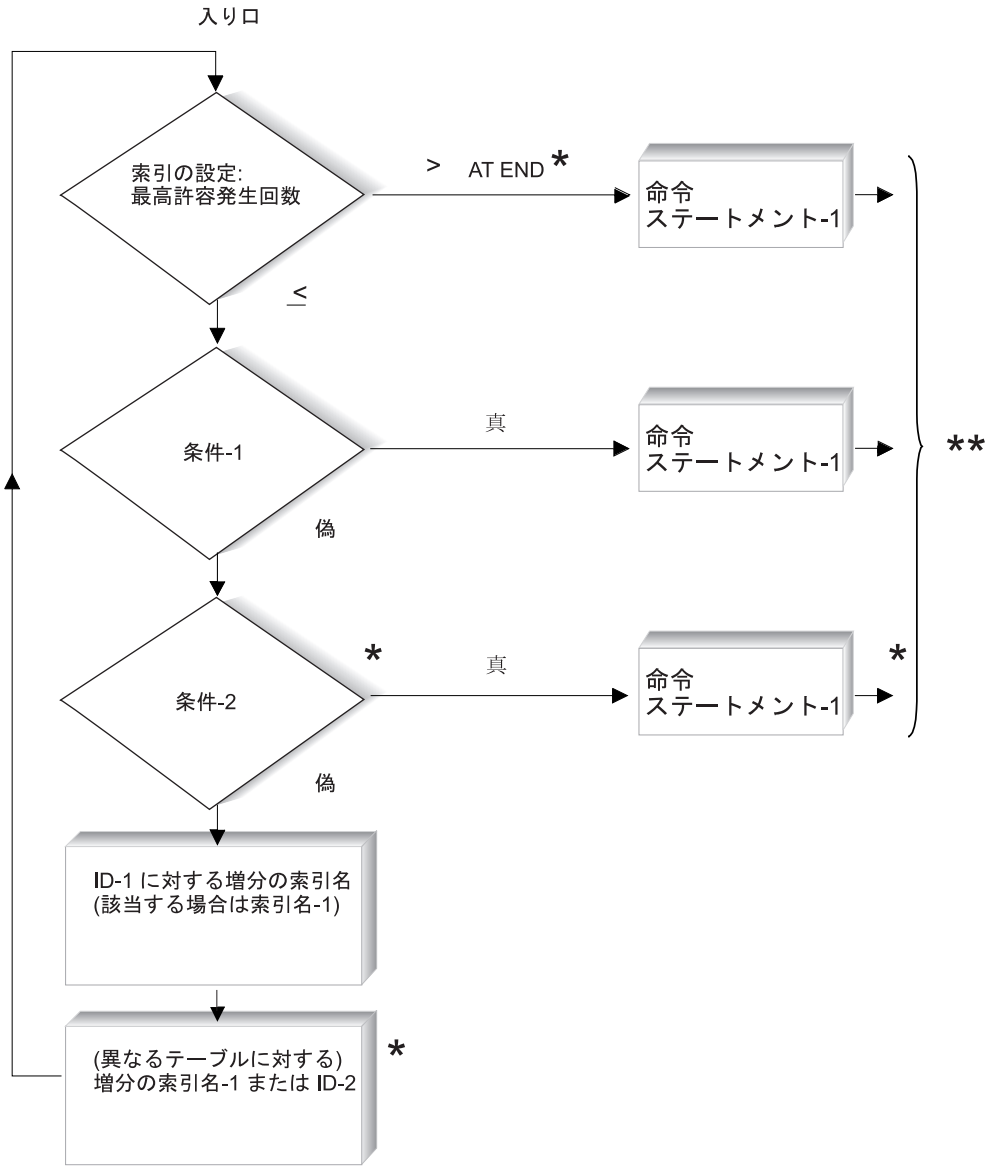
VARYING 指標名-1 句が省略されている場合には、ID-1 についての最初の (または唯一の) 指標名が検索に使用されます。

ID-2

ID-2 は、指標データ項目または基本整数項目のいずれかでなければなりません。検索中には、次のいずれかの処理が適用されます。

- ID-2 が指標データ項目である場合には、検索指標が増やされるたびに、指定された指標データ項目が同時に同じだけ増やされます。
- ID-2 が整数データ項目である場合には、検索指標が増やされるたびに、指定されたデータ項目が同時に 1 だけ増やされます。

図 7-10 は、2 つの WHEN 句を持つ形式 1 の SEARCH 操作を示しています。



* この操作は、ステートメントの中で要求された場合に限り含まれます。

** 命令ステートメントが GO TO ステートメントで終了しない限り、制御は次の文に転送されます。

* この操作は、ステートメントの中で要求された場合に限り含まれます。

** 命令ステートメントが GO TO ステートメントで終了しない限り、制御は次の文に転送されます。

図 7-10. 2 つの WHEN 句を持つ形式 1 の SEARCH

二分検索

形式 2 の SEARCH ALL ステートメントでは、二分検索が実行されます。 検索指標は、その設定値が最初のテーブル・エレメントの値より小さくならず、また最後のテーブル・エレメントの値より大きくならないように検索操作中に変化するので、SET ステートメントによって初期設定する必要はありません。使用される指標は、常に、OCCURS 文節で指定された最初の指標名に関連する指標です。

ID-1

OCCURS 文節を含むデータ項目に従属するデータ項目を指定できます。つまり、2 ～ 7 次元テーブルの一部を指定できます。この場合には、データ記述記入項目で、テーブルの各次元について INDEXED BY 句が指定されていなければなりません。

検索を行う前に、特定のテーブルの ID-1 のエレメントを定義するために、すべての指標の値をより高い次元のテーブル用に設定することが必要です。

ID-1 は、テーブル・エレメント内のすべてのオカレンスを指していなければなりません。つまり、添え字付けまたは指標付けされてはなりません。

ID-1 は、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・データ項目にすることはできません。

IBM Extension

ID-1 は、浮動小数点データ項目にすることができません。

End of IBM Extension

IBM Extension

ASCENDING/DESCENDING KEY が DBCS データ項目として定義される場合、 ID-1 は DBCS データ項目にすることができます。

End of IBM Extension

IBM Extension

ASCENDING/DESCENDING KEY が日時データ項目として定義される場合、 ID-1 は日時データ項目にすることができます。

End of IBM Extension

ID-1 のデータ部の記述には、INDEXED BY オプションを伴う OCCURS 文節が入っていなければなりません。形式-2 の場合は、データ部の記述の OCCURS 文節に KEY IS 句も入っていなければなりません。

WHEN 句

WHEN 句がこの範囲内の指標の設定値について**満たされない**場合には、検索は失敗します。

WHEN オプションが**満たされた**場合には、制御が命令ステートメント-2 に渡され、WHEN 条件を満たしたオカレンスを示す値が指標に入ります。

条件名-1、条件名-2
指定される各条件名は、値を 1 つだけもたなければならず、また、それぞれがこのテーブル・エレメントについての ASCENDING/DESCENDING KEY ID と関連付けられなければなりません。

データ名-1、データ名-2
ID-1 のテーブル・エレメント内の ASCENDING/DESCENDING KEY データ項目を指定しなければなりません。さらに、ID-1 の最初の指標名と、他の指標またはリテラル (必要であれば) によって指標付けされなければなりません。各データ名は修飾できます。

IBM Extension

データ名-1またはデータ名-2 を浮動小数点データ項目にすることはできません。データ名-1 またはデータ名-2 は、日時データ項目にすることができます。

End of IBM Extension

ID-3、ID-4
ID-1 についての ASCENDING/DESCENDING KEY データ項目、または ID-1 についての最初の指標名によって指標付けされた項目であってはなりません。

ポインターまたはプロシージャー・ポインター・データ項目であってはなりません。

IBM Extension

浮動小数点データ項目にすることも、日時データ項目にすることもできます。

End of IBM Extension

算術式-1、算術式-2
7-8 ページの『算術式』で定義されている任意の式にすることができますが、次の制約があります。つまり、算術式内の ID は、ID-1 についての ASCENDING/DESCENDING KEY データ項目、または ID-1 についての最初の指標名によって指標付けされた項目であってはなりません。

WHEN 句の中で ASCENDING/DESCENDING KEY データ項目が暗黙または明示的に指定される場合には、 ID-1 についてのすべての先行 ASCENDING/DESCENDING KEY データ名も指定されなければなりません。

SEARCH ALL 操作の結果は、次の場合に**だけ**予測できます。

- テーブル内のデータが ASCENDING/DESCENDING KEY 句に従って順序付けられている。
- WHEN 文節で指定された ASCENDING/DESCENDING KEY の内容によって、固有なテーブル参照がもたらされる。

SEARCH ステートメントに関する考慮事項

指標データ項目は、それに対する直接参照の制約があるために、添え字として使用することはできません。

同じ指標名に対して相対指標付け参照と一緒に直接指標付け参照も使用することによって、テーブル・エレメントの 2 つの異なるオカレンスを比較の目的で参照できます。

VARYING オプションの対象が他のテーブル・エレメントについての指標名である場合には、形式 1 の SEARCH ステートメント 1 つで、2 つのテーブル・エレメントを同時に進むことができます。

可変長テーブルに対して SEARCH ステートメントが正しく実行されるようにするためには、OCCURS DEPENDING ON 文節の対象 (データ名-1) に、テーブルの現在の長さを指定する値を含めるようにしなければなりません。

SEARCH ステートメントの範囲は、次のいずれかによって終了させることができます。

- 同じネスト・レベルの END-SEARCH 句
- 分離文字ピリオド
- 直前の IF ステートメントに関連する ELSE または END-IF 句

SEARCH の例

次の例では、入力データと一致する品目を見つけるために在庫テーブルを検索します。キーは ITEM-NUMBER です。

```
.. 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

 DATA DIVISION.
 FILE SECTION.
 FD  SALES-DATA
     BLOCK CONTAINS 1 RECORDS
     RECORD CONTAINS 80 CHARACTERS
     LABEL RECORDS STANDARD
     DATA RECORD IS SALES-REPORTS.
 01  SALES-REPORTS                  PIC X(80).
 FD  PRINTED-REPORT
     BLOCK CONTAINS 1 RECORDS
     RECORD CONTAINS 132 CHARACTERS
     LABEL RECORDS OMITTED
     DATA RECORD IS PRINTER-OUTPUT.
 01  PRINTER-OUTPUT                 PIC X(132).
 FD  INVENTORY-DATA
     BLOCK CONTAINS 1 RECORDS
     RECORD CONTAINS 40 CHARACTERS
     LABEL RECORDS STANDARD
     DATA RECORD IS INVENTORY-RECORD.
 01  INVENTORY-RECORD.
     03  I-NUMBER                  PIC 9(4).
     03  INV-ID                    PIC X(26).
     03  I-COST                    PIC 9(8)V99.
 WORKING-STORAGE SECTION.
 01  EOF-SW              PIC X      VALUE "N".
 01  EOF-SW2             PIC X      VALUE "N".
 01  SUB1                PIC 99.
 01  RECORDS-NOT-FOUND   PIC 9(5)   VALUE ZEROS.
 01  TOTAL-COSTS         PIC 9(10)  VALUE ZEROS.
 01  HOLD-INPUT-DATA.
     03  INVENTORY-NUMBER          PIC 9999.
     03  PURCHASE-COST             PIC 9(4)V99.
     03  PURCHASE-DATE             PIC 9(6).
     03  FILLER                    PIC X(64).
 01  PRINTER-SPECS.
     03  PRINT-LINE.
         05  OUTPUT-ITEM-NUMBER    PIC ZZZ9.
         05  FILLER                PIC X(48) VALUE SPACES.
         05  TOTAL-COSTS-0         PIC $(8).99.
 01  PRODUCT-TABLE.
     05  INVENTORY-NUMBERS         OCCURS 50 TIMES
                                   ASCENDING KEY ITEM-NUMBER
                                   INDEXED BY INDEX-1.
             07  ITEM-NUMBER       PIC 9(4).
             07  ITEM-DESCRIPTION  PIC X(26).
             07  ITEM-COST         PIC 9(8)V99.
```

```
.. 1  ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ...7

 PROCEDURE DIVISION.
 100-START-IT.
     OPEN INPUT SALES-DATA INVENTORY-DATA OUTPUT PRINTED-REPORT.
     MOVE HIGH-VALUES TO PRODUCT-TABLE.
     PERFORM READ-INVENTORY-DATA.
 LOAD-TABLE-ROUTINE.
     PERFORM LOAD-IT VARYING SUB1 FROM 1 BY 1 UNTIL SUB1 > 50
                     OR EOF-SW2 = "Y".
     PERFORM 110-READ-IT.
 200-MAIN-ROUTINE.
     PERFORM PROCESS-DATA UNTIL EOF-SW = "Y".
     MOVE TOTAL-COSTS TO TOTAL-COSTS-0.
     PERFORM WRITE-REPORT THRU WRITE-REPORT-EXIT.
     DISPLAY "RECORDS NOT FOUND - " RECORDS-NOT-FOUND
       UPON MYTUBE.
 STOP RUN.
 PROCESS-DATA.
     SEARCH ALL INVENTORY-NUMBERS
       AT END PERFORM KEY-NOT-FOUND THRU NOT-FOUND-EXIT
       WHEN ITEM-NUMBER (INDEX-1) = INVENTORY-NUMBER
          MOVE ITEM-NUMBER (INDEX-1) TO OUTPUT-ITEM-NUMBER
          MOVE ITEM-COST (INDEX-1) TO TOTAL-COSTS-0
          ADD ITEM-COST (INDEX-1) TO TOTAL-COSTS
          PERFORM WRITE-REPORT THRU WRITE-REPORT-EXIT.
     PERFORM 110-READ-IT.
 KEY-NOT-FOUND.
     ADD 1 TO RECORDS-NOT-FOUND.
 NOT-FOUND-EXIT.
     EXIT.
 LOAD-IT.
     MOVE INVENTORY-RECORD TO INVENTORY-NUMBERS (SUB1).
     PERFORM READ-INVENTORY-DATA.
 WRITE-REPORT.
     WRITE PRINTER-OUTPUT FROM PRINTER-SPECS.
 WRITE-REPORT-EXIT.
     EXIT.
 READ-INVENTORY-DATA.
     READ INVENTORY-DATA
       AT END MOVE "Y" TO EOF-SW2.
 110-READ-IT.
     READ SALES-DATA INTO HOLD-INPUT-DATA
       AT END MOVE "Y" TO EOF-SW.
```

SET ステートメント

SET ステートメントは、以下のことを行うために使用できます。

- テーブル・エレメントを参照するのに使用される指標名または ID の値を初期設定する
- 指標名の値を増加または減少させる
- 外部スイッチの状況を ON または OFF に設定する
- 条件変数条件を真にするためにデータを移動する

IBM Extension

- ポインターおよびプロシージャー・ポインター・データ項目と ADDRESS OF 特殊レジスターを設定する
- 現行ロケールのロケール・カテゴリーを設定および照会する。

End of IBM Extension

SET ステートメント内の送り出しフィールドと受け入れフィールドが記憶域の一部を共有している (すなわち、オペランドがオーバーラップしている) 場合には、その SET ステートメントの実行結果は未定義です。

IBM Extension

- 形式 5 - ポインター・データ項目
- 形式 6 - プロシージャー・ポインター・データ項目
- 形式 7 - ポインターの調整
- 形式 8 - ロケール

End of IBM Extension

形式 1 - 指標名、ID の初期設定

形式 1 の SET ステートメントが実行されると、受け入れフィールドの現行値が送り出しフィールドの値によって置き換えられます (変換を伴います)。

SET ステートメント - 形式 1

指標名-1、ID-1

受け入れフィールド。

指標データ項目または基本数字整数項目のいずれかでなければなりません。

IBM Extension

ID-1 は、浮動小数点データ項目にすることができません。

End of IBM Extension

指標名-2

送り出しフィールド。

SET ステートメント実行前の指標名-2 の値は、それに関連するテーブルのオカレンス番号と一致していなければなりません。

ID-2

送り出しフィールド。

指標データ項目または基本数字整数項目のいずれかでなければなりません。

IBM Extension

ID-2 は、浮動小数点データ項目にすることができません。

End of IBM Extension

整数-1
送り出しフィールド。

正の整数でなければなりません。

形式 1 の SET ステートメントの実行方法は、次のように受け入れフィールドのタイプによって異なります。

- 指標名の受け入れフィールド (指標名-1 など) は、通常、送り出しフィールドによって指示されたオカレンス番号を表す変位値に変換されます。結果としての指標名の値が有効になるのは、関連するテーブル・エレメントのオカレンス番号と一致している場合です。 1 つの例外として、送り出しフィールドが指標データ項目である場合には、指標データ項目の値は変換されずに指標名へ入れられます。
- 指標データ項目の受け入れフィールド (ID-1 など) は、送り出しフィールドの内容 (指標名または指標データ項目でなければならない) と等しく設定されます。変換は行われません。数字整数またはリテラルの送り出しフィールドを指定してはなりません。
- 整数データ項目の受け入れフィールド (ID-1 など) は、送り出しフィールド (指標名でなければならない) に関連付けられたオカレンス番号に設定されます。整数データ項目、指標データ項目、またはリテラルの送り出しフィールドを指定してはなりません。

表 7-14 に、形式 1 の SET ステートメントでの送り出しフィールドと受け入れフィールドの有効な組み合わせを示します。

表 7-14. 形式 1 の SET ステートメントに関する送り出しフィールドおよび受け入れフィールド

送り出しフィールド	**受け入れフィールド**		
	指標名	**指標データ項目**	**整数データ項目**
指標名	有効	有効	有効
指標データ項目	有効*	有効*	—
整数データ項目	有効	—	—
整数リテラル	有効	—	—
* 変換は行われません。			

受け入れフィールドは、それらが指定された順序で左から右に処理されます。 ID の受け入れフィールドに関連する添え字付けまたは指標付けは、フィールドが処理される直前に評価されます。

送り出しフィールドに使用される値は、SET ステートメントの実行開始時の値です。

SEARCH または PERFORM ステートメントの実行後には、指標名の値が未定義になることがあります。このため、他のテーブル処理操作が試みられる前に、形式 1 の SET ステートメントによってそのような指標名を再び初期設定しなければなりません。

IBM Extension

指標名-2 が OCCURS DEPENDING ON 文節を含む従属項目を持つテーブルを参照する場合、指標名-1 は未定義の値を受け取る可能性があります。詳細については 9-46 ページの『付録 H. 複合 OCCURS DEPENDING ON』を参照してください。

End of IBM Extension

形式 2 - 指標値の調整

形式 2 の SET ステートメントが実行されると、受け入れフィールドの値が、送り出しフィールド内の値と一致する値だけ増加 (UP BY) または減少 (DOWN BY) されます。

SET ステートメント - 形式 2

指標名-3
この指標名の値は、SET ステートメントの実行前と実行後の両方で、関連するテーブル内のオカレンス番号と一致していなければなりません。

ID-3
この送り出しフィールドは基本整数データ項目でなければなりません。

IBM Extension

ID-3 は、浮動小数点データ項目にすることができません。

End of IBM Extension

整数-2
この送り出しフィールドは整数でなければなりません。

形式 2 の SET ステートメントが実行されると、受け入れフィールドの値が、 ID-3 または整数-2 の値によって表されているオカレンスの番号と一致する値だけ増加 (UP BY) または減少 (DOWN BY) されます。

IBM Extension

指標名-3 が OCCURS DEPENDING ON 文節を含む従属項目を持つテーブルを参照し、形式 2 の SET ステートメントの実行前に、この ODO オブジェクトが変更された場合、指標名-3 には、関連するテーブルのオカレンス番号に対応する値が含まれていない可能性があります。詳細については 9-46 ページの『付録 H. 複合 OCCURS DEPENDING ON』を参照してください。

End of IBM Extension

受け入れフィールドは、それらが指定された順序で左から右に処理されます。 SET ステートメントの実行開始時点でのフィールドの増加または減少の値は、すべての受け入れフィールドで使用されます。

形式 3 - 外部スイッチの設定

形式 3 の SET ステートメントが実行されると、指定された簡略名に関連するそれぞれの外部スイッチの状況が ON または OFF になります。

SET ステートメント - 形式 3

簡略名

状況を変更できる外部スイッチと関連付けられなければなりません。

形式 3 の場合、各簡略名は、状況を変更できる外部スイッチと関連付けられなければなりません。使用できる外部スイッチは、UPSI-0 から UPSI-7 までの UPSI スイッチだけです。

指定された簡略名と関連する各外部スイッチの状況は、そのスイッチと関連する条件名の評価の結果としての真理値が、ON 句が指定された場合にはオンの状況、OFF 句が指定された場合にはオフの状況を反映するように変更されます。

形式 4 - 条件名

形式 4 の SET ステートメントが実行されると、条件名に関連する値がそれの条件変数に入れられます。

SET ステートメント - 形式 4

条件名-1

条件変数に関連付けられなければなりません。

条件名-1 の VALUE 文節に複数のリテラルが指定されると、それに関連する条件変数は最初のリテラルに設定されます。

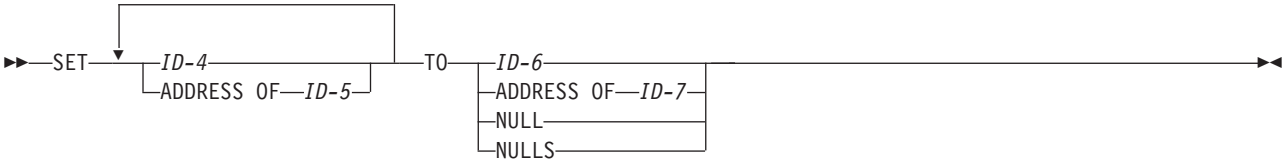
複数の条件名が指定された場合、結果は、それぞれの条件名について、それらが指定されているのと同じ順序で別々の SET ステートメントが書かれた場合と同じになります。

形式 5 - ポインター・データ項目

IBM Extension

形式 5 の SET ステートメントが実行されると、 受け入れフィールドの現行値が送り出しフィールドに含まれるアドレス値によって置き換えられます。

SET ステートメント - 形式 5



ID-4

受け入れフィールド。

これは、USAGE IS POINTER として記述されていなければなりません。

ADDRESS OF ID-5
受け入れフィールド。

これは ADDRESS OF 特殊レジスターです。

リンケージ・セクションで定義されたレベル 01 またはレベル 77 の項目でなければなりません。 TO 句で指定されたオペランドの値に設定されます。添え字付けまたは参照変更することはできません。

ID-6
送り出しフィールド。

これは、USAGE IS POINTER として記述されていなければなりません。

プログラム自体の作業用ストレージ・セクション、ローカル・ストレージ・セクションまたはファイル・セクション内のアドレスが入っていてはなりません。

ADDRESS OF ID-7
送り出しフィールド。

データ部のセクションのレベル 66 または 88 以外の項目でなければなりません。

ADDRESS OF ID-7 には、ID の内容ではなく、そのアドレスが入ります。 ID-7 は、添え字付けまたは参照変更 (あるいはその両方) が可能です。

NULL、NULLS
送り出しフィールド。

受け入れフィールドを、無効なアドレスの値を含むように設定します。

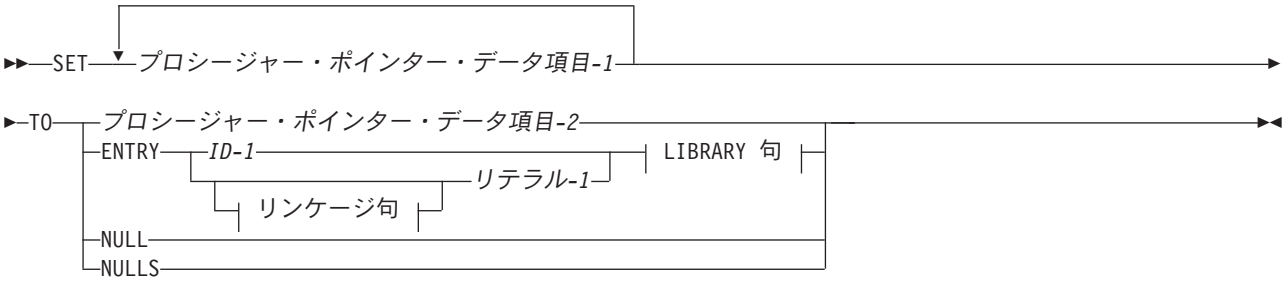
End of IBM Extension

形式 6 - プロシージャー・ポインター・データ項目

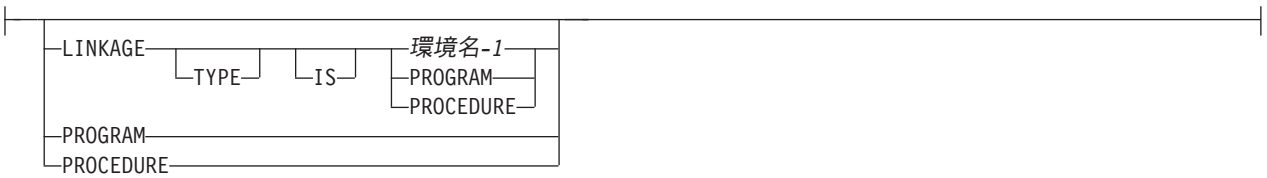
IBM Extension

SET ステートメントの形式 6 が実行されると、 受け入れフィールドの現行値が送り出しフィールドに含まれるアドレス値によって置き換えられます。

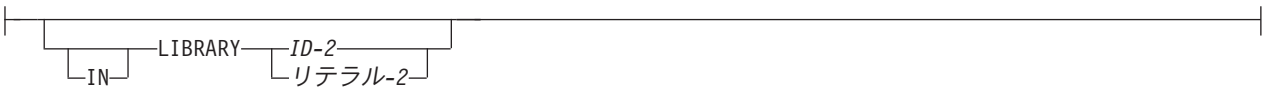
SET ステートメント - 形式 6



リンケージ句:



LIBRARY 句:



プロシージャー・ポインター・データ項目-1、プロシージャー・ポインター・データ項目-2
プロシージャー・ポインター・データ項目-1 は受け入れフィールドです。

USAGE IS PROCEDURE-POINTER として記述されていなければなりません。

ID-1
値をプログラム名にすることができるように、英数字項目として定義されていなければなりません。(詳細は 4-2 ページの『PROGRAM-ID 段落』を参照してください。) プロシージャー・ポインター・データ項目は、同じコンパイル単位の最外部の COBOL プログラム (ILE プロシージャー)、または ID-1 で指定されたプログラム・オブジェクト (*PGM) に設定されます。 ID の内容は、CRTCBLMOD または CRTBNDCBL コマンドの *MONOPRC オプションによって影響されます。

リテラル-1
非数字でなければならず、プログラム名の形成に関する規則に従っていなければなりません。リテラルは、CRTCBLMOD または CRTBNDCBL コマンドの *MONOPRC オプションによって影響されます。プロシージャー・ポインター・データ項目は、同じコンパイル単位の最外部の COBOL プログラム (ILE プロシージャー)、他のコンパイル単位内の最外部の COBOL プログラム (ILE プロシージャー)、ILE プロシージャー (他の ILE 言語で書かれた)、またはプログラム・オブジェクト (*PGM) に設定できます。プロシージャー・ポインター・データ項目は、指定された名前のネストされた COBOL プログラムが SET のポイントから可視である場合でも、ネストされた COBOL プログラム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・データ項目が設定されるオブジェクトのタイプは、ENTRY 文節の LINKAGE TYPE 句と、SPECIAL-NAMES 段落の LINKAGE TYPE 文節および CRTCBLMOD または CRTBNDCBL コマンドの LINKLIT パラメーターによって判別されます。

LINKAGE TYPE 句: LINKAGE TYPE 句は、プロシージャー・ポインター・データ項目が設定されるプログラムのタイプを指定するために使用されます。個別にコンパイルされたプログラム・オブジェクト (*PGM) のアドレスまたはプログラム内のプロシージャーに設定できます。

環境名-1
プロシージャー・ポインター・データ項目-1 が設定されるプログラムのタイプ。環境名-1 は以下のものとして定義できます。

- PGM (プログラム・オブジェクト、すなわち *PGM)
- PRC (プロシージャー)

PROGRAM
プロシージャー・ポインター・データ項目-1 はプログラム・オブジェクト (*PGM) に設定されます。

PROCEDURE
プロシージャー・ポインター・データ項目-1 はプロシージャーに設定されます。

NULL(S)
受け入れフィールドを、無効なアドレスの値を含むように設定します。

IN LIBRARY 句: IN LIBRARY 句は、IBM i プログラム・オブジェクトにプロシージャー・ポインター・データ項目を設定する場合のみ有効です。つまり、プログラムのリンケージは、SET ステートメントで暗黙または明示的に指定されなければなりません。

ID-2
英数字データ項目でなければなりません。 ID-2 の内容は、有効な IBM i ライブラリー名を表していなければなりません。 IBM i ライブラリー名は、最大 10 文字の長さです。したがって ID-2 の最初の 10 文字が、ライブラリー名を形成するために使用されます。

リテラル-2
非数字でなくてはならず、最大 10 文字まで可能です。

ID-2 およびリテラル-2 は *MONOPRC コンパイラー・オプションの影響を受けず、また、IBM i 拡張名を含むことができます。

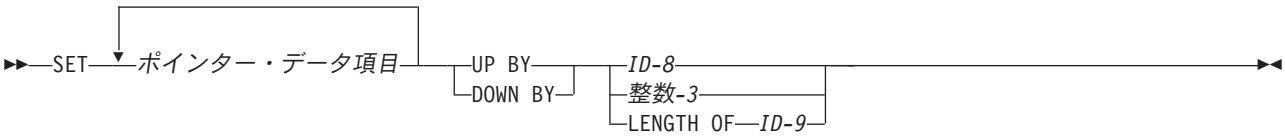
End of IBM Extension

形式 7 - ポインターの調整

IBM Extension

形式 7 の SET ステートメントが実行されると、ポインター・データ項目に含まれるアドレスが、送り出しフィールドの値に一致する値だけ増加 (UP BY) または減少 (DOWN BY) されます。

SET ステートメント - 形式 7



ポインター・データ項目
受け入れフィールドは、USAGE IS POINTER を持つ基本データ項目でなければなりません。

ID-8
この送り出しフィールドは基本整数データ項目でなければなりません。

ID-8 は、浮動小数点データ項目にすることができません。

整数-3
この送り出しフィールドは整数でなければなりません。

ID-9
ID-9 に関する規則の詳細は 7-78 ページの『LENGTH OF 特殊レジスター』を参照してください。

End of IBM Extension

形式 8 - ロケール

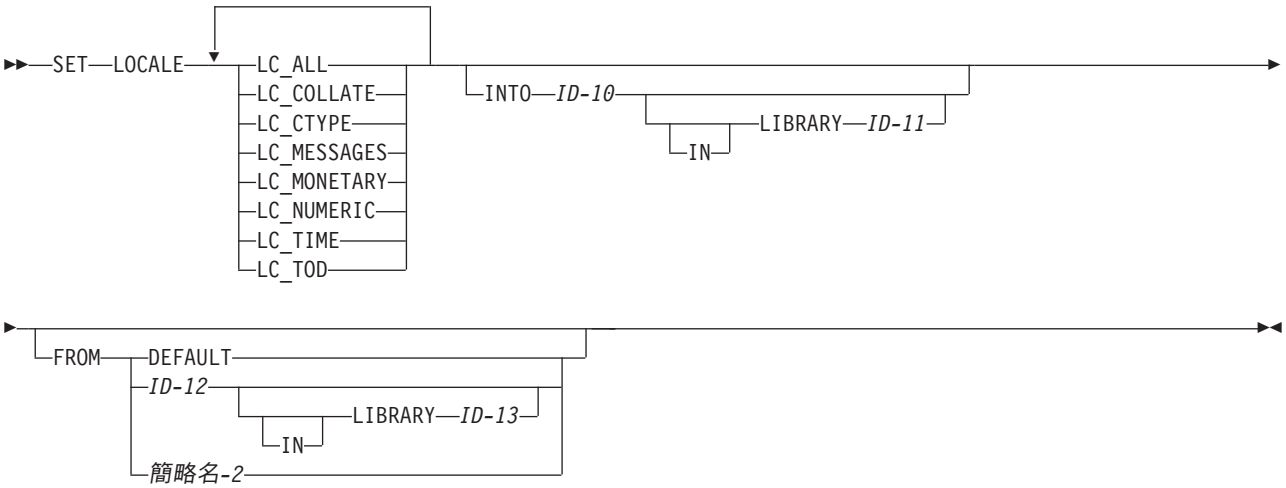
IBM Extension

SET ステートメントの形式 8 によって、現行ロケールのロケール・カテゴリーの設定と照会が可能となります。**ロケール**は、言語や特定の文化圏固有の情報を含むシステム・オブジェクトです。例えば、ロケールには世界の特定の地域の日付と時刻についての適切な形式が入っています。ロケール内の情報は、**ロケール・カテゴリー**に分割されます。例えば、ロケール・カテゴリー LC_TIME には、日付と時刻の形式に関する情報が入っています。実行単位ごとに、1 つの DEFAULT ロケール、1 つの現行ロケール、およびゼロないし多数の特定のロケールが存在します。現行ロケールは、そのロケール・カテゴリーの一部またはすべてを DEFAULT または特定のロケールに設定することにより変更されます。ロケール・カテゴリー (現行ロケールの) が設定された特定ロケールの名前を ID の中で使用できます。ロケール・カテゴリーの内容は、下記からロケール・カテゴリーを設定することにより変更できます。

- システム・デフォルト
- 英数字基本データ項目において定義されたロケール
- SPECIAL-NAMES 段落で指定された簡略名

指定された各ロケール・カテゴリーは、実行単位の期間中またはカテゴリーを指定する別の SET ステートメントが処理されるまでの間効力をもちます。

SET ステートメント - 形式 8



LC_ALL
ロケール・カテゴリーの LC_COLLATE、LC_CTYPE、LC_MESSAGES、LC_MONETARY、LC_NUMERIC、LC_TIME、および LC_TOD (ロケールに含まれるこれ以外のすべてのカテゴリーも含まれます。)

LC_COLLATE
照合順序を定義するロケール・カテゴリー。

LC_CTYPE
文字種別と文字タイプを定義するロケール・カテゴリー。

LC_MESSAGES
通知メッセージと診断メッセージのフォーマット設定および対話式応答のフォーマット設定を定義するロケール・カテゴリー。

LC_MONETARY
通貨のフォーマット設定を定義するロケール・カテゴリー。

LC_NUMERIC
数字のフォーマット設定を定義するロケール・カテゴリー。

LC_TIME
日時フォーマット設定を定義するロケール・カテゴリー。

LC_TOD
時間帯差、時間帯名、および夏時間開始点と終点を定義するロケール・カテゴリー。

ID-10
ID-10 の値はロケール・カテゴリーを参照します。 ID-10 は基本英数字データ項目でなければなりません。 INTO 句が指定されている場合は、指定されたカテゴリーに対する現行ロケールの識別は、ID-10 が参照するデータ項目の中に保管されます。 INTO 句は、英数字から英数字への移動用 MOVE ステートメントの規則を使用して、FROM 句の前に処理されます。

DEFAULT
ロケール・カテゴリーを現行のデフォルトに設定します。デフォルトのロケールは、実行単位が起動する時点で発生し、実行単位の期間中デフォルトのロケールのまま残ります。デフォルトのロケールはまた、実行単位が活動化される時点で現行ロケールにもなり、SET ステートメントの形式 8 を使用して切り換えられるまで現行ロケールのまま残ります。

ID-12
ID-10 の値はロケール・カテゴリーを参照します。 ID-12 は基本英数字データ項目を参照する必要があります。 ID-12 で指定されたロケールが利用不能である場合は、オペレーティング・システムのエスケープ・メッセージが出されます。 FROM 句が指定されている場合は、指定されたカテゴリーの現行ロケールは、ID-12 が参照するデータ項目の内容に設定されます。現行ロケール識別は、英数字から英数字への移動用 MOVE ステートメントの規則を使用して保管されます。

簡略名-2
簡略名-2 で指定されたロケールが利用不能である場合は、オペレーティング・システムのエスケープ・メッセージが出されます。FROM 句が指定されている場合は、指定されたカテゴリーの現行ロケールは、簡略名-2 が識別するロケール・カテゴリーに設定されます。

IN LIBRARY 句: IN LIBRARY 句を使用して、ロケール・オブジェクトが置かれる IBM i ライブラリーを指定します。 INTO 文節の場合、ID-11 は指定されたロケール・カテゴリーのライブラリー名を使用して更新されます。 FROM 文節の場合は、ID-12 を使用してロケール・カテゴリーの設定先であるロケール・オブジェクトを位置指定します。

ID-11、ID-13
基本英数字データ項目でなければなりません。 ID-11 または ID-13 の内容は、有効な IBM i ライブラリー名を表していなければなりません。 IBM i ライブラリー名は、最大 10 文字の長さです。したがって ID-2 の最初の 10 文字が、ライブラリー名を形成するために使用されます。

ID-13 が指定されていない場合は、ライブラリー *LIBL が想定されます。そうでない場合は、ID-13 には、ID-12 で指定されたロケール・オブジェクト名が存在するライブラリーが入っていなければなりません。 ID-11 が指定される場合は、この ID には現行のロケール・カテゴリーの最後の設定先であるロケール・オブジェクトのライブラリー名が入っています。 ID-10 内のロケール名が DEFAULT である場合は、ID-11 はスペースに設定されます。

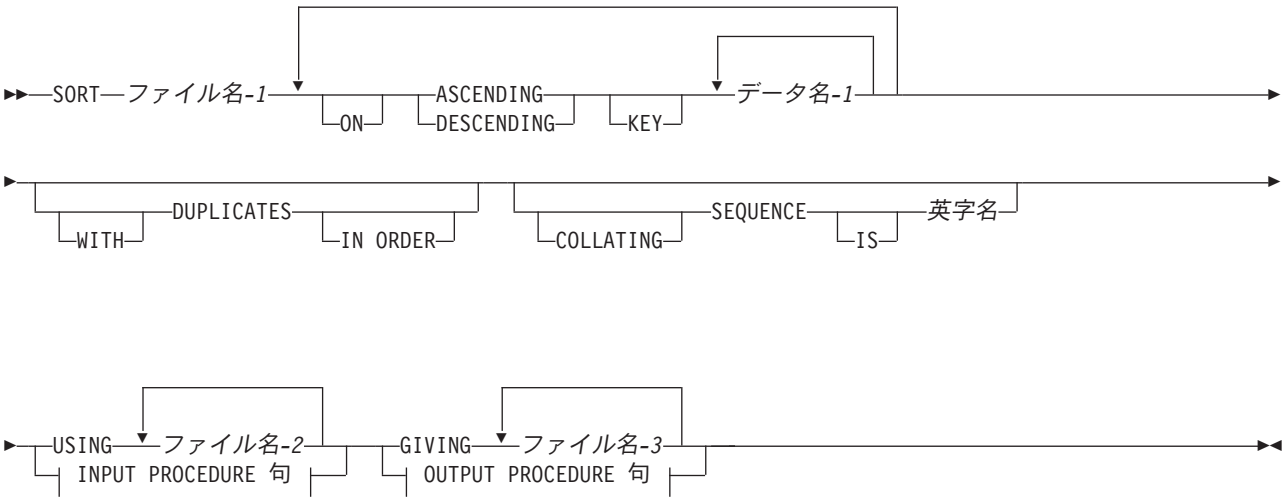
ID-11 および ID-13 は *MONOPRC コンパイラー・オプションの影響を受けず、また、IBM i 拡張名を含むことができます。

End of IBM Extension

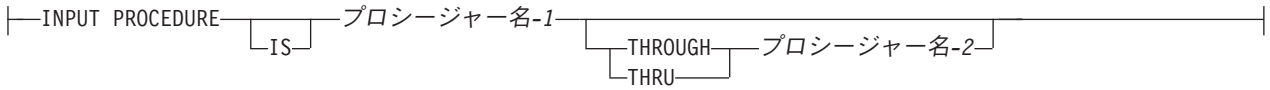
SORT ステートメント

SORT ステートメントは、1 つまたは複数のファイルからレコードを受け入れ、指定されたキーに従ってそれらをソートし、ソートしたレコードを OUTPUT PROCEDURE や出力ファイルで使用できるようにします。 SORT ステートメントは、宣言セクション内でなければ手続き部のどこに記入してもかまいません。 USING または GIVING ファイルの最大数は 32 です。

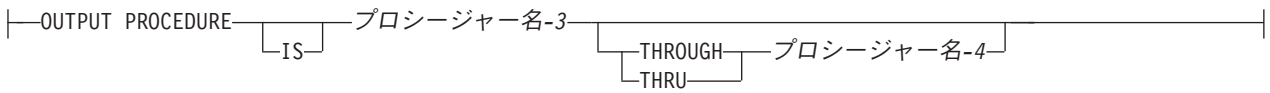
SORT ステートメント - 形式



INPUT PROCEDURE 句:



OUTPUT PROCEDURE 句:



ファイル名-1
SD 記入項目で指定された、ソートされるレコードを記述する名前。

ヌル可能フィールドはサポートされますが、ヌル値は ASSIGN 文節で ALWNULL が指定された DATABASE ファイルにだけサポートされます。 ALWNULL が指定されず、フィールドにヌル値が含まれている場合は、SORT 操作は失敗し、ファイル状況の 90 が戻されます。

ASCENDING/DESCENDING KEY 句

この句は、指定されたソート・キーに基づいてレコードが昇順または降順 (指定された句に従います) に処理されるように指定します。

データ名-1

データ名-1 は、ソートの基礎となる KEY データ項目を指定します。各データ名は、**ファイル名-1** に関連付けられたレコード内のデータ項目を識別しなければなりません。次の規則が適用されます。

- 特定の KEY データ項目は、各入力ファイル内で物理的に同じ位置に存在し、同じデータ・フォーマットをもっていなければなりません。ただし、同じデータ名を持つ必要はありません。
- ファイル名-1 が複数のレコード記述をもっている場合には、KEY データ項目は 1 つだけのレコード記述中で記述されている必要があります。
- ファイル名-1 に可変長レコードが含まれている場合には、すべての KEY データ項目はレコードの最初の *n* 文字の位置に入っていなければなりません。ここで *n* は、ファイル名-1 について指定された最大レコード・サイズです。
- KEY データ項目は、OCCURS 文節を含んではならず、また、OCCURS 文節を含む項目に従属してもなりません。
- KEY データ項目の全長が 2000 バイトを超えてはなりません。
- KEY データ項目を修飾することはできますが、添え字付けまたは指標付けすることはできません。
- KEY データ項目の位置は可変であってはなりません。
- SORT キーの中で可変長フィールドを可変長フィールドとして使用することはできません。可変長フィールドは、ILE COBOLによってグループ項目に変換されます。可変長フィールドはグループ項目に変換されるため、SORT キーの中で使用されると英数字データ項目として比較されます。

SORT ステートメントは、KEY データ項目を、それらの項目がどのように KEY 句に分割されているかにかかわりなく、重要度の高いものから順に左から右に列挙します。左端のデータ名が主キーであり、次のデータ名がその次に重要度の高いキーというようになります。

ソート操作の方向は、キーワード ASCENDING または DESCENDING の指定によって、次のように異なります。

- ASCENDING を指定すると、最下位キー値から最上位キー値への順序となります。
- DESCENDING を指定すると、最上位キー値から最下位キー値への順序となります。
- KEY データ項目が英字、英数字、英数字編集、または数字編集である場合には、キー値の順序は使用される照合順序によって決まります (7-216 ページの『COLLATING SEQUENCE 句』を参照してください)。
- KEY データ項目が DBCS、DBCS 編集、または国別である場合には、キー値の順序は DBCS 文字または国別文字の 16 進数値の 2 進数照合順序に基づきます。 COLLATING SEQUENCE 句は無視されます。

IBM Extension

- KEY データ項目は、浮動小数点項目または日時項目にすることができます。
- KEY データ項目を参照変更することはできますが、添え字付けまたは指標付けすることはできません。
- KEY が外部浮動小数点項目である場合、コンパイラーはデータ項目を、数字データではなく文字データとして取り扱います。レコードがソートされる順序は、使用される照合順序によって異なります。
- KEY データ項目が内部浮動小数点である場合、キー値の順序は数字の順序になります。
- KEY が日時項目である場合は、いくつかの形式だけが日付項目または時刻項目としてソートされます。ILE COBOL は、IBM i DDS より多くの日時形式をサポートします。一般的に、IBM i DDS 形式に一

致する ILE COBOL の日時形式は、日付項目または時刻項目としてソートされます。その他のすべての形式は英数字項目として扱われ、その 16 進値に基づいてソートされます。

End of IBM Extension

- キーの比較は、比較条件のオペランドの比較に関する規則に従って実行されます (7-10 ページの『条件式』の『比較条件』を参照してください)。

DUPLICATES 句

DUPLICATES 句が指定されており、1 つのレコードに関連するすべてのキー・エレメントの内容が、他の 1 つまたは複数のレコード内の対応するキー・エレメントと等しい場合には、これらのレコードの順序は次のようになります。

- SORT ステートメント内に指定されている関連する入力ファイルの順序。特定のファイル内では、順序はレコードがそのファイルの中でアクセスされる順序です。
- これらのレコードが入力プロシージャーによって解放される順序 (入力プロシージャーが指定されている場合)。

DUPLICATES 句が指定されていない場合には、これらのレコードの順序は未定義です。

COLLATING SEQUENCE 句

この句は、このソート操作において KEY データ項目の非数字比較に使用される照合順序を指定します。

英字名
SPECIAL-NAMES 段落の英字名文節の中で指定されていなければなりません。英字名文節オプションの任意の 1 つを指定できます。英字名文節オプションおよびその意味のリストについては 5-4 ページの『SPECIAL-NAMES 段落』を参照してください。

COLLATING SEQUENCE 句を省略した場合には、OBJECT-COMPUTER 段落内の PROGRAM COLLATING SEQUENCE 文節 (指定されている場合) によって、使用される照合順序が決まります。

COLLATING SEQUENCE 句および PROGRAM COLLATING SEQUENCE 文節の両方を省略した場合には、EBCDIC 照合順序が使用されます。

USING 句

ファイル名-2, ...
入力ファイル。

USING 句を使用した場合には、**ファイル名-2**、... (つまり、入力ファイル) のすべてのレコードが自動的にファイル名-1 に転送されます。 SORT ステートメントの実行時には、これらのファイルがオープンされていてはなりません。コンパイラーは、これらのファイルを自動的にオープンし、読み取り、レコードを使用可能にします。これらのファイルについて EXCEPTION/ERROR プロシージャーが指定されている場合には、コンパイラーはこれらのプロシージャーに対して必要なリンケージを行います。入力ファイルは、順次、相対、または索引付きファイルでなければなりません。

すべての入力ファイルは、順次または動的アクセス・モードを指定していなければならず、また、データ部の FD 記入項目で記述されていなければなりません。

INPUT PROCEDURE 句

この句では、ソート操作の開始前に入力レコードを選択または変更するためのプロシージャーの名前を指定します。

プロシージャー名-1
入力プロシージャー内の最初の (また唯一の) セクションまたは段落を指定します。

プロシージャー名-2
入力プロシージャー内の最後のセクションまたは段落を指定します。

入力プロシージャーは、ファイル名-1 で参照されるファイルに対して RELEASE ステートメントによって一度に 1 つずつ使用可能にされるレコードを選択、変更、またはコピーするのに必要な任意のプロシージャーから構成できます。この中には、入力プロシージャー内の CALL、EXIT、GO TO、および PERFORM ステートメントによる制御の転送の結果として実行されるすべてのステートメントが含まれます。また、入力プロシージャー内のステートメントの実行の結果として実行される宣言プロシージャー内のすべてのステートメントも含まれます。入力プロシージャー内では、MERGE、RETURN、または SORT ステートメントの実行を起こしてはなりません。

入力プロシージャーを指定すると、SORT ステートメントによってファイル名-1 が順序付けられる前に、制御がこの入力プロシージャーに渡されます。コンパイラーは、この入力プロシージャー内の最後のステートメントの終わりに、戻りメカニズムを挿入します。また、入力プロシージャー内の最後のステートメントに制御が渡るとファイル名-1 にリリースされたレコードがソートされます。

GIVING 句

ファイル名-3, ...
出力ファイル。

GIVING 句を指定した場合には、ファイル名-1 の中のソート済みの全レコードは自動的に出力ファイル (ファイル名-3、...) に転送されます。 SORT ステートメントの実行時には、このファイルがオープンされていてはなりません。

出力ファイルに可変長レコードが含まれている場合には、ファイル名-1 内のレコードのサイズは、出力ファイルについて記述されている最小レコードよりも小さくてはならず、最大レコードよりも大きくてはなりません。出力ファイルに固定長レコードが含まれている場合には、ファイル名-1 内のレコードのサイズは、出力ファイルについて記述されている最大レコードよりも大きくてはなりません。

ファイル名-3 によって参照されるファイルのおのおのに対して、SORT ステートメントの実行時に次のような処置が取られます。

- ファイルの処理が開始されます。この開始処理は、OUTPUT 句を指定した OPEN ステートメントが実行されたかのように行われます。
- ソート済みの論理レコードが戻され、ファイルに書き込まれます。おのおののレコードは、オプショナルの句を指定しない WRITE ステートメントが実行されたかのように書き込まれます。レコードは、ファイルの以前の内容 (もしあれば) を上書きします。

IBM Extension

ファイル名-3 が論理データベース・ファイルである場合には、レコードはファイルの終わりに追加されます。

End of IBM Extension

ファイル名-3 によって参照されるファイルが INDEXED (索引付き) ファイルである場合には、このファイルに関連するキー・データ名が、SORT ステートメント中に ASCENDING KEY 句をもっていなけ

ればなりません。これと同じデータ名は、それのレコード内で、ファイルの基本レコード・キーに関連するデータ項目と同じ文字位置を占めていなければなりません。

相対ファイルの場合には、戻される最初のレコードについての相対キー・データ項目に値 '1' が入り、戻される 2 番目のレコードについての相対キー・データ項目に値 '2' が入り、... というようになります。SORT ステートメントの実行後には、相対キー・データ項目の内容は、ファイルに戻された最後のレコードを示します。

- ファイルの処理が終了します。この終了処理は、オプショナルの句を指定しない CLOSE ステートメントが実行されたかのように行われます。

注: 索引付きファイルへの書き込み中に重複キーが見つかると、SORT ステートメントの実行が終了し、すべての GIVING ファイル内のソート済みデータは未完成のままとなります。

これらの暗黙の機能は、関連する USE AFTER EXCEPTION/ERROR プロシージャーが実行されるように実行されます。ただし、そのような USE プロシージャーの実行により、ファイル名-3 によって参照されているファイルを操作するか、あるいはファイル名-3 に関連付けられているレコード域にアクセスするステートメントの実行が生じてはなりません。

このファイルの外部定義境界を超えてデータを書き込もうとする最初の試みが行われると、このファイルについて指定されている USE AFTER STANDARD EXCEPTION/ERROR プロシージャーが実行されます。この USE プロシージャーから制御が返された場合、あるいはこのような USE プロシージャーが指定されていない場合は、このファイルの処理が終了します。

すべての出力ファイルは、順次または動的アクセス・モードを指定していなければならず、また、データ部の FD 記入項目で記述されていなければなりません。

出力ファイルは、索引付き、相対、または順次ファイルでなければなりません。

また、出力ファイルは、キー順アクセス・パスなしで作成されたものでなければなりません。出力ファイルがそのようなパスをもっている場合、SORT ステートメントでは、データ記述仕様 (DDS) で指定された照合順序を指定変更することができません。

OUTPUT PROCEDURE 句

この句は、ソート操作からの出力レコードを選択または変更するためのプロシージャーの名前を指定します。

プロシージャー名-3
出力プロシージャー内の最初の (または唯一の) セクションまたは段落を指定します。

プロシージャー名-4
出力プロシージャー内の最後のセクションまたは段落を識別します。

出力プロシージャーは、ファイル名-1 からのソートされた順序で、RETURN ステートメントによって一度に 1 つずつ使用可能にされるレコードを選択、変更、またはコピーするのに必要な任意のプロシージャーから構成できます。出力プロシージャーの範囲には、出力プロシージャー内の CALL、EXIT、GO TO、および PERFORM ステートメントによる制御権の移動の結果として実行されるすべてのステートメントが含まれます。また、出力プロシージャー内のステートメントの実行の結果として実行される宣言プロシージャー内のすべてのステートメントも含まれます。出力プロシージャーの範囲には、MERGE、RELEASE、または SORT ステートメントを含めることはできません。

出力プロシージャーが指定されると、ファイル名-1 が SORT ステートメントによって順序付けられた後で、制御がこの出力プロシージャーに渡されます。コンパイラーは、この出力プロシージャー内の最後のス

テートメントの後に、戻りメカニズムを挿入します。制御がこのステートメントに渡されると、戻りメカニズムがソート操作を終了させ、SORT ステートメントの後の次の実行可能ステートメントに制御を渡します。出力プロシージャーに入る前に、ソート・プロシージャーは、ソートされた順序で次のレコードを選択できるポイントに到達します (要求された場合)。次のレコードを要求するのは、出力プロシージャー内の RETURN ステートメントです。

注: INPUT PROCEDURE 句および OUTPUT PROCEDURE 句は、基本 PERFORM ステートメントのものと類似しています。例えば、OUTPUT PROCEDURE 句でプロシージャーを指定すると、そのプロシージャーは、PERFORM ステートメントで指定された場合と同じように、ソート操作時に実行されます。

PERFORM ステートメントの場合と同様に、プロシージャーの実行は最後のステートメントが実行された後で終了されます。入力または出力プロシージャーの最後のステートメントは、EXIT ステートメントにすることができます (7-116 ページの『EXIT ステートメント』を参照してください)。

IBM Extension

SORT-RETURN 特殊レジスターには、SORT 操作の成功 (または失敗) を示す戻りコードが入ります。詳しくは、7-138 ページの『SORT-RETURN 特殊レジスター』を参照してください。

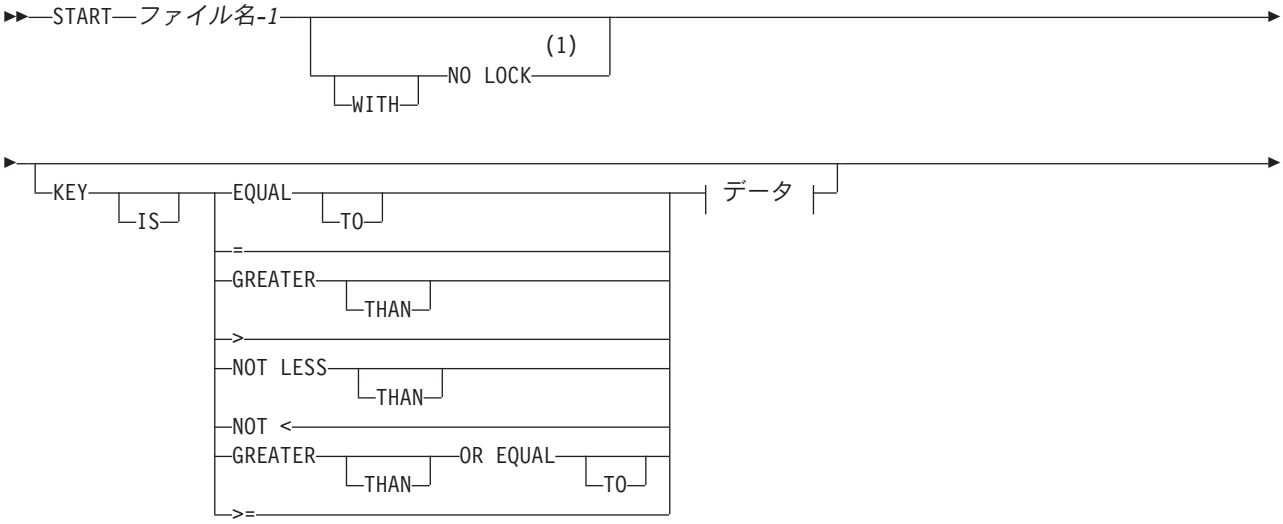
End of IBM Extension

START ステートメント

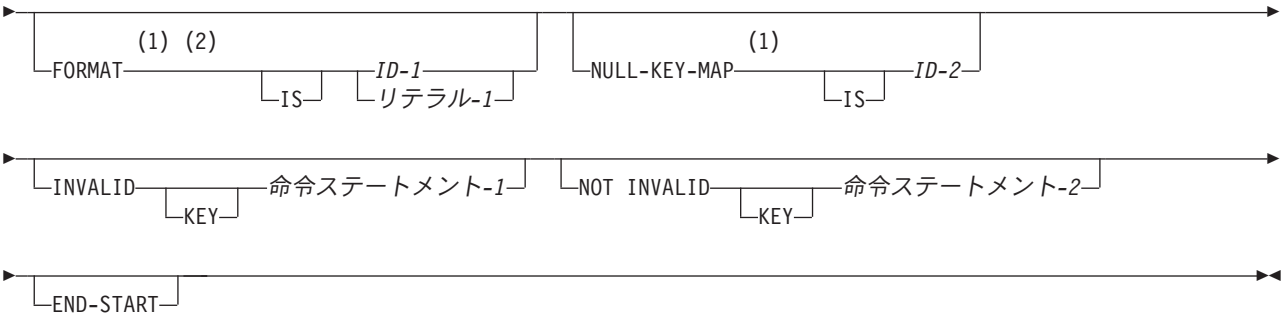
START ステートメントは、後続の順次レコード検索のために索引付きファイルまたは相対ファイル内の位置設定を行う手段となります。この位置設定は、ファイル内のレコードのキー値を、START ステートメントの実行に先立ってファイルのレコード域の RECORD KEY 部分 (索引付きファイルの場合)、あるいは RELATIVE KEY データ項目 (相対ファイルの場合) に入れられた値と比較することによって行われます。

注: START ステートメントの実行時には、関連する索引付きファイルまたは相対ファイルは、INPUT モードまたは I-O モードでオープンされていなければなりません。

START ステートメント - 形式



START ステートメント



データ:

注:

1 IBM 拡張

2 DATABASE 装置上の索引付きファイルにだけ適用されます。

NO LOCK 句

IBM Extension

NO LOCK 句は、I-O (更新) モードでオープンするファイルに対するレコード・ロックが START 操作によって取得できないようにします。また、NO LOCK 句を持つ START ステートメントは、キー値の比較を満たしたレコードが他のジョブによりロックされていても成功します。さらに、この句を持つ START ステートメントは、前の START 操作によってロックされたレコードを解放します。

I-O モードでオープンされていないファイルにこの句を使用すると、エラー・メッセージが出されます。

ファイル・ロックおよびレコード・ロックについては、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

End of IBM Extension

ファイル名-1
順次または動的アクセスを持つファイルでなければなりません。ファイル名-1 は、データ部の FD 記入項目で定義されていなければならず、また、ソート・ファイルの名前であってはなりません。

KEY 句

KEY 句が指定されている場合には、ファイル位置標識は、比較を満たすキー・フィールドを持つ論理レコードに設定されます。

KEY 句が指定されていない場合には、KEY IS EQUAL (TO 基本レコード・キー) が暗黙指定されます。

データ名-1
修飾または参照変更することはできますが、添え字付けすることはできません。

IBM Extension

データ名-1 は、内部浮動小数点データ項目または外部浮動小数点データ項目、DBCS データ項目、または日時データ項目のどれでもかまいません。

End of IBM Extension

IBM Extension

複数のデータ名を指定できます。最初のデータ名に続くすべてのデータ名は、構文検査されるだけです。

End of IBM Extension

START ステートメントが実行される際に、キー・データ名内の現行値とファイルの索引内の対応するキー・フィールドが比較されます。

FILE-CONTROL 記入項目に FILE STATUS 文節が指定されている場合には、START ステートメントの実行時に関連する状況キーが更新されます。 (7-37 ページの『状況キー』を参照。)

FORMAT 句

IBM Extension

FORMAT 句で指定する値には、この入出力操作で使用するレコード形式の名前が入ります。システムはこれを使用して、どのレコード形式に対して操作を行うかを指定または選択します。

ID-1 を指定する場合には、10 文字以下の英数字データ項目にしなければなりません。

リテラル-1 を指定する場合は、10 文字以下の大文字の文字ストリングにしなければなりません。

全桁がブランクの値は、FORMAT 句が指定されなかった場合と同じ扱いになります。その値がファイルに対して有効でない場合は、9K の FILE STATUS が戻され、(そのファイルに適用可能であれば) USE プロシージャーが呼び出されます。

この句を指定する場合、ファイル位置標識は、指定されたレコード形式をもち、比較を満たす最初のレコードに設定されます。この句を省略した場合には、現行レコード・ポインターは、レコード形式を問わず、比較を満たす最初のレコードに設定されます。

FORMAT 句と EXTERNALLY-DESCRIBED-KEY および KEY IS 句の相互作用については 7-224 ページの表 7-15を参照してください。

End of IBM Extension

NULL-KEY-MAP IS 句

IBM Extension

ID-2 に 指定された値に従って、START 操作のためのレコードのヌル・キー・マップを指定します。ID-2 はブール項目または英数字項目でなければなりません。

ID-2 は、添え字を付けたり、参照変更できます。

ファイルに代替キーがある場合、ID-2 は、参照の現行キーのヌル・キー・マップに関連しています。

この句は、ALWNULL 属性をもち、ASSIGN 文節で DATABASE が指定された装置タイプのファイルにだけ指定できます。キー・フィールドの 1 つがヌル可能で、NULL-KEY-MAP 句が使用されない場合は、すべてのブール・ゼロをもったヌル・キー・マップが使用されます。

NULL-KEY-MAP IS 句の例: この例では、以下の値はファイルの中のキーを表しており、これにはそれぞれが 2 バイトでできた 3 個のフィールドが入っています。キーは、ファイル・セクションで以下のコードによって定義されます。

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT FILE-1 ASSIGN to DATABASE-FILE1-ALWNULL
    ACCESS is DYNAMIC RECORD KEY IS FULL-PRODUCT-CODE IN FILE-1
    ORGANIZATION IS INDEXED.
FD FILE-1.
01 FULL-PRODUCT-CODE.
   05 TYPE-CODE      PIC X(2).
   05 COLOR-CODE     PIC X(2).
   05 LOCATION-CODE  PIC X(2).
WORKING-STORAGE SECTION.
01 FILE1-N.
   05 FULL-PRODUCT-CODE-NKM.
      06 FILLER             PIC X VALUE ZERO.
      06 COLOR-CODE-NF      PIC 1 VALUE B"0".
      06 LOCATION-CODE-NF   PIC 1 VALUE B"0".
```

フィールド 2 および 3 はヌル可能フィールドであり、'-' はヌルを示し、xx は任意の値を示します。以下は、ファイルの中のレコードを表しています。

```
NN----
NN--xx
NNxx--
```

以下の START ステートメントを考えてみます。

```
  START FILE-1
      NULL-KEY-MAP IS FULL-PRODUCT-CODE-NKM
      INVALID KEY DISPLAY "No data in system for product code " TYPE-CODE
      GO TO ERROR-ROUTINE
      END-START.
```

START ステートメントのヌル・キー・マップの値が 010 の場合は、ポインターはキー NN--xx をもったレコードを指すように設定されます。 START ステートメントのヌル・キー・マップの値が 011 の場合は、ポインターはキー NN---- をもったレコードを指すように設定されます。

ヌル可能フィールドの使用について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

End of IBM Extension

INVALID KEY 句

ファイル内のレコードによって比較が満たされない場合には、無効キー条件が発生します。ファイル位置標識の値は未定義となり、INVALID KEY 命令ステートメント (指定されている場合) が実行されます。
(7-38 ページの『INVALID KEY 条件』を参照。)

このファイルについて EXCEPTION/ERROR プロシージャーが明示的または暗黙に指定されていない場合は、INVALID KEY 句を指定しなければなりません。

NOT INVALID KEY 句

NOT INVALID KEY 句を指定した START ステートメントが正常に完了すると、この句に関連付けられている命令ステートメントに制御が渡されます。

END-START 句

この明示範囲終了符号は、START ステートメントの範囲を区切る働きをします。 END-START 句を使用することによって、START 条件ステートメントを別の条件ステートメントにネストすることができます。また、END-START 句は、START 命令ステートメントで使用することもできます。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

索引付きファイル

KEY 句が指定されている場合には、比較に使用されるキー・データ項目はデータ名です。

KEY 句が指定されていない場合、ファイル位置標識は、RECORD KEY データ項目に入っている値と等しいキーを持つレコードに設定されます。

START ステートメントが正常に実行された場合には、データ名-1 の関連付けられている RECORD KEY または ALTERNATE RECORD KEY が、後続の READ ステートメントの参照キーになります。

KEY 句が指定されている場合、比較に使用される検索引数はデータ名-1 であり、次のいずれかにすることができます。

- 基本 RECORD KEY 自体
- 任意の ALTERNATE RECORD KEY
- ファイルのレコード記述内の英数字データ項目で、左端の文字位置がレコード域内のキー・フィールドの左端の文字位置に対応しているもの。このデータ項目の長さは、そのファイルのレコード・キーの長さより小さいかあるいは等しくなければなりません。

  このデータ項目は、修飾または参照変更できます。キーそのものを使用しない場合は、左端の文字位置に参照変更開始位置を加算した位置が、キー・フィールドの左端の文字位置に対応していなければなりません。

注: RECORD KEY が COMP、COMP-3、COMP-4、または COMP-5 として定義されている場合、キー・
データ項目は RECORD KEY そのものでなければなりません。レコード域の部分キー・フィールドを使用
することはできません。

ファイル位置標識は、ファイル内で比較を満たすレコード・キーを持つ、ある形式の最初のレコードに設定されます。比較するオペランドの長さが同じでない場合には、長い方のフィールドの右側が短いフィールドの長さに合わせて切り捨てられたかのように比較が実行されます。 PROGRAM COLLATING SEQUENCE 文節が指定されても無効になる点を除き、他の数字比較および非数字比較のすべての規則が適用されます。

IBM Extension

RECORD KEY IS EXTERNALLY-DESCRIBED-KEY を指定したファイルの場合は、次のような追加の考慮事項が適用されます。

- 予約語 EXTERNALLY-DESCRIBED-KEY を指定できます。これは、レコード域内の完全なキーが比較に使用されなければならないことを示します。
- 一連のデータ名を指定できます。これによって、レコード域中の部分キー・フィールドが使用できます (総称 START)。これらのデータ名は以下の規則に従わなければなりません。

- 指定するデータ名のうち最後のもの以外はすべて、ファイルについてコピーされた単一の形式用のレコード・キーでなければなりません。データ名が入っているレコード形式は、FORMAT 句によって指定できるものである必要はありません。
- これらのデータ名 (キー・フィールド) の順序は、DDS で定義されているキーの順序と一致しなければなりません。つまり、重要度の高いフィールドから順に指定しなければなりません。
- データ名の総数は、そのレコード形式用に定義されたキー・フィールドの数を超えることはできません。
- 一連のデータ名の最後のデータ名がレコード域内のキー・フィールドでない場合、そのデータ名の左バイトは、その相対位置に定義されているキー・フィールドと同じスペースを占有しなければなりません。レコード域内でこの位置にあるキー・フィールドが COMP、COMP-3、COMP-4、または COMP-5 フィールドである場合には、キー・フィールドそのものだけがデータ名として使用可能です。
- 最後のキーだけが参照変更することができ、参照変更開始位置は 1 と等しくなければなりません。

• 表 7-15 は、KEY IS と FORMAT 句の間の処理を示しています。

表 7-15. KEY IS と FORMAT 句の関係

FORMAT 句の指定	KEY 句		
	一連のデータ名	省略	EXTERNALLY-DESCRIBED-KEY
Yes	A、B	C、D	C、B
No	A、E	F、G	F、E

検索引数は、指定されたデータ項目を使用して作成されます。

B ファイル位置標識は、指定された形式をもち、KEY 句で指定された比較を満たすレコード・キーを持つ、ファイル内の最初のレコードに設定されます。

C 検索引数は、FORMAT 句で指定された形式用のレコード域内のキー・フィールドを使用して作成されます。

D ファイル位置標識は、指定された形式をもち、検索引数と等しいレコード・キーを持つ、ファイル内の最初のレコードに設定されます。

E ファイル位置標識は、ファイルの共通キーをもち、KEY 句で指定された比較を満たす、ファイル内の最初のレコードに設定されます。共通キーがない場合、ファイル位置標識はファイル内の先頭レコードに設定されます。

F 検索引数は、ファイルについての最初のレコード形式 (プログラムで定義された) 用のレコード域のキー・フィールドを使用して作成されます。

G ファイル位置標識は、ファイルの共通キーをもち、検索引数と等しい、ファイル内の最初のレコードに設定されます。共通キーがない場合、ファイル位置標識はファイル内の先頭レコードに設定されます。

End of IBM Extension

KEY 句が指定されていない場合には、EQUAL TO 比較に使用されるキー・データ項目は基本 RECORD KEY です。

データ名-1
次のいずれかにすることができます。

• 基本 RECORD KEY

- ファイルのレコード記述内の英数字データ項目で、左端の文字位置がそのレコード・キーの左端の文字位置と対応しているもの。これは修飾できます。このデータ項目の長さは、このファイルのレコード・キーの長さと等しいか、あるいはそれより小さくなければなりません。

ファイル位置標識は、比較を満たすキー・フィールドを持つ、ファイル内の最初のレコードに設定されます。比較するオペランドの長さが同じでない場合には、長い方のフィールドの右側が短いフィールドの長さに合わせて切り捨てられたかのように比較が実行されます。 PROGRAM COLLATING SEQUENCE 文節が指定されても無効になる点を除き、他の数字比較および非数字比較のすべての規則が適用されます。

START ステートメントが正常に実行された場合には、データ名-1 の関連付けられている RECORD KEY が、後続の READ ステートメントの参照キーになります。

START ステートメントの実行が失敗すると、参照キーが未定義になります。

IBM Extension

装置タイプ DATABASE の索引付きファイルの場合、比較の意味は、ファイル用に定義されたレコード域内のキー・フィールドのタイプの影響を受けることがあります。このシステムでのキー・フィールドは、複数フィールドとして定義することができ、それぞれのフィールドは昇順または降順にすることができます。システムは、レコード・キーに入っている値と DDS 内で指定された形式と順序に基づいて、レコードについての順序 (キー順アクセス・パス) を確立します。 START ステートメントが処理される際には、要求は次のように解釈されます。

COBOL 比較	**システムの結果**
GREATER THAN	AFTER
NOT LESS THAN	EQUAL TO または AFTER

例えば、GREATER THAN という比較を用いてステートメントが処理される場合は、START ステートメントによって指定された検索引数の後の最初のレコードを見つけるために、順序付けられたレコードの検索が行われます。ファイルが降順キーで順序付けられた場合、ファイル位置標識は、指定されたキーより小さく、かつ、START ステートメントに指定されたキーよりも大きくないキーを持つレコードを指します。

End of IBM Extension

相対ファイル

KEY 句が指定されていない場合、ファイル位置標識は、RELATIVE KEY データ項目と等しいキー (相対レコード番号) を持つ、ファイル内のレコードに設定されます。

KEY 句を指定する場合、データ名-1 には RELATIVE KEY を指定しなければなりません。ファイル位置標識は、RELATIVE KEY データ項目との比較を満たすキー (相対レコード番号) を持つ、ファイル内の最初の論理レコードに設定されます。

KEY 句が指定されていない場合には、KEY IS EQUAL (TO 基本レコード・キー) が暗黙指定されます。

データ名-1 は修飾できますが、添え字付けすることはできません。

START ステートメントの実行時には、相対キーの現行値とファイル内の既存のレコードの相対レコード番号が比較されます。

FILE-CONTROL 記入項目に FILE STATUS 文節が指定されている場合には、START ステートメントの実行時に関連する状況キーが更新されます。 (7-37 ページの『共通の処理機能』の『状況キー』を参照してください。)

KEY 句が指定されているかどうかにかかわらず、比較に使用されるキー・データ項目は RELATIVE KEY データ項目になります。 START ステートメントが正常に実行された場合、ファイル位置標識は、ファイル内で比較を満たすキーを持つ論理レコードを指します。このキーは、後続の READ ステートメントの参照キーとなります。

START ステートメントが正常に実行されなかった場合には、参照キーおよびファイル位置標識が未定義となります。

STOP ステートメント

STOP ステートメントは、オブジェクト・プログラムの実行を一時的にまたは永続的に停止させます。

STOP ステートメント - 形式

リテラル

数字、非数字、またはブールにすることができ、また、ALL リテラル以外の表意定数にすることもできます。リテラルが数字である場合は、符号なしの整数でなければなりません。

IBM Extension

浮動小数点リテラルにすることはできません。

End of IBM Extension

STOP リテラルを指定すると、そのリテラルが、バッチ・ジョブの場合にはシステム・オペレーターに、対話式ジョブの場合にはワークステーションに知らされます。プログラムの実行は中断されます。プログラムの実行は、オペレーターが介入しなければ再開されません。

オペレーターの応答によって、取られる処置が決定されます。

オペレーター
応答処置

G (デフォルト)
次の命令から続行します。

C 最も近い制御境界にあるプログラムまでのすべてのプログラムの実行を終了します。最も近い制御境界がハード制御境界である場合には、COBOL 実行単位の呼び出し側にエスケープ・メッセージ CEE9901 が出されます。バッチ・ジョブの場合、ジョブについての ENDSEV パラメーター (CRTJOBD CL コマンドを参照してください) にメッセージの重大度より小さい値が入っていると、ジョブが取り消されます。

D COBOL ID をダンプした後、C と同じ処置を実行します。

F COBOL ID およびファイル情報をダンプした後、C と同じ処置を実行します。

STOP リテラルの出力には、プログラム名が入ります。リテラルは、2 次レベル・テキストに入れられ、ヘルプ・キーが使用された場合に表示されます。

STOP リテラル・ステートメントは、プログラムの実行中にオペレーターの介入が必要とされる場合に、特別な状況 (特別なテープまたはディスクを取り付けなければならない場合や、特定の日付コードを入力しなければならない場合など) で役立ちます。ただし、オペレーターの介入が必要なときには、ACCEPT ステートメントおよび DISPLAY ステートメントを使用することをお勧めします。

STOP RUN を指定すると、 最も近い制御境界にあるプログラム (これも含む) までのすべてのプログラムが終了され、制御が制御境界の前のプログラムに戻されます。最も近い制御境界がハード制御境界である場合には、STOP RUN によって活動化グループ (実行単位) が終了され、次に、活動化グループに範囲が限定されているすべてのファイルがクローズされます。 STOP RUN ステートメントを一連の命令ステートメントで指定する場合には、その中の最後または唯一のステートメントにしなければなりません。

上記のどの場合も、呼び出し側プログラムはシステムである可能性があります。その場合は、実行単位の実行が終了し、オペレーティング・システムに制御が渡されます。

また、COBOL リンケージ規則に従わない言語で書かれたプログラムで主プログラムを呼び出すと、この呼び出し側プログラムに制御が戻ります。

各種条件における STOP RUN ステートメントの動作について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き* 」の『ILE COBOL プログラムからの戻り』を参照してください。

RETURN-CODE 特殊レジスター

IBM Extension

RETURN-CODE 特殊レジスターは、プログラムから呼び出し側 (呼び出し側プログラムまたはシステム) に戻りコード情報 (つまり、数値) を渡すために使用できます。

RETURN-CODE 特殊レジスターは、EXIT PROGRAM、GOBACK、または STOP RUN ステートメントを実行する前に設定できます。

RETURN-CODE には、次の暗黙定義があります。

```
01 RETURN-CODE GLOBAL PICTURE S9999 USAGE BINARY VALUE 0
```

この特殊レジスターは、PICTURE S9999 USAGE BINARY のデータ定義を持つデータ項目が許容される場所であれば、プログラム内のどこででも使用できます。ネストされたプログラムの中で使用される際には、RETURN-CODE 特殊レジスターは、最外部のプログラム内で GLOBAL として暗黙に定義されます。COBOL サブプログラムが終了すると、そのサブプログラムの 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。主 COBOL プログラムが終了し、制御権がオペレーティング・システムに戻される際には、特殊レジスターの内容がユーザー戻りコードとしてオペレーティング・システムに戻されます。

主 COBOL プログラムが活動化グループの最初のプログラムでなければならないことに注意してください。そのために RETURN-CODE 特殊レジスターの内容をジョブのユーザー戻りコードとして戻す場合は、通常オプション ACTGRP(*CALLER) を使ってこの COBOL プログラムをコンパイルしてはいけません。呼び出し側プログラムにより、形式 JOBI0600 を持つ API QUSRJOBI を呼び出すことによって、ユーザー戻りコードを取り出すことができます。

プログラムの最初の呼び出しの場合、RETURN-CODE 特殊レジスターはゼロに初期設定されます。これは、正常終了を表す正常な戻りコードです。フィールドは、取り消されているかまたは INITIAL 属性を持つプログラムの後続の呼び出しの際にゼロにリセットされます。それ以外の場合には、RETURN-CODE 特殊レジスターはリセットされず、前の呼び出しの後に入った値のまま変更されません。

整数の引数が認められている関数内ではどこでも、RETURN-CODE 特殊レジスターを指定できます。

戻りコード情報の引き渡しについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

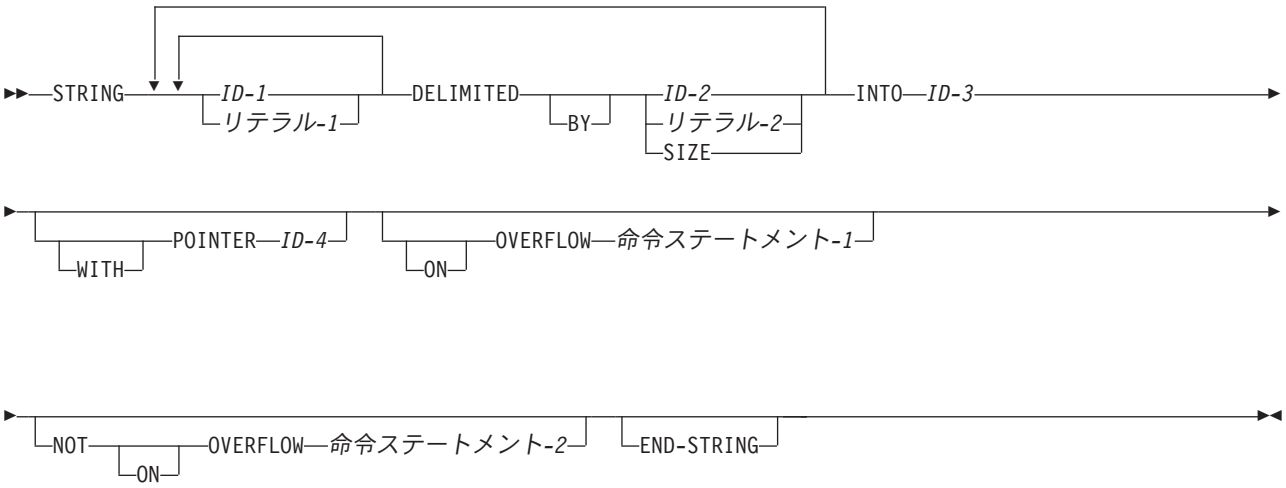
End of IBM Extension

STRING ステートメント

STRING ステートメント・ストリングは、複数のデータ項目またはリテラルの一部または全部の内容を連結して、単一のデータ項目にします。

一連の MOVE ステートメントの代わりに、1 つの STRING ステートメントを書くことができます。

STRING ステートメント - 形式



注: ID-4 (POINTER 項目) 以外のすべての ID は、明示的または暗黙に USAGE DISPLAY を持っていなければなりません。

ID-1

1 つまたは複数の送り出しフィールドを表します。送り出しフィールドまたは分離文字が基本数字項目である場合には、それは整数として記述されていなければなりません。さらに、その PICTURE 文字ストリングに記号 P が含まれていてはなりません。

リテラル-1

1 つまたは複数の送り出しフィールドを表します。すべてのリテラルは非数字リテラルでなければなりません。それぞれは、ALL リテラルをもたない表意定数にすることができます。表意定数を指定すると、それは 1 文字の非数字リテラルと見なされます。

IBM Extension

ID-1 から ID-3 まで
外部浮動小数点項目にすることはできません。

End of IBM Extension

IBM Extension

ID-1、**ID-2**、または **ID-3** のいずれかが DBCS データ項目である場合は、それらの ID はすべて DBCS データ項目なければならず、リテラルはすべて DBCS リテラルでなければなりません。

ID-1、**ID-2**、または **ID-3** のいずれかが国別データ項目である場合は、それらのすべてが国別データ項目でなければなりません。

リテラル-1 または **リテラル-2** のいずれかが DBCS リテラルである場合は、両方とも DBCS リテラルでなければならず、リテラル-1 からリテラル-3 は DBCS データ項目でなければなりません。

SPACE は、DBCS 項目に使用できる唯一の表意定数です。

End of IBM Extension

DELIMITED BY 句

DELIMITED BY 句は、ストリングの区切りを設定します。

ID-2、リテラル-2
分離文字、すなわち転送されるデータを区切る文字です。

ID-1 または ID-2 が ID-3 または ID-4 と同じ記憶域を占有している場合には、これらの ID が同じデータ記述記入項目によって定義されている場合でも、結果は未定義になります。

表意定数を指定すると、それは 1 文字の非数字リテラルと見なされます。

SIZE
送り出し区域全体を転送します。

INTO 句

ID-3
受け入れフィールドを表します。

これは、編集データ項目を表してはならず、JUSTIFIED 文節で記述されていてはなりません。また、参照変更することはできません。

ID-3 と ID-4 が同じ記憶域を占有している場合には、これらの ID が同じデータ記述記入項目によって定義されている場合でも、結果は未定義になります。

IBM Extension

外部浮動小数点データ項目を表していてはなりません。

End of IBM Extension

POINTER 句

ID-4

受け入れフィールド内の文字位置を指す**ポインター・フィールド**を表します。

この ID は、受け入れ区域の長さに 1 を加えた値を含むために十分な大きさを持つ基本整数データ項目でなければなりません。ポインター・フィールドの PICTURE 文字ストリング内に記号 P を含むことはできません。

IBM Extension

ID-3 が DBCS データ項目の場合、ID-4 は受け入れフィールド内の相対 DBCS 文字位置を示します。

End of IBM Extension

ON OVERFLOW 句

ポインター値 (明示または暗黙) が次のものになると、制御が命令ステートメント-1 に移されます。

- ゼロまたは 1 よりも小さい値
- 受け入れフィールドの長さを超える値

上記のいずれかの状態が起こると、オーバーフロー条件が発生し、それ以上データが転送されません。STRING 操作が終了し、制御権が命令ステートメント-1 に移されます (ON OVERFLOW 句が指定されている場合)。それ以外の場合には、制御権が STRING ステートメントの終わりに移されます。 NOT ON OVERFLOW ステートメントは指定されていても無視されます。

制御権が命令ステートメント-1 に渡されると、命令ステートメント-1 に指定されている各ステートメントについての規則に従って実行が継続されます。制御権の明示的な移動をもたらすプロシージャー分岐ステートメントまたは条件ステートメントが実行される場合には、制御権はそのステートメントの規則に従って移されます。

STRING ステートメントの実行中にオーバーフロー条件が発生しないと、制御は STRING ステートメントの終わりに移されます。オーバーフロー条件が発生せず、NOT ON OVERFLOW 句が指定されている場合には、制御は命令ステートメント-2 に移されます。ON OVERFLOW 句は指定されていても無視されます。

制御権が命令ステートメント-2 に渡された場合は、命令ステートメント-2 に指定されている各ステートメントの規則に従って実行が継続されます。制御権の明示的な移動をもたらすプロシージャー分岐ステートメントまたは条件ステートメントが実行される場合には、制御権はそのステートメントについての規則に従って移動されます。それ以外の場合には、制御権は命令ステートメント-2 の実行完了時に STRING ステートメントの終わりに移されます。

ON OVERFLOW ステートメントは、ID-3 の終わりを超えて 1 つまたは複数の文字を入れようとしているか、または POINTER の初期値が 1 よりも小さい場合を除き、実行されません。

END-STRING 句

この明示範囲終了符号は、STRING ステートメントの範囲を区切る働きをします。 END-STRING 句を使用することによって、STRING 条件ステートメントを別の条件ステートメントにネストすることができます。また、END-STRING 句は、STRING 命令ステートメントで使用することもできます。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

データの流れ

STRING ステートメントが実行されると、データが送り出しフィールドから受け入れフィールドに移動されます。送り出しフィールドが処理される順序は、それらが指定されている順序です。次の規則が適用されます。

- 送り出しフィールドの文字は、英数字基本項目間の移動に関する規則に従って受け入れフィールドに転送されます。ただし、このときにスペースの埋め込みは行われません (7-139 ページの『MOVE ステートメント』を参照してください)。
- DELIMITED BY ID / リテラルを指定した場合には、各送り出し項目の内容は、左端の文字から、次のいずれかになるまで 1 文字ずつ転送されます。
  - この送り出しフィールドの分離文字に到達する (分離文字そのものは転送されません)。
  - この送り出しフィールドの右端の文字が転送される。
- DELIMITED BY SIZE ID を指定した場合には、それぞれの送り出しフィールド全体が受け入れフィールドへ転送されます。
- 受け入れフィールドが満杯になるかまたはすべての送り出しフィールドの処理が完了すると、操作が終了します。
- POINTER 句を指定した場合には、COBOL ユーザーは受け入れフィールド内のデータの配置を制御するための明示ポインター・フィールドを使用できます。ユーザーは明示ポインターの初期値を設定しなければなりません。初期値は、1 より小さいかまたは受け入れフィールドの文字カウントより大きい値であってはなりません。 (ポインター・フィールドは、受け入れフィールドの長さに 1 を加えた値を含むために十分な大きさのフィールドとして定義されていなければなりません。これは、転送の終了時にシステムがポインターを更新する際に算術オーバーフローが起こらないようにするためです。)
- POINTER 句を指定しない場合には、ユーザーはポインターを使用することができません。ただし、初期値 1 を持つ概念上の暗黙ポインターがシステムによって使用されます。
- 概念上、STRING ステートメント実行時の (明示または暗黙) ポインターの初期値は、データが転送される受け入れフィールドの先頭の文字位置です。その位置から始めて、データは左から右へ 1 文字ずつ配置されます。その位置から始めて、データは左から右へ 1 文字ずつ配置されます。各文字が配置されるたびに、明示または暗黙ポインターが 1 ずつ増やされます。ポインター・フィールド中の値はこの方法によってだけ変更されます。処理の終了時には、ポインター値は必ず、受け入れフィールドに転送された最後の文字の次の文字と等しい値を指します。

添え字付け、参照変更、可変長の計算、または関数の評価は、STRING ステートメントの処理の開始時に 1 回だけ実行されます。このため、ID-3 または ID-4 が、STRING ステートメント内で添え字、参照修飾子、または関数引数として使用されるか、または STRING ステートメントの任意の ID の長さまたは位置に影響する場合、これらの値は STRING ステートメントの始まりにおいて判別され、STRING ステートメントの結果には影響されません。

ID-1 または ID-2 が ID-3 または ID-4 と同じ記憶域を占有している場合、または ID-3 と ID-4 が同じ記憶域を占有している場合には、STRING ステートメントの実行の結果は未定義です。

STRING ステートメントの実行が完了すると、受け入れフィールドのうち、データが転送された部分だけが変更されます。受け入れフィールドの残りの部分には、この STRING ステートメントの実行前に入っていたデータがそのまま残っています。

次の STRING ステートメントが実行された場合、得られる結果は 7-232 ページの図 7-11 に示されているようになります。

STRING ステートメント

```
STRING ID-1 ID-2 DELIMITED BY ID-3
       ID-4 ID-5 DELIMITED BY SIZE
  INTO ID-7 WITH POINTER ID-8
END-STRING
```

FD 記入項目:

```
FD  INPUT-FILE LABEL RECORDS OMITTED.

01  RECORD-1  PICTURE X(30).

01  RECORD-2  PICTURE X(20).
```

READ ステートメント実行時の入力域の内容:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ1234
```

(RECORD-2) で読み取られているレコードの内容:

```
01234567890123456789
```

READ 実行後の入力域の内容:

```
01234567890123456789??????????
                     └────┘
          (入力域のこれらの文字は未定義)
```

図 7-11. STRING ステートメントの実行結果

STRING ステートメントの例

次の例では、STRING ステートメントに適用されるいくつかの考慮事項を示します。

データ部で、プログラマーは次のフィールドを定義したものとします。

```
   01 RPT-LINE     PICTURE X(120).
   01 LINE-POS     PICTURE 99.
   01 LINE-NO      PICTURE 9(5) VALUE 1.
   01 DEC-POINT    PICTURE X VALUE ".".
```

また、FILE セクションで、次の入力レコードを定義したものとします。

```
01 RCD-01.
   05 CUST-INFO.
      10 CUST-NAME  PICTURE X(15).
      10 CUST-ADDR  PICTURE X(34).
   05 BILL-INFO.
      10 INV-NO     PICTURE X(6).
      10 INV-AMT    PICTURE $$,$$$.99.
      10 AMT-PAID   PICTURE $$,$$$.99.
      10 DATE-PAID  PICTURE X(8).
      10 BAL-DUE    PICTURE $$,$$$.99.
      10 DATE-DUE   PICTURE X(8).
```

プログラマーが、RCD-01 の情報の一部から成る出力行を作成したいとします。この行は、行番号、顧客の名前と住所、請求書番号、支払期日、および支払金額 (示された円の桁数に切り捨てられた) で構成されます。

読み取られるレコードには次の情報が入っています。

J.B.␣SMITH␣␣␣␣␣
444␣SPRING␣ST.,␣CHICAGO,␣ILL.␣␣␣␣␣
A14275
\4,736.85
\2,400.00
09/22/76
\2,336.85
09/09/94

手続き部で、プログラマーは RPT-LINE を SPACES に初期設定し、LINE-POS (ポインター・フィールドとして使用される) を 4 に設定します。その後で、次の STRING ステートメントを出します。

```
STRING LINE-NO SPACE
    CUST-INFO SPACE
    INV-NO SPACE
    DATE-DUE SPACE
  DELIMITED BY SIZE,
    BAL-DUE
  DELIMITED BY DEC-POINT
  INTO RPT-LINE
  WITH POINTER LINE-POS.
```

ステートメントの実行時には、次の処理が行われます。

1. フィールド LINE-NO が RPT-LINE の桁 4 ～ 8 に移動されます。
2. スペースが桁 9 に移動されます。
3. グループ項目 CUST-INFO が桁 10 ～ 58 に移動されます。
4. スペースが桁 59 に移動されます。
5. INV-NO が桁 60 ～ 65 に移動されます。
6. スペースが桁 66 に移動されます。
7. DATE-DUE が桁 67 ～ 74 に移動されます。
8. スペースが桁 75 に移動されます。
9. BAL-DUE の小数点の前にある部分が桁 76 ～ 81 に移動されます。

STRING ステートメントの実行後は、次のようになります。

- RPT-LINE は 7-234 ページの図 7-12 で示されているようになります。
- LINE-POS には値 82 が入ります。

注: 一連の MOVE ステートメントの代わりに、1 つの STRING ステートメントを書くことができます。

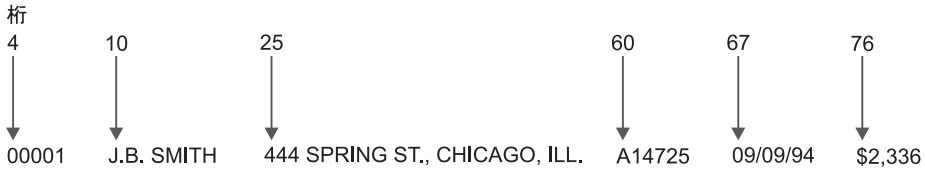
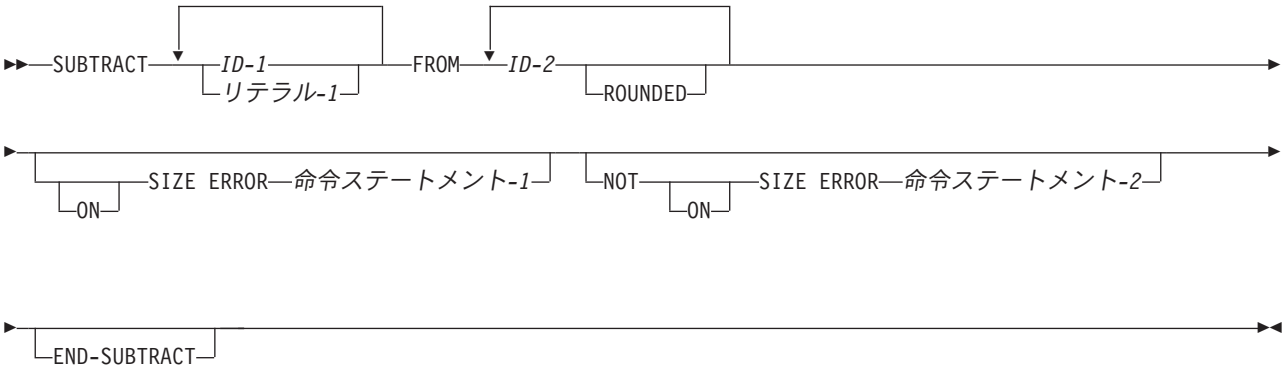


図 7-12. STRING ステートメントの例の出力データ

SUBTRACT ステートメント

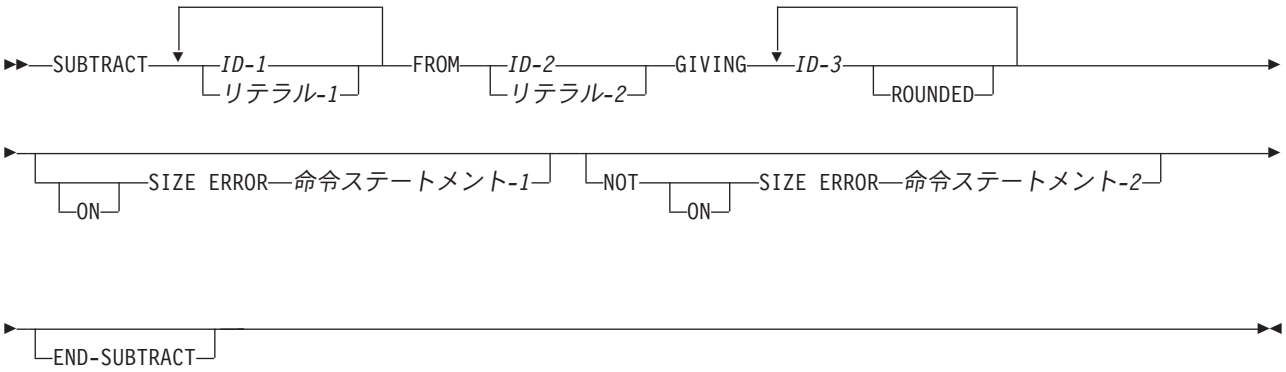
SUBTRACT ステートメントは、1 つまたは複数の数字項目から、1 つの数字項目または複数の数字項目の和を減算し、結果を保管します。

SUBTRACT ステートメント - 形式 1



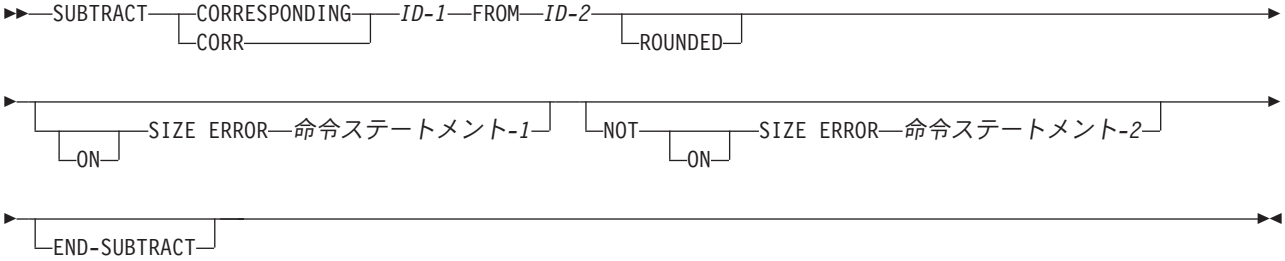
形式 1 では、キーワード FROM の前のすべての ID またはリテラルが加算され、この初期の和が ID-2 から減算され、ID-2 に保管されます。初期の和は、その後、ID-2 が見つかるたびに、ID-2 が指定されている順序で左から右に減算され、保管されます。

SUBTRACT ステートメント - 形式 2 - GIVING



形式 2 では、キーワード FROM の前のすべての ID またはリテラルが加算され、この和が ID-2 またはリテラル-2 から減算されます。減算の結果は、ID-3 によって参照されているデータ項目に保管されます。ID-2 またはリテラル-2 は未変更のままです。

SUBTRACT ステートメント - 形式 3 - CORRESPONDING



形式 3 では、ID-1 内の基本データ項目が、ID-2 内の対応する基本データ項目から減算され、結果が ID-2 に保管されます。

すべての形式において、以下が適用されます。

ID-1、ID-2、ID-3
形式 1 および 2 では、ID-1 および ID-2 は基本数字項目でなければなりません。

形式 2 では、GIVING の後のそれぞれの ID-3 は数字または数字編集基本項目でなければなりません。

形式 3 では、ID-1 はグループ項目でなければなりません。

リテラル-1、リテラル-2
数字リテラルでなければなりません。

語 GIVING に続くデータ項目を除く該当のステートメント内のすべてのオペランドを使用して、オペランドの合成内容が決定されます。オペランドの合成に関する詳細は 7-34 ページの『オペランドのサイズ』を参照してください。

IBM Extension

浮動小数点データ項目および浮動小数点リテラルは、数字データ項目または数字リテラルが指定できるところではどこでも使用できます。

End of IBM Extension

ROUNDED 句

ROUNDED 句に関する情報、およびオペランドに関する考慮事項については、 7-33 ページの『ROUNDED 句』を参照してください。

SIZE ERROR 句

SIZE ERROR 句に関する情報、およびオペランドに関する考慮事項については 7-33 ページの『SIZE ERROR 句』を参照してください。

CORRESPONDING 句 (形式 3)

CORRESPONDING 句 (CORR) を使用すると、同じ名前の基本数字データ項目を、それらが属するグループ項目を指定することによって操作できます。

END-SUBTRACT 句

この明示範囲終了符号は、SUBTRACT ステートメントの範囲を区切ります。 END-SUBTRACT は、SUBTRACT 条件ステートメントを命令ステートメントに交換します。この句を使用することによって、SUBTRACT 条件ステートメントを別の条件ステートメントにネストすることができます。

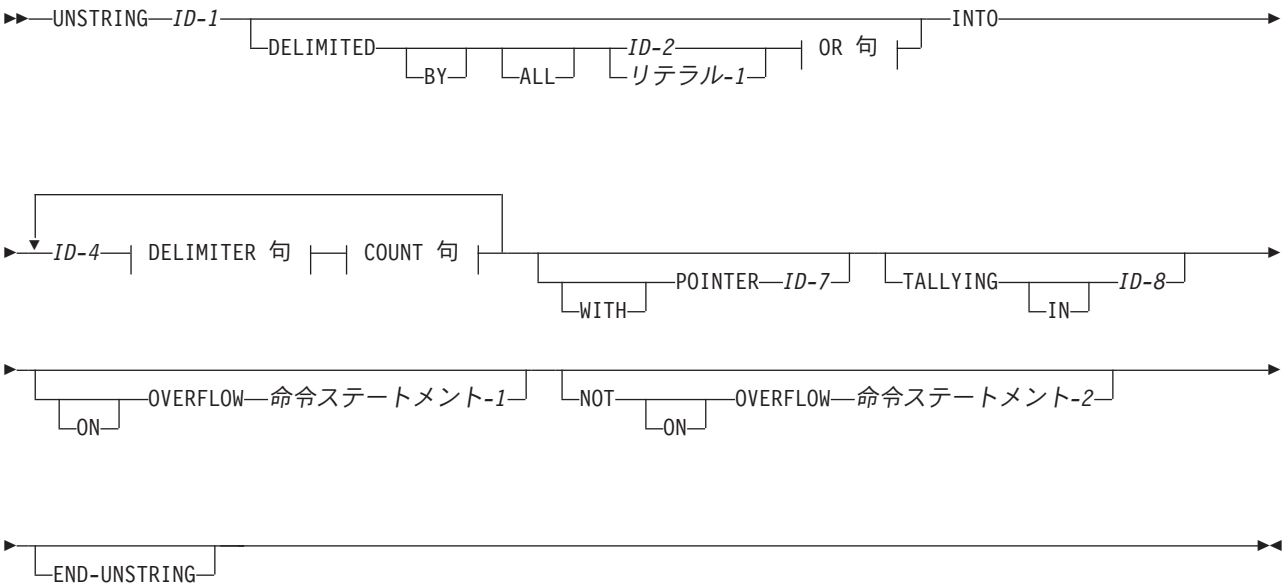
詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

UNSTRING ステートメント

UNSTRING ステートメントは、送り出しフィールド中の連続したデータを分離し、複数の受け入れフィールドへ入れます。

一連の MOVE ステートメントの代わりに、1 つの UNSTRING ステートメントを書くことができます。

UNSTRING ステートメント - 形式



OR 句:

DELIMITER 句:

COUNT 句:

ID-1
送り出しフィールドを表します。

これは、英数字データ項目でなければならず、参照変更することはできません。データはこのフィールドから受け入れフィールドへ転送されます。

IBM Extension

ID-1 は DBCS または国別データ項目にすることができます。

End of IBM Extension

DELIMITED BY 句

この句では、データ転送を制御する、 データ内の分離文字を指定します。

分離文字は、ID-2、ID-3、またはそれと対応するリテラルです。指定される各 ID またはリテラルが 1 つの分離文字を表します。それぞれが英数字データ項目でなければなりません。

DELIMITED BY 句を指定しなければ、DELIMITER IN 句および COUNT IN 句を指定してはなりません。

ID-2、ID-3
各 ID は 1 つの分離文字を表します。それぞれが英数字データ項目でなければなりません。

IBM Extension

いずれか一方が DBCS 項目である場合は、両方を DBCS 項目にしなければなりません。いずれか一方が国別項目である場合は、両方を国別項目にしなければなりません。

End of IBM Extension

リテラル-1、リテラル-2
各リテラルは非数字リテラルでなければなりません。それぞれ、ALL リテラルを除く任意の表意定数にすることができます。 表意定数が指定されている場合には、1 文字の非数字リテラルと見なされます。

IBM Extension

いずれか一方のリテラルが DBCS リテラルである場合は、両方を DBCS リテラルにしなければなりません。いずれか一方のリテラルが国別リテラルである場合は、両方を国別リテラルにしなければなり

| ません。表意定数 SPACE は DBCS リテラルまたは国別リテラルとして使用できます。

End of IBM Extension

ALL
連続した分離文字が 1 つ以上あっても、1 つだけであるかのように取り扱われ、その 1 つが分離文字受け入れフィールド (指定された場合) へ移されます。送り出しフィールドの中の分離文字は、基本英数字項目として取り扱われ、MOVE ステートメントの規則に従って現行の分離文字受け入れフィールドへ移動されます。

IBM Extension

ALL が DBCS または国別 ID またはリテラルで使用されると、送り出しフィールド内の分離文字は同じタイプ (DBCS または国別) として扱われます。これらは MOVE ステートメントの規則に従って移動されます。

End of IBM Extension

DELIMITED BY ALL が指定されて**いない**で、2 つ以上の連続する分離文字が検出されると、現行のデータ受け入れフィールドは、データ受け入れフィールドの記述に従ってスペースまたはゼロで埋められます。

IBM Extension

現行受け入れフィールドが国別項目の場合、必要に応じて、NTLPADCHAR コンパイラーまたは PROCESS ステートメント・オプションで指定されている「国間」埋め込み文字で埋められます。

End of IBM Extension

1 つの分離文字に複数の文字が含まれている場合、それが分離文字として認識されるのは、分離文字が連続しており、送り出しフィールドで指定されたとおりの順序になっている場合だけです。

複数の分離文字が指定されている場合には、OR 条件が存在し、重なり合わない各分離文字は、指定された順序で送り出しフィールドの中で認識されます。例えば、DELIMITED BY "AB" または "BC" が指定された場合には、送り出しフィールド内に AB または BC があれば分離文字と見なされ、ABC があれば AB があったと見なされます。データ・カウント・フィールド、ポインター・フィールド、およびフィールド・カウント・フィールドは、それぞれ PICTURE 文字ストリング中に記号 P をもたない整数項目でなければなりません。

INTO 句

ID-4
データ受け入れフィールドを表します。

それぞれが USAGE DISPLAY をもっていなければなりません。これらのフィールドは次のように定義できます。

- 英字
- 英数字
- 数字 (PICTURE 文字ストリング中に記号 P をもたない)

IBM Extension

ID-4 は浮動小数点項目として定義できません。

End of IBM Extension

IBM Extension

ID-4 は DBCS データ項目または国別データ項目にすることができます。

End of IBM Extension

DELIMITER IN

ID-5 は**分離文字受け入れフィールド**を表します。 ID-5 は英数字でなければなりません。

IBM Extension

ID-5 は DBCS データ項目または国別データ項目にすることができます。

End of IBM Extension

DELIMITER IN 句を指定できるのは、DELIMITED BY 句を指定する場合だけです。 ID は、英数字編集または数字編集項目として定義されていてはなりません。 ID-1 に区切り文字がない場合は、ID-5 はスペースで埋められます。

IBM Extension

ID-5 が国別項目の場合、必要に応じて、NTLPADCHAR コンパイラーまたは PROCESS ステートメント・オプションで指定されている「国間」埋め込み文字で埋められます。

End of IBM Extension

COUNT IN

ID-6 は、各データ転送用の**データ・カウント・フィールド**であり、PICTURE 文字ストリング内に記号 P をもたない整数データ項目でなければなりません。各フィールドでは、データ受け入れフィールドに移動される、送り出しフィールド内の調べられた文字のカウント (分離文字または送り出しフィールドの終わりによって終了される) が保持されます。分離文字はこのカウントに含まれません。

IBM Extension

ID-1 (送り出しフィールド) が DBCS データ項目または国別データ項目である場合、 ID-6 は、送り出しフィールド内で調べられた文字数 (バイト数ではない) を示します。

End of IBM Extension

DELIMITED BY 句を指定する場合を除き、COUNT IN 句を指定してはなりません。

POINTER 句

ID-7

ID-7 は、PICTURE 文字ストリング内に記号 P をもたない整数データ項目であり、送り出しフィールド内の相対位置を示す値を含みます。この句を指定する場合には、UNSTRING ステートメントの実行前にこのフィールドを初期設定しなければなりません。

TALLYING IN 句

ID-8

ID-8 は、**フィールド・カウント・フィールド**です。これは、PICTURE 文字ストリング内に記号 P をもたない整数データ項目を介してユーザーによって初期設定され、この UNSTRING ステートメントの実行で影響されたデータ受け入れフィールドの数だけ増やされます。

ON OVERFLOW 句

命令ステートメント-1 は、次の場合に実行されます。

- ポインター値 (明示または暗黙) が 1 より小さい場合。
- ポインター値 (明示または暗黙) が、送り出しフィールドの長さと等しい値を超えている場合。
- すべてのデータ受け入れフィールドが影響された後、送り出しフィールドにまだ未調査の文字が含まれている場合。

上記のいずれかの条件が生じると、次のようになります。

1. オーバーフロー条件が発生し、それ以上データが転送されません。
2. UNSTRING 操作が終了されます。
3. NOT ON OVERFLOW 句が指定されていても無視されます。
4. 制御権が UNSTRING ステートメントの終わり、または命令ステートメント-1 (ON OVERFLOW 句が指定されている場合) に移動されます。

制御権が命令ステートメント-1 に渡されると、命令ステートメント-1 に指定されている各ステートメントについての規則に従って実行が継続されます。制御権の明示的な移動をもたらすプロシージャー分岐ステートメントまたは条件ステートメントが実行される場合には、制御権はそのステートメントについての規則に従って移動されます。それ以外の場合には、制御権は命令ステートメント-1 の実行完了時に UNSTRING ステートメントの終わりに移動されます。

オーバーフロー条件をもたらす条件が検出されなかった場合、ON OVERFLOW 句 (指定されている場合) は無視されます。 NOT ON OVERFLOW 句が指定されている場合には、制御権は命令ステートメント-2 に移動されます。それ以外の場合には、制御権は UNSTRING ステートメントの終わりに移動されます。

制御権が命令ステートメント-2 に移動されると、命令ステートメント-2 に指定されている各ステートメントについての規則に従って実行が継続されます。制御権の明示的な移動をもたらすプロシージャー分岐ステートメントまたは条件ステートメントが実行される場合には、制御権はそのステートメントについての規則に従って移動されます。それ以外の場合には、制御権は命令ステートメント-2 の実行完了時に UNSTRING ステートメントの終わりに移動されます。

END-UNSTRING 句

この明示範囲終了符号は、UNSTRING ステートメントの範囲を区切る働きをします。 END-UNSTRING 句を使用することによって、UNSTRING 条件ステートメントを別の条件ステートメントにネストすることができます。また、END-UNSTRING 句は、UNSTRING 命令ステートメントで使用することもできます。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

データの流れ

UNSTRING ステートメントが開始されると、データは、次の規則に従って送り出しフィールドから現行のデータ受け入れフィールドへ転送されます (現行のデータ受け入れフィールドは ID-4 です)。

1. POINTER 句が指定されていない場合には、送り出しフィールドの文字ストリングは左端の文字から調査されます。 POINTER 句が指定されている場合には、ポインター・フィールド内の値によって指定された相対文字位置からフィールドが調査されます。
2. DELIMITED BY 句が指定されている場合には、調査は左から右へ 1 文字ずつ、分離文字が検出されるまで続けられます。分離文字が検出される前に送り出しフィールドの終わりに達した場合には、調査は送り出しフィールドの最後の文字で終了します。この時点でまだ受け入れフィールドが残っている場合には、次のフィールドが選択されます。それ以外の場合には、オーバーフロー条件が起こります。
3. DELIMITED BY 句が指定されていない場合には、調査される文字数は現行のデータ受け入れフィールドのサイズと等しくなります。これは、フィールドのデータ・カテゴリーによって異なります。
   a. 受け入れフィールドが英数字または英字である場合には、調査される文字数は現行の受け入れフィールド内の文字数と等しくなります。
   b. 受け入れフィールドが数字である場合には、調査される文字数は現行の受け入れフィールドの整数部分の文字数と等しくなります。
   c. 受け入れフィールドが SIGN IS SEPARATE 文節で記述されている場合には、調査される文字数は現行の受け入れフィールドのサイズより 1 文字少なくなります。
   d. 受け入れフィールドが可変長データ項目として記述されている場合には、調査される文字数は UNSTRING 操作開始時の現行の受け入れフィールドのサイズによって判別されます。
4. 調査された文字 (すべての分離文字を除く) は英数字基本項目として取り扱われ、MOVE ステートメントの規則 (7-139 ページの『MOVE ステートメント』を参照) に従って現行のデータ受け入れフィールドに移動されます。
5. DELIMITER IN 句が指定されている場合には、送り出しフィールド内の分離文字は基本英数字項目として取り扱われ、MOVE ステートメントの規則に従って現行の分離文字受け入れフィールドに移動されます。区切り条件が送り出しフィールドの終わりである場合には、現行の分離文字受け入れフィールドはスペースで埋められます。
6. COUNT IN 句が指定されている場合には、調査された文字 (すべての分離文字を除く) 数と等しい値が、基本項目移動の規則に従ってデータ・カウント・フィールドに転送されます。
7. DELIMITED BY 句が指定されている場合には、送り出しフィールドが、分離文字の右側にある最初の文字からさらに調査されます。
8. DELIMITED BY 句が指定されていない場合には、送り出しフィールドが、調査された最後の文字の右側にある最初の文字からさらに調査されます。
9. 後続のデータ受け入れフィールドごとに、送り出しフィールド内のすべての文字が転送され終わるまで、あるいはデータの入っていないデータ受け入れフィールドがなくなるまで、上記のプロシージャーが繰り返されます。
10. POINTER 句が指定されている場合には、ポインター・フィールドの値は、送り出しフィールド内の文字が調査されるごとに 1 ずつ増やされるかのように変化します。 UNSTRING ステートメントの実行完了時には、ポインター・フィールドに、初期値と送り出しフィールド内の調査された文字数とを加算した値が入ります。
11. TALLYING 句が指定されていて、UNSTRING ステートメントの実行が完了したときには、フィールド・カウント・フィールドに、初期値と影響されたデータ受け入れ域の数とを加算した値が入ります。

注: すべての添え字付け、参照変更、可変長の計算、または関数の評価は、UNSTRING ステートメントの処理の開始時に 1 回だけ実行されます。

UNSTRING ステートメントの ID のいずれかが添え字付きまたは指標付きである場合には、添え字および指標は次のように評価されます。

- 送り出しフィールド、ポインター・フィールド、またはフィールド・カウント・フィールドに関連付けられている添え字または指標は、データが転送される直前に 1 回だけ評価されます。
- 分離文字、データ受け入れフィールド、分離文字受け入れフィールド、またはデータ・カウント・フィールドに関連付けられている添え字または指標は、データが当該データ項目へ転送される直前に評価されます。

図 7-13 に、UNSTRING ステートメントの実行規則を示します。

```
UNSTRING ID-SEND DELIMITED BY DEL-ID  OR   ALL " * "
      INTO ID-R1  DELIMITER  IN ID-D1  COUNT  IN ID-C1
           ID-R2  DELIMITER  IN ID-D2
           ID-R3  DELIMITER  IN ID-D3    COUNT  IN ID-C3
           ID-R4  COUNT  IN ID-C4
      WITH POINTER ID-P
      TALLYING  IN  ID-T
      ON OVERFLOW  GO TO  OFLOW-EXIT.
```

(すべてのデータ受信フィールドは、英数字として定義されます)

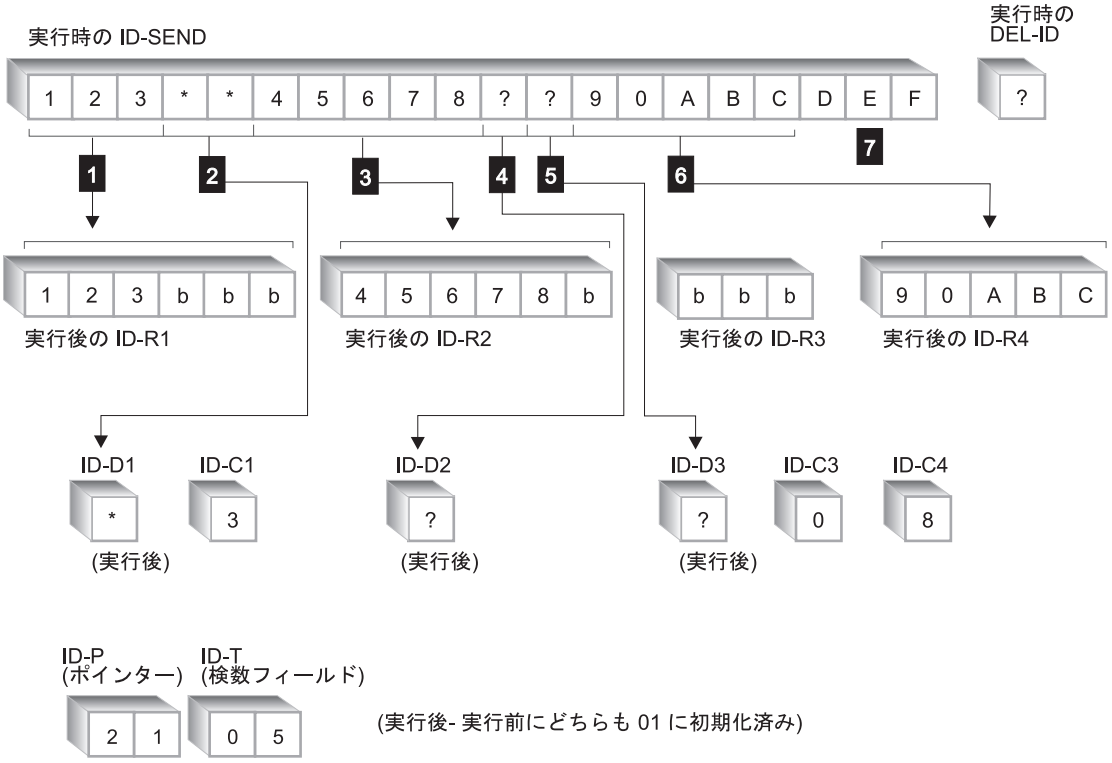


図 7-13. UNSTRING ステートメントの実行結果

1 ID-R1 に 3 文字が入れられます。

2 ALL * が指定されているため、連続するすべてのアスタリスクが処理されますが、ID-D1 には 1 つのアスタリスクだけが入れられます。

3 ID-R2 に 5 文字が入れられます。

4 ID-D2 に ? が入れられます。現行の受け入れフィールドは ID-R3 になります。

5 ID-D3 に ? が入れられます。 ID-R3 はスペースで埋められます。文字は転送されないため、ID-C3 に 0 が入れられます。

6 ID-R4 に 5 文字が入れられる前に、分離文字が検出されません。ID-C4 に 8 が入れられます。これは、最後の分離文字以降に調べられた文字の数を表します。

7 ID-P が 21 に更新されます。これは、送り出しフィールドの全長 + 1 です。ID-T が 5 に更新されます。これは、影響を受けたフィールド数 + 1 です。 ID-SEND 内に未調査の文字がないため、OVERFLOW EXIT は使用されません。

UNSTRING ステートメントの例

次の例では、UNSTRING ステートメントに適用されるいくつかの考慮事項を示します。

データ部では、UNSTRING ステートメントの影響を受ける以下の入力レコードが定義されているものとします。

```
01 INV-RCD.
   05 CONTROL-CHARS   PIC XX.
   05 ITEM-INDENT     PIC X(20).
   05 FILLER          PIC X.
   05 INV-CODE        PIC X(10).
   05 FILLER          PIC X.
   05 NO-UNITS        PIC 9(6).
   05 FILLER          PIC X.
   05 PRICE-PER-M     PIC 99999.
   05 FILLER          PIC X.
   05 RTL-AMT         PIC 9(6).99.
```

次の 2 つのレコードが UNSTRING ステートメントの受け入れフィールドとして定義されています。DISPLAY-REC は印刷出力用に使用されます。 WORK-REC は内部処理用に使用されます。

```
01 DISPLAY-REC
   05 INV-NO       PIC X(6).
   05 FILLER       PIC X VALUE SPACE
   05 ITEM-NAME    PIC X(20).
   05 FILLER       PIC X VALUE SPACE
   05 DISPLAY-DOLS PIC 9(6).

01 WORK-REC
   05 M-UNITS      PIC 9(6).
   05 FIELD-A      PIC 9(6).
   05 WK-PRICE
      REDEFINES
      FIELD-A      PIC 9999V99.
   05 INV-CLASS    PIC X(3).
```

UNSTRING ステートメントの中で制御フィールドとして使用するために、次のフィールドも定義されています。

```
01 DBY-1            PIC X, VALUE IS ".".
01 CTR-1            PIC 99, VALUE IS ZERO.
01 CTR-2            PIC 99, VALUE IS ZERO.
01 CTR-3            PIC 99, VALUE IS ZERO.
01 CTR-4            PIC 99, VALUE IS ZERO.
01 DLTR-1           PIC X.
01 DLTR-2           PIC X.
01 CHAR-CT          PIC 99, VALUE IS 3.
01 FLDS-FILLED      PIC 99, VALUE IS ZERO.
```

手続き部では、INV-RCD のサブフィールドを DISPLAY-REC および WORK-REC のサブフィールドに移動するために、次の UNSTRING ステートメントが書かれています。

```
UNSTRING INV-RCD
 DELIMITED BY ALL SPACES
   OR "/"
   OR DBY-1
```

```
INTO ITEM-NAME COUNT IN CTR-1,
INV-NO DELIMITER IN DLTR-1
  COUNT IN CTR-2,
INV-CLASS,
M-UNITS COUNT IN CTR-3,
DISPLAY-DOLS DELIMITER IN DLTR-2
  COUNT IN CTR-4
WITH POINTER CHAR-CT
TALLYING IN FLDS-FILLED
ON OVERFLOW
  GO TO UNSTRING-COMPLETE.
```

UNSTRING ステートメントを出す前に、CHAR-CT (ポインター項目) に値 3 を入れます。これは INV-RCD の始まりにある 2 文字の制御文字を処理の対象としないためです。 DBY-1 には分離文字として使用されるピリオドが入れられ、FLDS-FILLED (フィールド・カウント項目) には値 0 が入れられます。さらに、 図 7-14 で示されているデータが INV-RCD の中へ読み込まれます。

図 7-14. UNSTRING ステートメントの例 - 入力データ

UNSTRING ステートメントの実行時には、次の処理が行われます。

1. INV-RCD の桁 3 ～ 18 (FOUR-PENNY-NAILS) が ITEM-NAME に入れられ、その区域の中で左寄せされます。未使用の文字位置はスペースで埋められます。値 16 が CTR-1 に入れられます。
2. 分離文字として ALL SPACES が指定されているので、 5 つの連続した SPACE 文字も 1 つの分離文字と見なされます。
3. 桁 24 ～ 29 (707890) が INV-NO に入れられます。分離文字 / が DLTR-1 に入れられ、値 6 が CTR-2 に入れられます。
4. 桁 31 ～ 33 が INV-CLASS に入れられます。分離文字は SPACE ですが、分離文字用の受け入れ域として定義されたフィールドがないため、SPACE はバイパスされます。
5. 桁 35 ～ 40 (475120) が調べられ、M-UNITS に入れられます。分離文字は SPACE ですが、分離文字用の受け入れ域として定義された受け入れフィールドがないため、SPACE はバイパスされます。値 6 が CTR-3 に入れられます。
6. 桁 42 ～ 46 (00122) が FIELD-A に入れられ、その区域の中で右寄せされます。高位の桁位置は 0 (ゼロ) で埋められます。分離文字は SPACE ですが、分離文字用の受け入れ域として定義されたフィールドがないため、SPACE はバイパスされます。
7. 桁 48 ～ 53 (000379) が DISPLAY-DOLS に入れられます。分離文字ピリオドが DLTR-2 に入れられ、値 6 が CTR-4 に入れられます。
8. すべての受け入れフィールドが影響され、INV-RCD 内のデータの 2 文字が調査されていないため、ON OVERFLOW 出口が使用され、UNSTRING ステートメントの実行が完了します。

UNSTRING ステートメントの実行終了時、DISPLAY-REC には次のデータが含まれます。

```
707890 FOUR-PENNY-NAILS     000379
```

WORK-REC には次のデータが含まれます。

```
475120000122BBA
```

CHAR-CT (ポインター・フィールド) には値 55 が入り、FLD-FILLED (フィールド・カウント・フィールド) には値 6 が入ります。

注: 一連の MOVE ステートメントの代わりに、1 つの UNSTRING ステートメントを書くことができます。

WRITE ステートメント

WRITE ステートメントは、出力ファイルまたは入出力ファイル用にレコードを解放します。

WRITE ステートメントの実行時には、関連する索引付きまたは相対ファイルが OUTPUT、I-O、または EXTEND モードでオープンされていなければなりません。関連する順次ファイルは、OUTPUT または EXTEND (装置タイプ TAPEFILE、 DISK、または DATABASE) モードでオープンされていなければなりません。

IBM Extension

- 形式 3 - FORMATFILE
- 形式 4 - TRANSACTION (非サブファイル)
- 形式 5 - TRANSACTION (サブファイル)

End of IBM Extension

IBM Extension

OVRDBF CL コマンドの INHWRT パラメーターによって、このステートメントの処置をプログラム実行時に禁止することができます。このパラメーターが指定された場合、データに依存するエラーに対しては、非ゼロのファイル状況コードが設定されません。データに依存するエラーの例として、重複キーおよびデータ変換エラーがあります。

このコマンドについて詳しくは、**IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『プログラミング』のセクション『*CL および API*』を参照してください。

End of IBM Extension

順次ファイル

ADVANCING 句および END-OF-PAGE 句は、印刷されるページ上の各行の垂直位置決めを制御します。印刷されるページが中間装置 (例えばディスク) 上で保持されると、それが編集またはブラウズされたときに、形式が予期した出力と異なることがあります。

注: 1 つの WRITE ステートメントの中に ADVANCING PAGE 句と END-OF-PAGE 句の両方を指定してはなりません。

外部で定義されたファイルの境界を超えて書き込みを試みると、WRITE ステートメントの処理が正常に完了せず、EXCEPTION/ERROR 条件が起こります。レコード名の内容は影響を受けません。その後 8-34 ページの『USE ステートメントのプログラミング上の注意事項』で説明されているエラー処理の規則に従って、処理が行われます。

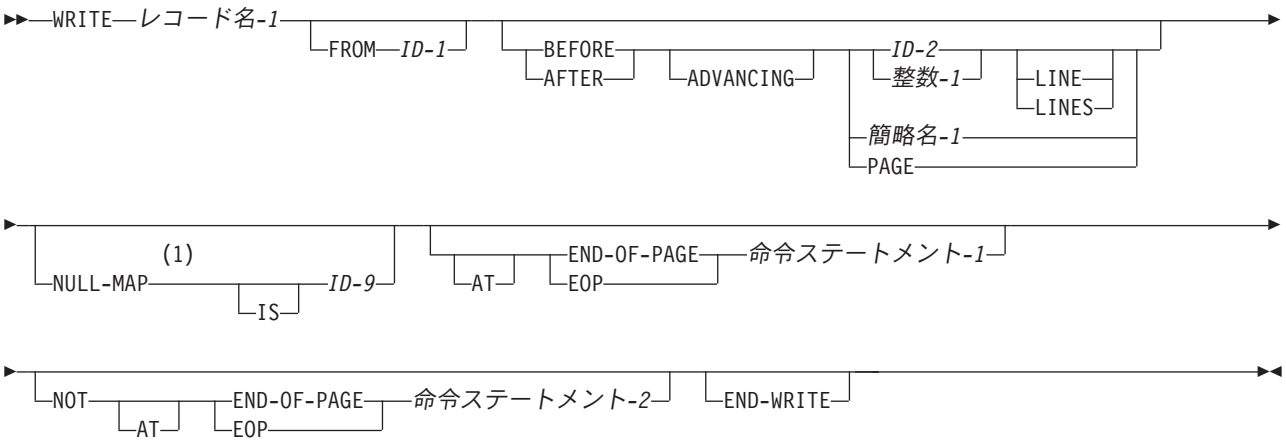
装置タイプ TAPEFILE または DISKETTE 上の順次ファイルの場合、マルチボリューム OUTPUT ファイルのボリュームの終わりが検出されると、WRITE ステートメントによって以下の操作が行われます。

1. 標準の終了ボリューム・ラベル・プロシージャーが実行されます。

2. ボリュームの切り替えが行われます。
3. 標準の開始ボリューム・ラベル・プロシージャーが実行されます。

ボリュームの終わりに達したという標識はプログラムに返されません。

WRITE ステートメント - 形式 - 順次ファイル



注:

1 IBM 拡張

レコード名-1

データ部の FD 記入項目で定義されていなければなりません。レコード名-1 は修飾できます。レコード名-1 をソート・ファイルまたはマージ・ファイルと関連付けてはなりません。

OCCURS DEPENDING ON (ODO) 配列が入ったレコードの WRITE を実行するには、その前に ODO
配列のサイズを有効な数値に確実に設定する必要があります。

IBM Extension

レコード名-1 は浮動小数点データ項目にすることができます。

End of IBM Extension

IBM Extension

レコード名-1 は DBCS データ項目を定義できます。レコード名-1 が DBCS データ項目の場合、ID-1 は、DBCS データ項目でなければなりません。

End of IBM Extension

FROM

FROM を指定すると、実行結果は次の場合と同じになります。

```
MOVE identifier-1 TO record-name-1
WRITE record-name-1
```

移動は、CORRESPONDING 句のない MOVE ステートメントの規則に従って実行されます。

WRITE ステートメントの実行後、情報は ID-1 ではまだ使用可能ですが、レコード名-1 では使用不可になります。 (7-38 ページの『INTO/FROM ID 句』を参照。)

ID-1
英数字データ項目または数字編集データ項目でなければなりません。データはこのフィールドから受け入れフィールドへ転送されます。

ID-1 は、英数字または DBCS 関数 ID の名前にすることができます。

IBM Extension

ID-1 は、浮動小数点にすることも、日時データ項目にすることもできます。

End of IBM Extension

ID-2
整数データ項目でなければなりません。

ファイルの最大レコード・サイズはファイルの作成時に設定され、後で変更することはできません。

レコード名-1 と ID-1 で同じ記憶域を参照してはなりません。

WRITE ステートメントの実行後には、次のいずれかの場合を除き、レコードはレコード名-1 で使用可能でなくなります。

- 関連するファイルが SAME RECORD AREA 文節の中で指定されている (この場合には、レコードは、SAME RECORD AREA 文節の中で指定されている他のファイルのレコードとしても使用可能です)。
- WRITE ステートメントが境界違反のために失敗する。

これらのいずれかの場合には、レコードはレコード名-1 でまだ使用できます。

ファイル位置標識は、WRITE ステートメントの実行によって影響されません。

ファイルの中にレコードを保管するのに必要な文字桁数は、COBOL プログラムの中でそのレコードの論理記述によって定義された文字桁数と同じであってもなくてもかまいません。 (6-69 ページの『PICTURE 文節の編集』および 6-89 ページの『USAGE 文節』を参照してください。)

ファイル制御記入項目で FILE STATUS 文節が指定されている場合には、WRITE ステートメントの実行時に、実行が成功するかどうかに関係なく、関連する状況キーが更新されます。

I-O モードでオープンされた順次ファイルに対しては、WRITE ステートメントを実行することはできません。

ADVANCING 句: ADVANCING 句は、ページ上の出力レコードの位置決めを制御します。 この句は、装置タイプ PRINTER にのみ適用されます。次の規則が適用されます。

1. BEFORE ADVANCING を指定した場合には、 ページ送りの前に行が印刷されます。
2. AFTER ADVANCING を指定した場合には、 ページ送りの後で行が印刷されます。
3. ID-2 を指定した場合には、ID-2 の現行値と等しい行数だけページ送りが行われます。 ID-2 は整数データ項目でなければなりません。
4. 整数-1 を指定した場合には、整数-1 の値と等しい行数だけページ送りが行われます。
5. 整数-1、または ID-2 の値はゼロにすることもできます。
6. 簡略名を指定した場合には、システム特有の処置が取られます。簡略名は、SPECIAL-NAMES 段落の環境名-1 と等価にする必要があります (有効な環境名は、 5-6 ページの表 5-1にリストされています)。簡略名として使用できる値の詳細は 5-4 ページの『SPECIAL-NAMES 段落』を参照してください。

7. PAGE を指定した場合には、装置が次の論理ページに位置付けられる前 (BEFORE) または後 (AFTER) (使用されている句による) に、レコードが論理ページに印刷されます。使用されている装置にとって PAGE が意味をもたない場合には、BEFORE または AFTER (指定されている句による) ADVANCING 1 LINE が使用されます。

   FD 記入項目に LINAGE 文節が入っている場合には、その文節で指定されているように、次のページの最初の印刷可能行への位置変更が行われます。 LINAGE 文節が省略されている場合には、次のページの 1 行目への位置変更が行われます。

   LINAGE-COUNTER の規則: このファイルについて LINAGE 文節が指定されている場合は、WRITE ステートメントの実行時に、関連する LINAGE-COUNTER 特殊レジスターが次の規則に従って変更されます。

   a. ADVANCING PAGE が指定されている場合には、LINAGE-COUNTER が 1 にリセットされます。
   b. ADVANCING ID-2 または整数-1 が指定されている場合には、LINAGE-COUNTER が ID-2 または整数-1 内の値だけ増やされます。
   c. ADVANCING 句が省略されている場合には、LINAGE-COUNTER が 1 だけ増やされます。
   d. 装置が新しいページの最初の使用可能行に位置変更されると、LINAGE-COUNTER が 1 にリセットされます。

この句が省略される場合には、AFTER ADVANCING 1 LINE が書かれている場合と同様に、自動的な行送りが行われます。

NULL-MAP IS 句:

IBM Extension

ページ『NULL-MAP IS 句』のこの句についての説明を参照してください。

End of IBM Extension

END-OF-PAGE 句: この句が指定されている (さらに、このファイルについての FD 記入項目に LINAGE 文節が入っている) 場合、WRITE ステートメントの実行中に、印刷されるページの論理的な終わりに達すると、命令ステートメントが実行されます。

NOT AT END-OF-PAGE 句が指定されている WRITE ステートメントの処理後に END-OF-PAGE 条件が存在しない場合には、制御はその句に関連する命令ステートメントに移動されます。

プリンター・ファイルに関する特別の考慮事項: キーワード END-OF-PAGE とキーワード EOP は同等です。END-OF-PAGE 句を指定する場合には、このファイルの FD 記入項目に LINAGE 文節が入っていなければなりません。 END-OF-PAGE を指定し、WRITE ステートメントの処理後に END-OF-PAGE 条件が存在する場合には、END-OF-PAGE 命令ステートメントが処理されます。印刷されるページの論理的な終わりは、レコード名に関連する LINAGE 文節の中で指定します。

プリンター・ファイルについての WRITE ステートメントの処理によってページ本体のフッター域での印刷または行送りが行われる場合には、そのファイルの END-OF-PAGE 条件が起こります。これは、WRITE ステートメントの実行により、LINAGE-COUNTER 中の値が、LINAGE 文節中の WITH FOOTING 句で指定された値と等しいかまたはそれより大きくなったときに起こります。 WRITE ステートメントが処理されてから、END-OF-PAGE 命令ステートメント (コーディングされている場合) が処理されます。

WRITE ステートメントの処理が現行のページ本体内で完全に実行できないときには、END-OF-PAGE 句が指定されているかどうかに関係なく、必ず自動ページ・オーバーフロー条件が起こります。これは、WRITE ステートメントを処理した場合に、LINAGE-COUNTER の値が LINAGE 文節で指定されているページ本体の行数を超えるときに起こります。このケースでは、装置が LINAGE 文節で指定されているように次の論理ページの最初の印刷可能行に位置変更される前または後 (指定されているオプションによる) に、行が印刷されます。

END-OF-PAGE 句が指定されている場合には、END-OF-PAGE 命令ステートメントが実行されます。次の場合には、END-OF-PAGE 条件と自動ページ・オーバーフロー条件が同時発生します。

- LINAGE 文節の WITH FOOTING 句が指定されていない場合。この場合には、END-OF-PAGE 条件と自動ページ・オーバーフロー条件の間に区別がなくなります。 FOOTING 句が指定されていない場合には、論理ページの下部にあるフッター情報を印刷することができません。
- WITH FOOTING 句が指定されているが、WRITE ステートメントの実行によって LINAGE-COUNTER が LINAGE 文節で指定されているフッターの値とページ本体の値の両方を超える場合。

キーワード END-OF-PAGE とキーワード EOP は同等です。

注: ADVANCING PAGE 句と END-OF-PAGE 句の両方を 1 つの WRITE ステートメント内に指定することはできません。

FORMATFILE に関する特別の考慮事項: キーワード END-OF-PAGE とキーワード EOP は同等です。END-OF-PAGE 句を指定し、FORMATFILE ファイルについての WRITE ステートメントの処理後に EOP 条件が存在する場合には、END-OF-PAGE 命令ステートメントが実行されます。 FORMATFILE ファイルの EOP 条件が起こるのは、そのファイルについての WRITE ステートメントの処理時にページの論理的な終わりに到達したときです。印刷されるページの論理的な終わりは、CRTPRTF コマンドまたは OVRPRTF コマンドのオーバーフロー行番号パラメーターで指定します。

END-WRITE 句: この明示範囲終了符号は、WRITE ステートメントの有効範囲を区切る働きをします。END-WRITE 句を使用することによって、WRITE 条件ステートメントを別の条件ステートメントにネストすることができます。また、END-WRITE 句は、WRITE 命令ステートメントで使用することもできます。

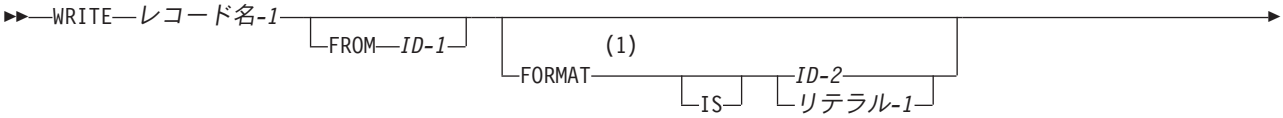
詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

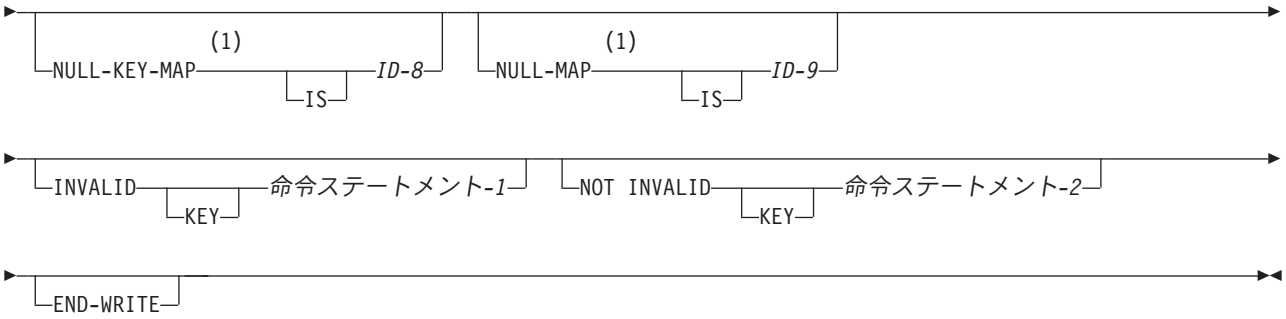
マルチボリューム・ファイル: マルチボリューム OUTPUT ファイル (テープまたは順次直接アクセス・ファイル) のボリュームの終わりが検出されると、WRITE ステートメントは次の操作を実行します。

- 標準の終了ボリューム・ラベル・プロシージャー
- ボリュームの切り替え
- 標準の開始ボリューム・ラベル・プロシージャー

索引付きファイルおよび相対ファイル

WRITE - 形式 2 - 索引付きおよび相対ファイル





注:

1 IBM 拡張

レコード名-1

データ部の FD 記入項目で定義されていなければなりません。レコード名-1 は修飾できます。レコード名-1 をソート・ファイルまたはマージ・ファイルと関連付けてはなりません。

OCCURS DEPENDING ON (ODO) 配列が入ったレコードの WRITE を実行するには、その前に ODO
配列のサイズを有効な数値に確実に設定する必要があります。

相対ファイルの場合のみ、レコード名-1 内の文字桁数は、置き換えられるレコード内の文字桁数と等しくなければなりません。レコード名-1 をソート・ファイルまたはマージ・ファイルと関連付けてはなりません。

FROM

FROM を指定すると、実行結果は次の場合と同じになります。

```
MOVE identifier-1 TO record-name-1
WRITE record-name-1
```

WRITE ステートメントの実行後、情報は ID-1 ではまだ使用可能ですが、レコード名-1 では使用不可になります。 (7-38 ページの『INTO/FROM ID 句』を参照。)

ID-1

英数字または数字編集データ項目でなければなりません。データはこのフィールドから受け入れフィールドへ転送されます。

レコード名-1 と ID-1 の両方で同じ記憶域を参照することはできません。

ID-1 は、英数字または DBCS 関数 ID の名前にすることができます。

索引付きファイルの書き込みに関する考慮事項: WRITE ステートメントを実行する前に、基本レコード・キー (ファイル制御記入項目で定義されている RECORD KEY データ項目) を希望の値に設定しなければなりません。 WRITE ステートメントが処理されると、システムはレコードを解放します。

IBM Extension

DUPLICATES 句を指定する場合には、ある形式用のレコード・キーの値が固有である必要はありません (5-40 ページの『RECORD KEY 文節』を参照してください)。この場合、システムは、レコードに対する後続の順次アクセスにおいて、DDS で指定された順序で検索を行えるようにレコードを保管します。

End of IBM Extension

SEQUENTIAL アクセスをもち、OUTPUT 用にオープンされており、ブロック化が有効である (BLOCK CONTAINS 文節が指定されている) 固定サイズの索引付きファイルにレコードが書き込まれる場合、ブロック化因数は、レコードのブロックによってファイルの終わりに到達するポイントで 1 に変わります。

ALTERNATE RECORD KEY 文節も、ファイル制御記入項目で指定されている場合、DUPLICATES 句が指定されない限り、それぞれの代替キーは固有でなければなりません。 DUPLICATES 句を指定する場合には、代替レコード・キーの値が固有である必要はありません。

この時点でのファイル内の残りのレコード数は、ブロック内のレコード数よりも少なくなります。

ファイル制御記入項目中に ACCESS IS SEQUENTIAL が指定されている場合には、レコードを RECORD KEY 値の昇順に解放しなければなりません。

ファイル制御記入項目中に ACCESS IS RANDOM または ACCESS IS DYNAMIC が指定されている場合には、プログラマーが指定する順序に従ってレコードを解放できます。索引付きファイルがランダム・アクセス・モードでアクセスされるときに、入出力ステートメントで FORMAT 句が指定されていない場合には、定義された最初の形式が使用されます。複数形式論理ファイルへの書き込み時には、WRITE ステートメントで形式を指定しなければなりません。

相対ファイルの書き込みに関する考慮事項: OUTPUT ファイルに対しては、WRITE ステートメントにより次のような処置が取られます。

- ACCESS IS SEQUENTIAL が指定されている場合。

  最初に解放されるレコードの相対レコード番号は 1 であり、2 番目に解放されるレコードの相対レコード番号は 2 であり、... というようになります。

  ファイル制御記入項目内に RELATIVE KEY が指定されている場合には、WRITE ステートメントの実行時に、解放されたばかりのレコードの相対レコード番号が RELATIVE KEY に入れられます。
- ACCESS IS RANDOM または ACCESS IS DYNAMIC が指定されている場合には、WRITE ステートメントを出す前に、このレコードについての希望の相対レコード番号を RELATIVE KEY に入れなければなりません。このレコードは、WRITE ステートメントの実行時に、ファイル内の指定された相対レコード番号の位置に入れられます。

I-O モードでオープンされるファイルについては、ACCESS IS RANDOM または ACCESS IS DYNAMIC が指定されていなければなりません。 WRITE ステートメントは、ファイルに新しいレコードを挿入します。 WRITE ステートメントを出す前に、このレコードについての希望の相対レコード番号を RELATIVE KEY に入れておかなければなりません。このレコードは、WRITE ステートメントの実行時に、ファイル内の指定された相対レコード番号の位置に入れられます。

レコードの DELETE 操作を許可しない (例えば、ALWDLT(*NO) パラメーターを指定した CRTPF を使用して) 物理ファイルについては、レコードの更新操作を許可 (つまり、ALWUPD(*YES) パラメーターを指定した CRTPF) しなければなりません。

FORMAT 句:

IBM Extension

ファイルに複数のレコード形式がある場合に必要です。

FORMAT 句で指定する値には、この入出力操作で使用するレコード形式の名前が入ります。システムはこれを使用して、どのレコード形式に対して操作を行うかを指定または選択します。

ID-2 を指定する場合には、10 文字以下の英数字データ項目にしなければなりません。

リテラル-1 を指定する場合は、10 文字以下の大文字の文字ストリングにしなければなりません。

索引付きファイルがランダム・アクセス・モードでアクセスされるときに、入出力ステートメントで FORMAT 句が指定されていない場合には、定義された最初の形式が使用されます。

End of IBM Extension

NULL-KEY-MAP IS 句:

IBM Extension

ページ『NULL-KEY-MAP IS 句』のこの句に関する説明を参照してください。

End of IBM Extension

NULL-MAP IS 句:

IBM Extension

ページ『NULL-MAP IS 句』のこの句についての説明を参照してください。

End of IBM Extension

INVALID KEY 句: このファイルに対して明示または暗黙の EXCEPTION/ERROR プロシージャーが指定されていない場合には、 INVALID KEY 句を指定しなければなりません。

外部で定義されたファイルの境界を超えて書き込みを試みると、WRITE ステートメントの実行が失敗し、EXCEPTION/ERROR 条件が起こります。

ランダムまたは動的アクセス・モードの相対ファイルの場合、INVALID KEY 条件が起こるのは、RELATIVE KEY に、すでにデータの入っているレコードが指定されたときです。

ランダムまたは動的アクセス・モードの索引付きファイルの場合、INVALID KEY 条件が起こるのは、レコード域内のキー・フィールドの値が、すでに存在するレコードのものと等しく、DUPLICATES が認められていないときです。

順次アクセス・モードの索引付きファイルの場合、INVALID KEY 条件が起こるのは、連続するレコードの基本レコード・キーの値が昇順になっていないときです。

IBM Extension

重複キーが認められているファイルの場合、INVALID KEY 条件が起こるのは、レコード・キーの値が前のレコードのものよりも小さいときだけです。

End of IBM Extension

無効キー条件が検出されると、WRITE ステートメントの実行が失敗し、レコードの内容は影響されません。プログラム実行は 7-38 ページの『INVALID KEY 条件』に説明されている規則に従って続行されます。

NOT INVALID KEY 句: NOT INVALID KEY 句が指定されていて、WRITE ステートメントの実行終了時に無効キー条件が存在しない場合には、制御はこの句に関連する命令ステートメントに渡されます。

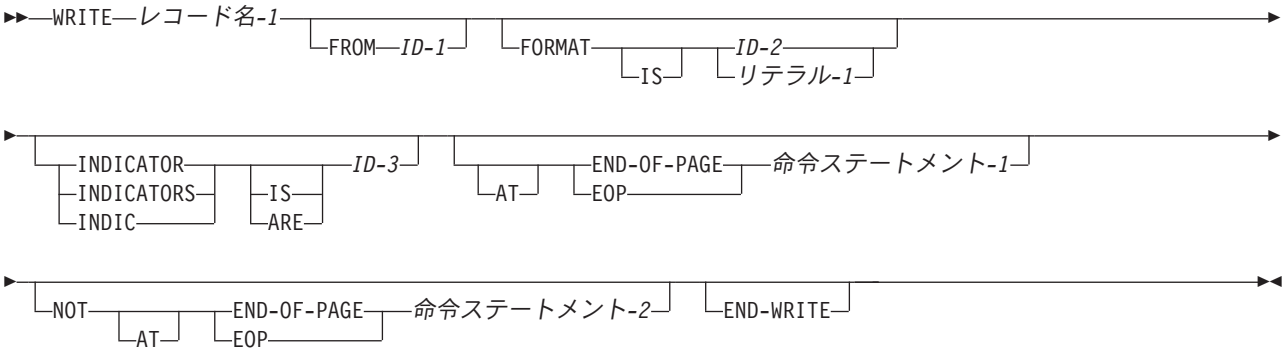
END-WRITE 句: この明示範囲終了符号は、WRITE ステートメントの有効範囲を区切る働きをします。END-WRITE 句を使用することによって、WRITE 条件ステートメントを別の条件ステートメントにネストすることができます。また、END-WRITE 句は、WRITE 命令ステートメントで使用することもできます。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

FORMATFILE:

IBM Extension

WRITE ステートメント - 形式 3 - FORMATFILE



FORMAT 句: ファイルに複数のレコード形式がある場合に必要です。

FORMAT 句で指定する値には、この入出力操作で使用するレコード形式の名前が入ります。システムはこれを使用して、どのレコード形式に対して操作を行うかを指定または選択します。

ID-2 を指定する場合には、10 文字以下の英数字データ項目にしなければなりません。

リテラル-1 を指定する場合は、10 文字以下の大文字の文字ストリングにしなければなりません。

全桁がブランクの値は、FORMAT 句が指定されなかった場合と同じ扱いになります。その値がファイルに対して有効でない場合は、9K の FILE STATUS が戻され、(そのファイルに適用可能であれば) USE プロシージャーが呼び出されます。

INDICATORS 句: データ・レコードが書き込まれるときに使用される標識を指定します。標識は、データ・レコードについての情報と、それがプログラムに入力された方法を渡すために使用できます。

INDICATORS 句について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

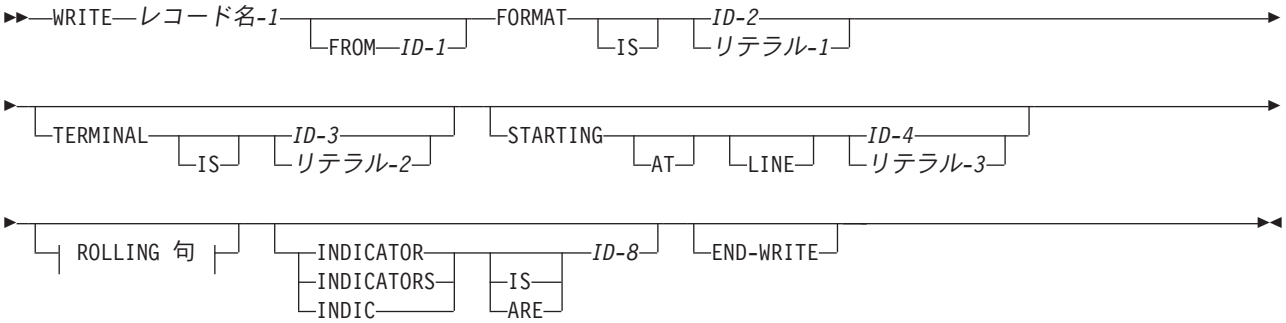
ID-3 は、OCCURS 文節なしで指定された基本ブール・データ項目か、または基本ブール・データ項目が従属しているグループ項目でなければなりません。

End of IBM Extension

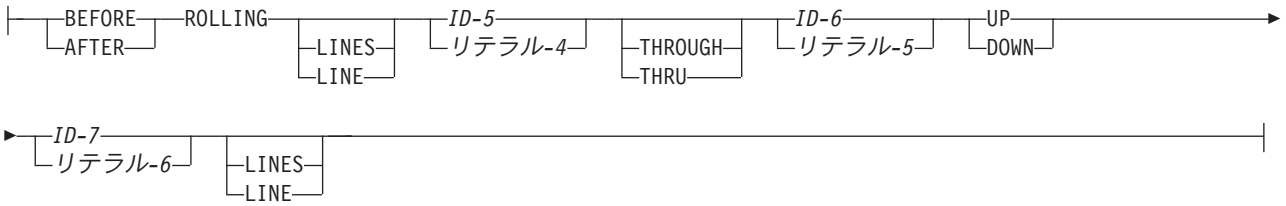
TRANSACTION (非サブファイル):

IBM Extension

WRITE - 形式 4 - TRANSACTION (非サブファイル)



ROLLING 句:



FORMAT 句: リテラル-1 または ID-2 では、書き込まれるレコード形式の名前を指定します。リテラル-1 を指定する場合は、非数字で、10 文字以下の大文字にしなければなりません。 ID-2 (指定されている場合) は、英数字データ項目で、10 文字以下の長さでなければなりません。 ID-2 がブランクである場合、WRITE ステートメントは、FORMAT 句が省略された場合と同じように実行されます。

TERMINAL 句: TERMINAL 句は、出力レコードの送信先プログラム装置を指定します。

リテラル-2 または ID-3 の内容は、ファイルによって以前に明示的または暗黙に獲得されたプログラム装置の名前でなければなりません。リテラル-2 を指定する場合は、10 文字以下の非数字にしなければなりません。 ID-3 を指定する場合は、10 文字以下の英数字データ項目を参照しなければなりません。ブランクの値は、TERMINAL 句が省略された場合と同様に扱われます。

TRANSACTION ファイルによって獲得されたプログラム装置が 1 台だけである場合は、TERMINAL 句を省略できます。そのプログラム装置は、常時、WRITE 操作用に使用できます。

複数のプログラム装置を獲得している TRANSACTION ファイルに対する WRITE 操作で TERMINAL 句が省略されると、デフォルトのプログラム装置が使用されます。

STARTING 句: STARTING 句は、可変開始行キーワードを使用するレコード形式の開始行番号を指定します。この句は、ディスプレイ装置についてのみ有効です。

フィールドが始まる実際の行番号は、次の式によって求めることができます。

実際の行 = 開始行 + DDS 開始行 - 1

ここで、

- **実際の行**は、実際の行番号
- **開始行**は、プログラム内で指定された開始行番号
- **DDS 開始行**は、データ記述仕様用紙の桁 39 ～ 41 に指定された行番号

以下の場合は、書き込み操作が正常に行われます。
- 上記の式の結果が正であり、ワークステーション画面の行数以下である。
- STARTING 句に指定された値が 0 である。この場合は、値 1 が想定されます。

以下の場合は、書き込み操作が失敗し、プログラムが終了します。
- 上記の式の結果がワークステーション画面の行数より大きい。
- STARTING 句に指定された値が負である。

STARTING 句に指定された値が画面区域内にある場合、画面区域の外側のフィールドは無視されます。

STARTING 句のリテラル-3 は数字リテラルでなければなりません。 ID-4 は基本数字項目でなければなりません。

STARTING 句を使用するには、書き込まれる形式について、DDS レコード・レベル・キーワード SLNO(*VAR) を指定しなければなりません。レコード形式でこのキーワードが指定されていない場合、STARTING 句は実行時に無視されます。

DDS キーワード CLRL も STARTING 句に影響を与えます。 CLRL は、WRITE ステートメントの実行時に画面がどの程度消去されるかを制御します。

SLNO(*VAR) および CLRL について詳しくは、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) 内のカテゴリー『データベースおよびファイル・システム』のセクション『*DB2 for i*』を参照してください。

ROLLING 句: ROLLING 句によって、ワークステーション画面に表示される行を移動できます。画面のすべてまたは一部の行を次ページ (ロールアップ) または前ページ (ロールダウン) することができます。行送りによって空けられる行は消去され、そこへ別の画面形式を書き出すことができます。この句は、ディスプレイ装置についてのみ有効です。

ROLLING は、ワークステーション画面に新しい形式を書き出そうとする WRITE ステートメントの中で指定します。書き出しを行送りの前と後のどちらに行うか、行送りしたい行の範囲、それらの行を何行送るか、および行送り操作を上方と下方のどちらに行うかを指定しなければなりません。

行送りの後、それらの行上のフィールドでは、その DDS 表示属性 (例えば、下線表示) は維持されますが、その DDS 使用属性 (例えば、入力可能性) は失われます。書き出されてから行送りされた (BEFORE ROLLING 句) 行上のフィールドも、その使用属性を失います。

形式の一部が行送りされる場合、形式全体がその使用属性を失います。複数の形式が存在する場合、行送りされた形式だけがその使用属性を失います。

ROLLING 句を指定する場合は、次の一般的な規則が適用されます。
- ROLLING 句を含む WRITE ステートメントで書き出されるそれぞれのレコード形式について、DDS レコード・レベル・キーワード ALWROL を指定しなければなりません。
- ALWROL キーワードと相互に排他的である別の DDS キーワードを使用してはなりません。

- 書き出されてから行送りされるレコード形式については、そのレコード形式が書き出されるときに表示画面が消去されるのを防ぐために、DDS キーワード CLRL または OVERLAY を指定しなければなりません。
- すべての ID およびリテラルは、正の整数値を表さなければなりません。
- 行送り開始行番号 (ID-5 またはリテラル-4) は、終了行番号 (ID-6 またはリテラル-5) を超えてはなりません。
- 開始行番号および終了行番号によって指定された、ウィンドウの外側に送られる行の内容は消えます。

詳しくは、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) 内のカテゴリー『データベースおよびファイル・システム』のセクション『*DB2 for i*』を参照してください。

INDICATORS 句: データ・レコードが書き込まれるときに使用される標識を指定します。標識は、データ・レコードについての情報と、それがプログラムに入力された方法を渡すために使用できます。

INDICATORS 句について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」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。

ID-8 は、OCCURS 文節なしで指定された基本ブール・データ項目か、または基本ブール・データ項目が従属しているグループ項目でなければなりません。

7-257 ページの図 7-15 に、行送り (ローリング) の例を示します。初期画面形式 FMT1 がワークステーション画面に書き出されています。プログラムは、この画面形式を処理し、現在、次の画面形式 FMT2 をワークステーション画面に書き出す準備ができています。 FMT2 がワークステーション画面に書き出される前に、FMT1 の一部が 2 行下方送りされます。

次の WRITE ステートメントの実行によって、FMT1 の一部が 2 行下方送りされ、FMT2 がワークステーション画面に書き出されます。

```
WRITE SCREENREC FORMAT "FMT2"
 AFTER ROLLING LINES 14 THROUGH 20
 DOWN 2 LINES
```

この WRITE ステートメントが実行されるとき、次のステップが行われます。

1. 行 14 ～ 20 の内容が 2 行下方送りされます。
   a. 行 14 ～ 18 の内容が行 16 ～ 20 に表示されるようになります。
   b. 行 14 および 15 の内容は空きになり、消去されます。
   c. 行 19 および 20 の内容はウィンドウの外側に送られて消えます。
2. 行送り操作が行われた後、FMT2 がワークステーション画面に書き出されます。
   a. FMT2 の一部は、行送り操作で空きとなった区域に書き出されます。
   b. FMT2 の一部は、FMT1 から残っているデータの上に書き出されます。
3. READ ステートメントによってワークステーション画面の内容がプログラムに戻されるときには、FMT2 の入力可能フィールドだけが戻されます。

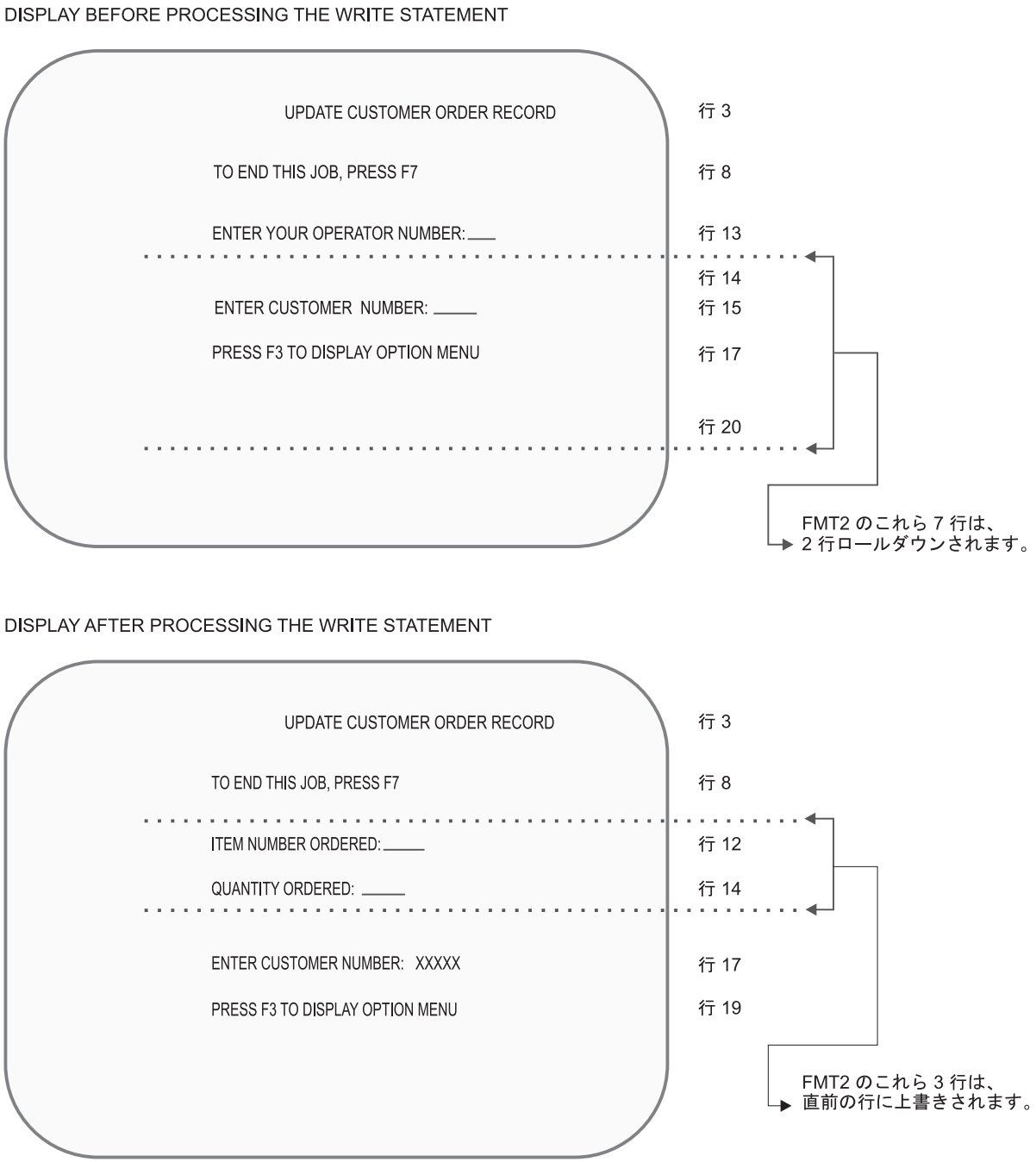


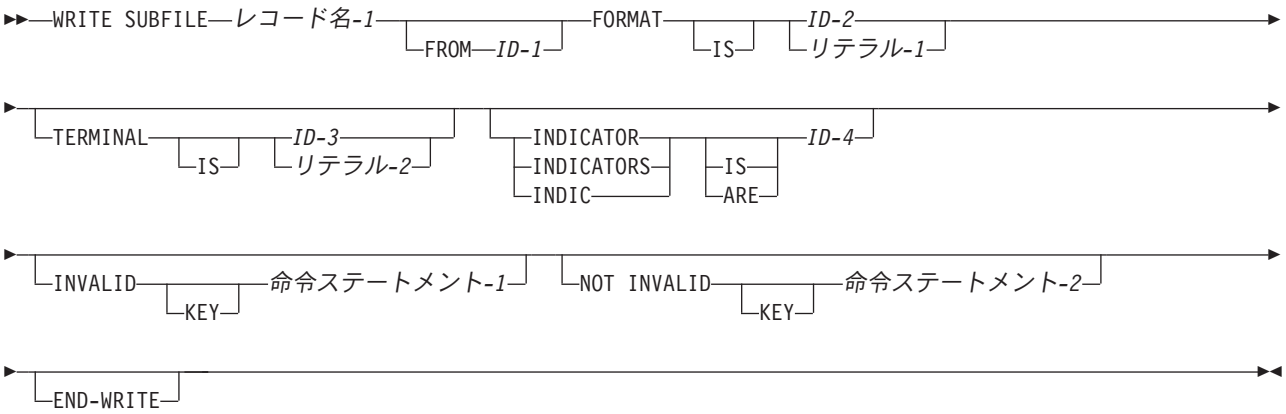
図 7-15. ROLLING 操作の例

End of IBM Extension

TRANSACTION (サブファイル):

IBM Extension

WRITE ステートメント - 形式 5 - TRANSACTION (サブファイル)



形式 5 は、ディスプレイ装置についてのみ 使用できます。その他のタイプの装置についてサブファイル形式の WRITE ステートメントを使用すると、WRITE 操作が失敗し、ファイル状況 90 が設定されます。

形式がサブファイル・レコードであり、SUBFILE が指定される場合には、書き込まれるファイルの SELECT 文節に RELATIVE KEY 文節が指定されていなければなりません。このサブファイルに書き込まれるレコードは、形式名によって識別されたサブファイル内のレコードで、RELATIVE KEY データ項目の値と等しい相対レコード番号を持つものです。

INDICATORS 句: データ・レコードが書き込まれるときに使用される標識を指定します。標識は、データ・レコードについての情報と、それがプログラムに入力された方法を渡すために使用できます。

INDICATORS 句について詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」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。

ID-4 は、OCCURS 文節なしで指定された基本ブール・データ項目か、または基本データ・ブールの項目が従属しているグループ項目でなければなりません。

TERMINAL 句: TERMINAL 句に関する一般的な考慮事項については、形式 4 を参照してください。

TERMINAL 句は、どのプログラム装置のサブファイルにレコードが書き込まれるかを指定します。TERMINAL 句を指定する場合、リテラル-2 または ID-3 は、TRANSACTION ファイルに関連付けられているワークステーションを参照しなければなりません。リテラル-2 または ID-3 にブランクの値が入っている場合、TERMINAL 句は、それが指定されなかった場合と同じように扱われます。 TERMINAL 句で指定するワークステーションは、明示的または暗黙に獲得されているものでなければなりません。

TERMINAL 句が省略されている場合、使用されるサブファイルは、デフォルトのプログラム装置に関連付けられているサブファイルです。

INVALID KEY 句: INVALID KEY 条件が発生するのは、サブファイルの中にそのレコード番号のレコードがすでに存在している場合、または指定された相対レコード番号が、許容される最大のサブファイル・レコード番号より大きい場合です。該当する USE プロシージャーが指定されていないすべてのファイルについて、WRITE SUBFILE ステートメントの中で INVALID KEY 句を指定しなければなりません。

NOT INVALID KEY 句: この句では、使用されるステートメントについて無効キー条件が存在しない場合に実行されるプロシージャーを指定できます。

END-WRITE 句: この明示範囲終了符号は、WRITE ステートメントの有効範囲を区切る働きをします。END-WRITE 句を使用することによって、WRITE 条件ステートメントを別の条件ステートメントにネストすることができます。また、END-WRITE 句は、WRITE 命令ステートメントで使用することもできます。

詳細については 7-29 ページの『範囲区切りステートメント』を参照してください。

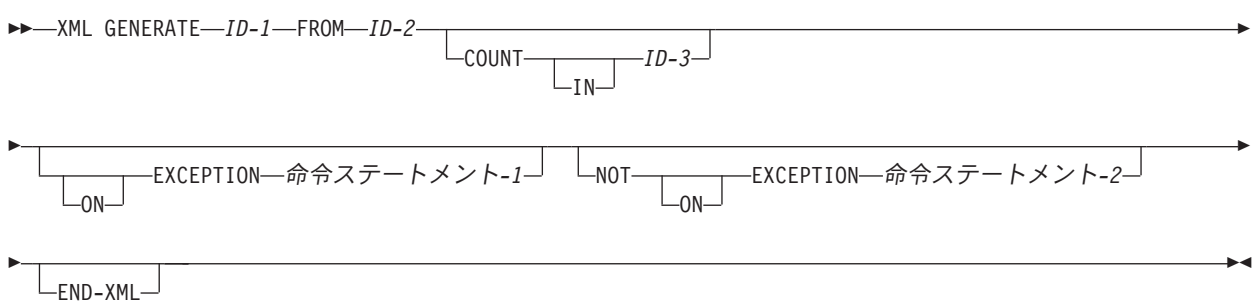
End of IBM Extension

XML GENERATE ステートメント

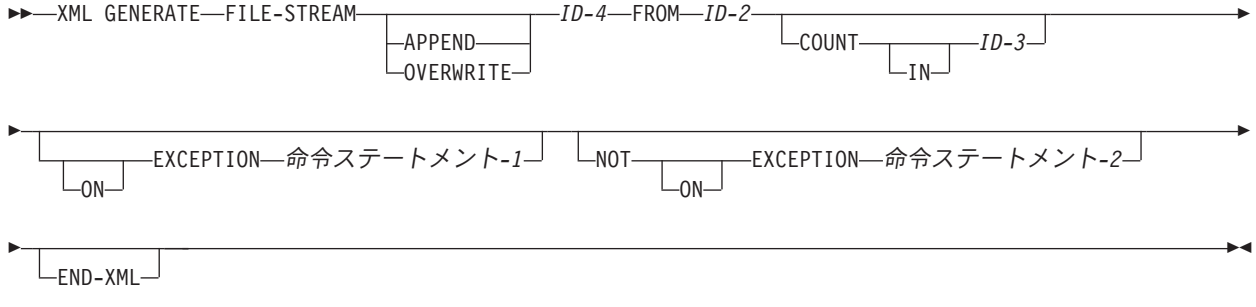
IBM Extension

XML GENERATE ステートメントは、データを XML 形式に変換します。

形式 1



形式 2



ID-1 生成された XML 文書の受信領域。*ID-1* は、以下のいずれかを参照する必要があります。

- カテゴリー英数字の基本データ項目
- 英数字グループ項目

- カテゴリー国別の基本データ項目

ID--1 が英数字グループ項目を参照する場合、*ID-1* は、カテゴリー英数字の基本データ項目であるかのように取り扱われます。

ID-1 は、JUSTIFIED 文節で記述されてはなりません。また、関数 ID であってはなりません。*ID-1* は、添え字を付けたり、参照変更できます。

ID-1 は、*ID-2* または *ID-3* とオーバーラップしないようにしてください。

ID-1 がカテゴリー英数字のデータ項目を参照する場合、生成された XML 文書は、PROCESS ステートメント CCSID オプション d で指定される CCSID でエンコードされます (ソース・コードのコンパイル時に有効な 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。

ID-1 がカテゴリー国別のデータ項目を参照する場合、生成された XML 文書は、UCS-2 でエンコードされます。PROCESS ステートメントの CCSID オプション d で国別 CCSID が指定されている場合、その CCSID が使用されます。それ以外の場合には、NTLCCSID PROCESS オプションで指定されている CCSID が使用されます。 バイト・オーダー・マークは生成されません。

ID-1 は、生成された XML が以下の *ID-2* からのデータを含む場合、カテゴリー国別のデータ項目を参照する必要があります。

- クラス国別またはクラス DBCS のすべてのデータ項目
- DBCS 名を持つすべてのデータ項目 (つまり、名前に DBCS 文字が含まれるデータ項目)

ID-1 は、生成された XML 文書を十分に含むことができる大きさである必要があります。通常、これは *ID-2* のサイズの 5 倍から 8 倍である必要があり、*identifier-2* 内の 1 つまたは複数のデータ名の長さによって異なります。 *ID-1* の大きさが十分でない場合、XML GENERATE ステートメントの終了時にエラー状態が発生します。

ID-2 XML 形式に変換されるグループまたは基本データ項目。

ID-2 は関数 ID にしたり参照変更にしたりすることはできませんが、添え字付きであってもかまいません。

ID-2 は、*ID-1* または *ID-3* とオーバーラップしてはいけません。

ID-2 は、RENAMES 節を指定してはいけません。

ID-2 で指定される以下のデータ項目は、XML GENERATE ステートメントによって無視されます。

- 名前が指定されていないすべての基本データ項目または基本 FILLER データ項目
- SYNCHRONIZED 項目用に挿入されるすべての遊びバイト
- REDEFINES 節で記述されるか、そのような再定義項目に従属する *ID-2* に従属するすべてのデータ項目
- RENAMES 節で記述される *ID-2* に従属するすべてのデータ項目
- すべての従属データ項目が無視されるすべてのグループ・データ項目

上記の規則に応じて無視されない *ID-2* によって指定されるすべてのデータ項目は、以下の条件を満たす必要があります。

- 各基本データ項目は、英字クラス、英数字クラス、数字クラス、または国別クラスのいずれかを持つか、指標データ項目である必要があります。 (つまり、基本データ項目は USAGE POINTER または USAGE PROCEDURE-POINTER 句を使用して記述することはできません。)
- そのような基本データ項目が少なくとも 1 つ存在する必要があります。

- FILLER 以外の各データ名は、隣接した上位のグループ・データ項目内で固有である必要があります。
- Unicode に変換される場合、すべての DBCS データ名は、XML 仕様、バージョン 1.0 で許可されている名前である必要があります。

例えば、次のようなデータ宣言を想定します。

```
01 STRUCT.
  02 STAT PIC X(4).
  02 IN-AREA PIC X(100).
  02 OK-AREA REDEFINES IN-AREA.
    03 FLAGS PIC X.
    03 PIC X(3).
    03 COUNTER USAGE COMP PIC S9(9).
    03 ASFNPTR REDEFINES COUNTER USAGE PROCEDURE-POINTER.
    03 UNREFERENCED PIC X(92).
  02 NG-AREA1 REDEFINES IN-AREA.
    03 FLAGS PIC X.
    03 PIC X(3).
    03 PTR USAGE POINTER.
    03 ASNUM REDEFINES PTR USAGE COMP PIC S9(9).
    03 PIC X(92).
  02 NG-AREA2 REDEFINES IN-AREA.
    03 FN-CODE PIC X.
    03 UNREFERENCED PIC X(3).
    03 QTYONHAND USAGE BINARY PIC 9(5).
    03 DESC USAGE NATIONAL PIC N(40).
    03 UNREFERENCED PIC X(12).
```

以下のデータ項目を *ID-2* として指定できます。

- STRUCT。従属データ項目 STAT および IN-AREA は XML 形式に変換されます。 (OK-AREA、NG-AREA1、および NG-AREA2 は、REDEFINES 節を指定するため無視されます。)
- OK-AREA。従属データ項目 FLAGS、COUNTER、および UNREFERENCED が変換されます。(03 PIC X(3) を指定するデータ記述項目を持つ項目は、基本 FILLER データ項目であるため、無視されます。ASFNPTR は、REDEFINES 節を指定するため、無視されます。)
- 以下を除く、STRUCT に従属するすべての基本データ項目。
  - ASFNPTR または PTR (許可されない使用法)
  - UNREFERENCED OF NG-AREA2 (非固有であるため適格ではないデータ項目名)
  - すべての FILLER データ項目

以下のデータ項目は *ID-2* として指定できません。

- NG-AREA1。従属データ項目 PTR は、USAGE POINTER を指定しますが、 REDEFINES 節を指定しないため。 (PTR は、REDEFINES 節を指定した場合、無視されます。)
- NG-AREA2。従属基本データ項目が非固有名 UNREFERENCED を持つため。

COUNT IN

COUNT IN 句が指定された場合、*ID-3* には、生成された XML 文字位置のカウント (XML GENERATE ステートメントの実行後) が含まれます。 *ID-1* (受信側) がカテゴリー国別を持つ場合、 そのカウントは国別文字位置 (UCS-2 文字エンコード単位) でのものです。それ以外の場合、カウントは、バイト単位です。

ID-3 データ・カウント・フィールド。ピクチャー・ストリングに記号 P を指定せずに定義した整数データ項目でなければなりません。

ID-3 は、*ID-1* または *ID-2* とオーバーラップしてはいけません。

FILE-STREAM 句

FILE-STREAM 句が指定される場合、変換された XML データは、ID-4 で指定される IFS ファイルに保存されます。以下のように、CCSID を使用して、XML ファイルはエンコードされます。

- Unicode UCS-2。生成された XML が (「ID-1」で説明したように) 以下に対する ID-2 からのデータを含む場合。
  - すべての国別データ項目または DBCS データ項目
  - DBCS 名を持つすべてのデータ項目
- これ以外の場合、PROCESS ステートメント CCSID オプション d で指定された CCSID (XML GENERATE 1 バイト・データ出力 CCSID、デフォルトは JOBRUN)。使用される CCSID は、7-269 ページの『XML 文書のためのコード化文字セット』にリストされている 1 バイト文字セット CCSID のいずれかである必要があります。

APPEND 句も OVERWRITE 句も使用されない場合、XML ファイル・エンコード CCSID を使用して新規ファイルが作成され、変換された XML データはそのファイルに保管されます。プログラムの実行中に同じ名前を持つファイルが存在する場合、XML 生成は停止し、特殊レジスター XML-CODE にはこのエラーを表す例外コードが含まれます。

APPEND 句が使用されるとき、ファイルが XML ファイル・エンコード CCSID を持つ場合には、変換された XML データは既存のファイルに追加されます。これ以外の場合、XML 生成は停止し、特殊レジスター XML-CODE にはこのエラーを表す例外コードが含まれます。

ただし、PROCESS オプション XMLGEN(KEEPFILEOPEN) が指定されていて、IFS ファイルが現在開かれている場合、APPEND および OVERWRITE 句なしで XML GENERATE を指定することで、IFS ファイルを閉じることができます (エラーおよび例外コードは発行されません)。

OVERWRITE 句が使用される場合、XML ファイル・エンコード CCSID を使用して既存のファイルは新規ファイルで置換されます。変換された XML データは新規ファイルに保管されます。

上述したエラーを除く他のすべてのファイル操作エラーは、ファイル操作エラー・メッセージを含むランタイム照会メッセージをトリガーします。操作を継続するために "G" で応答された場合は、特殊レジスター XML-CODE に例外コードが設定されます。

ID-4 IFS ファイル名であり、英字または英数字データ項目でなければなりません。変換された XML コンテントを保持する IFS ファイルのパス名を含みます。

ON EXCEPTION

XML 文書の生成時にエラーが発生する場合は、例外条件が存在します。例えば、*ID-1* が生成された XML 文書を含むのに十分大きくない場合などです。 この場合、XML 生成は停止し、受信側 *ID-1* のコンテントは未定義です。COUNT IN 句が指定された場合、*ID-3* には、0 から *ID-1* の長さの範囲を取ることができる、生成された文字位置の数が含まれます。

ON EXCEPTION 句が指定されている場合には、制御が *命令ステートメント-1* に転送されます。ON EXCEPTION 句が指定されていない場合、NOT ON EXCEPTION 句は、存在したとしても無視され、制御は、XML GENERATE ステートメントの終わりに転送されます。特殊レジスター XML-CODE には、例外コードが含まれます (「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の説明を参照)。

NOT ON EXCEPTION

XML 文書の生成時に例外条件が発生しない場合、指定されていれば、制御は *命令ステートメント-2* に渡され、指定されていない場合には、XML GENERATE ステートメントの最後に渡されます。ON EXCEPTION 句は、指定されていても無視されます。特殊レジスター XML-CODE は、XML GENERATE ステートメントの実行後、ゼロを含みます。

END-XML 句

この明示範囲終了符号は、XML GENERATE または XML PARSE ステートメントの有効範囲を区切ります。 END-XML では、条件 XML GENERATE または XML PARSE ステートメント (つまり、ON EXCEPTION または NOT ON EXCEPTION 句を指定する XML GENERATE または XML PARSE ステートメント) を別の条件ステートメント内でネストすることができます。

条件 XML GENERATE または XML PARSE ステートメントの有効範囲を以下を使用して終了することができます。

- 同じネスト・レベルにある END-XML 句
- 分離文字ピリオド

END-XML は、ON EXCEPTION または NOT ON EXCEPTION 句のいずれも指定していない XML GENERATE または XML PARSE ステートメントで使用することも可能です。

ネストされた XML GENERATE または XML PARSE ステートメント

指定された XML GENERATE または XML PARSE ステートメントが *命令ステートメント-1* または *命令ステートメント-2* として表示された場合、または別の XML GENERATE または XML PARSE ステートメントの *命令ステートメント-1* または*命令ステートメント-2* の一部として表示された場合、その指定された XML GENERATE または XML PARSE ステートメントは、*ネストされた* XML GENERATE または XML PARSE ステートメントです。

ネストされた XML GENERATE または XML PARSE ステートメントは、マッチングされた XML GENERATE および END-XML、または左から右に進む XML PARSE および END-XML の組み合わせとみなされます。 したがって、検出される任意の END-XML 句は、暗黙的にまたは明示的に終了していない最も近い先行する XML GENERATE または XML PARSE ステートメントにマッチングしています。

XML GENERATE の操作

ID-2 内の適格な各基本データ項目のコンテントは、7-264 ページの『基本データのフォーマット変換』 および 7-265 ページの『生成された XML データのトリミング』 で説明されているような文字フォーマットに変換されます。各ストレージ域の最初の定義のみが処理されます。データ項目の再定義は含まれません。RENAMES 節で効果的に定義されたデータ項目も含まれません。

次に、変換されたコンテントは、XML マークアップで要素文字コンテントとして挿入されます。XML 要素名は、7-265 ページの『XML 要素名情報』で説明されているように *ID-2* 内のデータ名から取得されます。選択された基本項目を含むグループ項目の名前は親要素として保存されます。生成された XML をより読みやすくするために、余分の空白 (改行、インデントなど) は挿入されません。XML 宣言は生成されません。

ID-1 で指定された受信領域が、作成された XML 文書を含むのに十分大きくない場合、エラー条件が存在します。詳細は、上述の ON EXCEPTION 句の説明を参照してください。

ID-1 が生成された XML 文書よりも長い場合、XML が生成される部分の *ID-1* のみが変更されます。*ID-1* の残りの部分には、この XML GENERATE ステートメントの実行前に入っていたデータがそのまま残ります。このデータを参照するのを避けるためには、*ID-1* を XML GENERATE ステートメントの前のスペースに初期設定するか、COUNT IN 句を指定します。

COUNT IN 句が指定された場合、*ID-3* には (XML GENERATE ステートメントの実行後)、生成された文字位置の総数 (UCS-2 エンコード単位またはバイト単位) が含まれます。 *ID-3* を参照変更長さフィールドとして使用し、生成された XML 文書を含む部分の *ID-2* を参照することができます。

XML GENERATE ステートメントの実行後、特殊レジスター XML-CODE は、正常な完了を示す 0 または、ゼロ以外の例外コードのいずれかを含みます。(詳しくは、「*ILE COBOL プログラマーの手引き*」も参照してください)。

XML PARSE ステートメントは、特殊レジスター XML-CODE も使用します。したがって、 XML PARSE ステートメントの処理手順で XML GENERATE ステートメントをコード化する場合は、その XML GENERATE ステートメントを実行する前に XML-CODE の値を保管し、XML GENERATE ステートメントの終了後に保管した値を復元します。

基本データのフォーマット変換: 基本データ項目は、以下のようにデータ項目のタイプに応じて文字フォーマットに変換されます。

- カテゴリー英字、英数字、英数字編集、DBCS、外部浮動小数点、国別、および数値編集のデータ項目は変換されません。
- NOSTDTRUNC コンパイラー・オプションを使用してコンパイルされた COMP-5 データ項目またはバイナリー・データ項目以外の固定小数点数値データ項目は、以下を持つ数値編集項目に移動されたかのように変換されます。
  - 少なくとも 1 つの整数位置を持つ、数値項目が持つ整数位置と同数の整数位置
  - 数値項目が少なくとも 1 つの小数点以下の桁数を持つ場合の明示的小数点
  - 数値項目が持つ小数点以下の桁数と同数の小数点以下の桁数
  - データ項目が符号付きの場合、先導する '-' ピクチャー記号 (PICTURE 節に S を持つ)
- NOSTDTRUNC コンパイラー・オプションを使用してコンパイルされた COMP-5 データ項目およびバイナリー・データ項目は、整数位置の数を除いて、他の固定小数点数値項目と同じように変換されます。整数位置の数は、以下のようにピクチャー文字ストリングの '9' 記号の数に応じて計算されます。
  - データ項目が 1 から 4 の '9' ピクチャー記号を持っている場合、5 から小数点以下の桁数を引く
  - データ項目が 5 から 9 の '9' ピクチャー記号を持っている場合、10 から小数点以下の桁数を引く
  - データ項目が 10 から 18 の '9' ピクチャー記号を持っている場合、20 から小数点以下の桁数を引く
- 内部浮動小数点データ項目は、以下のデータ項目に移動されたかのように変換されます。
  - COMP-1 の場合: PICTURE -9.9(8)E+99 を持つ外部浮動小数点データ項目
  - COMP-2 の場合: PICTURE -9.9(17)E+99 を持つ外部浮動小数点データ項目 (桁位置数のため適格ではない)
- 指標データ項目は、USAGE BINARY PICTURE S9(9) と宣言されているかのように変換されます。

7-265 ページの『生成された XML データのトリミング』 で説明されているように、文字フォーマットに変換後、前後のスペースおよび先行ゼロは除去されます。

変換後、データ項目が、XML コンテントで許可されていない文字を含む場合、関連する XML 仕様で指定したように、元のデータ値 (つまり、変換またはトリミングの前のデータ項目中の値) は 16 進数で表され、接頭部 'hex' を持つ要素タグ名は、通常のタグ名で置換されます。 例えば、データ項目 Customer-Name が実行時に LOW-VALUES を含んでいることが分かった場合、通常の 'Customer-Name' の代わりに XML 要素タグ名 'hex.Customer-Name' が使用され、コンテントはゼロ数字のペアのストリングとして表されます。

5 つの文字 & (アンパーサンド)、' (アポストロフィ)、> (より大きい)、< (より小さい)、および " (引用符) の残ったインスタンスは、同等の XML 参照 '&'、'''、'>'、 '<'、および '"' にそれぞれ変換されます。

したがって、*ID-1* が、カテゴリー国別のデータ項目である場合、 すべての国別以外の値は国別フォーマットに変換されます。

生成された XML データのトリミング: データ値へのトリミングは、文字フォーマットに変換後に行われます。 (変換は、 7-264 ページの『基本データのフォーマット変換』の下に説明されます。)

符号付き数値から変換される値の場合、値が正のとき、先行スペースは除去されます。

数値項目から変換される値の場合、実際のまたは暗黙の小数点のすぐ左の桁より前にある先行ゼロ (開始負符号 (-) の後) は除去されます。小数点より右にある後続ゼロは保持されます。例えば、次のとおりです。

- -012.340 は、-12.340 になります。
- 0000.45 は、0.45 になります。
- 0013 は、13 になります。
- 0000 は、0 になります。

英字クラス、英数字クラス、DBCS クラス、および国別クラスのデータ項目からの文字値は、対応するデータ項目が左 (デフォルト) か右に位置調整されているかに応じて、後続または先行スペースが除去されます。 つまり、JUSTIFIED 節を指定しない対応するデータ項目を持つ値の場合、後続スペースが除去されます。JUSTIFIED 節を指定するデータ項目を持つ値の場合、先行スペースが除去されます。文字値がスペースのみから構成される場合、トリミングの完了後、スペースが 1 つ値として残ります。

XML 要素名情報

ID-2 から生成される XML 文書では、XML 要素タグ名は、*ID-2* で指定されるデータ項目の名前、および *ID-2* に従属する適格なデータ名から、以下のように取得されます。

- データ記述項目からのデータ名の正確な大/小文字混合スペルが保持されます。そのデータ項目への任意の参照からのスペル (例えば、OCCURS DEPENDING ON 節) は使用されません。
- 数字で開始するデータ名は、アンダースコアーで接頭部が付けられます。例えば、データ名 '3D' は、XML タグ名 '_3D' になります。
- 大文字および小文字の任意の組み合わせで、文字 'xml' で開始するデータ名は、アンダースコアーで接頭部が付けられます。例えば、データ名 'Xml' は、XML タグ名 '_Xml' になります。
- XML バージョン 1.0 コンテントでは許可されていない文字を含むことが実行時に検出されたデータ項目の名前は、'hex' で接頭部が付けられ、コンテント自身は 16 進数で表示されます。

Unicode に変換されるときに、DBCS データ名は、XML 仕様、バージョン 1.0 で許可されている名前である必要があります。

XML GENERATE ステートメントの実行後に特殊レジスター XML-CODE が含むことができる例外コードの説明については、「*ILE COBOL プログラミング・ガイド*」を参照してください。

End of IBM Extension

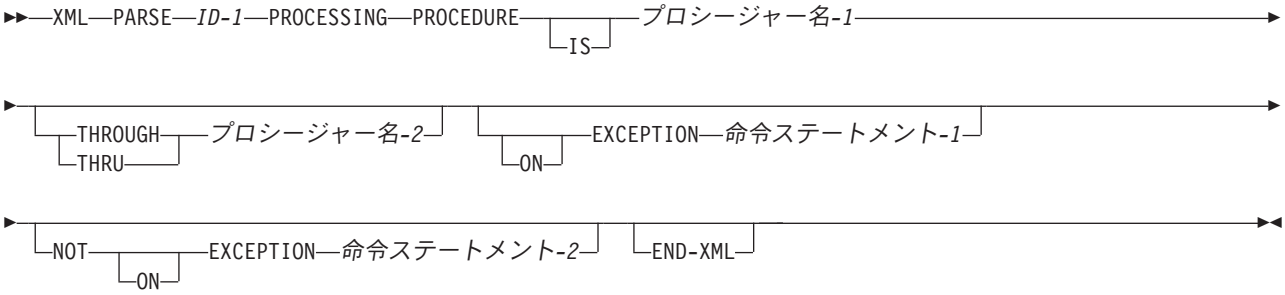
XML PARSE ステートメント

IBM Extension

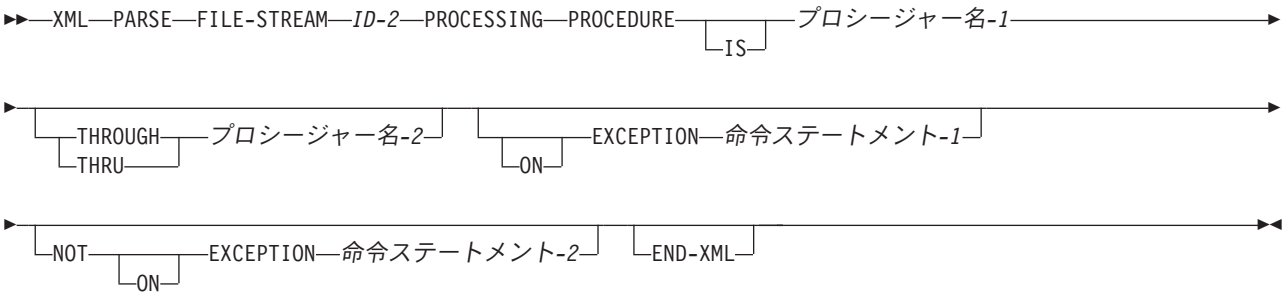
XML PARSE ステートメントは、COBOL 実行時の一部である高速 XML パーサーに対する ILE COBOL 言語インターフェースです。XML PARSE ステートメントは、XML 文書を構文解析し、1 つずつユーザー作成の処理プロシージャーに渡します。

XML PARSE ステートメント

XML 解析ステートメント - 形式 1



XML 解析ステートメント - 形式 2



ID-1 XML 文書文字ストリームを含む英数字または国別データ項目でなければなりません。 ID-1 は関数 ID とすることはできません。

ID-1 が英数字の場合、その内容は、 7-269 ページの『XML 文書のためのコード化文字セット』でリストされている、1 バイト文字セットの 1 つを使用してエンコードされなければなりません。エンコード宣言を含まない EBCDIC XML 文書は、ソース・メンバーのコード化文字セットで解析されます。

ID-1 が国別の場合、その内容は、国別 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 を使用してエンコードされなければなりません。

ID-2 ID-2 は、XML 文書を含むストリーム・ファイルの絶対または相対パス名を含む英数字データ項目でなければなりません。 絶対名は「/」で始まり、「/u/user1/myxml」のようになります。 相対パス名は「/」では始まらないので、現行ディレクトリーと連結されます。エンコード宣言を含まない指定されたストリーム・ファイルに配置された XML 文書 (ASCII XML 文書を含む) は、ストリーム・ファイルのコード化文字セットで解析されます。

PROCESSING PROCEDURE 句
プロシージャーの名前を指定し、XML パーサーが生成するさまざまなイベントを処理します。

プロシージャー名-1
処理プロシージャー内の、最初または唯一の、セクションまたは段落を指定します。

プロシージャー名-2
処理プロシージャー内の、最後のセクションまたは段落を指定します。

処理プロシージャーは、XML イベントが処理されるステートメントで構成されます。処理プロシージャーの中には、処理プロシージャー内の、CALL、EXIT、GO TO、GOBACK、および PERFORM ステートメントによって実行されるすべてのステートメントも含まれます。

処理プロシージャーは、XML PARSE ステートメントを*直接* 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制御を最外部プログラムに渡す場合、ターゲット・プログラムは、同じまたは異なる XML PARSE ステートメントを実行できます。複数のスレッドで実行するプログラムは、同じ XML ステートメントまたは異なる XML ステートメントを同時に実行できます。

コンパイラーは、処理プロシージャー内の最後のステートメントの後ろに、戻りメカニズムを挿入します。処理プロシージャーは、STOP RUN ステートメントによって実行単位を終了できます。GOBACK または EXIT PROGRAM ステートメントによってパーサーに戻ろうとしてはなりません。

処理プロシージャーについて詳しくは、7-268 ページの『制御フロー』および 7-269 ページの『処理プロシージャー』を参照してください。

ON EXCEPTION

ON EXCEPTION 句は、XML PARSE ステートメントが例外条件を発生させた場合に実行される命令ステートメントを指定します。

例外条件は、XML パーサーが、XML 文書の処理でエラーを検出した場合に発生します。まずパーサーは、制御を処理プロシージャーに渡し、特殊レジスター XML-EVENT が「EXCEPTION」を含むように設定して、例外 XML イベントをシグナル通知します。パーサーは、特殊レジスター XML-CODE 内に数値エラー・コードを規定します。詳しくは、「*ILE COBOL プログラマーの手引き*」で説明しています。

例外条件は、通常の XML イベントからパーサーに戻る前に XML-CODE を -1 に設定することにより、処理プロシージャーが解析を故意に終了した場合にも発生します。 このケースでは、パーサーは、EXCEPTION XML イベントをシグナル通知しません。

ON EXCEPTION 句が指定されている場合、パーサーは制御を命令ステートメント-1 に転送します。ON EXCEPTION が指定されていない場合、NOT ON EXCEPTION 句は、あるとしても無視され、制御は、XML PARSE ステートメントの終わりに転送されます。

XML 処理プロシージャーが、例外 XML イベントを処理し、制御がパーサーに戻る前に XML-CODE をゼロに設定した場合、例外条件はもはや存在しません。処理されていない他の例外が、パーサーの終了前に発生した場合、制御は、指定されていれば、NOT ON EXCEPTION 句の命令ステートメント-2 に転送されます。

NOT ON EXCEPTION

NOT ON EXCEPTION 句は、XML PARSE 処理の終了時に例外条件が存在しない場合に実行される命令ステートメントを指定します。

例外条件が XML PARSE 処理の終了時に存在しない場合、制御は、指定されていれば、NOT ON EXCEPTION 句の命令ステートメント-2 に転送されます。NOT ON EXCEPTION 句が指定されていない場合は、制御は XML PARSE ステートメントの終わりに転送されます。 ON EXCEPTION 句は、指定されていても無視されます。

特殊レジスター XML-CODE は、XML PARSE ステートメントの実行後、ゼロを含みます。

END-XML 句

この明示範囲終了符号は、XML PARSE ステートメントの範囲を区切る働きをします。END-XML 句を使用することによって、XML PARSE 条件ステートメントを別の条件ステートメントにネストすることができます。 END-XML は、ON EXCEPTION または NOT ON EXCEPTION 句のいずれも指定していない XML PARSE ステートメントで使用されることも可能です。

制御フロー

XML パーサーが XML PARSE ステートメントから制御を受け取ると、パーサーは XML 文書を解析し、処理中の以下のポイントで、プロシージャー名-1 に制御を転送します。

- 解析処理の開始
- 文書のフラグメントが検出された時点
- パーサーが XML 文書の解析中にエラーを検出した時点
- XML 文書の処理の終了

制御は、処理プロシージャーが終わりに達した時点で、XML パーサーに戻ります。

パーサーと処理プロシージャー間での制御の交換は、以下のいずれかになるまで続きます。

- XML 文書全体が解析され、END-OF-DOCUMENT イベントで終了した。
- パーサーが例外を検出し、パーサーに戻る前に、処理プロシージャーが特殊レジスター XML-CODE をゼロにリセットしなかった。
- パーサーに戻る前に XML-CODE を -1 に設定して、処理プロシージャーが故意に解析を終了した。

その後、パーサーは終了して、パーサーまたは処理プロシージャーが設定した最も新しい値を含む XML-CODE 特殊レジスターによって、制御を XML PARSE ステートメントに戻します。

処理プロシージャーに渡される各 XML イベントについては、XML-CODE、 XML-EVENT、および XML-TEXT または XML-NTEXT 特殊レジスターが、個々のイベントに関する情報を含んでいます。XML-CODE 特殊レジスターの内容は、XML PARSE ステートメントの実行中および実行後に定義されます。他のすべての XML 特殊レジスターの内容は、処理プロシージャーの範囲外では未定義です。

通常のイベントに対して、処理プロシージャーが制御を受け取った場合、特殊レジスター XML-CODE はゼロを含んでいます。 EXCEPTION イベントに対しては、XML-CODE は、「*ILE COBOL プログラマーの手引き*」で指定されている XML 例外コードのうち 1 つを含んでいます。特殊レジスター XML-EVENT は、「START-OF-DOCUMENT」などのイベント名に設定されます。 XML-TEXT または XML-NTEXT のいずれかが、『XML-EVENT 特殊レジスター』で説明するように、イベントに対応する文書の部分を含んでいます。

XML 特殊レジスターに関して詳しくは、『特殊レジスター』を参照してください。

すべての種類の XML イベントに対しては、処理プロシージャーが制御をパーサーに戻したとき、XML-CODE がゼロでない場合、それ以上の EXCEPTION イベントなしにパーサーは終了します。処理プロシージャーからパーサーに戻る前に、EXCEPTION でないイベントに対して XML-CODE を -1 に設定すると、ユーザーが開始した例外条件でパーサーは強制的に終了します。いくつかの EXCEPTION イベントに対しては、処理プロシージャーは XML-CODE をゼロに設定し、強制的にパーサーを続行させることができますが、以降の結果は予測不能です。 XML-CODE がゼロの場合、XML 文書全体が解析されてしまうか未処理の例外条件が発生するまで、解析は続行します。

解析中、XML PARSE ステートメントを指定したプログラムが、繰り返し呼び出されてはなりません。

XML 特殊レジスターはネストされたプログラムで参照できますが、XML PARSE ステートメントは、ネストされたプログラムで指定してはなりません。

EXCEPTION イベントおよび例外処理に関して詳しくは、「*ILE COBOL プログラマーの手引き*」を参照してください。

処理プロシージャー

処理プロシージャーのコーディングする場合、以下に十分注意してください。

- XML 処理プロシージャーは、EXIT PROGRAM または GOBACK ステートメントを含んではなりません。
- 処理プロシージャーの ALTER、GO TO、および PERFORM ステートメントを使用して、処理プロシージャー外部のプロシージャー名に、制御を転送できます。ただし、制御は、GO TO または PERFORM ステートメントの後、処理プロシージャーに戻らなければなりません。
- 処理プロシージャーは、CALL ステートメントを含むことができます。ターゲット・プログラムは、XML PARSE ステートメントを含むことができます。

「*ILE COBOL プログラマーの手引き*」には、XML PARSE ステートメントおよび処理プロシージャーの使用に関する詳細があります。

XML 文書のためのコード化文字セット

XML PARSE は、国別データ項目、英数字データ項目、および UCS-2 および 1 バイト CCSID を持つ IFS ファイルの XML 文書をサポートしています。国別データ項目の文書は、国別 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 Unicode UCS-2 CCSID を使用してエンコードされなければなりません。 英数字データ項目の文書は、XML 文書のための「XML 文書でサポートされている EBCDIC CCSID」 (表 7-16) で示されている単一バイトの EBCDIC CCSID で明示的にサポートされているものの 1 つ、または「XML 文書でサポートされている ASCII CCSID」(表 7-17) で示されている ASCII CCSID のうち 1 つを使用して、エンコードされなければなりません。

表 7-16. **XML 文書でサポートされている EBCDIC CCSID**

CCSID	内容
1140、37	米国、カナダなど。ユーロ国別拡張 CCSID (ECECP)、国別拡張 CCSID
1141、273	オーストリア、ドイツ ECECP、CECP
1142、277	デンマーク、ノルウェー ECECP、CECP
1143、278	フィンランド、スウェーデン ECECP 、CECP
1144、280	イタリア ECECP、CECP
1145、284	スペイン、ラテンアメリカ (スペイン語) ECECP、CECP
1146、285	英国 ECECP、CECP
1147、297	フランス ECECP、CECP
1148、500	国際 ECECP、CECP
1149、871	アイスランド ECECP、CECP

表 7-17. **XML 文書でサポートされている ASCII CCSID**

CCSID	内容
813	ISO 8859-7 ギリシャ語/Latin
819	ISO 8859-1 Latin 1/オープン・システム
920	ISO 8859-9 Latin 5 (ECMA-128、トルコ TS-5881)

ASCII XML 文書を解析する場合、特殊レジスター XML-TEXT にある処理プロシージャーに渡される文書のフラグメントは、ASCII でエンコードされます。移動や比較のような ILE COBOL 操作は、適切な操作に対する EBCDIC エンコードまたは国別文字に依拠しているので、文書フラグメントを変換してから、それらを使用しなければなりません。 XML 文書が COBOL プログラムにあるときにこれを行うには、

MOVE ステートメントを使用して、まず XML 文書の ASCII CCSID から国別文字に変換します。 次に、必要な場合、MOVE ステートメントを使用して、その結果を国別文字から EBCDIC に変換します。

他の CCSID でエンコードされた COBOL プログラムの XML 文書については、MOVE ステートメントを使用して、それらを国別文字に変換することで、解析できます。 特殊レジスター XML-NTEXT にある処理プロシージャーに渡された文書テキストの個々の部分については、その後、必要に応じて、MOVE ステートメントを使用して元の CCSID にもう一度変換できます。

XML 文書が IFS ファイルにある場合、オブジェクト・コピー (CPY) コマンドを使用して、CCSID 変換を実行します。パーサーから戻される文書フラグメントでの作業をより簡単にするには、XML PARSE で文書を使用する前に以下を実行することをお勧めします。

1. 各 xml レコードの開始で '<' タグに先行する文字を除去する必要があります。
2. IFS ファイルでの各行の終了は、LF (改行) ではなく、CR (復帰) のみを持つ必要があります。
3. XML 文書を国別 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 に変換するか、または XML 文書をジョブの CCSID に変換します。
4. XML 文書のエンコード宣言を手動で変更し、文書の実際の CCSID を指定します。

文書のエンコードの指定およびパーサーがエンコードを決定する方法について詳しくは、「*ILE COBOL プログラマーの手引き*」を参照してください。

特殊レジスター

- XML-CODE
- XML-EVENT
- XML-NTEXT
- XML-TEXT

XML-CODE 特殊レジスター: XML-CODE 特殊レジスターは、以下の目的に使用されます。

- XML パーサーと、XML PARSE ステートメントで識別された処理プロシージャー間で、状況を通信するため
- XML GENERATE ステートメントが正常に実行されたか、XML 実行時に例外が発生したかを示すため

XML パーサーが XML-CODE を設定してから、パーサーの終了時に、各イベントごとに制御が処理プロシージャーに転送されます。 XML-CODE をリセットしてから、制御を処理プロシージャーから XML パーサーへ戻すことができます。

XML-CODE 特殊レジスターには、次の暗黙定義があります。

```
01 XML-CODE PICTURE S9(9) USAGE BINARY VALUE 0.
```

ネストされたプログラムの中で使用される際には、この特殊レジスターは、最外部のプログラム内で、グローバル属性によって暗黙に定義されます。

XML パーサーは XML イベントを検出すると、XML-CODE を設定してから制御を処理プロシージャーに渡します。 EXCEPTION イベントを除くすべてのイベントに対して、処理プロシージャーが制御を受け取った場合、XML-CODE はゼロを含んでいます。

EXCEPTION イベントに対して、パーサーは、XML-CODE を例外の性質を示す例外コードに設定します。例外コードについては、「*ILE COBOL プログラマーの手引き*」で詳しく説明しています。

以下のように、パーサーに戻る前に、XML-CODE を設定することができます。

- -1 に設定すると、通常のイベント後、パーサーが EXCEPTION イベントを発生せずに終了することを示します。
- ゼロに設定すると、継続が許可されている EXCEPTION イベント後、パーサーが処理を続けることを示します。 パーサーは XML 文書の処理を続けようとしますが、結果は未定義です。

XML-CODE をその他の値に設定してからパーサーに戻る場合、結果は未定義です。

パーサーが制御を XML PARSE ステートメントに戻すと、XML-CODE は、パーサーまたは処理プロシージャーのいずれかによって設定された最も新しい値を含みます。

XML GENERATE ステートメントの終了時に、XML-CODE には、XML 生成の正常完了を示す 0、または XML 生成時に例外が発生したことを示すゼロ以外のエラー・コードが含まれます。XML GENERATE 例外コードは、「*ILE COBOL プログラマーズ・ガイド*」で説明されています。

XML-EVENT 特殊レジスター: XML-EVENT 特殊レジスターは、XML パーサーから XML PARSE ステートメントで識別された処理プロシージャーへ、イベント情報を通信するために使用されます。制御が処理プロシージャーへ渡される前に、XML パーサーは、XML-EVENT 特殊レジスターを、 表 7-18 にあるような XML イベントの名前に設定します。

XML-CODE 特殊レジスターには、次の暗黙定義があります。

```
01 XML-EVENT USAGE DISPLAY PICTURE X(30) VALUE SPACE.
```

ネストされたプログラムの中で使用される際には、この特殊レジスターは、最外部のプログラム内で、グローバル属性によって暗黙に定義されます。

XML-EVENT を、受信データ項目として使用することはできません。

表 7-18. XML-EVENT および XML-TEXT または XML-NTEXT 特殊レジスターの内容

XML イベント (XML-EVENT の内容)	**XML-TEXT または XML-NTEXT の内容**
ATTRIBUTE-CHARACTER	属性値にある定義済みエンティティー参照に相当する単一の文字。
ATTRIBUTE-CHARACTERS	引用符またはアポストロフィーで囲まれた値。値がエンティティー参照を含む場合、これは属性値のサブストリングとすることができます。
ATTRIBUTE-NAME	属性名、= の左側にあるストリング
ATTRIBUTE-NATIONAL-CHARACTER	XML PARSE ステートメントの ID-1 によって指定された XML 文書のタイプにかかわらず、XML-TEXT は空であり、 XML-NTEXT は、(数値) 文字参照に相当する単一の国別文字を含む。
COMMENT	開始文字シーケンス「<!—」と終了文字シーケンス「—>」の間にあるコメントのテキスト
CONTENT-CHARACTER	要素コンテントにある定義済みエンティティー参照に相当する単一の文字。
CONTENT-CHARACTERS	開始タグと終了タグの間にある要素コンテント。コンテントがエンティティー参照または別の要素を含む場合、これを要素コンテントのサブストリングとすることができます。

表 7-18. XML-EVENT および XML-TEXT または XML-NTEXT 特殊レジスターの内容 (続き)

XML イベント (XML-EVENT の内容)	XML-TEXT または XML-NTEXT の内容
CONTENT-NATIONAL-CHARACTER	XML PARSE ステートメントの ID-1 によって指定された XML 文書のタイプにかかわらず、XML-TEXT は空であり、 XML-NTEXT は、(数値) 文字参照に相当する単一の国別文字を含む。
DOCUMENT-TYPE-DECLARATION	開始文字シーケンス「<!DOCTYPE」および終了文字シーケンス「>」を含む、文書タイプ宣言全体。
ENCODING-DECLARATION	引用符またはアポストロフィーで囲まれた、 XML 宣言内のエンコード宣言の値。
END-OF-CDATA-SECTION	常時、ストリング「]]>」を含む。
END-OF-DOCUMENT	Null、長さゼロ
END-OF-ELEMENT	終了要素タグまたは空の要素タグの名前。
EXCEPTION	例外が検出されたポイントまでを含む、正常にスキャンされた文書の部分。[1]
PROCESSING-INSTRUCTION-DATA	処理命令の残り、終了シーケンス「?>」は含まないが、末尾の空白文字は含み、先頭の空白文字は含まない。
PROCESSING-INSTRUCTION-TARGET	処理命令ターゲット名、処理命令である開始シーケンス「<?」の直後にある。
STANDALONE-DECLARATION	引用符またはアポストロフィーで囲まれた、 XML 宣言内のスタンドアロン宣言の値。
START-OF-CDATA-SECTION	常時、ストリング「<![CDATA[」を含む。
START-OF-DOCUMENT	文書全体。
START-OF-ELEMENT	開始要素タグまたは空の要素タグの名前、エレメント・タイプとも言う。
UNKNOWN-REFERENCE-IN-CONTENT	エンティティー参照名で、「&」および「;」区切り文字を含まない。
UNKNOWN-REFERENCE-IN-ATTRIBUTE	エンティティー参照名で、「&」および「;」区切り文字を含まない。
VERSION-INFORMATION	引用符またはアポストロフィーで囲まれた、XML 宣言内のバージョン宣言の値。これは、現行では常時「1.0」です。

注:

1. エンコードの矛盾に関する例外がシグナル通知されてから、解析が開始します。これらの例外について、XML-TEXT は、長さゼロであるか、文書のエンコード宣言値のみを含みます。XML 例外コードについては、「*ILE COBOL プログラマーの手引き*」を参照してください。

XML-NTEXT 特殊レジスター: XML-NTEXT 特殊レジスターは、USAGE NATIONAL である文書フラグメントを含むように、XML 解析中に定義されます。

XML-NTEXT は、含まれている XML 文書フラグメントの長さの基本国別データ項目です。 XML-NTEXT の長さは、ゼロから 8,000,000 までの国別文字位置をとることができます。 最大バイト長は 16,000,000 です。

注: COBOL データ記述記入項目に相当するものはありません。

ネストされたプログラムの中で使用される際には、この特殊レジスターは、最外部のプログラム内で、グローバル属性によって暗黙に定義されます。

以下のケースでは、パーサーは、XML-NTEXT を、イベントに関連する文書フラグメントに設定してから、制御を処理プロシージャーに転送します。

- XML PARSE ステートメントのオペランドが国別データ項目の場合
- ATTRIBUTE-NATIONAL-CHARACTER イベントの場合
- CONTENT-NATIONAL-CHARACTER イベントの場合

XML-NTEXT が設定されている場合、 **XML-TEXT** 特殊レジスターの長さはゼロです。いつでも、2 つの特殊レジスター XML-NTEXT および XML-TEXT のうち 1 つだけが、ゼロ以外の長さとなります。

LENGTH 関数を使用して、XML-NTEXT が含む国別文字の数を決定してください。XML-NTEXT に対する LENGTH OF 特殊レジスターには、国別文字の数ではなく、XML-NTEXT に含まれているバイト数があります。

注: START-DOCUMENT イベントは、8,000,000 国別文字を超えることがあります。この場合、特殊レジスター XML-NTEXT には、イベントの最初の 8,000,000 文字のみが含まれ、その LENGTH は 16,000,000 バイトに設定されます。

XML-NTEXT を、受け入れ項目として使用することはできません。

XML-TEXT 特殊レジスター: XML-TEXT 特殊レジスターは、クラス英数字である文書フラグメントを含むように、XML 解析中に定義されます。

XML-TEXT は、含まれている XML 文書フラグメントの長さの基本英数字データ項目です。 XML-TEXT の長さは、ゼロから 16,000,000 バイトまでをとることができます。

注: COBOL データ記述記入項目に相当するものはありません。

ネストされたプログラムの中で使用される際には、この特殊レジスターは、最外部のプログラム内で、グローバル属性によって暗黙に定義されます。

XML PARSE ステートメントのオペランドが、 ATTRIBUTE-NATIONAL-CHARACTER イベントおよび CONTENT-NATIONAL-CHARACTER イベントを除く英数字データ項目の場合、パーサーは、XML-TEXT を、イベントに関連する文書フラグメントに設定してから、制御を処理プロシージャーに転送します。

XML-TEXT が設定されている場合、 **XML-NTEXT** 特殊レジスターの長さはゼロです。いつでも、2 つの特殊レジスター XML-NTEXT および XML-TEXT のうち 1 つだけが、ゼロ以外の長さとなります。

LENGTH 関数または XML-TEXT に対する LENGTH OF 特殊レジスターを使用して、XML-TEXT が含むバイト数を決定します。

注: START-DOCUMENT イベントは、16,000,000 文字を超えることがあります。この場合、特殊レジスター XML-TEXT には、イベントの最初の 16,000,000 文字のみが含まれ、その LENGTH は 16,000,000 バイトに設定されます。

XML-TEXT を、受け入れ項目として使用することはできません。

End of IBM Extension

組み込み関数

データ処理を行う上で、オブジェクト・プログラムに関連するデータ・ストレージにおいては必要な値が直接アクセスできず、代わりに他のデータを操作することでその値を得なければならなくなることがしばしばあります。組み込み関数は、数学演算、文字演算、論理演算を実行する関数です。したがって、組み込み関数を使うことにより、オブジェクト・プログラムの実行中に自動的に値を得るデータ項目を参照できます。

関数は、実行するサービスのタイプに基づいて、数理、統計、日付 / 時刻、財務、文字処理、および一般の 6 つのカテゴリーに分けることができます。

関数は、手続き部ステートメントで関数名を必要な引数とともに指定することで参照できます。

関数は基本データ項目であり、英数字、DBCS、数字、整数、ブール、または日時の値を戻します。関数を、受け入れオペランドとして使用することはできません。

関数定義と評価

関数のクラスおよび特性や、関数が必要とする引数の数およびタイプは、その関数の関数定義によって決まります。これらの特性には、次のものが含まれます。

- 関数によってはそのクラスおよび特性が、その関数に対する引数によって決まる
- 英数字関数の場合は、戻り値のサイズ

IBM Extension

- DBCS 関数の場合は、戻り値のサイズ
- 日時関数の場合は、戻り値の長さ
- 数字および整数関数の場合は、戻り値の符号およびその関数が整数かどうか
- その関数によって戻される実際の値

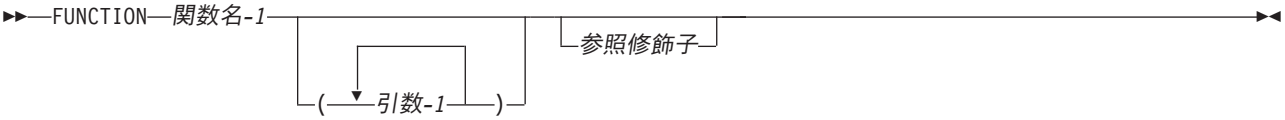
End of IBM Extension

組み込み関数の評価はどれも、それが現れるコンテキストの影響を受けません。つまり、関数の評価は、その関数の外側にある演算およびオペランドの影響を受けないということです。ただし、関数の評価が、その引数の属性の影響を受けることはあります。

関数の指定

関数 ID の一般形式は次のようになります。

フォーマット



関数名-1

関数名-1 は、組み込み関数名の 1 つでなければなりません。

引数-1

引数-1 は、ID、リテラル (表意定数以外)、または算術式でなければなりません。

参照修飾子
英数字または DBCS のカテゴリーの関数に対してだけ指定できます。

英数字ソース・ステートメントと数字ソース・ステートメントで組み込み関数を呼び出す例が、次に示されています。

以下の英数字ソース・ステートメントは、

```
MOVE FUNCTION UPPER-CASE("hello") TO DATA-NAME.
```

引数内の小文字をそれぞれに対応する大文字に置き換え、結果的に HELLO を DATA-NAME に移動します。

以下の数字ソース・ステートメントは、

```
COMPUTE NUM-ITEM = FUNCTION MEAN(A B C)
```

A、B、および C の値を加算し、次に 3 で除算して、その結果を NUM-ITEM に入れます。

手続き部ステートメント内で、各関数 ID は、それと同じ位置にある ID に関連する任意の参照変更または添え字付けが評価されるのと同時に評価されます。

関数のタイプ

関数には次の 7 つのタイプがあります。

- 英数字
- 数字

IBM Extension

- DBCS
- 国別
- 日時
- ブール

End of IBM Extension

- 整数

英数字関数のクラスおよびカテゴリーは英数字です。戻り値は DISPLAY を暗黙に使用し、標準データ・フォーマットの文字です。戻り値の文字位置の数は、関数定義によって決まります。

数字関数のクラスおよびカテゴリーは数字です。戻り値は、常に演算符号を持っているものと見なされ、数字中間結果となります。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

IBM Extension

DBCS 関数のクラスおよびカテゴリーは DBCS です。戻り値は、DISPLAY-1 を暗黙に使用します。戻り値の文字位置の数は、関数定義によって決まります。

国別関数のクラスおよびカテゴリーは NATIONAL です。戻り値は、NATIONAL を暗黙に使用します。戻り値の文字位置の数は、関数定義によって決まります。

日時関数は日時クラスの関数であり、そのカテゴリーは日付、時刻、またはタイム・スタンプです。戻り値は DISPLAY を暗黙に使用します。戻り値の文字位置の数は、関数定義によって決まります。

ブール関数のクラスおよびカテゴリーはブールです。戻り値は DISPLAY を暗黙に使用します。ブール値の真 (B"1") または偽 (B"0") のいずれかが戻されます。

End of IBM Extension

整数関数のクラスおよびカテゴリーは数字です。戻り値は、常に演算符号を持っているものと見なされ、整数の中間結果となります。戻り値の桁位置の個数は、関数定義によって決まります。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

使用上の規則

英数字関数

一般形式において ID が認められていて、かつ一般形式に関する規則が関数への参照を特に禁じていないところではどこでも、英数字関数を指定できます。ただし、次の指定はできません。

- 任意のステートメントの受け入れオペランドとして指定する場合。
- 一般形式に関する規則で参照されているデータ項目が特定の特性 (クラスおよびカテゴリー、使用法、サイズ、および暗黙的値など) を持つ必要があり、定義に従った関数の評価および指定された特定の引数が、これらの特性を持たない場合。

英数字関数の参照変更が認められています。関数に参照変更が指定されている場合には、参照変更の評価は、関数の評価の直後に行われます。

英数字関数を、英数字の引数が認められている関数に対する引数として参照できます。

数字関数

数字関数は、算術式を指定できるところでのみ使用できます。

数字関数は、数字の引数が認められている関数に対する引数として参照できます。

数字関数は、整数オペランドが必要なところでは、たとえその特定の参照が整数値を生成するとしても、使用できません。 INTEGER または INTEGER-PART 関数を使用して、数字の引数のタイプを強制的に整数にすることができます。

IBM Extension

DBCS 関数

一般形式において DBCS ID が認められていて、一般形式に関する規則が関数への参照を特に禁じていないところではどこでも、DBCS 関数を指定できます。ただし、次の指定はできません。

- 任意のステートメントの受け入れオペランドとして指定。
- 一般形式に関する規則で参照されているデータ項目が特定の特性 (クラスおよびカテゴリー、使用法、サイズ、および暗黙的値など) を持つ必要があり、定義に従った関数の評価および指定された特定の引数が、これらの特性を持たない場合。

DBCS 関数の参照変更が認められています。関数に参照変更が指定されている場合には、参照変更の評価は、関数の評価の直後に行われます。

DBCS 関数は、DBCS 引数を認めている関数の引数として参照できます。

国別関数

一般形式において国別 ID が認められていて、一般形式に関する規則が関数への参照を特に禁じていないところではどこでも、国別関数を指定できます。ただし、次の指定はできません。

- 任意のステートメントの受け入れオペランドとして指定。
- 一般形式に関する規則で参照されているデータ項目が特定の特性 (クラスおよびカテゴリー、使用法、サイズ、および暗黙的値など) を持つ必要があり、定義に従った関数の評価および指定された特定の引数が、これらの特性を持たない場合。

国別関数は、国別引数を認めている関数の引数として参照できます。

日時関数

一般形式において日時 ID が認められていて、一般形式に関する規則が関数への参照を特に禁じていない場合であれば、日時関数を指定できます。ただし、次の指定はできません。

- 任意のステートメントの受け入れオペランドとして指定。
- 一般形式に関する規則において、参照されるデータ項目が特定の特性 (例えばクラスおよびカテゴリー、使用法、サイズ、暗黙の値) を持つことが求められていて、関数定義および指定された特定の引数による関数の評価が、これらの特性を持たない場合の指定。

日時関数は、比較条件の一部として使用できます。日時関数を比較条件の中に指定すると、その比較条件の評価はこの関数の評価の直後に実行されます。

日時関数は、日時を引数として取ることができる関数の引数として参照できます。

ブール関数

一般形式においてブール ID が認められていて、一般形式に関する規則が関数への参照を特に禁じていない場合であれば、ブール関数を指定できます。ただし、次の指定はできません。

- 任意のステートメントの受け入れオペランドとして指定。
- 一般形式に関する規則において、参照されるデータ項目が特定の特性 (例えばクラスおよびカテゴリー、使用法、サイズ、暗黙の値) を持つことが求められていて、関数定義および指定された特定の引数による関数の評価が、これらの特性を持たない場合の指定。

ブール関数は、比較条件の一部として使用できます。ブール関数を比較条件の中に指定すると、その比較条件の評価はこの関数の評価の直後に実行されます。

ブール関数は、ブール値を引数として取ることができる関数の引数として参照できます。

End of IBM Extension

整数関数

整数関数は、算術式を指定できるところでのみ使用できます。

整数関数は、整数の引数を認めている関数に対する引数として参照できます。

特別な使用に関する注意

CALL ステートメントの USING 句中の ID は、関数 ID であってはなりません。

COPY ステートメントの REPLACING 句中では、すべてのタイプの関数 ID が認められています。

引数

関数によっては、戻り値が、その関数の評価時に関数 ID 内で指定された引数によって決まるものもあります。引数を必要としない関数もあれば、固定数の引数を必要とする関数もあり、そしてまた、可変数の引数を認める関数もあります。

引数は次のいずれかでなければなりません。

- ID
- 算術式

- 関数 ID
- 表意定数以外のリテラル
- 特殊レジスター
- 簡略名
- キーワード

関数への引数として、結果がその引数のカテゴリー要件を満たす任意の関数または関数を含む式 (同じ関数の別の評価も含めて) が認められています。

関数に特有の引数指定については 7-281 ページの『関数定義』を参照してください。

引数のタイプには次のものがあります。

英字 英字クラスまたは英字のみを含む非数字リテラルの基本データ項目。引数の内容は、関数の値を決定するのに使用されます。引数の長さを使用して、関数の値を決定することもできます。

英数字 英字クラスまたは英数字クラス、または非数字リテラルのデータ項目。引数の内容は、関数の値を決定するのに使用されます。引数の長さを使用して、関数の値を決定することもできます。

IBM Extension

ブール ブール・クラスのデータ項目またはブール・リテラル。

DBCS DBCS クラスまたは DBCS リテラルのデータ項目。引数の内容は、関数の値を決定するのに使用されます。引数の長さを使用して、関数の値を決定することもできます。

国別 NATIONAL クラスまたは国別リテラルのデータ項目。引数の内容は、関数の値を決定するのに使用されます。引数の長さを使用して、関数の値を決定することもできます。

日時 日時クラスのデータ項目。引数の内容は、関数の値を決定するのに使用されます。引数の長さを使用して、関数の値を決定することもできます。

End of IBM Extension

指標 指標データ項目。引数に関連付けられているサイズをその引数の値の判別に使用できます。

整数 結果が常に整数値になる算術式。この式の、符号も含めた値は、関数の値を決定するのに使用されます。

数字 算術式の、符号も含めた値は、関数の値を決定するのに使用されます。

IBM Extension

キーワード
キーワードは関数定義に従って指定する必要があります。

簡略名 SPECIAL-NAMES 段落で定義した簡略名が指定されます。簡略名に関連付けられている機能を関数の値の判別に使用できます。

ポインター
ポインター ID。引数に関連付けられているサイズを関数の値の判別に使用できます。

タイプ宣言
タイプ名が指定されます。タイプ宣言に関連付けられているサイズを関数の値の判別に使用できます。

特殊レジスター
特殊レジスターは関数定義に従って指定する必要があります。特殊レジスターに関連付けられている情報を関数の値の判別に使用できます。

End of IBM Extension

関数によっては引数に、例えば受け入れ値の範囲といった制約を定めているものもあります。関数の引数として割り当てられた値が指定された制約にしたがっていない場合、戻り値は未定義になります。

ネストされた関数が引数として使用された場合、その関数の引数の評価は、外側の関数の引数の影響を受けません。

同じ関数レベルの引数のみが互いに影響し合います。この相互の影響は、次の 2 つの場合に生じます。
- 関数の引数として現れる算術式の計算は、その関数の別の引数の影響を受けます。
- 関数の評価においては、そのすべての引数の属性が考慮されます。

IBM Extension

浮動小数点リテラルは、数字引数の使用が認められている場合、および数字引数を取ることができる関数内で使用される算術式の中では、どこでも使用できます。浮動小数点リテラルは、整数の引数が必要であるところでは認められていません。

外部浮動小数点項目は、数字引数の使用が認められている場合、および数字引数を取ることができる関数内で使用される算術式の中では、どこでも使用できます。

外部浮動小数点項目は、整数の引数が必要であるところや、関数 ID で英数字クラスの引数が認められているところ (例えば LOWER-CASE、REVERSE、 UPPER-CASE、NUMVAL、および NUMVAL-C 関数) では、認められていません。

End of IBM Extension

関数の引数を評価する際の優先順序

関数が評価されるとき、その引数は、一連の引数が指定されている順序 (左から右) で個別に評価されます。評価される引数は、関数 ID であっても、関数 ID を含む式であってもかまいません。

算術式が引数として指定され、式の最初の演算子が単項正または単項負である場合、そのすぐ左端に左括弧が必要です。例えば、関数 `MEAN(x-y z)` は、2 つの引数 `x-y` と `z` の平均になります。

`y` の単項負の平均を得るには、括弧を次のように追加します。

```
MEAN((x) (-y) z)
```

注: 上記の例のように、単項負の平均を計算していて、単項負が複数の引数の最初の引数でない場合、前にある一連の引数も、括弧で囲む必要があります。これによって、この例の単項負 (-y) は添え字として解釈されないことが保証されます。

ALL 添え字付け

関数が、ある引数を不定の回数繰り返すことを認めている場合、データ名とテーブルを識別する任意の修飾子を指定することで、そのテーブルを参照できます。この後にすぐ添え字付けを続けて、添え字のうちの 1 つまたは複数を語 ALL にすることができます。

注: ALL 添え字の評価結果は、少なくとも 1 つの引数にならなくてなりません。そうでなければ関数によって戻される値が未定義となります。ただし、*RANGE コンパイラー・オプションを指定することで、状況を実行時に診断できます。

ALL を 1 つの添え字として指定すると、すべての可能なテーブル・エレメントを、その添え字位置におけるすべての有効な添え字を使用して指定することと同等になります。

「テーブル名 (ALL)」として指定されているテーブル引数に関しては、各テーブル・エレメントが引数として暗黙に指定される順序は左から右で、最初 (または左端) の引数は「テーブル名 (1)」となり、ALL は 1 に置き換えられます。次の引数は「テーブル名 (2)」で、添え字は 1 増やされています。この処理は、添え字が 1 ずつ増やされて暗黙の引数を生成しながら、ALL 添え字がその値の範囲に増えるまで継続されます。

例えば、

```
FUNCTION MEAN(Table(ALL))
```

は、次と同じです。

```
FUNCTION MEAN(Table(1) Table(2) Table(3)... Table(n))
```

ここで、n はテーブル内のエレメントの数です。

複数の ALL 添え字、例えば「テーブル名 (ALL, ALL, ALL)」がある場合、最初の暗黙の引数は「テーブル名 (1, 1, 1)」で、それぞれの ALL は 1 に置き換えられます。次の引数は「テーブル名 (1, 1, 2)」で、右端の添え字が 1 増やされます。右端の ALL で表されている添え字はその値の範囲まで増やされ続けて、それぞれの値の暗黙の引数を生成します。

ALL として指定された添え字がいったん値の範囲まで増えると、ALL と指定された次の左の添え字が 1 増やされます。新たに増やされた添え字の右にある、ALL として指定されている各添え字が、暗黙の引数を生成するために 1 に設定されます。再度右端の ALL で表されている添え字が、その値の範囲まで増え続け、それぞれの値のための暗黙の引数を生成します。この処理は、ALL と指定されているそれぞれの添え字が、それぞれの値の範囲まで増え続けるまで継続されます。

例えば、

```
FUNCTION MEAN(Table(ALL, ALL))
```

は、次と同じです。

```
FUNCTION MEAN(Table(1, 1) Table(1, 2) Table(1, 3)... Table(1, n)
              Table(2, 1) Table(2, 2) Table(2, 3)... Table(2, n)
              Table(3, 1) Table(3, 2) Table(3, 3)... Table(3, n)
                                  .
                                  .
                                  .
              Table(m, 1) Table(m, 2) Table(m, 3)... Table(m, n))
```

ここで、n はテーブルの列数をあらわす次元のエレメント数で、m はテーブルの行数をあらわす次元のエレメント数です。

ALL 添え字は、リテラル添え字、データ名添え字、または指標名添え字と結合して、多次元テーブルを参照できます。

例えば、

```
FUNCTION MEAN(Table(ALL, 2))
```

は、次と同じです。

```
FUNCTION MEAN(Table(1, 2)
              Table(2, 2)
              Table(3, 2)
                  .
                  .
                  .
              Table(m, 2))
```

ここで、m はテーブルの行次元のエレメント数です。

ALL 添え字が引数として指定され、その引数が参照変更されている場合、その参照修飾子はテーブルの暗黙に指定されたそれぞれのエレメントに対して適用されます。

ALL 添え字が参照変更されるオペランドとして指定されている場合、その参照修飾子は、テーブルの暗黙に指定されたそれぞれのエレメントに対して適用されます。

ALL 添え字が OCCURS DEPENDING ON 文節と関連している場合、値の範囲は、OCCURS DEPENDING ON 文節のオブジェクトによって決定されます。

例えば、次のような給与計算のレコード定義が与えられたとします。

```
01 PAYROLL.
  02 PAYROLL-WEEK   PIC 99.
  02 PAYROLL-HOURS  PIC 999 OCCURS 1 TO 52
     DEPENDING ON PAYROLL-WEEK.
```

次の COMPUTE ステートメントを使用して、任意の週の平均作業時間を割り出すことができます。

次の COMPUTE ステートメントを使用して、今日までの 1 年間の合計時間、任意の週の最長作業時間、および最長時間に対応する特定の週を判別できます。

```
    COMPUTE YTD-HOURS = FUNCTION SUM (PAYROLL-HOURS(ALL))
    COMPUTE MAX-HOURS = FUNCTION MAX (PAYROLL-HOURS(ALL))
    COMPUTE MAX-WEEK  = FUNCTION ORD-MAX (PAYROLL-DAYS(ALL))
```

この関数呼び出しでは、ALL 添え字を使用して PAYROLL-HOURS 配列のすべてのエレメント (PAYROLL-WEEK フィールドの実行時間値によって異なる) を参照できます。

関数定義

7-282 ページの表 7-19 に、それぞれの組み込み関数の、引数タイプ、関数タイプ、および戻り値の概説を記載してあります。引数タイプおよび関数タイプは次のように省略されています。

- A = 英字

IBM Extension

- B = ブール
- D = DBCS
- DA = 日時

End of IBM Extension

- I = 整数
- IX = 指標

IBM Extension

- K = キーワード
- M = 簡略名

End of IBM Extension

- N = 数字
- NL = 国別

IBM Extension

- P = ポインター
- S = 特殊レジスター
- T = タイプ名

End of IBM Extension

- X = 英数字
- U = 国別 (汎用文字セットとユニコード)

注: これらの省略形に関連する番号によって、関数のどの引数を参照しているかを識別します。

表 7-19. 関数の表

関数名	引数タイプ	関数タイプ	戻り値
ACOS	N1	N	N1 のアークコサイン
ADD-DURATION[1]	DA1、 K2、I3	DA	期間が加算された日時項目
ANNUITY	N1、I2	N	I2 の期間に、N1 の利率で支払われた年金を概算する。
ASIN	N1	N	N1 のアークサイン
ATAN	N1	N	N1 のアークタンジェント
CHAR	I1	X	プログラム照合順序位置 I1 の文字
CONVERT-DATE-TIME[1]	DA1 または X1 または I1、 K2、X3 または K3 または S3、M4 または S4	DA	変換された日時項目
COS	N1	N	N1 のコサイン
CURRENT-DATE	なし	X	現在の日付と時刻、およびグリニッジ標準時との時差
DATE-OF-INTEGER	I1	I	整数日付と等しい標準日付 (YYYYMMDD)
DATE-TO-YYYYMMDD[1]	I1 または I1、I2	I	標準日付の年を 2 桁から 4 桁に変換する
DAY-OF-INTEGER	I1	I	整数日付と等しい年間通算日 (YYYYDDD)
DAY-TO-YYYYDDD[1]	I1 または I1、I2	I	年間通算日の年を 2 桁から 4 桁に変換する

表 7-19. 関数の表 (続き)

関数名	引数タイプ	関数タイプ	戻り値
DISPLAY-OF	NL1	X	指定された場合、I2 で識別されるコード・ページを使用して対応する文字表現に変換される NL1 の各文字、または I2 が指定されていない場合、コンパイル時に選択されるデフォルトのコード・ページ
	NL1、I2		
	NL1、X2 または D2		
EXTRACT-DATE-TIME[1]	DA1、X2 または K2	I または X	抽出された日付項目、時刻項目、またはタイム・スタンプ項目の一部
FACTORIAL	I1	I	I1 の階乗
FIND-DURATION[1]	DA1、 DA2、 K3	I	2 つの日時項目間の整数の期間
INTEGER	N1	I	N1 より大きくない最大の整数
INTEGER-OF-DATE	I1	I	標準日付 (YYYYMMDD) と等しい整数日付
INTEGER-OF-DAY	I1	I	年間通算日の日付 (YYYYDDD) と等しい整数日付
INTEGER-PART	N1	I	引数の整数部分
LENGTH	A1、N1、 X1、D1、B1、T1、 DA1、P1、IX1、 NL	I	引数の長さ
LOCALE-DATE[1]	X1 または D1、M2	X	指定したロケールに従って形式化された日付ストリング
LOCALE-TIME[1]	X1 または D1、M2	X	指定したロケールに従って形式化された時刻ストリング
LOG	N1	N	N1 の自然対数
LOG10	N1	N	N1 の 10 を底とする常用対数
LOWER-CASE	A1 または X1	X	引数中の文字をすべて小文字に設定する
	D1	D	
	NL	NL	
MAX	A1...	X	最大引数の値; 関数のタイプは引数によって異なります。
	I1...	I	
	N1...	N	
	X1...	X	
	U1...	U	
MEAN	N1...	N	引数の算術平均
MEDIAN	N1...	N	引数の中央値
MIDRANGE	N1...	N	最大引数と最小引数の平均値
MIN	A1...または	X	最小引数の値; 関数のタイプは引数によって異なります。
	I1...または	I	
	N1...または	N	
	X1...または	X	
	U1...	U	
MOD	I1、I2	I	I1 モジュロ I2

表 7-19. 関数の表 (続き)

関数名	引数タイプ	関数タイプ	戻り値
NATIONAL-OF	A1 または X1 または D1	NL	指定された場合、I2 で識別されるコード・ページを使用して、国別文字に変換される引数-1 の文字、または I2 が指定されていない場合、コンパイル時に選択されるデフォルトのコード・ページ
	A1 または X1 または D1、I2		
	A1 または X1 または D1、NL2		
NUMVAL	X1	N	単純な数字ストリングの数字
NUMVAL-C	X1 または X1、X2	N	オプショナルのコンマと通貨記号が付いた数字ストリングの数字
ORD	A1 または X1	I	照合順序における引数の順序位置
ORD-MAX	A1...、 N1...、	I	最大引数の順序位置
	X1...、または U1...		
ORD-MIN	A1...、 N1...、	I	最小引数の順序位置
	X1...、または U1...		
PRESENT-VALUE	N1、N2...	N	N1 の割引率での、一連の将来の期間終了時の量 N2 の現在値
RANDOM	I1 または、なし	N	乱数
RANGE	I1...	I	最大引数から最小引数を引いた値; 関数のタイプは引数によって異なります。
	N1...	N	
REM	N1、N2 (N2 はゼロ以外)	N	N1 を N2 で割ったときの剰余
REVERSE	A1 または X1	X	引数の文字の逆順
	D1	D	
	NL	NL	
SIN	N1	N	N1 のサイン
SQRT	N1	N	N1 の平方根
STANDARD DEVIATION	N1...	N	引数の標準偏差
SUM	I1...	I	引数の合計; 関数のタイプは引数によって異なります。
	N1...	N	
SUBTRACT-DURATION[1]	DA1、 K2、I3	DA	期間が減算された日時項目
TAN	N1	N	N1 のタンジェント
TEST-DATE-TIME[1]	DA1 または X1 または I1、 K2、X3 または K3 または S3、M4 または S4	B	日時項目が有効な場合は真 (B"1")、その他の場合は偽
TRIM[1]	A1 または X1	X	左ブランクおよび右ブランクを持つ、または指定した文字を切り取ったストリング
	A1、A2 または X1、X2		
	D1 または D1、D2	D	
	NL1 または NL1、NL2	NL	

表 7-19. 関数の表 (続き)

関数名	引数タイプ	関数タイプ	戻り値
TRIML[1]	A1 または X1	X	左ブランクを持つ、または指定した文字を切り取ったストリング
	A1、A2 または X1、X2		
	D1 または D1、D2	D	
	NL1 または NL1、NL2	NL	
TRIMR[1]	A1 または X1	X	右ブランクを持つ、または指定した文字を切り取ったストリング
	A1、A2 または X1、X2		
	D1 または D1、D2	D	
	NL1 または NL1、NL2	NL	
UPPER-CASE	A1 または X1	X	引数中の文字をすべて大文字に設定する
	D1	D	
	NL	NL	
UTF8STRING[1]	A1、X1、 D1 または NL1		可変長 UTF-8 ストリング
VARIANCE	N1...	N	引数の差異
WHEN-COMPILED	なし	X	プログラムがコンパイルされた日付と時刻
YEAR-TO-YYYY[1]	I1 または I1、I2	I	年を 2 桁から 4 桁に変換する
注: [1] IBM 拡張			

以下の各セクションには 7-282 ページの表 7-19 に要約されている組み込み関数のそれぞれの関数定義が記載されています。

ACOS

ACOS 関数は、指定された引数のアークコサインの近似値をラジアンで戻します。

関数タイプは数字です。

フォーマット

引数-1
数字クラスでなければなりません。引数-1 の値は、-1 以上 +1 以下でなければなりません。

戻り値は引数のアークコサインの近似値であり、ゼロ以上パイ (π) 以下の値です。

ADD-DURATION

IBM Extension

ADD-DURATION 関数は、日付、時刻、またはタイム・スタンプの項目に期間を加算し、その変更した項目を戻します。

関数タイプは日時です。

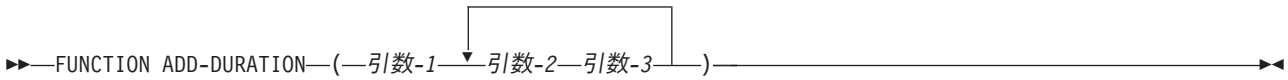
戻り値の長さは、引数-1 に指定する日付項目、時刻項目、またはタイム・スタンプ項目の長さによって決まります。戻り値は、引数-1 の長さまで切り捨てられます。

日付項目に期間を加算する場合は、戻される日付は、以下に示す特定の範囲内の値でなければなりません。

- 4 桁の日付の場合は 0001/01/01 から 9999/12/31 までの範囲内でなければならない。
- 2 桁の日付の場合は 0001/01/01 から 9999/12/31 までの範囲内でなければならない。ただし、年の部分は 2 桁に切り捨てられます。
- 3 桁の年 (1 桁の世紀と 2 桁の年) の場合は 1900/01/01 から 2899/12/31 (デフォルト) までの範囲内でなければならない。この範囲は DATTIM PROCESS ステートメント・オプションを指定することによって変更できます。

2 桁の日付項目に期間を加算する場合の戻り値の範囲は 4 桁の年の場合と同じです。ただし、戻り値の年の部分は 2 桁に切り捨てられます。

フォーマット



引数-1

日付、時刻、またはタイム・スタンプのデータ項目でなければなりません。

引数-1 は、期間を加算される値が入っているデータ項目です。期間は引数-2 と引数-3 に指定します。

引数-2

引数-2 は期間を表すキーワードです。有効な期間キーワードを以下に示します。

- YEARS
- MONTHS
- DAYS
- HOURS
- MINUTES
- SECONDS
- MICROSECONDS
- PICOSECONDS

期間キーワードと引数-1 との間で整合性がとれている必要があります。例えば、期間キーワードは以下の規則に従わなければなりません。

1. YEARS、MONTHS、および DAYS は、日付項目またはタイム・スタンプ項目に対してのみ加算することができる。

2. HOURS、MINUTES、SECONDS、および MICROSECONDS は、時刻項目またはタイム・スタンプ項目に対してのみ加算することができる。
3. PICOSECONDS は、タイム・スタンプ項目にのみ加算できます。

引数-3

整数算術式でなければなりません。引数-3 は、引数-2 で指定した期間の単位数であり、引数-1 に加算されます。

引数-3 は負の整数でも構いませんが、この関数が取るのはその絶対値だけです。引数-3 が 9 桁よりも長い場合は切り捨てが行われます。

引数-2 および引数-3 は反復可能です。ただし、1 つの組み込み関数の中に重複する引数-2 があってはなりません。

日付に期間を加算した結果が無効となった場合は、その日付に対する調整が行われます。例えば、1997 年 3 月 31 日という日付に期間 1 月を加算すると、その結果は 1997 年 4 月 31 日という無効な日付となります。このような場合、この日付は 1997 年 4 月 30 日という有効な日付に調整されます。

例

以下の例で ADD-DURATION 組み込み関数の使用法を示します。

```
MOVE FUNCTION ADD-DURATION (date-3 MONTHS 1)
     TO  date-2.
MOVE FUNCTION ADD-DURATION (date-3 MONTHS int-1 * 2)
     TO  date-1.
MOVE FUNCTION ADD-DURATION (date-1  YEARS 1 MONTHS 5 DAYS 23)
     TO date-2.
```

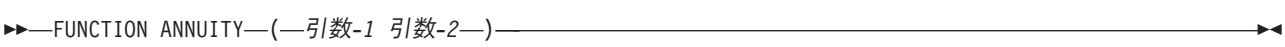
End of IBM Extension

ANNUITY

ANNUITY 関数は、初期値 1 に対して、一定の利率で一定の期間数の間、それぞれの期間終了時に支払われる年金の比率の近似値を戻します。期間数は引数-2 で指定し、利率は引数-1 で指定します。例えば、引数-1 がゼロで引数-2 が 4 の場合、戻り値は比率 1/4 の近似値になります。

関数タイプは数字です。

フォーマット



引数-1

数字クラスでなければなりません。引数-1 の値はゼロ以上でなければなりません。

引数-2

正の整数でなければなりません。

引数-1 の値がゼロの場合、関数によって戻される値は次の近似値になります。

```
1 / ARGUMENT-2
```

引数-1 の値がゼロでない場合、関数の値は次の近似値になります。

```
ARGUMENT-1 / (1 - (1 + ARGUMENT-1) ** (- ARGUMENT-2))
```

ASIN

ASIN 関数は、指定された引数のアークサインの近似値をラジアンで戻します。

関数タイプは数字です。

フォーマット

引数-1
数字クラスでなければなりません。引数-1 の値は、-1 以上 +1 以下でなければなりません。

戻り値は引数-1 のアークサインの近似値であり、-π/2 以上 +π/2 以下の値になります。

ATAN

ATAN 関数は、指定された引数のアークタンジェントの近似値をラジアンで戻します。

関数タイプは数字です。

フォーマット

引数-1
数字クラスでなければなりません。

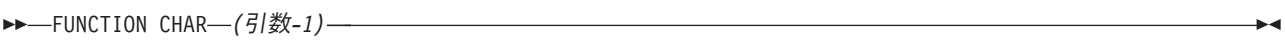
戻り値は引数-1 のアークタンジェントの近似値で、-π/2 より大きく +π/2 より小さな値になります。

CHAR

CHAR 関数は、プログラム照合順序において、指定された引数の値に等しい順序位置にある 1 文字の英数字を戻します。

関数タイプは英数字です。

フォーマット



引数-1
整数でなければなりません。値は 1 以上で、かつ照合順序内の位置の個数以下でなくてはなりません。

プログラム照合順序において、同じ位置に複数の文字があるときには、関数値として戻される文字は、ALPHABET 文節内でその文字位置に対して指定された最初のリテラルです。

現行のプログラム照合順序が ALPHABET 文節によって指定されていない場合は、EBCDIC 照合順序が使用されます。

CONVERT-DATE-TIME

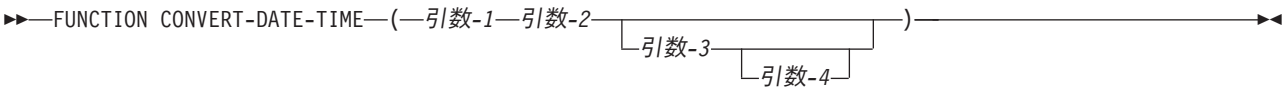
IBM Extension

CONVERT-DATE-TIME 関数は、英数字クラス、数字クラス、または日時クラスの項目をとり、日時項目を戻します。

関数タイプは日時です。

戻り値の長さは、引数-2 から引数-4 で指定した日付項目、時刻項目、またはタイム・スタンプ項目の形式で認められている長さによって決まります。

フォーマット



引数-1

リテラル-3 は、下記のものになります。

- 日付、時刻、またはタイム・スタンプの項目
- 英数字クラスの項目
- 非数字リテラル
- 整数値クラスの項目

引数-2

戻り値のカテゴリーを指定します。以下のキーワードのいずれか 1 つでなければなりません。

- DATE
- TIME
- TIMESTAMP

引数-1 が日付、時刻、またはタイム・スタンプの項目である場合、CONVERT-DATE-TIME で可能な変換は以下のものだけです。

- 日付から日付またはタイム・スタンプへの変換
- 時刻から時刻またはタイム・スタンプへの変換
- タイム・スタンプから日付、時刻、またはタイム・スタンプへの変換

引数-2 が TIMESTAMP の場合、引数-3 は、FORMAT OF 特殊レジスターでのみ指定でき、引数-4 は指定できません。

引数-1 が日時項目である場合は、日時の移動が行われます。

引数-1 が整数値であり、戻される日時項目が引数-3 で指定した日時形式で認められている長さよりも長い場合は、戻される日時項目に対する右寄せおよび切り捨てが行われます。

引数-1 が整数値以外のものあり、戻される日時項目が引数-3 で指定した日時形式で認められている長さよりも長い場合は、戻される日時項目に対する左寄せおよび切り捨てが行われます。

引数-3

日付項目または時刻項目の形式を指定します。以下のものでなければなりません。

- 長さが最小でも 2 文字の非数字リテラル

- キーワード LOCALE
- FORMAT OF 特殊レジスター

有効なリテラルのリストおよびこの引数が従わなければならない規則については 5-16 ページの『FORMAT 文節』に記載されている SPECIAL-NAMES FORMAT 文節の説明を参照してください。

引数-3 は引数-2 によって参照されるカテゴリーを表していなければなりません。

引数-3 がキーワード LOCALE である場合は、日付または時刻の形式は LOCALE に基づきます。引数-4 を指定しないと、現行ロケールが使用されます。その他の場合は、簡略名または LOCALE OF 特殊レジスターと関連付けられているロケールが使用されます。

| 引数-3 を指定しないと、戻り値の形式は SPECIAL-NAMES FORMAT 文節に依存します。
| SPECIAL-NAMES 段落内に形式が 1 つも定義されていない場合は、*ISO 形式が使用されます。
| TIMESTAMP では、引数-3 が指定されていない場合、デフォルトの形式 @Y-%m-%d-%H%M%S.@Sm
| が使用されます。

引数-4

LOCALE と関連付けられている簡略名または LOCALE OF 特殊レジスターでなければなりません。

引数-4 が従わなければならない規則を以下に示します。

- 引数-4 が指定されており、かつ引数-3 がロケールに基づく形式リテラルである (例えば %p を含んでいる) 場合は、そのロケールに基づく形式リテラルは引数-4 で指定されているロケールを使用して変換指定子の実際の値を判別する。
- 引数-3 がロケールに基づく形式リテラルであり (例えば %p を含んでいる)、かつ引数-4 が指定されていない場合は、そのロケールに基づく形式リテラルは現行ロケールを使用して変換指定子の実際の値を判別する。
- 引数-3 がロケールに基づく形式リテラルであり (例えば %p を含んでいる)、かつ LOCALE OF 特殊レジスターが非ロケール項目を参照するために使用されている場合は、そのロケールに基づく形式リテラルはデフォルトのロケールを使用して変換指定子の実際の値を判別する。

例

以下の例で CONVERT-DATE-TIME 組み込み関数の使用法を示します。

```
MOVE FUNCTION CONVERT-DATE-TIME ('95/05/30' DATE)
     TO  date-1.
MOVE FUNCTION CONVERT-DATE-TIME
     ('95/05/30' DATE '%y/%m/%d')
     TO  date-1.
MOVE FUNCTION CONVERT-DATE-TIME
     ('95/05/30' DATE '%y/%m/%d' my-locale)
     TO  date-1.
MOVE FUNCTION CONVERT-DATE-TIME
     ('95/05/30' DATE LOCALE my-locale)
     TO  date-1.
```

End of IBM Extension

COS

COS 関数は、引数にラジアンで指定された角度または弧のコサインの近似値を戻します。

関数タイプは数字です。

フォーマット

引数-1
数字クラスでなければなりません。

戻り値は引数のコサインの近似値であり、-1 以上 +1 以下の値になります。

CURRENT-DATE

CURRENT-DATE 関数は、その関数を評価するシステムによって提供される、カレンダー日付、その日の時刻、およびグリニッジ標準時との時差を示す 21 文字の英数字の値を戻します。

関数タイプは英数字です。

フォーマット

左から右へ読んだ場合、戻り値の 21 の文字位置は次のように解釈します。

文字位置
内容

1 〜 4 西暦の年を表す 4 桁の数字

5 〜 6 01 〜 12 の範囲の月を表す 2 桁の数字

7 〜 8 01 〜 31 の範囲の日を表す 2 桁の数字

9 〜 10
00 〜 23 の範囲の時を表す 2 桁の数字

11 〜 12
00 〜 59 の範囲の分を表す 2 桁の数字

13 〜 14
00 〜 59 の範囲の秒を表す 2 桁の数字

15 〜 16
00 〜 99 の範囲の 1/100 秒を表す 2 桁の数字

17 文字 '-' または文字 '+'。前の文字位置で示されているローカル時刻がグリニッジ標準時より遅れている場合、文字 '-' が戻されます。示されているローカル時刻がグリニッジ標準時と同じかそれより進んでいる場合、文字 '+' が戻されます。

18 〜 19
文字位置 17 が '-' である場合には、00 〜 12 の範囲の 2 桁の数字が戻され、通知された時刻がグリニッジ標準時よりその時間数だけ遅れていることを示します。文字位置 17 が '+' である場合には、00 〜 13 の範囲の 2 桁の数字が戻され、通知された時刻がグリニッジ標準時よりその時間数だけ進んでいることを示します。

20 ～ 21
00 ～ 59 の範囲の 2 桁の数字が戻され、通知された時刻のグリニッジ標準時からの進み、または遅れの分数を示します。進んでいるか遅れているかは、文字位置 17 が '+' であるか '-' であるかによって決まります。

詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

DATE-OF-INTEGER

DATE-OF-INTEGER 関数は、西暦の日付を、整数日付形式から標準日付形式 (YYYYMMDD) に変換します。

関数タイプは整数です。

関数結果は 8 桁の整数です。

フォーマット

引数-1
西暦の 1600 年 12 月 31 日以後の日数を表す正の整数。有効範囲は 1 ～ 3,067,671 で、これは 1601 年 1 月 1 日から 9999 年 12 月 31 日に対応しています。

戻り値は、引数-1 として指定されている整数と等しい国際標準化機構 (ISO) 標準日付を表します。

戻り値は YYYYMMDD 形式の整数で、YYYY が西暦の年、MM がその年の月、DD がその月の日を表します。

DAY-OF-INTEGER

DAY-OF-INTEGER 関数は、西暦の日付を、整数日付形式から年間通算日形式 (YYYYDDD) に変換します。

関数タイプは整数です。

関数結果は 7 桁の整数です。

フォーマット

引数-1
西暦の 1600 年 12 月 31 日以後の日数を表す正の整数。有効範囲は 1 ～ 3,067,671 で、これは 1601 年 1 月 1 日から 9999 年 12 月 31 日に対応しています。

戻り値は、引数-1 として指定されている整数日付に等しい年間通算日を表します。戻り値は YYYYDDD 形式の整数で、YYYY は西暦の年、DDD はその年の日を表しています。

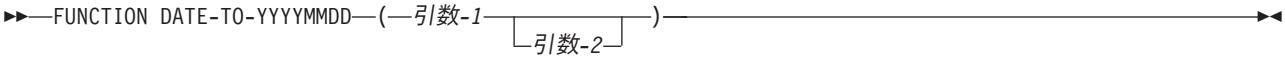
DATE-TO-YYYYMMDD

IBM Extension

DATE-TO-YYYYMMDD 関数は、引数-1 を YYnnnn 形式から YYYYnnnn 形式に変換します。実行時に引数-2 が年に加算された場合、引数-2 は 100 年間隔の終了年、すなわち引数-1 の年が当てはまるスライド・ウィンドウを定義します。

関数のタイプは整数です。

フォーマット



引数-1
1000000 より小さい正の整数でなければなりません。

引数-2
整数でなければなりません。引数-2 が省略された場合、関数は引数-2 の値を 50 として評価します。
実行時に、引数-2 と年との和は、10000 より小さく、1699 より大きい値になります。

同等の算術式は、次のようになります。

```
(FUNCTION YEAR-TO-YYYY (YY, argument-2) * 10000 + nnnn)
```

ここで、

```
YY = (argument-1/10000) truncated to an integer value
nnnn = argument-1 modulus 10000
```

この関数は、スライド・ウィンドウのアルゴリズムをサポートします。固定ウィンドウを指定する方法については 7-326 ページの『YEAR-TO-YYYY』を参照してください。

例

2002 年には次の関数の戻り値は、

```
FUNCTION DATE-TO-YYYYMMDD(851003, 10)
```

19851003 になります。

1994 年には次の関数の戻り値は、

```
FUNCTION DATE-TO-YYYYMMDD(981002,(-10))
```

18981002 になります。

End of IBM Extension

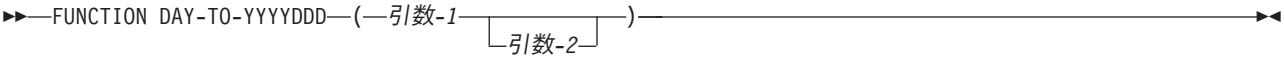
DAY-TO-YYYYDDD

IBM Extension

DAY-TO-YYYYDDD 関数は引数-1 を YYnnn 形式から YYYYnnn 形式に変換します。実行時に引数-2 が年に加算された場合、引数-2 は 100 年間隔の終了年、すなわち引数-1 の年が当てはまるスライド・ウィンドウを定義します。

関数のタイプは整数です。

フォーマット



引数-1
100000 より小さい正の整数でなければなりません。

引数-2
整数でなければなりません。引数-2 が省略された場合、関数は引数-2 の値を 50 として評価します。
実行時に、引数-2 と年との和は、10000 より小さく、1699 より大きい値になります。

同等の算術式は、次のようになります。

```
(FUNCTION YEAR-TO-YYYY (YY, argument-2) * 1000 + nnn)
```

ここで、

```
YY = (argument-1/1000) truncated to an integer value
nnn = argument-1 modulus 1000
```

この関数は、スライド・ウィンドウのアルゴリズムをサポートします。固定ウィンドウを指定する方法については 7-326 ページの『YEAR-TO-YYYY』を参照してください。

例

1996 年には次の関数の戻り値は、

```
FUNCTION DAY-TO-YYYYDDD(85003, 10)
```

1985003 になります。

2013 年には次の関数の戻り値は、

```
FUNCTION DAY-TO-YYYYDDD(95005,(-10))
```

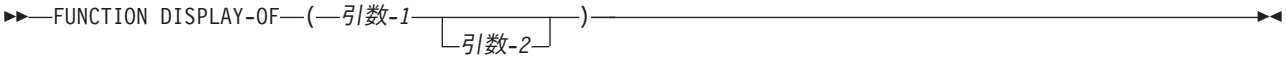
1995005 になります。

End of IBM Extension

DISPLAY-OF

DISPLAY-OF 関数は、特定コード・ページ表現に変換される引数-1 のコンテントで構成される英数字ストリングを戻します。関数のタイプは英数字です。

形式 1: ターゲット CCSID の指定



引数-1
クラス国別でなければなりません。引数-1 は、変換のソース・ストリングを識別します。

IBM Extension

引数-2
整数でなければなりません。引数-2 は、変換のターゲット・コード・ページを識別します。引数-2 は、EBCDIC、 ASCII、UTF-8、または EUC コード・ページを識別する有効な CCSID 番号でなければなりません。EBCDIC または ASCII CCSID は、SBCS、DBCS、または混合 SBCS/DBCS であるコード・ページを識別できます。

End of IBM Extension

引数-2 が省略された場合、ターゲット・コード・ページは、ソース・コードがコンパイルされたときの CCSID コンパイラー・オプション用に有効になるターゲット・コード・ページです。有効なターゲット・コード・ページが 65535 である場合、デフォルトの CCSID 37 が使用されます。

戻り値は、ターゲット・コード・ページ表現に変換される引数-1 の文字で構成される英数字ストリングです。ソース文字をターゲット・コード・ページの文字に変換できない場合、ソース文字は、1 バイト EBCDIC ターゲット・コード・ページに対してはシステム定義の置換文字 X'3F' で、DBCS コード・ページに対しては X'FEFE' で置換されます。例外条件は発生しません。

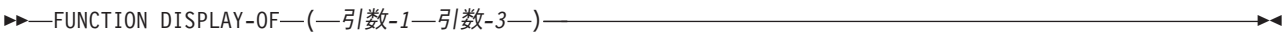
戻り値の長さは、引数-1 のコンテントおよびターゲット・コード・ページの特性によって異なります。

例外: 変換が失敗すると、重大なランタイム・エラーが発生します。国別データ (国別 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 を使用してコード化された) からターゲット CCSID への変換がオペレーティング・システムでサポートされていることを確認してください。

使用上の注意

1. ターゲット・コード・ページが混合 SBCS/DBCS EBCDIC コード・ページである場合、戻り値は、シフトアウトおよびシフトイン制御文字で区切られている DBCS サブストリングを含むことができます。
2. 整数引数-2 を指定した DISPLAY-OF 関数は、CCSID コンパイラー・オプションで指定されたものと異なるコード・ページで表示される文字データを生成するために使用されます。そのデータへの後続の COBOL 操作は、データが CCSID コンパイラー・オプションで指定される EBCDIC コード・ページで表示されることを想定する暗黙の変換を伴うことができるので、細心の注意が必要です。

形式 2: ユーザー置換文字の指定



引数-1
クラス国別でなければなりません。引数-1 は、変換のソース・ストリングを識別します。

引数-3

1 文字位置の長さを持つ、クラス英字、英数字または DBCS のリテラルまたはデータ項目でなければなりません。引数-3 は、対応する英数字または DBCS 文字を持たない国別文字の変換で使用される英数字または DBCS 置換文字を指定します。ユーザー置換文字は、EBCDIC および ASCII コード・ページでのみサポートされます。EUC などの別のコード・ページ用に使用される場合、コンパイル・エラー・メッセージ (重大度 30) が発行され、引数-3 は無視されます。

ユーザー置換文字は、有効なターゲット・コード・ページ CCSID と同じ CCSID を持ちます。有効なターゲット・コード・ページ CCSID が 2 バイトである場合、2 バイトの置換文字 (DBCS) のみを引数-3 として指定できます。有効なターゲット・コード・ページ CCSID が 1 バイトまたは混合バイトである場合は、1 バイトの置換文字 (英字または英数字) のみを引数-3 として指定できますが、混合バイト CCSID 用の 2 バイト置換文字をユーザー置換文字で置換することはできません。ターゲット CCSID タイプおよび引数-3 クラスの間の不整合がコンパイル時に検出された場合、コンパイル・エラー・メッセージ (重大度 30) が発行されます。不整合がランタイム時に検出された場合、重大なランタイム ・メッセージが発行されます。プログラムを継続するために、ユーザーがメッセージに 'G' で応答すると、引数-3 は無視されます。

ターゲット・コード・ページは、ソース・コードのコンパイル時に CCSID コンパイラー・オプション用に有効になるターゲット・コード・ページです。有効な CCSID コンパイラー・オプションが 65535 である場合、デフォルトの CCSID 37 が使用されます。

戻り値は、ターゲット・コード・ページ表現に変換される引数-1 の文字で構成される英数字ストリングです。ソース文字をターゲット・コード・ページの文字に変換できない場合、ソース文字は、ユーザー置換文字引数-3 で置換されます。例外条件は発生しません。

戻り値の長さは、引数-1 のコンテントおよびターゲット・コード・ページの特性によって異なります。

例外: 変換が失敗すると、重大なランタイム・エラーが発生します。ソース ccsid (UCS-2) からターゲット CCSID への変換がオペレーティング・システムでサポートされていることを確認してください。

DISPLAY-OF および NATIONAL-OF 組み込み関数をネストして、任意の CCSID から任意の他の CCSID に簡単に変換することができます。

例えば、以下のコードは、EBCDIC ストリングを ASCII ストリングに変換します。

```
  77  EBCDIC-CCSID PIC 9(4) BINARY VALUE 1140.
  77  ASCII-CCSID PIC 9(4) BINARY VALUE 819.
  77  Input-EBCDIC PIC X(80).
  77  ASCII-Output PIC X(80).
  . . .
* Convert EBCDIC to ASCII
      Move Function
        Display-of
          (Function National-of
             (Input-EBCDIC EBCDIC-CCSID)
           ASCII-CCSID
          )
      to ASCII-output
```

EXTRACT-DATE-TIME

IBM Extension

EXTRACT-DATE-TIME 関数は、日付、時刻、またはタイム・スタンプ項目の一部を戻します。

関数タイプは整数または英数字です。引数-2 がキーワード (MONTHS または DAYS など) である場合または数字指定子のみから成る場合は、整数が戻されます。その他の場合は、英数字データ項目が戻されます。

結果の長さは、日時項目から抽出される値によって決まります。

フォーマット

引数-1
日付、時刻、またはタイム・スタンプの項目でなければなりません。

引数-2
EXTRACT-DATE-TIME 関数によって戻される値を指定します。

引数-2 は、1 つまたは複数の分離文字と変換指定を含む期間または非数字リテラルを表すキーワードです。

この非数字リテラルの内容が数字変換指定子だけの場合は、EXTRACT-DATE-TIME 関数の戻り値は整数です。非数字リテラルの内容に分離文字または英数字変換指定子が含まれている場合は、戻り値は英数字となります。

引数-2 がキーワードの場合は整数が戻されます。

有効な期間およびそれらと等価の変換指定を以下に示します。

- YEARS ('@Y')
- MONTHS ('%m')
- DAYS ('%d')
- HOURS ('%H')
- MINUTES ('%M')
- SECONDS ('%S')
- MICROSECONDS ('@Sm')
- PICOSECONDS ('@Sp')

他の有効な変換指定のリストは SPECIAL-NAMES 段落の FORMAT 文節について説明している 5-17 ページの表 5-4を参照してください。

使用する期間キーワードまたは変換指定子は、引数-1 と整合性がとれている必要があります。例えば、期間キーワードは以下の規則に従わなければなりません。

1. YEARS、MONTHS、および DAYS は、日付項目またはタイム・スタンプ項目からのみ抽出することができる。
2. HOURS、MINUTES、SECONDS、および MICROSECONDS は、時刻項目またはタイム・スタンプ項目からのみ抽出することができる。
3. 引数-1 がロケールに基づくデータ項目であり、引数-2 がロケールに基づく変換指定子 (%p など) を含むときは、そのロケールに基づく変換指定子 (この場合は %p) は引数-1 のロケールを使用する。

   引数-1 がロケールに基づくデータ項目ではない ときは、ロケールに基づく変換指定子 (この場合は %p) はロケールに基づかない変換指定子として扱われ、% は @ に置き換えられます。この例では %p が @p になるということです。ここで、@p はロケールに基づかない場合の %p と等価です。

4. PICOSECONDS は、タイム・スタンプ項目からのみ抽出できます。

例

以下の例で EXTRACT-DATE-TIME 組み込み関数の使用法を示します。

```
COMPUTE integer-1 = FUNCTION EXTRACT-DATE-TIME (date-3 MONTHS).
COMPUTE integer-1 = FUNCTION EXTRACT-DATE-TIME (date-3 '%m').
MOVE FUNCTION EXTRACT-DATE-TIME (date-2 '%m/%d') to alphanum-1.
```

End of IBM Extension

FACTORIAL

FACTORIAL 関数は、指定された引数の階乗である整数を戻します。

関数タイプは整数です。

フォーマット

引数-1

*NOEXTEND コンパイラー・オプションが有効である場合の引数-1 は、ゼロ以上 28 以下の整数でなければなりません。*EXTEND31 コンパイラー・オプションが有効である場合、引数-1 はゼロ以上 29 以下の整数でなければなりません。引数-1 の値がゼロである場合、値 1 が戻り値で、その他の場合は、その階乗が戻されます。 *EXTEND31FULL または *EXTEND63 コンパイラー・オプションが有効である場合、引数-1 は、ゼロ以上 49 以下の整数でなければなりません。

FIND-DURATION

IBM Extension

FIND-DURATION 関数は、以下の 2 つの間の期間を計算するために使用します。

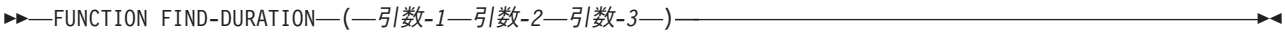
- 日付と日付
- 日付とタイム・スタンプ
- 時刻と時刻
- 時刻とタイム・スタンプ
- タイム・スタンプとタイム・スタンプ

FIND-DURATION 関数は、指定した期間を完全単位とする形式の整数を戻します。丸めはすべて下方向に行われます。期間計算ではマイクロ秒の計算も行われます。

関数タイプは整数です。

関数結果は 9 桁の整数です。この関数結果が 9 桁 (999,999,999) よりも大きい場合は、マシン・チェックが発生します。

フォーマット



引数-1、引数-2
日付、時刻、またはタイム・スタンプの項目でなければなりません。

引数-2 から引数-1 が減算されます。戻り値は、引数-3 で指定する期間を単位とする整数です。引数-1 が引数-2 よりも時間的に後の場合は、結果は負となります。引数-1 が引数-2 よりも時間的に前の場合は、結果は正となります。

引数-3
期間を表すキーワードです。有効な期間キーワードを以下に示します。

- YEARS
- MONTHS
- DAYS
- HOURS
- MINUTES
- SECONDS
- MICROSECONDS
- PICOSECONDS

有効な期間キーワードを判別するために、以下の規則が適用されます。

1. 引数-1 または引数-2 が日付項目の場合は、指定する期間は日付と整合性がとれていなければならない。
2. 引数-1 または引数-2 が時刻項目の場合は、指定する期間は時刻と整合性がとれていなければならない。
3. 戻り値が整数でない場合は切り捨てが行われる。例えば 1997 年 3 月 17 日と 1997 年 5 月 2 日との間の期間は 1.5 か月です。 FIND-DURATION は整数だけを戻すため、.5 は切り捨てられ、実際に戻される値は 1 となります。
4. PICOSECONDS 期間は、引数-1 および引数-2 がタイム・スタンプ項目の場合にのみ要求できます。

例

以下の例で FIND-DURATION 組み込み関数の使用法を示します。

```
COMPUTE integer-1 = FUNCTION FIND-DURATION (date-3 date-4 MONTHS).
COMPUTE integer-1 = FUNCTION FIND-DURATION (timestamp-1 date-5 MONTHS).
```

End of IBM Extension

INTEGER

INTEGER 関数は、引数以下の最大の整数値を戻します。

関数タイプは整数です。

フォーマット

引数-1
数字クラスでなければなりません。

戻り値は、引数-1 の値以下の最大の整数です。例えば、

```
FUNCTION INTEGER (2.5)
```

は、値 2 を戻し、

```
FUNCTION INTEGER (-2.5)
```

は、値 -3 を戻します。

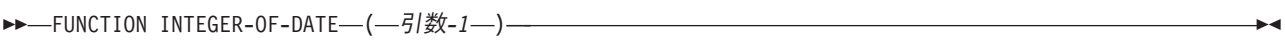
INTEGER-OF-DATE

INTEGER-OF-DATE 関数は、西暦の日付を標準日付形式 (YYYYMMDD) から整数日付形式に変換します。

関数タイプは整数です。

関数結果は、1 ～ 3,067,671 の範囲の 7 桁の整数です。

フォーマット



引数-1
(YYYY * 10,000) + (MM * 100) + DD を計算して得られる値を持つ YYYYMMDD 形式の整数でなくてはなりません。

- YYYY は西暦の年を表します。1600 より大きく 9999 以下の整数でなくてはなりません。
- MM は月を表し、13 より小さい正の整数でなくてはなりません。
- DD は日を表し、32 より小さい正の整数で、かつ指定された年と月との組み合わせが有効であることを前提にしています。

戻り値は、引数-1 で表される日付が、西暦の 1600 年 12 月 31 日から数えて何日目であるかを示す整数です。

INTEGER-OF-DAY

INTEGER-OF-DAY 関数は、西暦の日付を年間通算日形式 (YYYYDDD) から整数日付形式に変換します。

関数タイプは整数です。

関数結果は 7 桁の整数です。

フォーマット

引数-1
(YYYY * 1000) + DDD を計算して得られる値を持つ YYYYDDD 形式の整数でなくてはなりません。

- YYYY は西暦の年を表します。1600 より大きく 9999 以下の整数でなくてはなりません。

- DDD はその年の日を表します。367 より小さい正の整数で、かつ指定された年に対して有効であることが前提です。

戻り値は、引数-1 で表される日付が、西暦の 1600 年 12 月 31 日から数えて何日目であるかを示す整数です。

INTEGER-PART

INTEGER-PART 関数は、指定された引数の整数部分の整数を戻します。

関数タイプは整数です。

フォーマット

引数-1
数字クラスでなければなりません。

引数-1 の値がゼロである場合には、戻り値はゼロです。引数-1 の値が正である場合には、戻り値は引数-1 の値以下の最大の整数です。引数-1 の値が負である場合には、戻り値は引数-1 の値以上の最小の整数です。

例えば、

```
FUNCTION INTEGER-PART (+1.5)
```

は、値 +1 を戻し、

```
FUNCTION INTEGER-PART (-1.5)
```

は、値 -1 を戻します。

LENGTH

LENGTH 関数は、引数の長さのバイト数に等しい整数を戻します。関数タイプは整数です。

関数結果は 9 桁の整数です。

フォーマット

引数-1
非数字、ブール、または DBCS のリテラルであっても構いません。つまり、あらゆるクラスまたはあらゆるカテゴリーのデータ項目を指定できるということです。

引数-1 または引数-1 に従属する任意のデータ項目が、OCCURS 文節の DEPENDING 句で記述されている場合は、DEPENDING 句で指定されたデータ名によって参照されるデータ項目の内容が、LENGTH 関数の評価時に使用されます。

引数-1 は、タイプ名またはタイプ名に従属する項目であってもかまいません。

USAGE IS POINTER または USAGE IS PROCEDURE-POINTER で記述されたデータ項目は、LENGTH 関数の引数-1 として使用できます。結果は常に 16 になります。

LENGTH

ADDRESS OF 特殊レジスターまたは LENGTH OF 特殊レジスターは、LENGTH 関数に対する引数-1 として使用できます。結果は、ADDRESS OF オブジェクトまたは LENGTH OF オブジェクトとは無関係に、それぞれが常に 16 と 4 になります。

引数-1 が非数字リテラル、基本データ項目、または可変のオカレンス・データ項目を含まないグループ・データ項目である場合、戻り値は、引数-1 の長さの文字位置の個数に等しい整数です。

引数-1 がヌル終了の非数値リテラルである場合、戻り値は、リテラルの最後のヌル文字を除いたリテラル内の英数字位置の数に等しくなります。1 バイトおよび 2 バイト文字の混合を含むヌル終了の非数値リテラルの長さは、各バイトが 1 バイト文字であるかのようにカウントされます。

IBM Extension

引数-1 が DBCS または国別文字のデータ項目またはリテラルである場合には、戻り値は、引数の長さの DBCS 文字位置または国別文字位置の個数です。例えば、`FUNCTION LENGTH (G"D1D2")` は、値 4 ではなく 2 を戻します。

End of IBM Extension

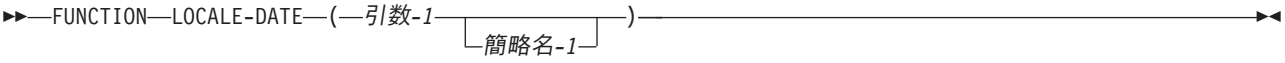
戻り値には、暗黙の FILLER 文字が (もしあれば) 含まれます。

LOCALE-DATE

IBM Extension

LOCALE-DATE 関数は、ロケールによって指定されている、各文化圏に適切な形式の日付を含む文字ストリングを戻します。

関数タイプは英数字です。



引数-1

引数-1 は、英数字または DBCS でなければならず、8 文字位置分の長さをもっていなければなりません。

引数-1 は、CURRENT-DATE 関数によって文字位置 1 ～ 8 に戻される年、月、日の形式と同じ形式の日付でなければなりません。 CURRENT-DATE 関数の詳細は 7-291 ページの『CURRENT-DATE』を参照してください。

簡略名-1

簡略名-1 は SPECIAL-NAMES 段落内のロケールと関連付けられていなければなりません。

戻り値

LOCALE-DATE 組み込み関数の戻り値は次のとおりです。

1. 簡略名-1 が指定されている場合は、簡略名-1 と関連付けられているロケールが日付のフォーマット設定に使用される。その他の場合は現行ロケールが使用されます。簡略名-1 と関連付けられているロケールが利用できない場合は、オペレーティング・システムのエスケープ・メッセージが発行されます。
2. 戻り値は、引数-1 で指定した日付を含む文字ストリングであり、ロケールに指定されている日付形式で戻される。

3. 戻り値の長さはロケールに指定されている形式によって決まる。

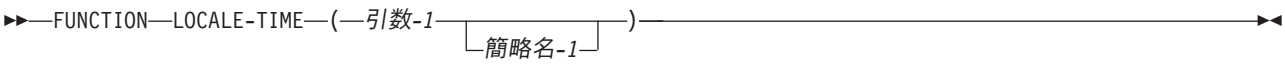
End of IBM Extension

LOCALE-TIME

IBM Extension

LOCALE-TIME 関数は、ロケールによって指定されている、特定の文化圏固有の適切な形式の時刻を含む文字ストリングを戻します。

関数タイプは英数字です。



引数-1

引数-1 は、英数字または DBCS でなければならず、13 文字位置分の長さをもっていなければなりません。

引数-1 の内容は、CURRENT-DATE 関数によって文字位置 9 ～ 21 に戻される時、分、秒の形式と同じ形式でなければなりません。 CURRENT-DATE 関数の詳細は 7-291 ページの『CURRENT-DATE』を参照してください。

簡略名-1

簡略名-1 は SPECIAL-NAMES 段落内のロケールと関連付けられていなければなりません。

戻り値

LOCALE-TIME 組み込み関数の戻り値は次のとおりです。

1. 簡略名-1 が指定されている場合は、時刻のフォーマット設定に使用されるロケールは、簡略名-1 と関連付けられているロケールになる。その他の場合は現行ロケールが使用されます。簡略名-1 と関連付けられているロケールが利用できない場合は、オペレーティング・システムのエスケープ・メッセージが発行されます。
2. 戻り値は、引数-1 によって指定されている時刻の時、分、秒を含む文字ストリングであり、ロケールに指示されている、特定の文化圏固有の適切な形式になっている。この値は、引数-1 の最後の 5 文字の桁に入れられている協定世界時 (グリニッジ標準時) からの相対時間と、ロケールの LC-TOD カテゴリーに指定されている相対時間との差によって調整されます。
3. 戻り値の長さはロケールに指示されている形式によって決まる。

End of IBM Extension

LOG

LOG 関数は、指定された引数の ***e*** を底とする対数 (自然対数) の近似値を戻します。

関数タイプは数字です。

LOG

フォーマット

引数-1
数字クラスでなければなりません。引数-1 の値はゼロより大きくなければなりません。

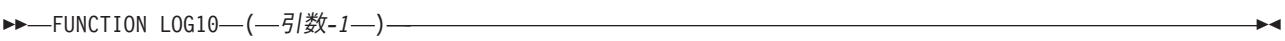
戻り値は、引数-1 の ***e*** を底とする対数の近似値です。

LOG10

LOG10 関数は、指定された引数の 10 を底とする対数の近似値を戻します。

関数タイプは数字です。

フォーマット



引数-1
数字クラスでなければなりません。引数-1 の値はゼロより大きくなければなりません。

戻り値は、引数-1 の 10 を底とする対数の近似値です。

LOWER-CASE

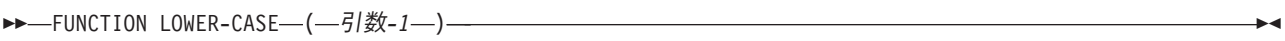
LOWER-CASE 関数は、指定された引数と同じ長さの文字ストリングを、それぞれの大文字を対応する小文字に置き換えて戻します。

関数タイプは、次のように引数のタイプに依存します。

引数タイプ	関数タイプ
英字	英数字
英数字	英数字
DBCS[1]	DBCS[1]

注: [1] IBM 拡張

フォーマット



引数-1
英字クラスまたは英数字クラスで、かつ長さが少なくとも 1 文字はなければなりません。

IBM Extension

引数-1 は、DBCS または国別文字であってもかまいません。

End of IBM Extension

それぞれの大文字が対応する小文字に置き換えられる以外は、引数-1 と同じ文字ストリングが戻されます。プログラム照合順序またはコード・ページは、戻り値に影響を与えません。

IBM Extension

引数-1 が DBCS の場合、DBCS の値は影響を受けません。引数-1 が混合リテラルの場合、そのリテラルの 1 バイトの部分のみが影響を受けます。

End of IBM Extension

戻される文字ストリングは、引数-1 と同じ長さになります。

MAX

MAX 関数は、最大値を含む引数の内容を戻します。

関数タイプは、次のように引数のタイプに依存します。

引数タイプ	関数タイプ
英字	英数字
英数字	英数字
国別	国別
指標	指標
整数であるすべての引数	整数
数字 (整数の引数があってもよい)	数字

フォーマット

引数-1
数字クラス、英数字クラス、英字クラス、または DBCS クラスであってもよく、ブール・クラスであってはいけません。

複数の引数-1 が指定されている場合には、英字の引数と英数字の引数の組み合わせが認められています。引数タイプの他の組み合わせは認められていません。例えば DBCS 引数は、英数字の引数と混合することはできません。

戻り値は、最大値を持つ引数-1 の内容です。最大値を判別するために使用される比較は、単純条件の規則に従って行われます。詳細については 7-10 ページの『条件式』を参照してください。

複数の引数が同じ最大値を持つ場合、その値を持つ左端の引数が戻されます。

関数のタイプが英数字または DBCS である場合、戻り値のサイズは、選択された引数のサイズと同じになります。

MEAN

MEAN 関数は、引数の算術平均の近似値を戻します。

関数タイプは数字です。

MEAN

フォーマット

引数-1
数字クラスでなければなりません。

戻り値は、一連の引数-1 の算術平均値です。戻り値は、一連の引数-1 の合計を、引数-1 によって参照されるオカレンスの個数で除算したものと定義されます。

同等の算術式は、次のようになります。

1. 引数-1 のオカレンスが 1 の場合

   (引数-1)

2. 引数-1 のオカレンスが 2 の場合

   ((引数-1_1 + 引数-1_2) / 2)

3. 引数-1 のオカレンスが n の場合

   ((引数-1_1 + 引数-1_2 + ... + 引数-1_n) / n)

MEDIAN

MEDIAN 関数は、引数をソートされた順序に並べることによって構成されたリストの中央の値を持つ引数の内容を戻します。

関数タイプは数字です。

フォーマット

引数-1
数字クラスでなければなりません。

戻り値は、すべての引数-1 の値をソートされた順序に並べることによって構成されたリストの中央の値を持つ引数-1 の内容を戻します。

引数-1 によって参照されるオカレンスの数が奇数である場合は、引数-1 によって参照されるオカレンスの少なくとも半数が戻り値よりも大きいか等しく、少なくとも半数が戻り値より小さいか等しくなります。引数-1 によって参照されるオカレンスの数が偶数の場合は、戻り値は、中央の 2 つのオカレンスによって参照される値の算術平均値になります。

引数の値をソートされた順序に並べるために使用される比較は、単純条件の規則に従って行われます。詳細については 7-10 ページの『条件式』を参照してください。

MIDRANGE

MIDRANGE 関数は、最小の引数と最大の引数の値の算術平均の近似値を戻します。

関数タイプは数字です。

フォーマット

引数-1
数字クラスでなければなりません。

戻り値は、最大の引数-1 の値と最小の引数-1 の値の算術平均値です。最大値および最小値を判別するために使用される比較は、単純条件の規則に従って行われます。詳細については 7-10 ページの『条件式』を参照してください。

MIN

MIN 関数は、最小値を含む引数の内容を戻します。

関数タイプは、次のように引数のタイプに依存します。

引数タイプ	関数タイプ
英字	英数字
英数字	英数字
国別	国別
指標	指標
整数であるすべての引数	整数
数字 (整数の引数があってもよい)	数字

フォーマット

引数-1
数字クラス、英数字クラス、英字クラス、または DBCS クラスであってもよく、ブール・クラスであってはいけません。

複数の引数-1 が指定されている場合には、英字の引数と英数字の引数の組み合わせが認められています。引数タイプの他の組み合わせは認められていません。例えば DBCS 引数は、英数字の引数と混合することはできません。

戻り値は、最小値を持つ引数-1 の内容です。最小値を判別するために使用される比較は、単純条件の規則に従って行われます。詳細については 7-10 ページの『条件式』を参照してください。

複数の引数-1 が同じ最小値を持つ場合、その値を持つ左端の引数-1 が戻されます。

MIN

関数のタイプが英数字または DBCS である場合、戻り値のサイズは、選択された引数-1 のサイズと同じになります。

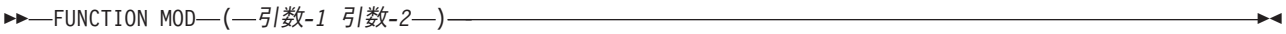
MOD

MOD 関数は、引数-1 モジュロ引数-2 の整数値を戻します。

関数タイプは整数です。

関数結果は、引数-1 と引数-2 のいずれか短い方と同じ桁数を持つ整数です。

フォーマット



引数-1
整数でなければなりません。

引数-2
整数でなければなりません。ゼロであってはなりません。

戻り値は、引数-1 モジュロ引数-2 です。戻り値は次のように定義されます。

引数-1 - (引数-2 * FUNCTION INTEGER (引数-1 / 引数-2))

以下に、引数-1 と引数-2 のいくつかの値に対して期待される結果を示します。

引数-1	引数-2	戻り
11	5	1
-11	5	4
11	-5	-4
-11	-5	-1

NATIONAL-OF

NATIONAL-OF 関数は、引数-1 内の UCS-2 表現の文字で構成される国別文字ストリングを戻します。関数のタイプは国別です。

形式 1: ソース CCSID の指定

引数-1
クラス英字、英数字、または DBCS でなければなりません。引数-1 は、変換のソース・ストリングを識別します。

IBM Extension

引数-2

整数でなければなりません。引数-2 は、変換のソース・コード・ページを識別します。引数-2 は、EBCDIC、 ASCII、UTF-8、または EUC コード・ページを識別する有効な CCSID 番号でなければなりません。EBCDIC または ASCII CCSID は、SBCS、DBCS、または混合 SBCS/DBCS であるコード・ページを識別できます。

引数-2 が省略された場合、ソース・コード・ページは、ソース・コードのコンパイル時に CCSID コンパイラー・オプション用に有効になるソース・コード・ページです。ソース・コード・ページが 65535 である場合、デフォルトの CCSID 37 が使用されます。

End of IBM Extension

戻り値は、国別文字表現 (国別 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) に変換される引数-1 の文字で構成される国別文字ストリングです。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の『変換と精度』を参照してください。

ソース文字を国別文字に変換できない場合、ソース文字は、システム定義の置換文字 X'FFFD' に変換されます。例外条件は発生しません。

戻り値の長さは、引数-1 のコンテントおよびソース・コード・ページの特性によって異なります。

例外: 変換が失敗すると、重大なランタイム・エラーが発生します。ソース CCSID からターゲット CCSID (国別 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) への変換がオペレーティング・システムでサポートされていることを確認してください。

形式 2: ユーザー置換文字の指定

引数-1

クラス英字、英数字、または DBCS でなければなりません。引数-1 は、変換のソース・ストリングを識別します。

引数-3

長さが 1 文字位置の国別リテラルまたは国別データ項目でなければなりません。

引数-3 は、対応する国別文字を持たない英数字の変換で使用される国別置換文字を指定します。

ソース・コード・ページは、ソース・コードのコンパイル時に CCSID コンパイラー・オプション用に有効になるソース・コード・ページです。CCSID コンパイラー・オプションが 65535 である場合、デフォルトの CCSID 37 が使用されます。

戻り値は、国別文字表現 (国別 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) に変換される引数-1 の文字で構成される国別文字ストリングです。ソース文字を国別文字に変換できない場合、ソース文字は、ユーザー置換文字引数-3 に変換されます。例外条件は発生しません。

戻り値の長さは、引数-1 のコンテントおよびソース・コード・ページの特性によって異なります。

例外: 変換が失敗すると、重大なランタイム・エラーが発生します。ソース ccsid からターゲット CCSID (UCS-2) への変換がオペレーティング・システムでサポートされていることを確認してください。

NUMVAL

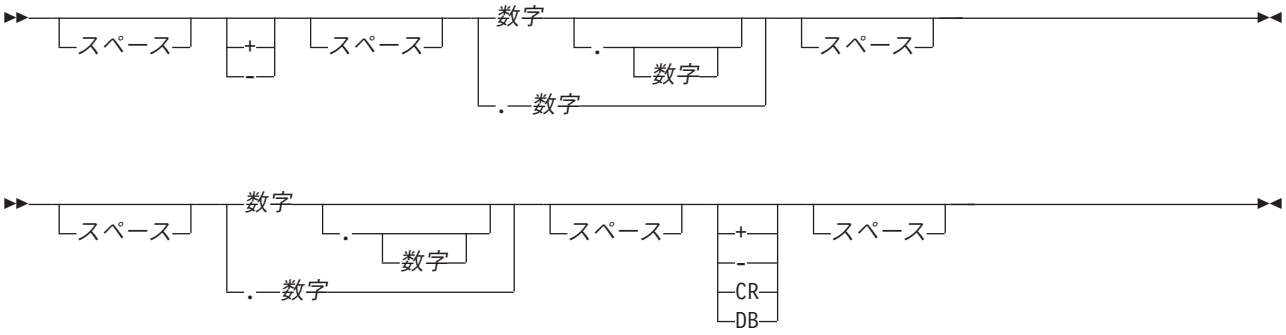
NUMVAL 関数は、引数で指定された英数字ストリングによって表される数字を戻します。この関数は、ストリングの前後のブランクをすべて取り除き、算術式で使用できる数字を生成します。

関数タイプは数字です。

フォーマット

引数-1

次の形式のいずれかの内容を持つ、非数字リテラルまたは英数字データ項目でなくてはなりません。



スペース

1 つまたは複数のスペースのストリング

数字

1 つまたは複数の数字のストリング。合計桁数が 18 を超えてはなりません。

SPECIAL-NAMES 段落に DECIMAL-POINT IS COMMA 文節を指定している場合、引数-1 では、小数点ではなくコンマを使用しなくてはなりません。

戻り値は、引数-1 で表される数値の浮動小数点の近似値になります。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の『変換と精度』を参照してください。

NUMVAL-C

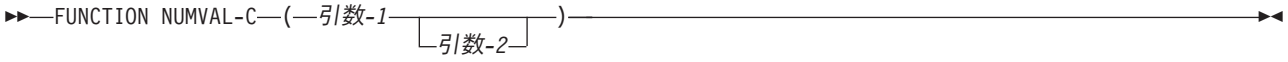
NUMVAL-C 関数は、引数-1 として指定された英数字ストリングによって表される数字を戻します。引数-2 によって指定されるすべてのオプショナルの通貨記号、および小数点の前にあるすべてのオプショナルのコンマは取り除かれ、算術式で使用できる数字を生成します。

NUMVAL-C 関数は、以下の条件で指定できない場合があります。

- プログラム内に複数の CURRENCY SIGN 文節が指定されている。
- WITH PICTURE SYMBOL 句が CURRENCY SIGN 文節に指定されている。
- 通貨記号に小文字が指定されている。

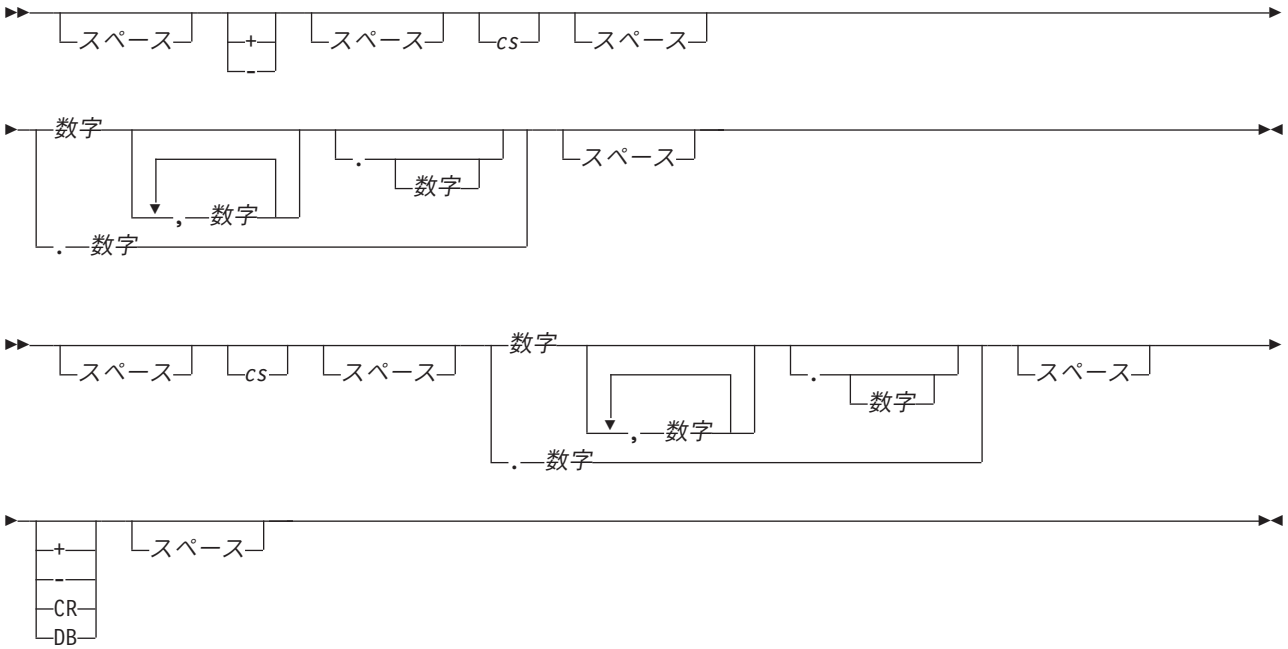
関数タイプは数字です。

フォーマット



引数-1

次の形式のいずれかの内容を持つ、非数字リテラルまたは英数字データ項目でなくてはなりません。



スペース

1 つまたは複数のスペースのストリング

cs 引数-2 によって指定される 1 つまたは複数の文字のストリング。 cs によって指定される文字のコピーは、引数-1 に最大で 1 つ現れることができます。

数字

1 つまたは複数の数字のストリング。合計桁数が 18 を超えてはなりません。

SPECIAL-NAMES 段落に DECIMAL-POINT IS COMMA 文節を指定した場合、引数-1 のコンマの機能と小数点の機能は逆になります。

引数-2

指定する場合は、非数字リテラルまたは英数字データ項目でなくてはならず、次の規則にしたがいます。

- 0 ～ 9 のどの数字も含んではならず、前または後ろにスペースが付いていても、+ - . , のどの特殊文字が付いていてもいけません。
- 基本データ項目またはグループ・データ項目として有効な、ゼロを含む任意の長さを取ることができます。
- 引数-2 のマッチングでは、大文字小文字の区別をします。例えば、引数-2 を 'Dm' として指定した場合、'DM'、'dm'、または 'dM' とは一致しません。

引数-2 を指定しない場合、cs として使用される文字は、そのプログラムに対して指定された通貨記号になります。

NUMVAL-C

戻り値は、引数-1 で表される数値の浮動小数点の近似値になります。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の『変換と精度』を参照してください。

ORD

ORD 関数は、その引数のプログラムの照合順序における順序位置を整数値で戻します。最下位の順序位置は 1 です。

関数タイプは整数です。

フォーマット

引数-1
長さが 1 文字でなければならず、英字クラスまたは英数字クラスでなければなりません。

戻り値は、引数-1 の、プログラムの照合順序における順序位置です。戻り値の範囲は 1 ～ 256 で、照合順序に依存します。

ORD-MAX

ORD-MAX 関数は、最大値を含む引数の、引数リストにおける順序数の位置である値を戻します。

関数タイプは整数です。

フォーマット

引数-1
数字クラス、英数字クラス、または英字クラスであってもよく、ブール・クラスであってはいけません。

複数の引数-1 が指定されている場合、すべての引数は同じクラスでなくてはなりませんが、英字の引数と英数字の引数との組み合わせだけは例外として認められています。

戻り値は、一連の引数-1 の中で最大値を持つ引数-1 の位置に相当する順序数です。

最大値を持つ引数-1 を判別するために使用される比較は、単純条件の規則に従って行われます。詳細については 7-10 ページの『条件式』を参照してください。

複数の引数-1 が同じ最大値を持つ場合、戻される数は、その値を持つ左端の引数-1 の位置に相当します。

ORD-MIN

ORD-MIN 関数は、最小値を含む引数の、順序数である値を戻します。

関数タイプは整数です。

フォーマット

引数-1
数字クラス、英数字クラス、または英字クラスであってもよく、ブール・クラスであってはいけません。

複数の引数-1 が指定されている場合、すべての引数は同じクラスでなくてはなりませんが、英字の引数と英数字の引数との組み合わせだけは例外として認められています。

戻り値は、一連の引数-1 における最小値を持つ引数-1 の位置に相当する順序数です。

最小値を持つ引数-1 を判別するために使用される比較は、単純条件の規則に従って行われます。詳細については 7-10 ページの『条件式』を参照してください。

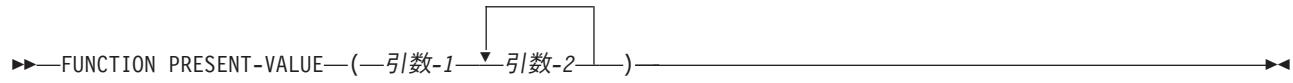
複数の引数-1 が同じ最小値を持つ場合、戻される数は、その値を持つ左端の引数-1 の位置に相当します。

PRESENT-VALUE

PRESENT-VALUE 関数は、引数-1 で指定した減価率に基づいて、引数-2 で指定した将来の各期末の総額に対する現在価値を近似値で戻します。

関数タイプは数字です。

フォーマット



引数-1
数字クラスでなければなりません。-1 より大きな値でなければなりません。

引数-2
数字クラスでなければなりません。

戻り値と等しい算術式は、次のようになります。

1. 引数-2 のオカレンスが 1 の場合

   (引数-2 / (1 + 引数-1))

2. 引数-2 のオカレンスが 2 の場合

   (引数-2_1 / (1 + 引数-1) + 引数-2_2 / (1 + 引数-1)**2)

3. 引数-2 のオカレンスが n の場合

   FUNCTION SUM (引数-2_1 / (1 + 引数-1) ... 引数-2_n / (1 + 引数-1)**n)

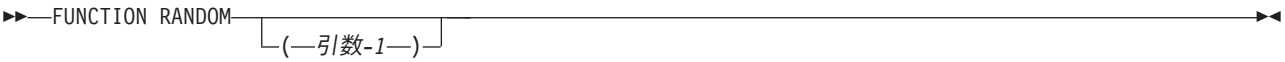
引数-2 の各オカレンスに対して 1 つの項があります。指数 n は、一連の項のそれぞれに対して 1 ずつ増加します。

RANDOM

RANDOM 関数は、一様分布からの疑似乱数の数字を戻します。

関数タイプは数字です。

フォーマット



引数-1

引数-1 が指定された場合、引数-1 はゼロまたは正の整数でなくてはならず、PIC9(18) 固定項目で指定可能な最大値である (10**18)-1 以下の値を持ちます。ただし、ゼロ以上 2,147,483,645 以下の値を持つ引数-1 のみが疑似乱数の特異な順序を生成します。

次の参照で引数-1 を指定する場合は、疑似乱数の新たな順序が開始されます。

実行単位におけるこの関数への最初の参照で引数-1 を指定しない場合、使用されるシード値はゼロになります。

それぞれの場合において、次に引数-1 を指定しないで参照すると、現行の順序における次の数が戻ります。

戻り値は、ゼロと 1 の間に限定されます。

与えられたシード値に対して、疑似乱数の順序は常に同じになります。

RANGE

RANGE 関数は、最大の引数の値から最小の引数の値を引いた値を戻します。

関数タイプは、次のように引数のタイプに依存します。

引数タイプ	関数タイプ
整数であるすべての引数	整数
数字 (整数の引数があってもよい)	数字

フォーマット

引数-1

数字クラスでなければなりません。

戻り値は、最大値を持つ引数-1 から最小値を持つ引数-1 を引いた値に等しくなります。最大値および最小値を判別するために使用される比較は、単純条件の規則に従って行われます。詳細については 7-10 ページの『条件式』を参照してください。

RANGE 関数と同等の算術式は、次のようになります。

```
(FUNCTION MAX (argument-1) - FUNCTION MIN (argument-1))
```

REM

REM 関数は、引数-1 を引数-2 で除算した場合の剰余にあたる数字を戻します。

関数タイプは数字です。

フォーマット

引数-1
数字クラスでなければなりません。

引数-2
数字クラスでなければなりません。ゼロであってはなりません。

戻り値は、引数-1 を引数-2 で除算した場合の剰余です。戻り値は、次の式で定義されます。

```
argument-1 - (argument-2 * FUNCTION INTEGER-PART (argument-1/argument-2))
```

REVERSE

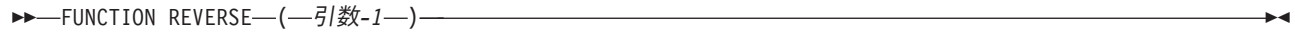
REVERSE 関数は、引数とまったく同じ長さの文字ストリングで、引数で指定されている文字と順序が逆であること以外はまったく同じ文字を持つ文字ストリングを戻します。

関数タイプは、次のように引数のタイプに依存します。

引数タイプ	関数タイプ
英字	英数字
英数字	英数字
DBCS[1]	DBCS[1]

注: [1] IBM 拡張

フォーマット



引数-1
英字クラスまたは英数字クラスで、かつ長さが少なくとも 1 文字はなければなりません。

IBM Extension

引数-1 は、DBCS または国別文字であってもかまいません。

End of IBM Extension

引数-1 が長さが n の文字ストリングである場合、戻り値は長さ n の文字ストリングで、例えば 1 <= j <= n の場合、戻り値の j の位置の文字は、引数-1 の n-j+1 の位置の文字になります。

SIN

SIN 関数は、引数のラジアンで指定した角度または弧のサインの近似値を戻します。

関数タイプは数字です。

フォーマット

引数-1
数字クラスでなければなりません。

戻り値は引数-1 のサインの近似値であり、-1 以上 +1 以下の値になります。

SQRT

SQRT 関数は、指定された引数の平方根の近似値を戻します。

関数タイプは数字です。

フォーマット

引数-1
数字クラスでなければなりません。引数-1 の値は、ゼロまたは正でなければなりません。

戻り値は、引数-1 の平方根の近似値の絶対値になります。

STANDARD-DEVIATION

STANDARD-DEVIATION 関数は、その引数の標準偏差の近似値を戻します。

関数タイプは数字です。

フォーマット

引数-1
数字クラスでなければなりません。

戻り値は、一連の引数-1 の標準偏差の近似値です。戻り値は次のように計算されます。

1. 各引数-1 と一連の引数-1 の算術平均との差を求め、2 乗する。
2. 得られた値をすべて加算する。この値を一連の引数-1 の個数で除算する。
3. 得られた商の平方根を求める。戻り値は、この平方根の絶対値になります。

一連の引数-1 がただ 1 つの値で構成されている場合、あるいは一連の引数-1 がすべて可変のオカレンス・データ項目で構成されていて、それらすべてのデータ項目に関してオカレンスの合計数が 1 である場合、戻り値はゼロになります。

FUNCTION STANDARD-DEVIATION と同等の算術式は、次のようになります。

```
FUNCTION SQRT (FUNCTION VARIANCE (argument-1))
```

SUBTRACT-DURATION

IBM Extension

SUBTRACT-DURATION 関数は、日付、時刻、またはタイム・スタンプの項目に期間を加算し、その変更した項目を戻します。

関数タイプは日時です。

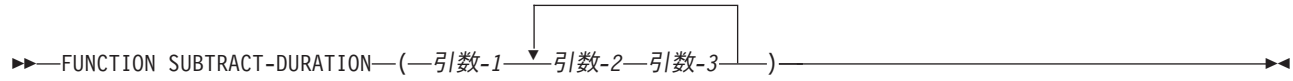
戻り値の長さは、引数-1 に指定する日付項目、時刻項目、またはタイム・スタンプ項目の長さによって決まります。戻り値は、引数-1 の長さまで切り捨てられます。

日付項目から期間を減算する場合は、戻される日付は以下に示す特定の範囲内の値でなければなりません。

- 4 桁の日付の場合は 0001/01/01 から 9999/12/31 までの範囲内でなければならない。
- 2 桁の日付の場合は 0001/01/01 から 9999/12/31 までの範囲内でなければならない。ただし、年の部分は 2 桁に切り捨てられます。
- 3 桁の年 (1 桁の世紀と 2 桁の年) の場合は 1900/01/01 から 2899/12/31 (デフォルト) までの範囲内でなければならない。この範囲は DATTIM PROCESS ステートメント・オプションを指定することによって変更できます。

2 桁の日付項目から期間を減算する場合の戻り値の範囲は 4 桁の年の場合と同じです。ただし、戻り値の年の部分は 2 桁に切り捨てられます。

フォーマット



引数-1

日付、時刻、またはタイム・スタンプの項目でなければなりません。

引数-1 は、期間を減算される値です。期間は引数-2 と引数-3 に指定します。

引数-2

引数-2 は期間を表すキーワードです。有効な期間を以下に示します。

- YEARS
- MONTHS
- DAYS
- HOURS
- MINUTES
- SECONDS
- MICROSECONDS
- PICOSECONDS

使用する期間キーワードまたは変換指定子は、引数-1 と整合性がとれている必要があります。例えば、期間キーワードは以下の規則に従わなければなりません。

1. YEARS、MONTHS、および DAYS は、日付項目またはタイム・スタンプ項目からのみ減算することができる。
2. HOURS、MINUTES、SECONDS、および MICROSECONDS は、時刻項目またはタイム・スタンプ項目からのみ減算することができる。
3. PICOSECONDS は、タイム・スタンプ項目からのみ減算できる。

引数-3

整数算術式でなければなりません。引数-3 は、引数-2 で指定した期間の単位数であり、引数-1 から減算されます。

引数-2 および引数-3 は反復可能です。ただし、1 つの組み込み関数の中に重複する引数-2 があってはなりません。

引数-3 は負の整数でも構いませんが、この関数が取るのはその絶対値だけです。引数-3 が 9 桁よりも長い場合は切り捨てが行われます。

日付から期間を減算した結果が無効となった場合は、その日付に対する調整が行われます。例えば、1997 年 5 月 31 日という日付から期間 1 月を減算すると、その結果は 1997 年 4 月 31 日という無効な日付となります。このような場合、この日付は 1997 年 4 月 30 日という有効な日付に調整されます。

例

以下の例で SUBTRACT-DURATION 組み込み関数の使用法を示します。

```
MOVE FUNCTION SUBTRACT-DURATION (date-1 MONTHS 1)
     TO date-2.
MOVE FUNCTION SUBTRACT-DURATION (date-2 MONTHS 1 + 2 * 3)
     TO date-1.
MOVE FUNCTION SUBTRACT-DURATION (date-3 MONTHS 5 DAYS 1000)
     TO date-1.
```

End of IBM Extension

SUM

SUM 関数は、引数の合計値を戻します。

関数タイプは、次のように引数のタイプに依存します。

引数タイプ	関数タイプ
整数であるすべての引数	整数
数字 (整数の引数があってもよい)	数字

フォーマット

引数-1

数字クラスでなければなりません。

戻り値は、引数の和になります。一連の引数-1 がすべて整数である場合、戻り値は整数になります。一連の引数-1 がすべて整数でない場合、ある数字が戻されます。

同等の算術式は、次のようになります。

1. 引数-1 のオカレンスが 1 の場合

   (引数-1)

2. 引数-1 のオカレンスが 2 の場合

   ($引数\text{-}1_1$ + $引数\text{-}1_2$)

3. 引数-1 のオカレンスが n の場合

   ($引数\text{-}1_1$ + $引数\text{-}1_2$ + ... + $引数\text{-}1_n$)

TAN

TAN 関数は、引数のラジアンで指定した角度または弧のタンジェントの近似値を戻します。

関数タイプは数字です。

フォーマット

引数-1
数字クラスでなければなりません。

戻り値は、引数-1 のタンジェントの近似値です。

TEST-DATE-TIME

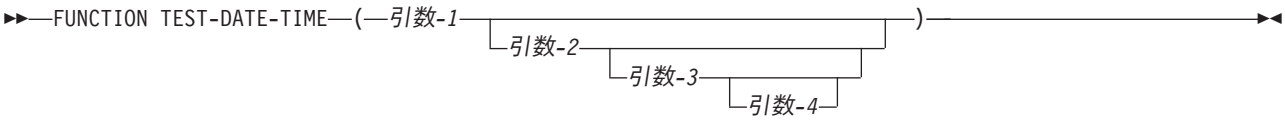
IBM Extension

TEST-DATE-TIME 関数は、日付、時刻、またはタイム・スタンプの項目、英数字項目、数字パックまたは数字ゾーンの項目をとり、その日付、時刻、またはタイム・スタンプが有効であるかどうかを判別します。有効な項目の場合は真 (B'1') を、無効な項目の場合は偽 (B'0') を戻します。

関数タイプはブールです。

戻り値の長さは 1 バイトです。

フォーマット



引数-1
リテラル-3 は、下記のものになります。

- 日付、時刻、またはタイム・スタンプの項目

- 英数字クラスの項目
- 非数字リテラル
- 整数値クラスの項目

引数-1 が日付、時刻、またはタイム・スタンプの項目である 場合は、引数-2 から引数-4 はオプショナルです。引数-1 が日付、時刻、またはタイム・スタンプの項目ではない 場合は、引数-2 を指定する必要があります (引数-3 および引数-4 はオプショナルです)。

引数-1 は、それが有効な項目であるかどうかを調べるために、そのタイプまたは引数-2 から引数-4 に基づいてテストされます。

引数-2

以下のキーワードのいずれか 1 つでなければなりません。

- DATE
- TIME
- TIMESTAMP

引数-2 が TIMESTAMP の場合、引数-3 は、FORMAT OF 特殊レジスターでのみ指定でき、引数-4 は指定できません。

引数-3

日付項目または時刻項目の形式を指定します。以下のものでなければなりません。

- 長さが最小でも 2 文字の非数字リテラル
- キーワード LOCALE
- FORMAT OF 特殊レジスター

有効なリテラルのリストが必要な場合は SPECIAL-NAMES FORMAT 文節を参照してください。

引数-3 がキーワード LOCALE である場合は、日付または時刻の形式は LOCALE に基づきます。引数-4 を指定しないと、現行ロケールが使用されます。その他の場合は、簡略名または LOCALE OF 特殊レジスターと関連付けられているロケールが使用されます。

引数-3 を指定しない場合は、テストには SPECIAL-NAMES FORMAT 文節内で定義した形式が使用されます。 TIMESTAMP では、引数-3 が指定されていない場合、デフォルトの形式 @Y-%m-%d-%H%M%S.@Sm が使用されます。

引数-4

LOCALE と関連付けられている簡略名または LOCALE OF 特殊レジスターでなければなりません。

引数-4 が従わなければならない規則を以下に示します。

- 引数-4 が指定されており、かつ引数-3 がロケールに基づく形式リテラルである (例えば %p を含んでいる) 場合は、そのロケールに基づく形式リテラルは引数-4 で指定されているロケールを使用して変換指定子の実際の値を判別する。
- 引数-3 がロケールに基づく形式リテラルであり (例えば %p を含んでいる)、かつ引数-4 が指定されていない場合は、そのロケールに基づく形式リテラルは現行ロケールを使用して変換指定子の実際の値を判別する。
- 引数-3 がロケールに基づく形式リテラルであり (例えば %p を含んでいる)、かつ LOCALE OF 特殊レジスターが非ロケール項目を参照するために使用されている場合は、そのロケールに基づく形式リテラルはデフォルトのロケールを使用して変換指定子の実際の値を判別する。

例

以下の例で TEST-DATE-TIME 組み込み関数の使用法を示します。

```
WORKING-STORAGE SECTION.
01 mydate1 PIC X(8) VALUE '07312013'.

PROCEDURE DIVISION.

IF FUNCTION TEST-DATE-TIME (mydate1 DATE '%d%m%Y') = B'1'
   DISPLAY 'Valid Date'
END-IF.
```

End of IBM Extension

TRIM

IBM Extension

TRIM 関数は、与えられたストリングを前後のブランクを除去して、または、与えられたストリングを前後の指定した文字を除去して戻します。

関数のタイプはその引数のクラスに応じて英数字、DBCS または国別です。

形式:

引数-1
非数値リテラル、またはクラス英字、英数字、DBCS または国別のデータ項目でなければなりません。引数-1 は、トリムのソース・ストリングを識別します。

引数-2
指定された場合、非数値リテラル、または引数-1 と同じクラスのデータ項目でなければなりません。切り取る文字を指定します。指定しない場合、トリム文字がブランクにデフォルト設定されます。

引数-2 が指定されない場合、戻り値は、すべての前後ブランクを除去した引数-1 の文字で構成される英数字、DBCS または国別文字ストリングです。ブランク文字は、引数-1 がクラス英数字である場合、1 バイト・スペース文字 (' ' または X'40')、引数-1 がクラス DBCS である場合、2 バイト・スペース (X'4040')、または引数-1 がクラス国別の場合、国別スペース (X'0020' または X'3000') です。

引数-2 が指定された場合、引数-2 のすべての文字はストリングの両側から切り取られます。戻り値は、引数-2 で指定したすべての前後文字を除去した引数-1 の文字で構成される英数字、DBCS または国別文字ストリングです。

戻されるストリングの長さは、コンテントおよび引数-1 のクラスによって異なります。文字位置の数で戻されるストリングの長さになります。引数-1 が DBCS または国別データ項目である場合には、長さは、DBCS 文字位置または国別文字位置の数です。

戻り値

引数-2 パラメーターの文字の順序は、操作の結果に影響しません。 その文字列は、1 つずつの文字のリストとみなされます。例えば、`FUNCTION TRIML(fld, "abc")` は、'a'、'b'、または 'c' でない任意の文字で始まる fld のサブストリングを戻します。fld が "caxyz" を含んでいる場合、`FUNCTION TRIM(fld, "abc")` は、"xyz" を戻します。文字は、2 番目のパラメーターで 2 回、エラーなしで表示することができます。例えば、`FUNCTION TRIM(fld, "aba")` は、有効です。これは、`FUNCTION TRIM(fld, "ab")` と同じ意味です。

TRIM

FUNCTION TRIM、TRIML または TRIMR の 2 番目のパラメーターが指定された場合、ブランクが引数-2 の一部として使用されない限り、ブランクは切り取られません。TRIM、 TRIML および TRIMR 関数は、混合 SBCS/DBCS ストリングであることを識別せず、引数-1 および引数-2 は両方とも、そのクラスが英数字である場合に SBCS として扱われます。

例:

```
FUNCTION TRIM("xxxABxCxxx", "x") // returns 'ABxC'
FUNCTION TRIMR(">>>>ABC<<<<<", "<>") // returns '>>>>ABC'
MOVE "xyz" TO tc.
FUNCTION TRIML("xxyyzzyyzzABCxyzyxzxy", tc) // returns 'ABCxyzyxzxy'
```

End of IBM Extension

TRIML

IBM Extension

TRIML 関数は、与えられたストリングを先行ブランクを除去して、または、与えられたストリングを先行の指定した文字を除去して戻します。

関数のタイプはその引数のクラスに応じて英数字、DBCS または国別です。

形式:

引数-1
非数値リテラル、またはクラス英字、英数字、DBCS または国別のデータ項目でなければなりません。引数-1 は、トリムのソース・ストリングを識別します。

引数-2
指定された場合、非数値リテラル、または引数-1 と同じクラスのデータ項目でなければなりません。切り取る文字を指定します。指定しない場合、トリム文字がブランクにデフォルト設定されます。

引数-2 が指定されない場合、戻り値は、すべての先行ブランクを除去した引数-1 の文字で構成される英数字、DBCS または国別文字ストリングです。ブランク文字は、引数-1 がクラス英数字である場合、1 バイト・スペース文字 (' ' または X'40')、引数-1 がクラス DBCS である場合、2 バイト・スペース (X'4040')、または引数-1 がクラス国別の場合、国別スペース (X'0020' または X'3000') です。

引数-2 が指定された場合、戻り値は、引数-2 で指定したすべての先行文字を除去した引数-1 の文字で構成される英数字、DBCS または国別文字ストリングです。

戻されるストリングの長さは、コンテントおよび引数-1 のクラスによって異なります。文字位置の数で戻されるストリングの長さになります。引数-1 が DBCS または国別データ項目である場合には、長さは、DBCS 文字位置または国別文字位置の数です。

戻り値および例について詳しくは、 7-321 ページの『TRIM』を参照してください。

End of IBM Extension

TRIMR

IBM Extension

TRIMR 関数は、与えられたストリングを末尾ブランクを除去して、または、与えられたストリングを末尾の指定した文字を除去して戻します。

関数のタイプはその引数のクラスに応じて英数字、DBCS または国別です。

形式:

引数-1
非数値リテラル、またはクラス英字、英数字、DBCS または国別のデータ項目でなければなりません。引数-1 は、トリムのソース・ストリングを識別します。

引数-2
指定された場合、非数値リテラル、または引数-1 と同じクラスのデータ項目でなければなりません。切り取る文字を指定します。指定しない場合、トリム文字がブランクにデフォルト設定されます。

引数-2 が指定されない場合、戻り値は、すべての末尾のブランクを除去した引数-1 の文字で構成される英数字、DBCS または国別文字ストリングです。ブランク文字は、引数-1 がクラス英数字である場合、1 バイト・スペース文字 (' ' または X'40')、引数-1 がクラス DBCS である場合、2 バイト・スペース (X'4040')、または引数-1 がクラス国別の場合、国別スペース (X'0020' または X'3000') です。

引数-2 が指定された場合、戻り値は、引数-2 で指定したすべての末尾の文字を除去した引数-1 の文字で構成される英数字、DBCS または国別文字ストリングです。

戻されるストリングの長さは、コンテントおよび引数-1 のクラスによって異なります。文字位置の数で戻されるストリングの長さになります。引数-1 が DBCS または国別データ項目である場合には、長さは、DBCS 文字位置または国別文字位置の数です。

戻り値および例について詳しくは、 7-321 ページの『TRIM』を参照してください。

End of IBM Extension

UPPER-CASE

UPPER-CASE 関数は、指定された引数と同じ長さの文字ストリングを、それぞれの小文字を対応する大文字に置き換えて戻します。

関数タイプは、次のように引数のタイプに依存します。

引数タイプ	関数タイプ
英字	英数字
英数字	英数字
DBCS[1]	DBCS[1]

注: [1] IBM 拡張

UPPER-CASE

フォーマット

引数-1
英字クラスまたは英数字クラスで、かつ長さが少なくとも 1 文字はなければなりません。

IBM Extension

引数-1 は、DBCS または国別文字であってもかまいません。

End of IBM Extension

それぞれの小文字が対応する大文字に置き換えられる以外は、引数-1 と同じ文字ストリングが戻されます。プログラム照合順序またはコード・ページは、戻り値に影響を与えません。

IBM Extension

引数-1 が DBCS の場合、DBCS の値は影響を受けません。引数-1 が混合リテラルの場合、そのリテラルの 1 バイトの部分のみが影響を受けます。

End of IBM Extension

戻される文字ストリングは、引数-1 と同じ長さになります。

UTF8STRING

IBM Extension

UTF8STRING 関数は、指定された引数を対応する UTF-8 ストリングに変換します。戻されるストリングは可変長です。この関数によって戻される受け取り引数について、十分な長さを見込んでおくようにしてください。戻り値の最大長は、元の引数の 2 倍の長さです。

関数タイプは英数字です。

フォーマット

引数-1
英字、英数字、DBCS、または国別文字であり、長さは 1 文字以上でなければなりません。

End of IBM Extension

VARIANCE

VARIANCE 関数は、その引数の平方偏差の近似値を戻します。

関数タイプは数字です。

フォーマット

引数-1
数字クラスでなければなりません。

戻り値は、一連の引数-1 の平方偏差の近似値になります。

戻り値は、一連の引数-1 の標準偏差の 2 乗として定義されます。この値は、次のように計算されます。

1. 各引数-1 と一連の引数-1 の算術平均との差を求め、2 乗する。
2. 得られた値をすべて加算する。この値を一連の引数-1 の個数で除算する。

一連の引数-1 がただ 1 つの値で構成されている場合、あるいは一連の引数-1 がすべて可変のオカレンス・データ項目で構成されていて、それらすべてのデータ項目に関してオカレンスの合計数が 1 である場合、戻り値はゼロになります。

FUNCTION VARIANCE と同等の算術式は、次のようになります。

1. 引数-1 のオカレンスが 1 の場合

   (0)
2. 引数-1 のオカレンスが 2 の場合

   (((引数-1_1 - FUNCTION MEAN (引数-1)) ** 2 + (引数-1_2 - FUNCTION MEAN (引数-1)) ** 2) / 2)
3. 引数-1 のオカレンスが n の場合

   (FUNCTION SUM (((引数-1_1 - FUNCTION MEAN (引数-1)) ** 2) ... ((引数-1_n - FUNCTION MEAN (引数-1)) ** 2)) / n)

WHEN-COMPILED

WHEN-COMPILED 関数は、プログラムのコンパイルされた日付と時刻を、そのプログラムをコンパイルしたシステムから提供されて戻します。

関数タイプは英数字です。

フォーマット

左から右へ読んだ場合、戻り値の 21 の文字位置は次のように解釈します。

文字 位置	内容
1 ～ 4	西暦の年を表す 4 桁の数字
5 ～ 6	01 ～ 12 の範囲の月を表す 2 桁の数字
7 ～ 8	01 ～ 31 の範囲の日を表す 2 桁の数字
9 ～ 10	00 ～ 23 の範囲の時を表す 2 桁の数字

文字 位置	内容
11 ～ 12	00 ～ 59 の範囲の分を表す 2 桁の数字
13 ～ 14	00 ～ 59 の範囲の秒を表す 2 桁の数字
15 ～ 16	00 ～ 99 の範囲の 1/100 秒を表す 2 桁の数字
17	文字 '-' または文字 '+'。前の文字位置で示されているローカル時刻がグリニッジ標準時より遅れている場合、文字 '-' が戻されます。示されているローカル時刻がグリニッジ標準時と同じかそれより進んでいる場合、文字 '+' が戻されます。
18 ～ 19	文字位置 17 が '-' である場合には、00 ～ 12 の範囲の 2 桁の数字が戻され、通知された時刻がグリニッジ標準時よりその時間数だけ遅れていることを示します。文字位置 17 が '+' である場合には、00 ～ 13 の範囲の 2 桁の数字が戻され、通知された時刻がグリニッジ標準時よりその時間数だけ進んでいることを示します。
20 ～ 21	00 ～ 59 の範囲の 2 桁の数字が戻され、通知された時刻のグリニッジ標準時からの進み、または遅れの分数を示します。進んでいるか遅れているかは、文字位置 17 が '+' であるか '-' であるかによって決まります。

戻り値は、この関数を含むソース・プログラムがコンパイルされた日付と時刻です。 ILE COBOL の場合、日付と時刻はコンパイルの開始時に計算され、各リスト・ページのヘッダー行、DATE-COMPILED 段落、および WHEN-COMPILED 特殊レジスターに入れられます。そのプログラムが、含まれるプログラムである場合、戻り値は、そのプログラムを含む、別にコンパイルされたプログラムのコンパイル日時になります。

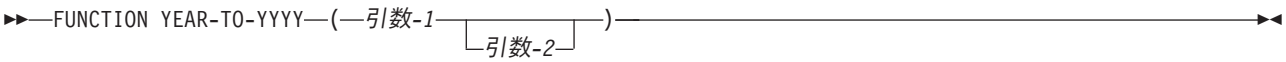
YEAR-TO-YYYY

IBM Extension

YEAR-TO-YYYY 関数は、引数-1 で与えられている、下位 2 桁の年を 4 桁の年へ変換します。実行時に引数-2 が年に加算された場合、引数-2 は 100 年間隔の終了年、すなわち引数-1 の年が当てはまるスライド・ウィンドウを定義します。

関数のタイプは整数です。

フォーマット



引数-1

100 より小さい、負でない整数でなくてはなりません。

引数-2

整数でなければなりません。引数-2 が省略された場合、関数は引数-2 の値を 50 として評価します。実行時に、引数-2 と年との和は、10000 より小さく、1699 より大きくなります。

コンパイラーが FUNCTION YEAR-TO-YYYY を計算するためには、スライド・ウィンドウの最終年 (すなわち「最大年」) を最初に計算することが必要です。「最大年」は、次のように計算されます。

```
(FUNCTION NUMVAL(FUNCTION CURRENT-DATE(1:4)) + 引数-2)
```

上記の「最大年」の場合、FUNCTION YEAR-TO-YYYY は 2 つの算術式のうちの 1 つと等しくなりますが、どちらと等しくなるかは次の条件によって決まります。

```
maximum-year modulus 100 >= argument-1
```

この条件が真の場合、同等の算術式は次のようになります。

```
(argument-1 + 100 * (truncated integer value of (maximum-year/100)))
```

偽の場合、同等の算術式は次のようになります。

```
(引数-1 + 100 * (最大年/100 の切り捨て整数値 - 1))
```

YEAR-TO-YYYY 関数は、スライド・ウィンドウのアルゴリズムを使用します。 YEAR-TO-YYYY 関数を固定ウィンドウに使用するためには、引数-2 は次のように指定できます。ここで固定最大年は、固定 100 年間隔での最大年です。

固定ウィンドウが 1973 ～ 2072 の場合、2009 年には引数-2 の値は 63 になり、2019 年には、値は 53 になります。

```
(固定最大年 - FUNCTION NUMVAL (FUNCTION CURRENT-DATE (1:4)))
```

例

1995 年に対して次の戻り値は、

```
FUNCTION YEAR-TO-YYYY(4, 23)
```

2004 になります。

2008 年に対して次の戻り値は、

```
FUNCTION YEAR-TO-YYYY(98,(-15))
```

1898 になります。

End of IBM Extension

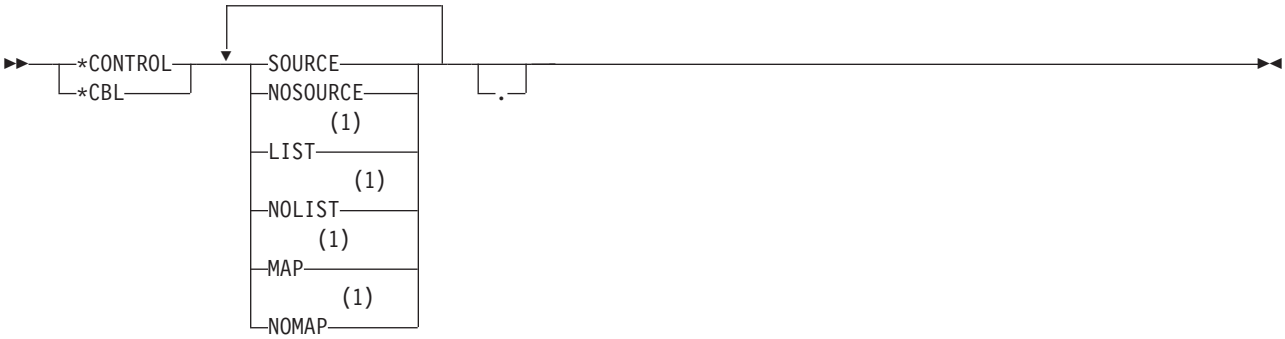
コンパイラー指示ステートメント

*CONTROL (*CBL) ステートメント

IBM Extension

*CONTROL (または *CBL) ステートメントにより、ソース・プログラム全体のソース・コードのリストを選択的に表示または抑止できます。

*CONTROL (*CBL) ステートメント - 形式



注:

1 構文検査だけ行われます。

コンパイラー出力の完全な説明については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

*CONTROL ステートメントと *CBL ステートメントは同義です。

文字 *CONTROL または *CBL は、8 桁目以降の任意の桁から始めることができ、その後に少なくとも 1 つのスペースまたはコンマと 1 つまたは複数のオプション・キーワードを続けます。オプション・キーワードは、1 つまたは複数のスペースまたはコンマで分離してください。このステートメントを指定する行には、それ以外のステートメントを指定することができず、また、継続行は許されません。ステートメントはピリオドで終わらせることができます。

要求するオプションは、次のように処理されます。

1. *CONTROL ステートメントに SOURCE または NOSOURCE が複数指定されると、最後のオプションが使用されます。
2. コンパイラー・オプションとして SOURCE ステートメントが要求されている場合に *CONTROL NOSOURCE ステートメントが検出されると、この時点からソース・リストの出力が抑止されます。ソースの出力が抑止されていることを告げる通知メッセージが出されます。

   その後、*CONTROL SOURCE を指定してソース・リストの出力を再開できます。

コンパイラー・オプションについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

3. コンパイラー・オプションとして *NOSOURCE が要求されている場合には、出力は常に禁止されます。
4. *CONTROL ステートメントは、それが書かれているソース・プログラムに対してだけ有効です。一連のソース・プログラムのバッチ・コンパイル全体にわたって有効ではありません。

*CONTROL (*CBL) ステートメントと COPY ステートメント

SUPPRESS 句を含む COPY ステートメントは、コピーされるメンバーに含まれている *CONTROL または *CBL オプションを無効にします。しかし、コンパイラーは、抑止された COPY メンバー内の *CONTROL および *CBL ステートメントを保管しています。 COPY メンバーの処理が終了すると、その中の最後の NOSOURCE または SOURCE オプションが実行されます。

COPY ステートメントに SUPPRESS 句が含まれていない場合は、コピーされたメンバー内の *CONTROL および *CBL ステートメントは直ちに実行されます。

End of IBM Extension

COPY ステートメント

COPY ステートメントは、事前に作成されたテキストを COBOL プログラムの中に組み入れるライブラリー・ステートメントです。

IBM Extension

- 形式 2 - DDS 変換
- 形式 3 - 基本 IFS

End of IBM Extension

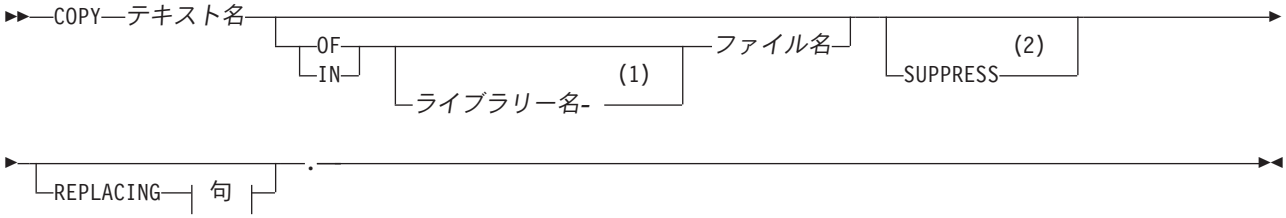
事前に作成されたソース・プログラム項目をコンパイル時にソース・プログラムに組み込むことができます。したがって、インストール・システムでは、標準のファイル記述、レコード記述、またはプロシージャーを、再コーディングせずに使用できます。これらの項目およびプロシージャーは、ユーザー作成のライブラリーに保管できます。その後、COPY ステートメントによってソース・プログラムに組み込むことができます。

COPY ステートメントを含むソース・プログラムのコンパイルは、結果として得られるソース・プログラムを処理する前にすべての COPY ステートメントを処理することと論理的に同じです。

COPY ステートメントを処理すると、テキスト名に関連付けられているライブラリー・テキストがソース・プログラムの中にコピーされて、COPY ステートメント全体 (語 COPY で始まりピリオドで終わる) が論理的に置き換えられます。 REPLACING 句が指定されていない場合には、ライブラリー・テキストは未変更のままコピーされます。

COPY ステートメント - 形式 1 - 基本

COPY ステートメント - 形式 1



注:

1 ファイル名を修飾する場合は、ライブラリー名とファイル名の間にハイフンが必要です。

2 IBM 拡張

テキスト名

テキスト名は、コピーされるメンバーの名前です。テキスト名は英字で始まらなければなりません。テキスト名の最初の 10 文字は、メンバー名として使用されます。したがって、これらの最初の 10 文字は 1 つのファイル内で固有でなければなりません。

テキスト名は、それが存在するライブラリー名およびファイル名によって修飾できます。ファイル名を指定しないと、ファイル名として QCBLLESRC が想定されます。コピー・メンバーがデフォルト・ファイル QCBLLESRC の中で見つからないと、次に QLBLSRC が検索されます。ファイル名がライブラリー名によって修飾されていない場合には、ファイル名はライブラリー・リスト内のファイルに存在するものと見なされます。ライブラリー名とファイル名の間には、スペースなしでハイフンを入れることが必要です。例えば、ライブラリー MYLIB 内のファイル MYFILE を修飾するには、MYLIB-MYFILE とコーディングします。

ライブラリー名、ファイル名、およびテキスト名は、プログラム名の形成に関する規則に従わなければなりません。

ライブラリー名

ライブラリー名の最初の 10 文字は、識別名として使用されます。したがって、これらの最初の 10 文字はシステム内で固有でなければなりません。ファイル名を修飾するには、ライブラリー名とファイル名の間にスペースなしでハイフンを入れることが必要です。

各 COPY ステートメントは、前にスペースがなければならず、分離文字ピリオドで終わらなければなりません。

COPY ステートメントは、文字ストリングまたは分離文字を指定できる場所であれば、ソース・プログラム内のどこにでも指定できます。ただし、COPY ステートメントを COPY ステートメントの中に指定してはなりません。コピーされた結果のテキストの中に COPY ステートメントが含まれていてはなりません。

IBM Extension

COPY ステートメントはネストすることができます。ただし、ネストされた COPY ステートメントには REPLACING 句を含めることができず、REPLACING 句を指定する COPY ステートメントにはネストされた COPY ステートメントを含めることができません。

COPY ステートメントは、再帰を起こすことができません。つまり、COPY メンバーは、その COPY メンバーについてのファイルの終わりに達するまで、一連のネストされた COPY ステートメントの中で一度し

か指定できません。

End of IBM Extension

ライブラリー・テキストおよび疑似テキストの中にデバッグ行があってもかまいません。デバッグ行内のテキスト語には、標識域に D が指定されなかった場合と同様にマッチング規則が適用されます。デバッグ行が疑似テキストの中に指定されるのは、そのデバッグ行がソース・プログラムの中で、始まりの疑似テキスト分離文字の後で、かつ対応する終わりの疑似テキスト分離文字の前から始まっている場合です。

COPY ステートメントがデバッグ行に指定された場合には、コピーされるテキストは、テキスト内のコメント行が結果のソース・プログラムの中でコメント行となること以外は、それがデバッグ行に指定された場合と同様に取り扱われます。

COPY という語をコメント記入項目の中、またはコメント記入項目を指定できる場所に指定すると、それはコメント記入項目の一部と見なされます。

SOURCE-COMPUTER 段落に WITH DEBUGGING MODE 文節が指定されていない場合には、すべての COPY および REPLACE ステートメントが処理された後、デバッグ行はコメント行のすべての特性を持つものと見なされます。

コメント行またはブランク行がライブラリー・テキストの中にあってもかまいません。 ライブラリー・テキスト内のコメント行またはブランク行は、未変更のまま結果のソース・プログラムにコピーされますが、次の例外があります。ライブラリー・テキスト内のコメント行またはブランク行は、疑似テキスト -1 と一致する一連のテキスト語の中にある場合にはコピーされません (8-6 ページの『置き換えおよび比較の規則』を参照してください)。

COBOL ソース・プログラム全体が構文的に正しいかどうかは、すべての COPY ステートメントが完全に処理されるまでは判断できません。これは、ライブラリー・テキストの構文の正しさが単独では判断できないためです。

ライブラリーからコピーされるライブラリー・テキストは、結果のプログラムにおいて、ライブラリーの中にあったときと同じ区域に置かれます。ライブラリー・テキストは、標準 COBOL 形式の規則に従っていなければなりません。

SUPPRESS 句

IBM Extension

SUPPRESS 句を指定すると、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。 SUPPRESS 句が継続している間は、このタイプの COPY ステートメントは *CONTROL ステートメントまたは *CBL ステートメントを無効にします。

コピーされたメンバーに *CONTROL ステートメントまたは *CBL ステートメントが含まれている場合、COPY メンバーの処理が終了すると、最後のステートメントが実行されます。

ネストされた COPY ステートメントの場合、SUPPRESS が有効なのは、次の COPY が検出されるまでの間だけです。抑止は、ネストされた COPY の完了後に再開されます。

End of IBM Extension

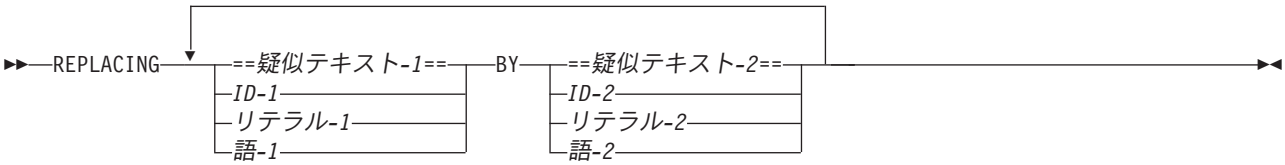
REPLACING 句

以下の説明では、各**オペランド**は、次のいずれかで構成できます。

- 疑似テキスト
- ID
- リテラル
- COBOL 語 (COPY を除く)
- 関数 ID

REPLACING 句を指定する場合には、ライブラリー・テキストがコピーされ、さらに、ライブラリー・テキスト内のオペランド -1 の正しく一致するものそれぞれが、対応するオペランド -2 によって置き換えられます。

REPLACING 句 - 形式



疑似テキスト -1、疑似テキスト -2

疑似テキストは、疑似テキスト分離文字 (==) によって結合された (ただし分離文字は含まない) 一連のテキスト語、コメント行、または区切りスペースです。

疑似テキスト -1 は、分離文字コンマまたは分離文字セミコロン以外の、最低 1 個のテキスト語を含んでいなければなりません。最初と最後のスペースは、テキスト比較処理には含まれません、複数の組み込みスペースはシングル・スペースであると見なされます。

疑似テキスト -2 は、テキスト語を含んでいる必要はなく、また、スペース文字またはコメント行 (あるいはその両方) だけから構成できます。例えば、ライブラリー・メンバー TEXTA が次の記入項目から構成される場合、

```
01  AA-DATA.
  10  AA-ID      PIC X(9).
  10  AA-TYPE    PIC X(1).
```

プログラマーは、次のように COPY ステートメントを使用して疑似テキストを置き換えることができます。

```
    COPY TEXTA REPLACING ==01 AA-DATA== BY ==05 EE-DATA==.
                         ==AA-ID==      BY ==EE-ID==.
                         ==AA-TYPE==    BY ==EE-TYPE==.
```

結果のテキストは、次のように書かれている場合と同様に扱われます。

```
05  EE-DATA.
  10  EE-ID      PIC X(9).
  10  EE-TYPE    PIC X(1).
```

疑似テキスト -1 には、少なくともテキスト語が含まれる必要があります。定義により、テキスト語は分離文字によって結合されるため、疑似テキストを使用して置換する語の一部を選択することはできません (例えば、データ名の中の接頭部)。一致するものを検出するために、完全なテキスト語を使用する必要があります。ただし、マッチングの目的のためだけに使用されており、ソース・プログラムにはコピーされない分離文字を使用して、ライブラリー・テキスト内にあるテキスト語を複数のテキスト語に

分割することによって、そのテキスト語の部分的な置き換えをシミュレートすることはできます。この方式の例については 8-7 ページの『コーディング例』を参照してください。

IBM Extension

疑似テキスト -1 または疑似テキスト -2 に、DBCS 文字ストリングまたは国別文字ストリングを含めることができます。 ただしこの場合、疑似テキストは 2 行にわたってはなりません。

End of IBM Extension

ID-1、ID-2
データ部のどのセクションでも定義できます。

関数 ID にすることができます。

リテラル-1、リテラル-2
数字または非数字にすることができます。

IBM Extension

浮動小数点リテラル、DBCS リテラル、または国別リテラルにすることもできます。

End of IBM Extension

語 -1、語 -2
任意の単一の COBOL 語 (COPY を除く) にすることができます。

マッチングのために、各 ID-1、リテラル -1、または語 -1 は、それぞれ ID-1、リテラル -1、または語 -1 だけを含む疑似テキストとして取り扱われます。

置き換えおよび比較の規則

1. ID の一部ではない算術および論理演算子は、テキスト語と見なされ、疑似テキスト・オプションを介してだけ置き換えることができます。
2. 表意定数がオペランド -1 である場合、それが一致するのは、それが指定されたとおりの正確な形で存在する場合だけです。例えば、ライブラリー・テキスト内に ALL 『AB』 が指定されている場合には、『ABAB』 は一致と見なされず、ALL 『AB』 だけが一致と見なされます。
3. 疑似テキスト -2 の位置付けの規則によって、置き換えで異なる区域への挿入が行われる場合を除き、オペランド -2 はオペランド -1 の位置にコピーされます。
4. ライブラリー・テキストの左端の語よりも前にある分離文字コンマ、セミコロン、またはスペース (あるいはこれらの組み合わせ) は、ソース・プログラムの中にコピーされます。左端のライブラリー・テキスト語と、REPLACING オプションで指定された最初のオペランド -1 から開始して、キーワード BY よりも前にある REPLACING オペランド全体が、等しい数の連続したライブラリー・テキスト語と比較されます。
5. オペランド -1 がライブラリー・テキストと一致するのは、オペランド -1 の中の一連のテキスト語が、文字単位で一連のライブラリー・テキスト語と等しい場合に限られます。マッチングのため、コンマまたはセミコロンの分離文字と、1 つ以上のスペース分離文字は、1 つのスペースと見なされます。
6. 一致が検出されない場合には、一致が検出されるかまたは REPLACING 句のオペランドがなくなるまで、連続するそれぞれのオペランド -1 について比較が繰り返されます。
7. オペランド -1 とライブラリー・テキストの間で一致が検出されると、対応するオペランド -2 がソース・プログラムにコピーされます。

8. すべてのオペランドが比較され、一致が検出されない場合、ライブラリー・テキストの左端の語がソース・プログラムにコピーされます。
9. その後、次に続くコピーされていないライブラリー・テキスト語が左端のテキスト語と見なされ、最初のオペランド -1 から比較処理が繰り返されます。この処理は、ライブラリー・テキストの右端の語が比較されるまで続けられます。
10. ライブラリー・テキストおよび疑似テキスト -1 の中のコメント行またはブランク行は、マッチングの目的で無視されます。ライブラリー・テキストおよび疑似テキスト -1 の中の一連の語は、参照形式に関する規則に従って判別されます。疑似テキスト -2 の中のコメント行またはブランク行は、疑似テキスト -2 がテキストの置き換えの結果としてソース・プログラムに入れられるときには常に、未変更のまま結果のプログラムにコピーされます。ライブラリー・テキスト内のコメント行またはブランク行は、未変更のまま結果のソース・プログラムにコピーされますが、次の例外があります。ライブラリー・テキスト内のコメント行またはブランク行は、疑似テキスト -1 と一致する一連のテキスト語の中にある場合にはコピーされません。
11. 置き換え後のテキスト語は、標準 COBOL 形式の規則に従ってソース・プログラムの中に入れられます。参照形式域の詳細は 2-22 ページの『参照形式』を参照してください。

    ライブラリーから更新せずにコピーされた各テキスト語は、結果のプログラムにおいて、元のライブラリーの中にあったと同じ行の区域から開始されます。しかし、この規則の例外として、更新されずにコピーされるテキスト語がライブラリー内の区域 A で開始し、かつ自分よりも長いテキストで置き換えられる区域 A 内の別のテキスト語の後に続く場合は、更新されずにコピーされたテキスト語は、区域 A で適合しなくなれば区域 B で開始されます。

    結果のプログラムに入れられる疑似テキスト -2 の各テキスト語は、結果のプログラムにおいて、元の疑似テキスト -2 の中にあったと同じ区域で開始します。結果のプログラムに入れられる各 ID-2、リテラル -2、および語 -2 は、結果のプログラムにおいて、置き換えられるライブラリー・テキストと同じ区域で開始します。

IBM Extension

12. COPY REPLACING は、EJECT、SKIP1/2/3、または TITLE コンパイラー指示ステートメントには影響を与えません。

End of IBM Extension

コーディング例

多くのプログラムに共通な一連のコード (ファイル記述とデータ記述、エラー・ルーチンと例外ルーチンなど) をライブラリーに保管し、COPY ステートメントと結合させて使用できます。そのような共通のコードについての命名規則が確立されている場合には、REPLACING 句を指定する必要はありません。名前がプログラムごとに異なる場合には、REPLACING 句を使用して、このプログラムにとって意味のある名前を提供できます。

例 1: この例では、ライブラリー・テキスト PAYLIB が次のデータ部の記入項目から構成されています。

```
01  A.
  02  B    PIC S99.
  02  C    PIC S9(5)V99.
  02  D    PIC S9999 OCCURS 1 TO 52 TIMES
      DEPENDING ON B OF A.
```

プログラマーは、プログラムのデータ部で次のように COPY ステートメントを使用することができます。

```
    COPY PAYLIB.
```

ライブラリー・テキストは、COPY ステートメントのすぐ後で未変更のまま COBOL プログラムにコピーされます。

例 2: 例 1 で使用したライブラリー・テキスト内のデータ名のいくつか (またはすべて) を変更するには、プログラマーは次のように REPLACING 句を使用できます。

```
  COPY PAYLIB REPLACING  A BY PAYROLL
                         B BY PAY-CODE
                         C BY GROSS-PAY
                         D BY HOURS.
```

ライブラリー・テキストがコピーされるときに、結果のテキストは次のように書かれている場合と同様に表示されます。

```
01  PAYROLL.
  02  PAY-CODE PIC S99.
  02  GROSS-PAY PIC S9(5)V99.
  02  HOURS PIC S9999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAY-CODE OF PAYROLL.
```

ここに示した変更は、このプログラムに対してだけ行われます。テキストはライブラリーの中では変更されていません。

例 3: この例では、ライブラリー・テキストを作成中にある特定の規則に従う場合に、データ名の一部を置き換えることができる方法を示します。この場合、ライブラリー・テキスト CONTACT には次のコードが含まれます。

```
01 (PRFX)-RECORD.
 03 (PRFX)-NAME PIC X(24).
 03 (PRFX)-PHONE PIC X(10).
 03 (PRFX)-EXTN PIC X(4).
```

プログラマーは、データ名の接頭部によってテキスト語 PRFX とそれを結合する括弧を置き換えて、このライブラリー・テキストをコピーできます。例えば、次の COPY ステートメントをプログラムのデータ部に書くことができます。

```
    COPY CONTACT REPLACING ==(PRFX)== BY ==CUST==.
```

ライブラリー・テキストがコピーされるときに、結果のテキストは次のように書かれている場合と同様に表示されます。

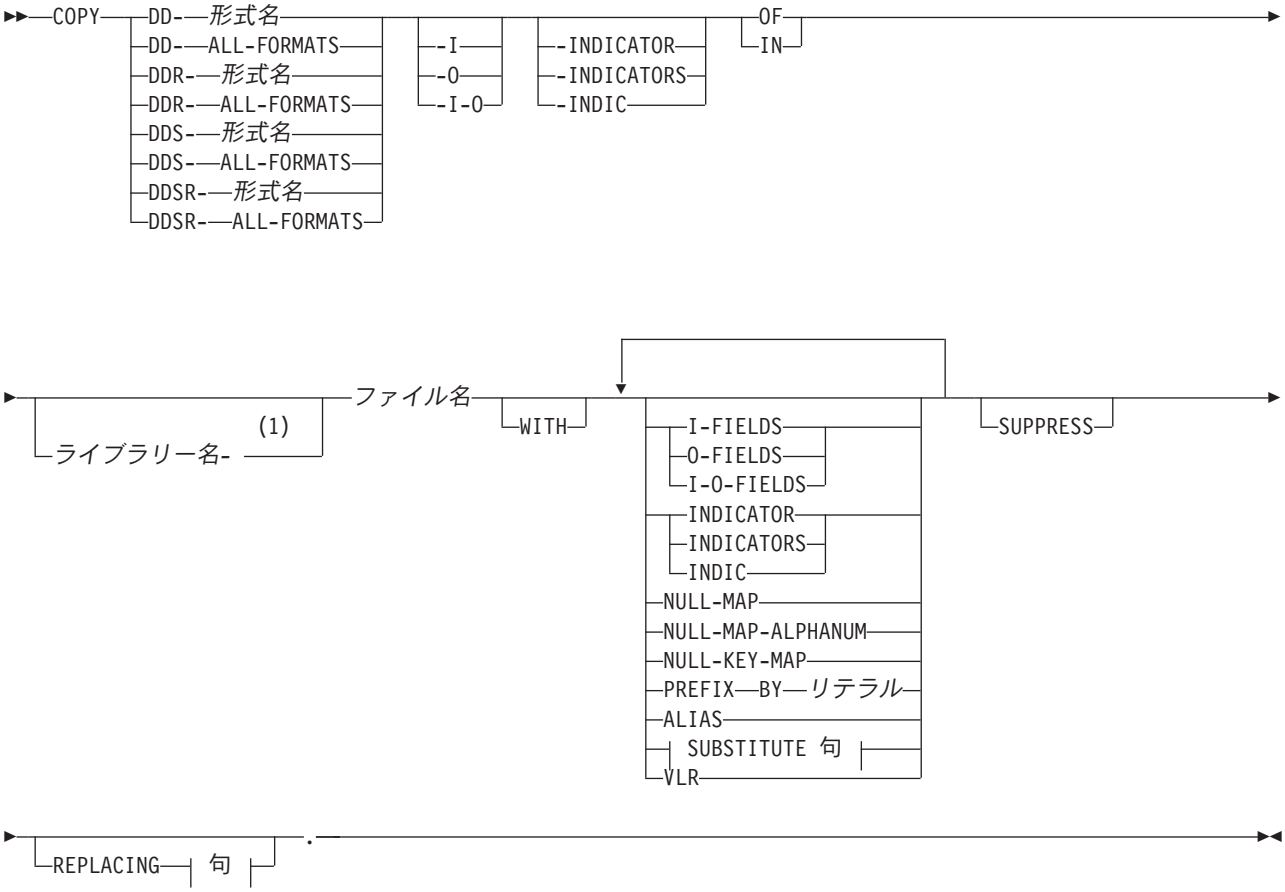
```
01 CUST-RECORD.
 03 CUST-NAME PIC X(24).
 03 CUST-PHONE PIC X(10).
 03 CUST-EXTN PIC X(4).
```

注: 多くの分離文字は COPY ステートメントを処理するときに特別に重要であるため、この方法でテキスト語の一部を区切るために使用できる値は括弧およびコロン記号に制限されます。さらに、ライブラリー・テキストを入力するときに、SEU 構文検査機能によってフラグが立てられたある特定のエラーを無視する必要があります。

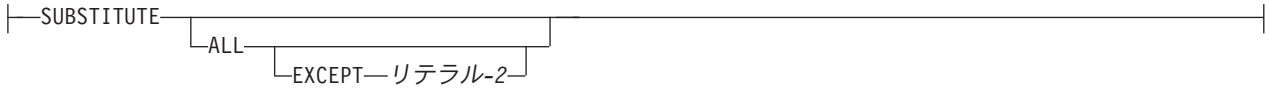
COPY ステートメント - 形式 2 - DDS 変換

IBM Extension

COPY ステートメント - 形式 2 - DDS 変換



SUBSTITUTE 句:



注:

1 ファイル名を修飾する場合は、ライブラリー名とファイル名の間にハイフンが必要です。

形式 2 に関する考慮事項

形式 2 の COPY ステートメント (DD、DDR、DDS、または DDSR オプション) は、システムに存在するファイルを記述する COBOL データ部ステートメントを作成するために使用できます。これらの記述は、コンパイル時に存在するファイルのバージョンに基づきます。これらの記述では、ファイルについての DDS ソース・ステートメントを使用しません。

REPLACE ステートメントが有効である場合、COPY ステートメントはコード行上の最初の項目でなければなりません。また、この行には、少なくとも最初のハイフンまでに、必要なオプションを指定するテキスト語が入っていなければなりません。

DDS は、DBCS の形式 J (DBCS データだけを含めることができるフィールドの場合)、E (DBCS または英数字データのいずれかを含めることができるフィールドの場合)、または O (DBCS および英数字データの両方を含めることができるフィールドの場合) をサポートします。また DDS は、形式 G のグラフィック・データ・タイプをサポートします。 *PICGGRAPHIC オプションは、形式 G の DDS 項目に対応する COBOL DBCS データ項目を作成するために使用されます。*PICNGRAPHIC オプションは、国別 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 を持つ COBOL NATIONAL データ項目を作成するために使用されます。 その他の環境ではすべて、データの正確なバイト数のデータを保持することのできる英数字データ項目が作成されます。

形式 2 の COPY ステートメントはデータ部でだけ使用でき、ユーザーはステートメントの前に 05 よりも小さいレベル番号のグループ・レベル項目を入れなければなりません。

DD および ALIAS オプション: DD オプションまたは ALIAS オプションは、**別名** (代替名) を参照するために使用されます。 DDS 内で別名を指定すると、最大 30 文字のデータ名を COBOL プログラムに組み込むことができます。

DD オプションまたは ALIAS オプションが使用されると、存在する別名が対応する DDS フィールド名を置き換えます。別名の中の下線はすべて、置き換えが行われる前にハイフンに変換されます。

DDR オプション: DDR オプションまたは SUBSTITUTE オプションは、DD オプションによって行われることをすべて行います。さらに、フィールド名 (あるいは、該当する場合は別名) 内の無効な COBOL 文字である @、#、\、および _ を対応する有効な COBOL 文字 A、N、D、および - に置き換えます。また、フィールド名の末尾から下線を除去します。

DDS オプション: DDS オプションは、指定した DDS 形式の内部 DDS のフィールド名をコピーします。

DDSR オプション: DDSR オプションは、DDS オプションによって行われることをすべて行います。さらに、無効な COBOL 文字である @、#、\、および _ をそれぞれ有効な COBOL 文字 A、N、D、および - に置き換えて、指定した DDS 形式で内部 DDS のフィールド名をコピーします。また、フィールド名の末尾から下線を除去します。

形式名および ALL-FORMATS オプション: 形式名は、ILE COBOL データ記述記入項目に変換される DDS レコード形式定義の名前です。形式名は、ILE COBOL データ名の形成に関する規則に従わなければなりません。

ALL-FORMATS オプションは、データ名の規則に従わない名前を含む、ファイルに定義されたすべての形式を変換します。そのような形式名を有効なデータ名に変更するには、REPLACING 句を使用しなければなりません。ただし、REPLACING 句は、EXTERNALLY-DESCRIBED-KEY によって定義された索引付きファイルの FD 記入項目内の形式名を変更する際には使用できません。このキーを RECORD KEY 文節内でデータ名を使用して定義できない場合は、そのファイルの DDS の指定内の形式名を変更する必要があります。

注: このコンテキストでは、コンパイラーは ALL-FORMATS と同様に ALL-FORMAT を受け入れます。

VLR オプション: VLR オプションは変数レコード・ファイルとともに使用する必要があります。このオプションは、可変長フィールドからのコピーを指定します。これは、CRTCBLMOD と CRTBNDCBL コマンドの CVTOPT (*VARCHAR) オプションを指定変更します。

PREFIX オプション: PREFIX オプションは、それぞれのフィールド名の前に挿入される接頭部 (リテラル) を指定するのに使用できます。これを使用すると、フィールドの内容または用途を識別する (つまり、記述する) のに役立ちます。リテラルは、一組のアポストロフィまたは一組の引用符の中に入れることができます。リテラルの最大長は 15 文字です。

I-O

-I または I-FIELDS も -O または O-FIELDS も指定しないと、-I-O または I-O-FIELDS が想定されます。 -I および O-FIELDS、または -O および I-FIELDS を指定すると、-I-O または I-O-FIELDS が想定されます。

形式名が標識属性なしで指定されており、-I と -O の両方の形式が生成される場合、それぞれのレコード形式は、生成される最大のレコード形式のサイズとして定義された 05 基本項目の再定義として生成されます。

ALL-FORMATS が標識属性なしで指定された場合には、それぞれのレコード形式は、次のいずれかとして定義された 05 基本項目の再定義として生成されます。

- COPY ステートメントが FILE SECTION にある場合には、ファイル内の最大レコード形式のサイズ。
- COPY ステートメントが FILE SECTION の外側にある場合には、生成される最大のレコード形式のサイズ。

標識属性が指定されている場合には、再定義は**行われません**。その代わりに、それぞれの形式が別々のデータ構造を生成します。詳細は 8-18 ページの『形式 2 の COPY ステートメントの INDICATOR 属性』を参照してください。

ファイルがデータベース・ファイルにある場合は、1 つの I-O 様式が生成されます。

その他のすべてのファイル・タイプの場合、生成される記述は次のようになります。

- -I が指定された場合、生成されるデータ記述記入項目には次のいずれかが含まれます。
  - 非サブファイル形式の場合、入力および入出力フィールド
  - サブファイル形式の場合、入力、出力、および入出力フィールド
- -O が指定された場合、生成されるデータ記述記入項目には出力フィールドおよび入出力フィールドが含まれます。

標識属性の使用法については 8-18 ページの『形式 2 の COPY ステートメントの INDICATOR 属性』で説明します。

ファイル名は、システム・ファイルの名前です。生成される DDS 記入項目は、ファイル内で定義されたレコード形式 (1 つまたは複数) を表します。ファイルは、プログラムのコンパイルの前に作成しなければなりません。

ライブラリー名の指定はオプションです。指定されていない場合は、現行のジョブ・ライブラリー・リストがデフォルト値として使用されます。

SUBSTITUTE 句

SUBSTITUTE 句 - 形式

SUBSTITUTE 句を使用することによって、プログラムで DDS の使用が可能となり、下線文字のような特定の文字を保存しておくことができます。下線文字は、標準の ILE COBOL 文字ではありませんが、ロケール・カテゴリーを指定するのに必要です。下線文字を、例えばコピーされた DDS に保存するためには、SUBSTITUTE 句を以下のように使用します。

```
...SUBSTITUTE ALL EXCEPT '_'.
```

リテラル-2

1 バイトの非数字リテラルでなければなりません。リテラル -2 で指定された文字は置き換えることはできません。

REPLACING 句

REPLACING 句は 8-5 ページの『REPLACING 句』で説明されています。

DDS ファイルでのヌル可能フィールドの使用

DDS でフィールドが ALWNULL として定義されている場合は、 COPY DDS ステートメントはそのフィールドをコメント付きでヌル可能と識別します。例えば、下記の 2 つの図には、ヌル可能フィールドの入っている DDS ファイル、およびこのフィールドが ILE COBOL プログラムの FILE-SECTION にコピーされるときにこれに対して作成される結果のコメントが示されています。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A* With the following physical file (TESTPF)
               R TESTING
                 FLD1          5S 0
                 FLD2          8       ALWNULL
                 FLD3          6
```

図 8-1. ヌル可能フィールドを示す DDS

```
* A COPY DDS-TESTING OF TESTPF.
*    I-O FORMAT:TESTING    FROM FILE TESTPF    OF LIBRARY QTEMP
*
     05  TESTING.
         06 FLD1                              PIC 9(5).
         06 FLD2                              PIC X(8).
                    (null-capable field)
         06 FLD3                              PIC X(6).
```

図 8-2. ヌル可能 DDS ファイルの ILE COBOL プログラムへのコピー後の結果

コピーされる DDS ヌル可能レコード形式のヌル・マップおよびヌル・キー・マップを生成するには、WITH NULL-MAP および WITH NULL-KEY-MAP 句を作業用ストレージ・セクションまたはローカル・ストレージ・セクションで新規 COPY DDS ステートメント上に指定する必要があります。 DDS の形式当たり 1 つだけ NULL-MAP のコピーが生成されます。例えば、形式に I (入力のみ) および B (入出力)

の両方が含まれる場合、生成されるヌル・マップのサイズは形式の中で指定されたすべてのフィールドに合うものです。換言すれば、これには I および B のすべてのフィールドが含まれています。

特定形式用の DDS で定義されたヌル可能フィールドごとに、データ項目定義が生成されます。生成されるデータ項目は、作業用ストレージ・セクションまたはローカル・ストレージ・セクションで COPY DDS ステートメント上に NULL-MAP または NULL-MAP-ALPHANUM を指定するかどうかによって異なります。

NULL-MAP を指定した場合は、ヌル・マップが、2 進ゼロ (0) に初期設定される PIC 1 値を指定して作成されます。以下のステートメントが、ヌル可能フィールドのソースで生成されます。

```
06  <field-name>-NF     PIC 1  VALUE B"0".
```

フィールドがヌル可能でない場合は、FILLER 項目が生成されます。

NULL-MAP-ALPHANUM を指定した場合は、ヌル・マップが、文字ゼロ (0) に初期設定される PIC X 値を指定して作成されます。以下のステートメントが、ヌル可能フィールドのソースで生成されます。

```
06  <field-name>-NF     PIC X  VALUE ZERO.
```

フィールドがヌル可能でない場合は、以下のステートメントがソースで生成されます。

```
06  <field-name>-AN     PIC X  VALUE ZERO.
```

NULL-MAP-ALPHANUM を使用して生成されたヌル・マップのサイズは、NULL-MAP を使用して生成されたヌル・マップのサイズと同じです。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A* Physical file for DDS
                R REC
                  FLD1          1A
                  FLD2          1A     ALWNULL
                  FLD3          1A
```

図 8-3. 一部のフィールドがヌル可能でない DDS ファイル

```
*DDS Generated
  05 REC-NM
     06 FILLER          PIC X VALUE ZERO.
     06 FLD2-NF         PIC 1 VALUE B"0".
     06 FILLER          PIC X VALUE ZERO.
```

図 8-4. NULL-MAP を使用して COPY DDS から生成された ILE COBOL コード

ヌル可能フィールド使用上の考慮事項

ヌル・マップ・フィールドに 1 または 0 以外の値を入れることは可能です。例えば、SQL がヌル・マップに値 2 を入れて、そのフィールドに 0 除算の結果が入っていることを示すことは可能です。

ヌル・マップで 0 または 1 以外の値を分かるようにするには、COPY DDS ステートメントに NULL-MAP-ALPHANUM を指定する必要があります。

NULL-MAP-ALPHANUM は、ヌル・マップとの間でやり取りされる値の範囲を、0 または 1 以外の値を含むよう拡張します。ヌル・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・マップで送信または受信することができる、0 または 1 以外の値について詳しくは、**IBM i Information Center** (Web サイト http://www.ibm.com/systems/i/infocenter/) のカテゴリー『データベースおよびファイル・システム』のセクション『*DB2 for i*』を参照してください。

日付データ・タイプを指定した COPY DDS の使用

表 8-1 および 表 8-2 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 8-1 には、IBM i DDS 日付データ・タイプと、それと同等の ILE COBOL 用の形式がリストされています。表 8-1 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 8-1. DDS 日付データ・タイプおよびそれらと等価の ILE COBOL の形式

IBM i 形式	COBOL が生成する形式	内容	フォーマット	有効な分離文字	長さ
*MDY	%m/%d/%y	月 / 日 / 年	mm/dd/yy	/-.,&	8
*DMY	%d/%m/%y	日 / 月 / 年	dd/mm/yy	/-.,&	8
*YMD	%y/%m/%d	年 / 月 / 日	yy/mm/dd	/-.,&	8
*JUL	%y/%j	ユリウス	yy/ddd	/-.,&	6
*ISO	@Y-%m-%d	ISO (国際標準化機構)	yyyy-mm-dd	-	10
*USA	%m/%d/@Y	IBM USA 標準規格	mm/dd/yyyy	/	10
*EUR	%d.%m.@Y	IBM 欧州標準規格	dd.mm.yyyy	.	10
*JIS	@Y-%m-%d	JIS (日本工業規格) 西暦	yyyy-mm-dd	-	10

6-41 ページの表 6-3 には、IBM i DDS 時間データ・タイプおよびそれと同等の ILE COBOL 形式をリストしています。表 8-2 は、パック 10 進数フィールド用で、USAGE PACKED-DECIMAL が生成されます。

表 8-2. DDS 時刻データ・タイプおよびそれらと等価の ILE COBOL の形式

IBM i 形式	COBOL が生成する形式	内容	フォーマット	有効な分離文字	長さ
*HMS	%H:%M:%S	時 : 分 : 秒	hh:mm:ss	:.,&	8
*ISO	%H.%M.%S	ISO (国際標準化機構)	hh.mm.ss	.	8
*USA	%I:%M @p	IBM USA 標準規格。AM および PM は、大文字小文字を任意に混合させることができます。	hh:mm AM または hh:mm PM	:	8
*EUR	%H.%M.%S	IBM 欧州標準規格	hh.mm.ss	.	8
*JIS	%H:%M:%S	JIS (日本工業規格) 西暦	hh:mm:ss	:	8

一般的な注意事項

- データベース・ファイルが標識を持つことはありません。
- RECORD KEY 文節で EXTERNALLY-DESCRIBED-KEY が指定されている場合には、 1 つの形式は 1 つの FD のもとで一度だけコピーできます。例えば、ある FD のもとで、あるファイルのすべての形式がコピーされると、その FD のもとで同一ファイルに対して他の形式 2 の COPY ステートメントを指定することはできません。

- WORKING-STORAGE または LOCAL-STORAGE の中でそれぞれの形式について別々のストレージが必要とされる場合には、それぞれの形式について別々の COPY ステートメントを指定する必要があります。

  例えば、ファイル CUSTMASTER に 2 つの形式 CUSTADR と CUSTDETL が含まれているとすると、次の COPY ステートメントを指定できます。

```
    SELECT FILE-X
    ASSIGN TO DATABASE-CUSTMASTER.
    .
    .
    .
FD  FILE-X
    LABEL RECORDS ARE STANDARD.
01  FILE-X-RECS.
     COPY DDS-ALL-FORMATS OF
        QGPL-CUSTMASTER.   (See Note 1.)
    .
    .
    .
WORKING-STORAGE SECTION.
01  ADR-REC.
     COPY DDS-CUSTADR OF
         CUSTMASTER.     (See Note 2.)
01  DETAIL-REC.
     COPY DDS-CUSTDETL OF
         CUSTMASTER.     (See Note 2.)
```

注:

1. この COPY ステートメントはすべての形式に対して記憶域を 1 つだけ生成します。
2. これらの COPY ステートメントは別々のストレージ域を生成します。

生成されるデータ構造

このセクションでは、COPY ステートメントによって生成される、以下のデータ構造について説明します。

- 形式 (レコード) レベルの構造
- データ・フィールドの構造
- 標識の構造

形式 (レコード) レベルの構造: ソース・プログラムのリストでは、各形式の先頭にコメント・テーブルが生成されます。これらのコメントは、プログラムのコンパイル中に使用されたファイルの詳細を示します。ファイルにレコード・キーがある場合には、DDS でキーが定義されている方法を示すコメントも生成されます。以下に、テーブルに入れられる可能性のあるレコード・キー記入項目と、テーブルのヘッディングを示します。

見出し	可能な記入項目
NUMBER	キー・フィールド番号
NAME	キー・フィールド名
RETRIEVAL	ASCENDING、DESCENDING
ALTSEQ	NO、YES

複数の形式の生成に備えて再定義が必要とされる場合、次のようにグループ・レベルの名前が生成されます。

```
05 file-name-RECORD
   PIC X(size of largest record).
```

各形式に対してグループ・レベルの名前は次のように割り当てられます。

- INPUT

  ```
  05 format-name-I
  ```

- OUTPUT

  ```
  05 format-name-O
  ```

- I-O 形式

  ```
  05 format-name
  ```

データ・フィールドの構造: フィールド名、PICTURE 定義、および数字の USAGE 文節は、内部 DDS 形式のフィールド名 (または、DD オプションの場合は別名) およびデータ・タイプ表現から直接に導出されます。フィールド名および PICTURE 定義は、以下のように構成されます。 `06 field-name PIC` (以下の表の注 1 を参照してください)。

注: 適切な COBOL 定義については表 8-3 を参照してください。

表 8-3. データ・フィールドの構造

DDS		COBOL データ部 n = フィールドの全長 (DDS の桁 30 ～ 34) m= 小数部の桁数 (DDS の桁 36 と 37)	
データ・タイプ (桁 35)	形式	DDS の桁 36 と 37 がブランクである場合	DDS の桁 36 と 37 がブランクでない場合
物理ファイル、論理ファイル、プリンター・ファイル、および通信ファイル			
b	デフォルト	PIC X(n)[3]	PIC S9(n-m)V9(m)
(ブランク)	パック 10 進数[5]	PIC S9(n) COMP-3	PIC S9(n-m)V9(m) COMP-3
P	ゾーン 10 進数/符号付き数字[4]	PIC S9	PIC S9(n-m)V9(m)
S	2 進数	PIC S9(n) COMP-4	PIC S9(n-m)V9(m) COMP-4
BF	浮動小数点[1]	PIC 9(5) COMP-4 または COMP-1	PIC 9(5) COMP-4 または COMP-1
	単精度	PIC 9(10) COMP-4 または COMP-2	PIC 9(10) COMP-4 または COMP-2
	倍精度	PIC X(n)[3]	
A	文字[4]	PIC X(n)	
H	16 進データ	PIC X(n) または FORMAT DATE	
L	日付[2]	PIC X(n) または FORMAT TIME	
T	時間[2]	PIC X(n) または	
ZE	タイム・スタンプ[2]	FORMAT TIMESTAMP	
J	DBCS 択一データ	PIC X(n)	
O	DBCS 専用データ	PIC X(n)	
G	DBCS 混用データ	PIC X(n)	
	DBCS グラフィック・データ	PIC X(2n) または PIC G(n)[3]	
	UCS2 グラフィック・データ	—	
		PIC N(2n)	
ディスプレイ・ファイル			

表 8-3. データ・フィールドの構造 (続き)

DDS		COBOL データ部 n = フィールドの全長 (DDS の桁 30 ～ 34) m= 小数部の桁数 (DDS の桁 36 と 37)	
b	デフォルト	PIC X(n)	PIC S9(n-m)V9(m)
(ブランク)	英字だけ	PIC X(n)	—
X	数字シフト	PIC X(n)	PIC S9(n-m)V9(m)
N	数字だけ	—	PIC S9(n-m)V9(m)
Y	キーボード入力禁止	PIC X(n)	PIC S9(n-m)V9(m)
I	カタカナ	PIC X(n)	—
W	英数字シフト	PIC X(n)	—
A	数字だけ	PIC X(n)	PIC S9(n)
D	浮動小数点[1]	PIC 9(5) COMP-4 または COMP-1	PIC 9(5) COMP-4 または COMP-1
F	単精度	PIC 9(10) COMP-4 または COMP-2	PIC 9(10) COMP-4 または COMP-2
	倍精度	PIC X(n)	—
	数字専用の文字	PIC X(n) または FORMAT DATE	—
M	日付[2]	PIC X(n) または FORMAT TIME	—
L	時間[2]	PIC X(n) または	—
T	タイム・スタンプ[2]	FORMAT TIMESTAMP	PIC S9(n-m)V9(m)
ZS	符号付き数字シフト	—	—
E	DBCS 択一	PIC X(n)	—
J	DBCS 専用	PIC X(n)	—
O	DBCS 混用	PIC X(n)	—
G	DBCS グラフィック	PIC X(2n) または PIC G(n)	
	UCS2 グラフィック	—	
		PIC N(2n)	

注:

1. CVTOPT パラメーターの *NOFLOAT 値が有効な場合には、浮動小数点フィールドは、BINARY の USAGE を持つ FILLER 項目として取り入れられます。 *FLOAT が指定された場合には、フィールドは、与えられたそれらの DDS 名を COMP-1 (単精度浮動小数点) または COMP-2 (倍精度浮動小数点) の USAGE で使用することによって、取り入れられます。 8-20 ページの『浮動小数点フィールド』を参照してください。
2. デフォルトで、FILLER 項目は英数字として宣言されます。 CRTBNDCBL または CRTCBLMOD コマンドの CVTOPT パラメーター上で *DATE、*TIME、または *TIMESTAMP を指定することにより、COBOL に日付、時刻、およびタイム・スタンプのフィールドを日時データ・タイプとして扱うようにすることもできます。 8-20 ページの『日付、時刻、およびタイム・スタンプのフィールド』を参照してください。
3. DDS では、フィールドが VARLEN の属性を持つ場合、結果はフィールドの先頭に 2 バイト追加されたものになります。
4. DDS 文字または DATFMT キーワードを指定したゾーン・データ・タイプが使用された場合には、CRTBNDCBL または CRTCBLMOD コマンドで CVTOPT パラメーターに *CVTTODATE 値が指定されると、ILE COBOL はこれを日付フィールドとして扱います。
5. DATFMT キーワードを指定した DDS パック・データ・タイプが使用された場合には、CRTBNDCBL または CRTCBLMOD コマンドで CVTOPT パラメーターに *CVTTODATE 値が指定されると、ILE COBOL はこれを日付フィールドとして扱います。
6. DDS では、データ・タイプが G で、属性が CCSID(n) (n は、国別 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) であるフィールドは、UCS-2 グラフィック・データ・タイプです。UCS-2 グラフィック・データ・タイプを COBOL NATIONAL データ・タイプとして取り入れるには、CRTBNDCBL または CRTCBLMOD コマンドの CVTOPT パラメーターに *PICNGRAPHIC を指定してください。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

標識の構造: 標識が要求され、形式の中に存在する場合には、構造の先頭に追加のグループ名 (06 レベル) が生成され、その後に該当する個々の標識用の記入項目 (07 レベル) が続きます。

```
06  format-name-(I or O)-INDIC.
    07  INxx PIC 1 INDIC xx.
```

ここで、xx は標識番号です。

例えば、次のとおりです。

```
06  SAMPLE1-I-INDIC.
    07  IN01 PIC 1 INDIC 01.
    07  IN04 PIC 1 INDIC 04.
    07  IN05 PIC 1 INDIC 05.
    07  IN07 PIC 1 INDIC 07.
06  FLD1 PIC  ...  .
06  FLD2 PIC  ...  .
```

形式 2 の COPY ステートメントの INDICATOR 属性: INDICATOR 属性は、標識用にデータ記述記入項目が生成されるかどうかを指定します。

INDICATOR 属性を指定すると、標識用にデータ記述記入項目が生成されますが、データ・フィールド用には生成されません。

次のように 05 グループ・レベル記入項目が生成されます。

- 単一構造について COPY を行う場合、

  ```
  COPY DDS-format-name-INDIC
  ```

  上記によって、以下が生成されます。

  ```
  05 形式名 -I (または -0)
  ```

- 複数構造について COPY を行う場合、

  ```
  COPY DDS-ALL-FORMATS-INDIC
  ```

  上記によって、以下が生成されます。

  ```
  05 file-name-RECORD.
  ```

生成されるデータ記述記入項目は、COPY ステートメントの中でどの用途属性 (I、O、または I-O) が指定または想定されるかによって決まります。

- ...I-INDICATOR... が指定された場合には、入力レコード域内で使用される標識として入力 (応答) 標識用のデータ記述記入項目が生成されます。
- ...O-INDICATOR... が指定された場合には、出力レコード域内で使用される標識として出力 (オプション) 標識用のデータ記述記入項目が生成されます。
- ...I-O-INDICATOR... が指定または想定された場合には、入力および出力レコード域内で使用される標識として、入力と出力 (応答とオプション) の両方の標識用の別個のデータ記述記入項目が生成されます。

個々の標識記述は 8-17 ページの『標識の構造』の説明に従って生成されます。

INDICATOR 属性が指定されない場合は、標識用にデータ記述記入項目が生成されるかどうかは、ファイルの作成時に DDS 内でキーワード INDARA が指定されたかどうかによって決まります。

- INDARA が指定されていない場合は、データ・フィールドと標識と両方のためにデータ記述記入項目が生成されます。
- INDARA が指定されている場合は、データ・フィールド用にだけデータ記述記入項目が生成され、標識用には生成されません。

I-O 形式の生成: フィールド記述がすべて同一であり、暗黙または明示的に INPUT または OUTPUT フィールドを要求した場合、生成されるフィールド記述は一組だけです。このタイプの記述には、「I-O FORMAT: 形式名」というコメント行が付けられます。レコード形式名には、-I と -O のどちらも付けられません。

注: データベース・ファイルの場合、データベース・ファイル内のフィールド記述はすべて同じであるため、必ずこのようになります。(図 8-5 を参照してください。)

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL         TESTLIB/STRTEXTD     I-SERIES1  06/02/15 11:27:50     Page  2
                           S o u r c e
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.                                                    02/02/21
  6     000700   OBJECT-COMPUTER. IBM-ISERIES.                                                    02/02/21
  7     000800 INPUT-OUTPUT SECTION.                                                             02/02/21
  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.                                                           02/02/21
        001800     COPY DDS-RDE OF NAMES.                                                       02/02/21
       +000001*    I-O FORMAT:RDE        FROM FILE NAMES      OF LIBRARY TESTLIB           RDE
       +000002*                          RECORD DESCRIPTION                                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*      .
        002700*      .
```

図 8-5. I-O 形式の生成

形式の再定義: ALL-FORMATS または -I-O 句に対して生成される可能性がある REDEFINES 文節には、特別な注意を払う必要があります。すべての形式が同一の区域 (一般にバッファー域) で再定義されるため、複数のフィールド名が同一のストレージを記述する可能性があり、形式の区域全体が各出力操作に先立って再初期設定されない場合には、予測できない結果が生じる可能性があります。

MOVE CORRESPONDING ステートメントで指定されたデータ項目に従属するデータ項目は、それが REDEFINES 文節を含むか、または再定義項目に従属する場合には一致せず、移動されません。

再初期設定を回避するには、-I および -O 接尾部を指定した複数の形式 2 の COPY ステートメント (DDS または DD) を使用して、作業用ストレージ・セクションまたはローカル・ストレージ・セクションで各形式または形式タイプ (入力または出力) 用の別々のストレージ域を作成できます。これらのレコード形式を READ INTO および WRITE FROM ステートメントで使用できます。例えば、次のとおりです。

```
FD ORDER-ENTRY-SCREEN . . .
01 ORDER-ENTRY-RECORD . . .
   .
   .
   .
WORKING-STORAGE SECTION.
01 ORDSFL-I-FORMAT.
   COPY DDS-ORDSFL-I OF DOESCR.
01 ORDSFL-O-FORMAT.
   COPY DDS-ORDSFL-O OF DOESCR.
   .
   .
   .
PROCEDURE DIVISION.
   .
   .
   .
READ SUBFILE ORDER-ENTRY-SCREEN NEXT MODIFIED RECORD
   INTO ORDSFL-I-FORMAT FORMAT IS "ORDSFL"
   AT END SET NO-MODIFIED-SUBFILE-RCD TO TRUE.
   .
   .
   .
MOVE CORR ORDSFL-I TO ORDSFL-O.
REWRITE SUBFILE ORDER-ENTRY-RECORD FROM ORDSFL-O-FORMAT
                           FORMAT IS "ORDSFL" . . .
   .
   .
   .
```

注: COPY ステートメントは、ファイル・セクション、作業用ストレージ・セクション、およびローカル・ストレージ・セクションで使用できますが、得られる結果が全く同じというわけではありません。詳細については 8-24 ページの『キー生成の例』を参照してください。

フィールドおよび形式名に関する追加の注意事項: 生成されたフィールド名が COBOL 予約語の場合、接尾部 -DDS がそのフィールド名に付け加えられます。生成されたフィールド名が物理ファイルから取られた (つまり、そのフィールドが CONCAT または RENAME キーワードの引数である) 場合にも、接尾部が付け加えられます。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

EXTERNALLY-DESCRIBED-KEY が使用されているときには、REPLACING 句を使用してキー・フィールド名または形式名を変更することはできません。

浮動小数点フィールド: ファイルは、内部浮動小数点フィールドを含むことができます。 CVTOPT パラメーターの *NOFLOAT 値 (デフォルト) が有効な場合には、浮動小数点フィールドは、BINARY の USAGE を持つ FILLER 項目として取り入れられます。 *FLOAT が指定された場合には、フィールドは、与えられたそれらの DDS 名を COMP-1 (単精度浮動小数点) または COMP-2 (倍精度浮動小数点) の USAGE で使用することによって、取り入れられます。

浮動小数点キー・フィールドが認められています。 KEY が内部浮動小数点数である場合には、キー値の順序は数字の順序になります。 KEY が外部浮動小数点数である場合には、キーは英数字であり、レコードの順序は、使用される照合順序によって異なります。

日付、時刻、およびタイム・スタンプのフィールド: このセクションでは、日付、時刻、およびタイム・スタンプのフィールドの以下のクラスについて説明します。

- 日時クラス
- 英数字のクラス

また、ここでは、DDS を使用して日時クラスの日付、時刻、タイム・スタンプのフィールドを定義する方法の例も示しています。

日時クラス: **日時フィールド**には、日時クラスの日付、時刻、およびタイム・スタンプ・データ項目が入っており、DATFMT キーワードを指定するゾーン、パック、および文字 DDS フィールドの使用が許可されています。日付データ・タイプは、ユーザー・プログラムに固定長文字フィールドとして取り入れられる日付、時刻、およびタイム・スタンプのフィールドと同じではありません。 ILE COBOL では、日付データ・タイプは USAGE DISPLAY または USAGE PACKED-DECIMAL データ項目に変換され、日付、時刻、およびタイム・スタンプ・フィールドは英数字データ項目に変換されます (『英数字クラス』に説明されています)。

日付データ・タイプは、*CVTTODATE 変換パラメーター・オプション (CVTOPT) が指定されている場合には、それと同等の ILE COBOL 用の形式に COPY DDS から変換されます。

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、このパラメーター用に COPY DDS から生成されるこれと同等の ILE COBOL の形式については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

英数字クラス: 英数字の日付、時刻、およびタイム・スタンプ・フィールドをプログラムに取り入れることができるのは、CRTCBLMOD または CRTBNDCBL コマンドの CVTOPT(*DATETIME) オプション、または PROCESS ステートメントの DATETIME オプションを指定する場合だけです。*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・フィールドは、固定長文字フィールドとして取り入れられます。プログラムでは、それらのフィールドに対して有効な文字操作を実行できます。

日付、時刻、およびタイム・スタンプ・データ・タイプには、それぞれ固有の形式があります。

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。

また、日付、時刻、またはタイム・スタンプ・フィールドに適切な値を移動する前にレコードの WRITE を試みると、WRITE 操作が失敗し、ファイル状況は 90 となります。

各データ・タイプの有効な形式については、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) 内のカテゴリー『データベースおよびファイル・システム』のセクション『*DB2 for i*』を参照してください。

日付、時刻、およびタイム・スタンプ DDS の例: 以下の例では、DDS で日付、時刻、およびタイム・スタンプ・フィールドの定義の方法について示しています。

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A               LOGICAL FILE LF1 FOR DATE, TIME, AND TIMESTAMP EXAMPLES
00010A
00020A          R RECORD1
00030A            DATFLD1         L         DATFMT(*JUL)
00040A                                      ALIAS(A_DATE_JUL)
00050A            DATFLD2         L         DATFMT(*EUR)
00060A                                      ALIAS(A_DATE_EUR)
00070A            DATFLD3         L         DATFMT(*DMY) DATSEP('-')
00080A                                      ALIAS(A_DATE_DMY)
00090A            DATFLD4         L         DATSEP(' ')
00100A            TIMFLD1         T         TIMFMT(*ISO)
00110A                                      ALIAS(A_DATE_ISO)
00120A            TIMFLD2         T         TIMFMT(*USA)
00130A                                      ALIAS(A_DATE_USA)
00140A            TIMFLD3         T         TIMSEP(' ')
00150A            TIMFLD4         T         TIMSEP('.')
00160A            TSFLD1          Z         DFT('1998-02-27-08.15.22.000000')
```

現在の日付が 1990 年 6 月 21 日で、現行システムの日付形式値が MDY で、システム日付の分離文字値が '/' の場合は、DATFLD3 には 21-06-90 が入ります。 DATFLD4 には 06 21 90 が入ります。

現在の日付が 1990 年 6 月 21 日で、現行システムの日付形式値が MDY で、現行システムの分離文字値が / の場合は、DATFLD1 には 90/172 が入ります (1990 年の 172 番目の日)。 DATFLD2 には 21.06.1990 が入ります。

現在の時刻が午後 2 時で、システム時刻形式が hhmmss で、システム時刻分離文字 ':' の場合は、TIMFLD1 に 14.00.00 が入ります。 TIMFLD2 には 2:00 PM が入ります。

現在の時刻が午後 2 時で、システム時刻形式が hhmmss で、システム時刻分離文字 ':' の場合は、TIMFLD3 に 14 00 00 が入ります。 TIMFLD4 には 14.00.00 が入ります。

タイム・スタンプ・フィールドを定義する場合は、以下の形式でデフォルト値を指定する必要があります。

```
DFT('YYYY-MM-DD-HH.MM.SS.UUUUUU')
```

DFT キーワードが指定されない場合は、デフォルト値は現在の時刻です。

図 8-6. 日付、時刻、およびタイム・スタンプ・フィールドが定義された DDS ファイル

可変長フィールド: CRTCBLMOD または CRTBNDCBL コマンドの CVTOPT(*VARCHAR) パラメーター、または PROCESS ステートメントの VARCHAR オプションを指定する場合には、プログラムに可変長フィールドを取り入れることができます。外部記述ファイルから抽出する可変長フィールドは、プログラムの中で固定長グループ項目になります。

これらのフィールドについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き* 」を参照してください。

レコードをレコード域に明示的に移動する前に WRITE 操作を実行すると、多くの場合、16 進値 40 (X'40') を持つブランクを書き込むことになります。可変長フィールドの場合、これは、現行のフィールド長として X'4040' が使用されることを意味します。

X'4040' は 10 進値 16448 に変換され、おそらく、可変長フィールドに関して定義された最大の長さを上回ります。このため、WRITE 操作または後続の CLOSE 操作は失敗し、ファイル状況は 90 となります。

形式 2 の COPY ステートメントで REPLACING 句を使用する場合の考慮事項: レベル番号を含め、生成された COBOL ソースを置き換えるには、REPLACING 句を使用できます。 (詳細は 8-5 ページの『REPLACING 句』を参照してください。) ただし、以下の例外に注意する必要があります。

- RECORD KEY IS EXTERNALLY-DESCRIBED-KEY が指定されている場合には、REPLACING 句によって形式名、またはキーであるフィールドの名前を変更することはできません。

図 8-7 は、REPLACING オプションを**指定しない**、形式 2 の COPY ステートメントを説明しています。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL  TESTLIB/STRTEXTD   I-SERIES1  06/02/15 11:27:50   Page   2
STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付


 11    000100 FD  CUST-MASTER.
 12    000200 01  CUSTOMER-RECORD.
       000300*
       000400* COPY DDS  W I T H O U T  REPLACING OPTION
       000500*
 13    000600     COPY DDS-CUSMST OF TESTLIB-CUSMSTP.
       +000001*    I-O FORMAT:CUSMST     FROM FILE CUSMSTP    OF LIBRARY TESTLIB           CUSMST
       +000002*                          ORDER HEADER RECORD                               CUSMST
 14    +000003       05  CUSMST.                                                           CUSMST
 15    +000004           06 CUST              PIC X(5).                                    CUSMST
       +000005*                          CUSTOMER NUMBER                                   CUSMST
 16    +000006           06 NAME              PIC X(25).                                   CUSMST
       +000007*                          CUSTOMER NAME                                     CUSMST
 17    +000008           06 ADDR              PIC X(20).                                   CUSMST
       +000009*                          CUSTOMER ADDRESS                                  CUSMST
 18    +000010           06 CITY              PIC X(20).                                   CUSMST
       +000011*                          CUSTOMER CITY                                     CUSMST
 19    +000012           06 STATE             PIC X(2).                                    CUSMST
       +000013*                          STATE                                             CUSMST
 20    +000014           06 ZIP               PIC S9(5)        COMP-3.                     CUSMST
       +000015*                          ZIP CODE                                          CUSMST
```

図 8-7. REPLACING オプションを指定しない形式 2 の COPY ステートメント

次の図は、REPLACING オプションを**指定する**、形式 2 の COPY ステートメントを示します。

```
5722WDS V5R4M0  060210 LN  IBM ILE COBOL     TESTLIB/STRTEXTD     I-SERIES1  06/02/15 11:27:50   Page   2
STMT PL SEQNBR -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S コピー名   変更日付


 30    001000 FD  CUST-MASTER.
 31    001100 01  CUSTOMER-RECORD.
       001200*
       001300* COPY DDS  W I T H  REPLACING OPTION
       001400*
 32    001500     COPY DDS-CUSMST OF TESTLIB-CUSMSTP
 33    001600          REPLACING NAME BY ADDR-LINE-1
 34    001700                    ADDR BY ADDR-LINE-2
 35    001800                    CITY BY ADDR-LINE-3.
       +000001*    I-O FORMAT:CUSMST     FROM FILE CUSMSTP    OF LIBRARY TESTLIB           CUSMST
       +000002*                          ORDER HEADER RECORD                               CUSMST
 36    +000003       05  CUSMST.                                                           CUSMST
 37    +000004           06 CUST              PIC X(5).                                    CUSMST
       +000005*                          CUSTOMER NUMBER                                   CUSMST
 38    +000006           06 ADDR-LINE-1             PIC X(25).                           CUSMST
       +000007*                          CUSTOMER NAME                                     CUSMST
 39    +000008           06 ADDR-LINE-2             PIC X(20).                           CUSMST
       +000009*                          CUSTOMER ADDRESS                                  CUSMST
 40    +000010           06 ADDR-LINE-3             PIC X(20).                           CUSMST
       +000011*                          CUSTOMER CITY                                     CUSMST
 41    +000012           06 STATE             PIC X(2).                                    CUSMST
       +000013*                          STATE                                             CUSMST
 42    +000014           06 ZIP               PIC S9(5)        COMP-3.                     CUSMST
       +000015*                          ZIP CODE                                          CUSMST
```

図 8-8. REPLACING オプションを指定する形式 2 の COPY ステートメント

キー生成の例

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A               PHYSICAL FILE PF1 FOR KEY GENERATION EXAMPLES
     A
     A          R PFRECORD
     A
     A            MTH            2
     A            DAY            2
     A            YEAR           4
     A            ITEM           5
     A
     A
     A          K MTH
     A          K DAY
```

図 8-9. 物理ファイルのデータ記述仕様

図 8-9 に示されている物理ファイルが、次の例の基礎になっています。それぞれの例において、SELECT 文節中で EXTERNALLY-DESCRIBED-KEY を指定している論理ファイル (物理ファイルから得られる) を参照しています。

CONCAT キーワードを使用した例

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A               LOGICAL FILE LF1 FOR CONCAT KEYWORD EXAMPLES
     A
     A          R RECORD1                   PFILE(PF1)
     A
     A            DATE                      CONCAT(MTH DAY YEAR)
     A
     A          K MTH
     A          K DAY
```

図 8-10. CONCAT キーワードを使用したデータ記述仕様

図 8-10 に示されている論理ファイルに対して COPY DDS は、物理ファイルから得られるキーおよびキー名を生成します。

```
    FD LF1 LABEL RECORDS ARE STANDARD.
    01  LOG-RECORD.
                COPY DDS-ALL-FORMATS OF LF1.
        05  LF1-RECORD PIC X(8).
   *  I-O FORMAT:RECORD-1   FROM FILE LF1        OF LIBRARY COPYDDS
   *
   *THE KEY DEFINITIONS FOR RECORD FORMAT  RECORD1
   *  NUMBER              NAME               RETRIEVAL    TYPE   ALTSEQ
   *   0001  MTH-DDS                          ASCENDING     AN     NO
   *         KEY NAME ORIGINATES FROM PHYSICAL FILE
   *   0002  DAY-DDS-DDS                      ASCENDING     AN     NO
   *         KEY NAME ORIGINATES FROM PHYSICAL FILE
        05 RECORD1       REDEFINES LF1-RECORD.
           06 DATE-DDS              PIC X(8).
           06 FILLER REDEFINES DATE-DDS.
              07 MTH-DDS            PIC X(2).
              07 DAY-DDS-DDS        PIC X(2).
              07 FILLER             PIC X(4).
```

図 8-11. CONCAT キーワードを使用した例

MTH は物理ファイルから得られるキーであり、DATE は ILE COBOL の予約語なので、COPY ステートメントは、フィールド名 MTH および DATE に接尾部 -DDS を追加します。 DAY は物理ファイルから得られるキーであり、また ILE COBOL の予約語でもあるので、COPY ステートメントは、フィールド名 DAY に接尾部 -DDS を 2 つ追加します。

COPY ステートメントをファイル・セクションから作業用ストレージ・セクション、またはリンケージ・セクションへ移動する場合は、DATE-DDS に従属するフィールドが利用できなくなることに注意してください。

```
    WORKING-STORAGE SECTION.
    01  WRK-RECORD.
                COPY DDS-ALL-FORMATS OF LF1.
       05  LF1-RECORD PIC X(8).
   *  I-O FORMAT:RECORD-1   FROM FILE LF1         OF LIBRARY COPYDDS
   *
       05  RECORD1      REDEFINES LF1-RECORD.
           06 DATE-DDS             PIC X(8).
```

図 8-12. CONCAT キーワードを使用した例—作業用ストレージ・セクション

RENAME キーワードを使用した例

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A               LOGICAL FILE LF2 FOR RENAME KEYWORD EXAMPLES
     A
     A          R RECORD2                   PFILE(PF1)
     A
     A            MONTH                     RENAME(MTH)
     A
     A          K MTH
```

図 8-13. RENAMES キーワードを使用したデータ記述仕様

8-24 ページの図 8-9 に示されている論理ファイルに対して COPY DDS は、物理ファイルから得られるキーおよびキー名を生成します。

```
    *
     FD LF2 LABEL RECORDS ARE STANDARD.
     01  LOG-RECORD.
                 COPY DDS-ALL-FORMATS OF LF2.
       05  LF2-RECORD PIC X(2).
    *  I-O FORMAT:RECORD2    FROM FILE LF2         OF LIBRARY COPYDDS
    *
    *THE KEY DEFINITIONS FOR RECORD FORMAT  RECORD2
    *  NUMBER              NAME              RETRIEVAL    TYPE   ALTSEQ
    *   0001  MTH-DDS                         ASCENDING     AN     NO
    *         KEY NAME ORIGINATES FROM PHYSICAL FILE
        05  RECORD2      REDEFINES LF2-RECORD.
            06 MONTH                PIC X(2).
            06 MTH-DDS REDEFINES MONTH PIC X(2).
```

図 8-14. RENAME キーワードを使用

MTH は物理ファイルから得られるキーなので、COPY ステートメントは、フィールド名 MTH に接尾部 -DDS を追加します。

SST キーワードを使用した例

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A               LOGICAL FILE LF3 FOR SST KEYWORD EXAMPLES
     A
     A          R RECORD3                   PFILE(PF1)
     A
     A            YY                I      SST(YEAR 2 2)
     A
     A          K YY
```

図 8-15. SST キーワードを使用したデータ記述仕様

8-24 ページの図 8-9 に示されている論理ファイルに対して COPY DDS は、次の仕様を生成します。

```
    *
     FD LF3 LABEL RECORDS ARE STANDARD.
     01  LOG-RECORD.
                 COPY DDS-ALL-FORMATS OF LF3.
         05  LF3-RECORD PIC X(2).
    *  I-O FORMAT:RECORD3    FROM FILE LF3         OF LIBRARY COPYDDS
    *
    *THE KEY DEFINITIONS FOR RECORD FORMAT  RECORD3
    *  NUMBER              NAME              RETRIEVAL    TYPE   ALTSEQ
    *   0001  YY                              ASCENDING     AN     NO
        05  RECORD3      REDEFINES LF3-RECORD.
            06 YY                   PIC X(2).
```

図 8-16. SST キーワードを使用

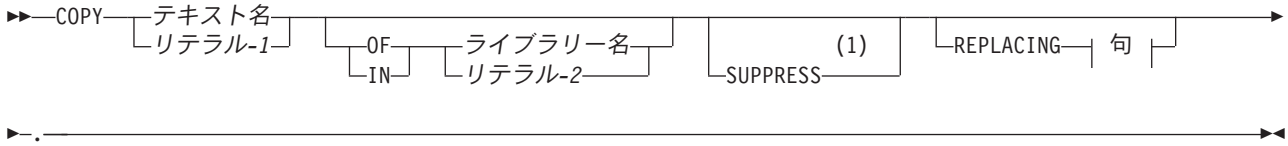
YY は物理ファイルから得られるキーではなく、また ILE COBOL の予約語でもないため、COPY ステートメントは、フィールド名 YY に接尾部を追加しません。

End of IBM Extension

COPY ステートメント - 形式 3 - 基本 IFS

IBM Extension

COPY ステートメント - 形式 3 - 基本 IFS



注:

1 IBM 拡張

literal-1 and literal-2

リテラル -1 は、コピーするストリーム・ファイルの名前です。ライブラリー名を省略すると、リテラルが直接、ファイル名、相対パス名、または絶対パス名として使用されます (最初の文字が「/」の場合)。例えば、次のとおりです。

```
COPY "MyInc"
COPY "x/MyInc"
COPY "/u/user1/MyInc"
COPY "/QSYS.LIB/QSYSINC.LIB/QCBLLESRC.FILE/JNI.MBR"
```

リテラル -2 は、コピー・ファイルのテキスト名またはリテラル -1 を探す、実際のパス、相対パスまたは絶対パスとして扱われます。

テキスト名

テキスト名がユーザー定義の COBOL ワードであり、その名前の環境変数が定義されている場合は、環境変数の値が、コピー・テキストを含むファイルの名前として使用されます。その名前の環境変数が定義されていない場合は、以下に指定された順番で、以下の名前に基づいて、コピー・テキストが検索されます。

1. text-name.cpy
2. text-name.CPY
3. text-name.cblle
4. text-name.CBLLE
5. text-name.cbllеinc
6. text-name.CBLLEINC
7. text-name.cbl
8. text-name.CBL
9. text-name.cob
10. text-name.COB
11. text-name.MBR
12. テキスト名

ライブラリー名

ライブラリー名がユーザー定義の COBOL ワードの場合は、その名前が環境変数として扱われます。環境変数の値は、コピー・ファイル、テキスト名、またはリテラル -1 を探すパスとして使用されます。環境変数を設定しないと、エラーが発生します。ライブラリー名とテキスト名の両方を指定すると、コンパイラーは、それら 2 つの値の間にパス分離文字 (/) を挿入し、ライブラリー名とテキスト名を連結することによって、コピー・テキストのパス名を作ります。例えば、COPY MYCOPY OF MYLIB について以下のように設定されているとします。

```
MYCOPY=mystuff/today.cpy
MYLIB=/u/user1
```

これらの設定は、以下のようになります。

```
/u/user1/mystuff/today.cpy
```

ライブラリー名が、コピー・テキストのコピー元のパスを示す環境変数の場合は、以下のような ADDENVVAR コマンドを使用して、ライブラリー名を定義します。

```
ADDENVVAR ENVVAR(COPYLIB) VALUE(/u/mystuff/copybooks)
```

環境変数の名前は英大文字でなければなりません。複数のコピー・ライブラリーを指定するには、環境変数を : (コロン) で区切られた複数のパス名に設定します。ライブラリー名が省略され、テキスト名が絶対パス名でない場合は、コピー・テキストは、以下の順番で検索されます。

1. 現行ディレクトリー
2. INCDIR パラメーターに指定したパス
3. SYSLIB 環境変数で指定したパス

End of IBM Extension

EJECT ステートメント

IBM Extension

EJECT ステートメントは、次のソース・ステートメントを次のページの一番上に印刷することを指定します。

EJECT ステートメント - 形式

EJECT ステートメントを指定する行には、他のステートメントを指定してはなりません。このステートメントは、区域 A と区域 B のどちらにでも書くことができ、分離文字ピリオドで終わらせることができます。

EJECT ステートメントはソース・プログラム自体のコンパイルには影響を与えません。

End of IBM Extension

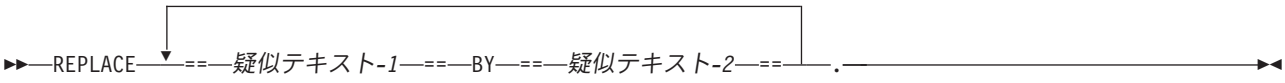
REPLACE ステートメント

REPLACE ステートメントは、ソース・プログラムのテキストを置き換えるために使用されます。

REPLACE ステートメントは、文字ストリングを指定できる場所であれば、ソース・プログラム内のどこにでも指定できます。このステートメントは、個別にコンパイルされるプログラムの最初のステートメントである場合を除き、前に分離文字ピリオドを付けなければなりません。分離文字ピリオドで終わらせる必要があります。

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。

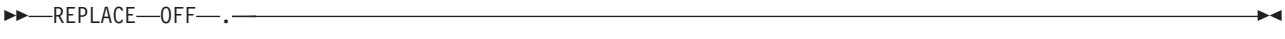
REPLACE ステートメント - 形式 1



一致するそれぞれの疑似テキスト -1 が、対応する疑似テキスト -2 によって置き換えられます。この処理は、次のいずれかが検出されるまで継続されます。

- REPLACE ステートメントの次のオカレンス
- プログラムの終わり
- REPLACE OFF (以下の形式 2 を参照してください)

REPLACE ステートメント - 形式 2



形式 2 は、形式 1 によって指定された現行のテキスト置き換えを終了させます。

疑似テキスト -1、疑似テキスト -2

疑似テキストは、疑似テキスト分離文字 (==) によって結合された (ただし分離文字は含まない) 一連のテキスト語、コメント行、または区切りスペースです。

疑似テキスト -1 は、分離文字コンマまたは分離文字セミコロン以外の、最低 1 個のテキスト語を含んでいなければなりません。最初と最後のスペースは、テキスト比較処理には含まれません、複数の組み込みスペースはシングル・スペースであると見なされます。

疑似テキスト -2 は、テキスト語を含んでいる必要はなく、また、スペース文字またはコメント行 (あるいはその両方) だけから構成できます。

疑似テキスト -1 には、分離文字で結合しなければならないテキスト語が必要であるため、疑似テキストをデータ名の一部 (例えば接頭部) を置き換えるのに使用することはできません。データ名全体が置き換えられる必要があります。

IBM Extension

疑似テキスト -1 または疑似テキスト -2 に、DBCS 文字ストリングまたは国別文字ストリングを含めることができます。 ただしこの場合、疑似テキストは 2 行にわたってはなりません。

End of IBM Extension

IBM Extension

REPLACE ステートメントが有効である場合には、形式 2 - DDS 変換 COPY ステートメントのレイアウトにはある特定の制限があります。 (8-9 ページの『COPY ステートメント - 形式 2 - DDS 変換』を参照。)

End of IBM Extension

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。
REPLACE ステートメントの処理の結果得られるテキストには、REPLACE ステートメントが入っていてはいけません。

置き換えのアルゴリズム

例えば、REPLACE ステートメントの中に 3 つの突き合わせられる疑似テキストの対があるとします。

1. 比較は、REPLACE ステートメントに続く左端のソース・プログラム・テキスト語と、最初の疑似テキスト -1 から開始されます。
2. 疑似テキスト -1 は、次の規則に従って、等しい数の連続したソース・プログラム・テキスト語と比較されます。
   - 比較は文字単位で行われます。
   - 大文字と小文字は同等です (リテラル内を除く)。
   - 分離文字コンマ、セミコロン、および 1 つまたは複数のスペースは、単一のスペースとして扱われます。
   - コメント行およびブランク行は、マッチングの目的では無視されます。

   IBM Extension

   - EJECT、SKIP 1/2/3、または TITLE ステートメントを含む行は、マッチングの目的では無視されます (コメント行として扱われます)。

   End of IBM Extension

   - デバッグ行は突き合わせのために処理されますが、標識域の D は無視されます。
3. 一致が検出されない場合には、一致が検出される (ステップ 5 に進む) まで、後続の疑似テキスト -1 それぞれについて比較が繰り返されます (この例では、3 つあります)。
4. 結局一致が検出されなかった場合には、次のソース・プログラム・テキスト語が左端のソース・プログラム・テキスト語として扱われ、サイクルが再びステップ 1 から始まります。
5. 一致が検出されると、対応する疑似テキスト -2 によってソース・プログラム内の一致したテキストが置き換えられます。
6. 一致に関与した右端のソース・プログラム・テキスト語の直後のテキスト語が、左端のソース・プログラム・テキスト語になります。サイクルは、最初に現れる疑似テキスト -1 から再び始まります (ステップ 1)。

プログラミング上の注意事項

SOURCE-COMPUTER 段落に WITH DEBUGGING MODE 文節が指定されていない場合には、すべての COPY および REPLACE ステートメントが処理された後、デバッグ行はコメント行のすべての特性を持つものと見なされます。

REPLACE ステートメントの処理の結果としてソース・プログラムに追加の行が取り入れられる場合、新しい行の標識域には、置き換えられるテキストの行と同じ文字が入ります。ただし、行にハイフンが入っていると、新しい行にはスペースが入ります。

疑似テキスト -2 の中のリテラルが、疑似テキスト -1 が入っている行に収まりきらず、そのリテラルがデバッグ行に置かれない場合には、リテラルの残りの部分を含む追加の継続行が取り入れられます。疑似テキスト -1 がデバッグ行にある場合には、プログラムがエラー状態になります。

SKIP1/2/3 ステートメント

IBM Extension

SKIP1/2/3 ステートメントは、ソース・リストを印刷するときにコンパイラーが追加しなければならないブランク行を指定します。 SKIP ステートメントはソース・プログラム自体のコンパイルには影響を与えません。

SKIP1/2/3 ステートメント - 形式

SKIP1
ブランク行を 1 つ指定します (2 行送り)。

SKIP2
ブランク行を 2 つ指定します (3 行送り)。

SKIP3
ブランク行を 3 つ指定します (4 行送り)。

SKIP1、SKIP2、または SKIP3 によって、2 行、3 行、または 4 行送りが 1 回発生します。

SKIP1、SKIP2、または SKIP3 は、区域 A または区域 B のどこにでも書くことができ、分離文字ピリオドで終わらせることができます。このステートメントを指定する行には、他のステートメントを指定してはなりません。

End of IBM Extension

TITLE ステートメント

IBM Extension

TITLE ステートメントは、コンパイル中に作り出されるソース・リストの各ページの上部に印刷されるタイトルを指定します。タイトル行は、コンパイラーと現行のリリース・レベルの ID が入った行の下に印刷されます。表題は表題行で左寄せされます。

TITLE ステートメント - 形式

リテラル
非数字でなければならず、後に分離文字ピリオドを書くことができます。表意定数であってはなりません。 DBCS リテラルまたは国別リテラルにすることができます。

TITLE ステートメントの働きを次に示します。

- 即座に強制的に改ページします。

- ソース・リストには印刷されません。
- コンパイルに対して他の影響は与えません。
- プログラムの実行には影響を与えません。

LIST オプションによって作り出されたリスト内の各ページごとに表題行が作り出されます。この表題行には、ソース・ステートメントの中で見つかった最後の TITLE ステートメントかまたはデフォルトが使用されます。

TITLE という語は区域 A と区域 B のいずれからでも始めることができます。

TITLE ステートメントは別の行へ継続させることはできません。

TITLE ステートメントはどの部のどの場所に書いてもかまいません。

TITLE ステートメントと同じ行に他のステートメントを書くことはできません。

End of IBM Extension

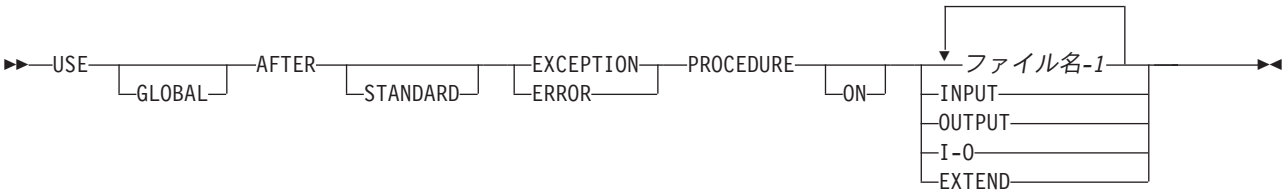
USE ステートメント

USE ステートメントは、システム定義のプロシージャーに加えて実行される、入出力例外またはエラー処理用のプロシージャーを指定します。 USE ステートメントはコンパイラー指示ステートメントですが、手続き部にしか指定できません。また、区域 B でだけ始めることができます。(GLOBAL 句の使用については 8-34 ページの『ネストされたプログラムについての優先順位規則』を参照してください。)

USE ステートメント - 形式 1 - EXCEPTION/ERROR

EXCEPTION と ERROR は同義であり、相互に交換して使用できます。

USE ステートメント - 形式



ファイル名-1

すべてのファイルに対して有効です。このオプションを指定すると、指定したファイルに対してだけプロシージャーが実行されます。ファイル名としてソート・ファイルまたはマージ・ファイルを参照することはできません。 1 つのファイルについては、1 つの EXCEPTION/ERROR プロシージャーしか指定できません。したがって、ファイル名の指定によって、複数の EXCEPTION/ERROR プロシージャーの実行要求が同時に起こってはなりません。ファイル名を指定した USE AFTER EXCEPTION/ERROR 宣言ステートメントは、ファイルのオープン・モードを指定した宣言ステートメントよりも優先されます。

IBM Extension

ファイル名句は TRANSACTION ファイルに対しても有効です。

End of IBM Extension

INPUT
すべてのファイルに対して有効です。このオプションを指定すると、INPUT モードでオープンされ、エラーが生じたすべてのファイルに対してプロシージャーが実行されます。

OUTPUT
すべてのファイルに対して有効です。このオプションを指定すると、OUTPUT モードでオープンされ、エラーが生じたすべてのファイルに対してプロシージャーが実行されます。

I-O
すべての直接アクセス・ファイルに対して有効です。このオプションを指定すると、I-O モードでオープンされ、エラーが生じたすべてのファイルに対してプロシージャーが実行されます。

IBM Extension

I-O 句は TRANSACTION ファイルに対しても有効です。

End of IBM Extension

EXTEND
このオプションを指定すると、EXTEND モードでオープンされ、エラーが生じたすべてのファイルに対してプロシージャーが実行されます。

EXCEPTION/ERROR プロシージャーが実行されるのは、次のいずれかのときです。

- システム定義の入出力エラー・ルーチンの完了後。
- INVALID KEY または AT END 句が入出力ステートメントの中で指定されていない場合に INVALID KEY または AT END 条件が検出されたとき。
- 状況キー 1 が 9 に設定されるような IBM 定義の条件が検出されたとき。 (7-37 ページの『状況キー』を参照。)

READ、WRITE、REWRITE、START、ACQUIRE、DROP、OPEN、または CLOSE ステートメントの実行中に入出力エラーが発生した場合には、EXCEPTION/ERROR プロシージャーが活動化されます。どの条件がエラーであるかを判別するには、 7-37 ページの『共通の処理機能』を参照してください。

EXCEPTION/ERROR 宣言プロシージャーの実行後は、エラーの原因となった入出力ステートメントの直後のステートメントに制御が戻されます。

宣言プロシージャーの中には、非宣言プロシージャーに対する参照があってはなりません。プログラムの非宣言部分には、EXCEPTION/ERROR 宣言プロシージャーの中で指定されているプロシージャー名に対する参照があってはなりません。ただし、例外として、PERFORM ステートメントでは、EXCEPTION/ERROR プロシージャーまたはそれに関連付けられているプロシージャーを参照できます。

EXCEPTION/ERROR 宣言プロシージャーの中には、以前に呼び出され、呼び出し側のルーチンへまだ制御を戻していない USE プロシージャーを実行させるステートメントを組み込むことはできません。

USE ステートメントのプログラミング上の注意事項

EXCEPTION/ERROR 宣言プロシージャーは、入出力エラーが発生するたびに状況キーの値をチェックするために使用できます。エラーの原因となったファイルに関する追加の情報は、簡略名、OPEN-FEEDBACK、および I-O-FEEDBACK からのデータを使用することによって得ることができます。

ファイルに対して EXCEPTION/ERROR 宣言プロシージャーを指定する際には、注意を払う必要があります。ファイルに対する最初の OPEN が正常に終了する前に、オブジェクト・プログラムによって現行の宣言が確立されることはありません。したがって、オープンされていないファイルに対して他の I-O ステートメントが実行された場合には、宣言は制御を受け取ることができません。ただし、このファイルが以前にオープンされたことがある場合には、直前に確立された宣言プロシージャーが制御を受け取ります。

例えば、OPEN OUTPUT ステートメントによってこのファイルに対する宣言プロシージャーが確立され、その後でエラーが起こらずにこのファイルがクローズされたものとします。後の処理の中で論理エラーが発生した場合には、このファイルが OUTPUT モードでオープンされたときに確立された宣言プロシージャーへ制御が渡されます。

エラー処理: 入出力エラーが発生したときに、適用可能なファイル状況文節がある (しかし、適用可能な USE プロシージャーはない) 場合には、ファイル状況が更新され、制御がプログラムに戻されます。エラーを処理するファイル状況文節、USE プロシージャー (暗黙または明示)、AT END 句、または INVALID KEY 句がない場合には、実行時メッセージが出されて、プログラムを終了させるかプログラムへ戻るかを選択することになります。

ネストされたプログラムについての優先順位規則

以下に、プログラムが他のプログラム内に含まれている場合の特殊な優先順位規則を示します。これらの規則を適用する際には、該当する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、その宣言の実行規則を満たさ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. 該当する条件を起こしたステートメントが入っているプログラム内のファイル固有の宣言 (GLOBAL 句を指定する、または指定しない *USE AFTER ERROR ON ファイル名-1* の形式)。
2. 該当する条件を起こしたステートメントが入っているプログラム内のモード固有の宣言 (GLOBAL 句を指定する、または指定しない *USE AFTER ERROR ON INPUT* の形式)。
3. GLOBAL 句が指定されており、該当する条件について最後に調べられたプログラムを直接に含むプログラム内にあるファイル特定宣言。
4. GLOBAL 句が指定されており、該当する条件について最後に調べられたプログラムを直接に含むプログラム内にあるモード特定宣言。
5. 規則 3 および 4 が、プログラムのネスト内の親に対して再帰的に適用されます。

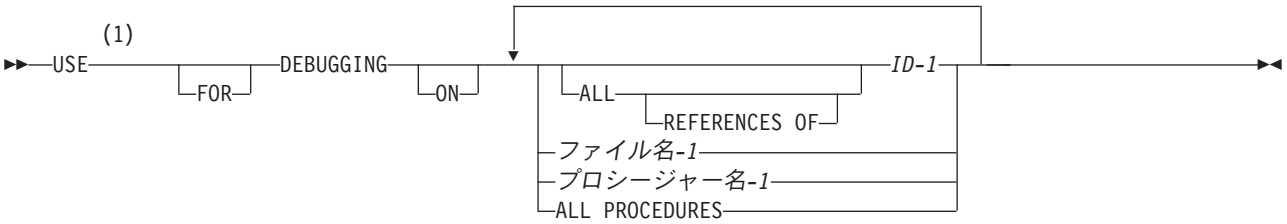
注: 各宣言プロシージャーは、同じ ILE COBOL プログラムの他の宣言プロシージャーおよび非宣言の呼び出しから独立する呼び出しとして実行されます。

USE FOR DEBUGGING

USE FOR DEBUGGING 宣言は、関連するデバッグ・プロシージャーによってモニターされる、ソース・プログラム内の項目を識別します。この宣言では、特定のエラーの発生時あるいは特定の項目またはファイルの変更時に実行すべきプロシージャーを確立します。

USE FOR DEBUGGING 宣言は、構文検査され、文書として扱われます。

USE FOR DEBUGGING 宣言 - 形式



注:

1 構文検査だけ行われます。

ID-1 を参照変更することはできません。

このステートメントは、デバッグ・モードにあるときにだけコンパイルされます。

コンパイラーは、次の有効な USE AFTER EXCEPTION ERROR ステートメントまたは END DECLARATIVES 分離文字に到達するまで、このステートメントに続くすべてのステートメントをコメントとして扱います。

付録

付録 A. ILE COBOL コンパイラー限界値

以下の表に、ILE COBOL コンパイラーでサポートされるコンパイラー限界値 をリストします。

表 9-1. ILE COBOL コンパイラー限界値

言語エレメント	ILE COBOL の限界値
一般	
以下の数: 一度にオープンされるファイル ネストされた COPY の中のネスト・レベル COPY の中の REPLACING オペランド	 実質的に限界はありません (1) 実質的に限界はありません (1) 実質的に限界はありません (1)
リテラルの全長	実質的に限界はありません (1)
VALUE 文節に使用可能な総ストレージ	実質的に限界はありません (1)
以下を識別する文字数: ライブラリー名 プログラム名 　　プログラム・オブジェクト 　　ILE プロシージャー テキスト名	 10 10 250 10
環境部 (ENVIRONMENT DIVISION)	
以下の数: 　SELECT ファイル名 1 つのファイル内の代替レコード・キー 代替レコード・キーを形成するために使用できる連続 する DDS フィールド	 実質的に限界はありません (1) 253 156
RESERVE 文節に指定されるバッファー (領域) の最大数	実質的に限界はありません (1)
以下の長さ: 1 つのファイル内の RECORD KEY 1 つのファイル内の ALTERNATE RECORD KEY	 2,000 バイト 2,000 バイト
データ部	

表 9-1. ILE COBOL コンパイラー限界値 (続き)

言語エレメント	ILE COBOL の限界値
以下の長さ: 作業用ストレージ・セクショングループ項目 リンケージ・セクションのグループ項目 ローカル・ストレージ・セクション 基本項目	 16,711,568 バイト 16,711,568 バイト 16,711,568 バイト 16,711,568 バイト
最大ブロック・サイズ	32,767 バイト
最大レコード長	32,767 バイト
以下の数: FD ファイル名 OCCURS レベル データ階層内のレベル SD ファイル名	 実質的に限界はありません (1) 7 49 実質的に限界はありません (1)
以下の数: 数字編集 (データ項目) の 　文字位置 ピクチャー文字ストリング ピクチャー複製	 127 90 16,711,568
OCCURS テーブル・サイズ (固定長) テーブル・サイズ (可変長) テーブル・エレメント・サイズ 1 つのテーブルの中の ASC/DESC KEY 文節 　の数 1 つのテーブルの中の ASC/DESC キー 　の数 指標名 (テーブル単位) INDEXED BY 文節 (テーブル単位) 1 つのテーブルの中のポインター	 16,711,568 バイト 16,711,568 バイト 16,711,568 バイト 実質的に限界はありません (1) 実質的に限界はありません (1) 実質的に限界はありません (1) 1 実質的に限界はありません (1)
手続き部	

表 9-1. ILE COBOL コンパイラー限界値 (続き)

言語エレメント	ILE COBOL の限界値
以下の数:	
GO TO プロシージャー名 DEPENDING ON	実質的に限界はありません (1)
ネストされた IF ステートメント	実質的に限界はありません (1)
IF のネスト・レベル	実質的に限界はありません (1)
ネストされた EVALUATE ステートメント	実質的に限界はありません (1)
CALL パラメーター	
プログラム・オブジェクトへの	255
ILE プロシージャーへの	400
組み込み関数の FUNCTION ネスト・レベルの限界	
SORT-MERGE 入力ファイル	123
SORT-MERGE 出力ファイル	32
SORT-MERGE キー	32
SEARCH ALL ... WHEN	2 000
比較条件	
UNSTRING 分離文字	実質的に限界はありません (1)
INSPECT TALLYING ID	実質的に限界はありません (1)
INSPECT REPLACING ID	実質的に限界はありません (1)
	実質的に限界はありません (1)
以下の長さ:	
SORT-MERGE キー	2 000 バイト

注:

1. 限界は、ハードウェア構成によって異なりますが、非常に大きな数です。ほとんどのアプリケーションではそれに達することはありません。

付録 B. 中間結果と算術精度

コンパイラーは、算術ステートメントを、演算子優先順位に従って実行される演算の連続として取り扱い、これらの演算の結果を入れるための中間フィールドを設定します。

中間結果が生成される可能性があるのは、次の場合です。

- ADD または SUBTRACT ステートメントで、verb の直後に複数のオペランドがある場合。
- COMPUTE ステートメントで、一連の算術演算または複数の結果フィールドを指定している場合。
- 算術式で、条件ステートメントおよび参照変更の指定に含まれている場合。
- GIVING オプションで、ADD、SUBTRACT、MULTIPLY、または DIVIDE ステートメントに対する複数の結果フィールドを持つ場合。
- ステートメントで、組み込み関数をオペランドとして使用している場合。

いつコンパイラーが固定小数点または浮動小数点演算を使用するのかの説明については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の章『データ項目の処理』を参照してください。

中間結果の精度の計算

コンパイラーはアルゴリズムを使用して、中間結果用に確保された整数桁および小数桁の数を判別します。

以下に、中間結果用に確保される整数桁および小数桁の数をコンパイラーが判別する方法について説明します。その中で次の省略語を使用します。

i 中間結果用に確保される整数桁の数。

d 中間結果用に確保される小数桁の数。

ROUNDED
ROUNDED オプションを使用する場合、より正確にするために、もう 1 つの整数または小数が、必要であれば付け加えられることがあります。最終結果だけが丸められます。中間結果は丸められません。正確に丸めることができる最大桁数は 62 桁です。

dmax 特定のステートメントの中で、次のうちの最大のもの。

- 最終結果フィールドに必要な小数桁の数。
- 任意のオペランド用に定義された小数桁の最大数。
- 任意の関数オペランドの outer-dmax。

inner-dmax
関数の inner-dmax は次のうちの最大のものです。

- 任意の基本引数用に定義された小数桁の数。
- 任意の算術式引数用の dmax。
- 任意の組み込み関数用の outer-dmax。

outer-dmax
関数結果が、それ自身の評価の外側にある演算に対して、どのような結果をもたらすかを決定する数 (例えば、その関数が算術式中のオペランドである、または他の関数への引数である場合)。

op1 生成された算術ステートメントの最初のオペランド。除算では、op1 は除数です。

op2 生成された算術ステートメントの 2 番目のオペランド。除算では、op2 は被除数です。

i1、i2 op1 と op2 のそれぞれの整数桁の数。

d1、d2
op1 または op2 用にそれぞれ定義された小数桁の数。

ir 生成された算術ステートメントまたは演算の処理から得られる中間結果フィールド。中間結果は、ir1、ir2 等で表されます。後続の中間結果が同じメモリー・ロケーションを共用することもあります。

COMPUTE ステートメントを使って、算術式での中間結果の使用方法を以下に示します。この場合、次のステートメントは、

```
COMPUTE Y = A + B * C - D / E + F ** G
```

以下のように置き換えられます。

F ** G		ir1 が生成される
MULTIPLY B	BY C	ir2 が生成される
DIVIDE E	INTO D	ir3 が生成される
ADD A	TO ir2	ir4 が生成される
SUBTRACT ir3	FROM ir4	ir5 が生成される
ADD ir5	TO ir1	Y が生成される

コンパイラーによる中間結果の計算

ir の中の整数桁の数は、次のように計算されます。

最初に、コンパイラーは、ir を生成するために使用されるオペランドのそれぞれに数字を割り当て、演算から得られる値を判別することにより、ir に入る可能性のある最大値を判別します。

- このステートメントのオペランドがデータ名の場合、そのデータ名に使用される値はそのデータ名の PICTURE の数字と等しくなります (すなわち、PICTURE 9V99 は値 9.99 をもちます)。
- オペランドがリテラルの場合、そのリテラルは PICTURE をもっているかのように取り扱われ、PICTURE の数字が使用されます (すなわち、リテラル +127.3 は暗黙の PICTURE S999V9 をもっています)。
- オペランドが中間結果の場合、前の演算で中間結果用に判別された PICTURE が使用されます。その PICTURE の数字が使用されます。
- 演算が除算である場合は、
  - op2 がデータ名である場合、op2 に使用される値は、そのデータ名の PICTURE 内の桁のゼロでない最小値となります (すなわち、PICTURE 9V99 は値 0.01 をもちます)。
  - op2 が中間結果である場合、その中間結果は PICTURE をもっているかのように取り扱われ、この PICTURE 内の桁のゼロでない最小値が使用されます。

上記のプロシージャーによって ir の最大値が判別されると、i は最大値の中の整数桁の数と等しく設定されます。

ir に含まれる小数桁の数は次のように計算されます。

表 9-2. 中間結果の精度の判別

演算	整数桁の数	小数桁の数
+ または -	(i1 または i2) + 1 のいずれか大きい方	d1 または d2 のいずれか大きい方
*	i1 + i2	d1 + d2
/	i2 + d1	(d2 - d1) または dmax のいずれか大きい方
**	**i2 が 0 と等しい場合** max(min(i1,18),1) (op2 が非整数の場合)[1] max(min(i1 * i1,18),1) (op2 が整数リテラルの場合)[1] **i2 が 0 と等しくない場合** max(min(i1 * (9 * i2),18),1) (op2 が非整数の場合)[1] max(min(i1 * i1 * (9 * i2),18),1) (op2 が整数リテラルの場合)[1]	op2 が非整数またはデータ名であれば、dmax、 op2 が整数リテラルであれば、 d1 * op2。
注: 1. これらの結果はそれ以降の処理の対象となります。		

最終結果で希望する精度を得るためには、任意の算術ステートメントのオペランドを十分な小数桁で定義しなければなりません。

9-6 ページの表 9-3 は、固定小数点数の中間結果を処理するときのコンパイラーの処置を示します。

表 9-3. コンパイラーが中間結果の切り捨てを行う時期の判別

i + d[2] の値	**d の値**	**i + dmax の値**	**処置**
< MAXLENGTH[1] = MAXLENGTH	任意の値	任意の値	整数桁 *i* と少数桁 *d* が ir 用に確保されます。
> MAXLENGTH[3]	< dmax = dmax	任意の値	MAXLENGTH - *d* 整数桁と *d* 小数桁が ir 用に確保されます。
	> dmax	< MAXLENGTH = MAXLENGTH	*i* 整数桁と MAXLENGTH - *i* 小数桁が ir 用に確保されます。
		> MAXLENGTH	MAXLENGTH - dmax 整数桁と dmax 小数桁が ir 用に確保されます。

注:

1. MAXLENGTH は次のいずれかの値です。
   - 10 進数の 18

   IBM Extension
   - (デフォルト) コンパイラー・オプション *NOEXTEND または PROCESS ステートメント・オプション NOEXTEND が指定されているときには、10 進数の 30。
   - 算術計算モード・コンパイラー・オプション *EXTEND31 または PROCESS ステートメント・オプション EXTEND31 が指定されている場合、10 進数の 31。
   - 算術計算モード・コンパイラー・オプション *EXTEND31FULL または PROCESS ステートメント・オプション EXTEND31FULL が指定されている場合、10 進数の 34。
   - 算術計算モード・コンパイラー・オプション *EXTEND63 または PROCESS ステートメント・オプション EXTEND63 が指定されているときには、10 進数の 63。

   End of IBM Extension
2. i + d の値が MAXLENGTH よりも小さい偶数である場合、コンパイラーは 1 を加算することによってそれを奇数に変換します。
3. i + d の値が 63 を超える場合は、ステートメントに SIZE ERROR 句が含まれていても、システム・メッセージ MCH1202 が出されることがあります。

中間結果フィールドが MAXLENGTH 桁を超える可能性があると考えられる場合には、浮動小数点オペランド (COMP-1 および COMP-2) を使用して、切り捨てを避けることができます。

整数関数

これらの関数は常に整数を戻し、outer-dmax は常にゼロになります。引数が整数でなければならない関数に関しては、inner-dmax は常にゼロになります。

表 9-4 は、関数結果の精度の要約です。

表 9-4. 整数組み込み関数の精度

機能	**Inner Dmax**	**Outer Dmax**	**関数結果**
DATE-OF-INTEGER	0	0	8 桁の整数
DATE-TO-YYYYMMDD	0	0	9 桁の整数
DAY-OF-INTEGER	0	0	7 桁の整数

表 9-4. 整数組み込み関数の精度 (続き)

機能	Inner Dmax	Outer Dmax	関数結果
DAY-TO-YYYYDDD	0	0	9 桁の整数
FIND-DURATION	適用外	0	9 桁の整数
INTEGER-OF-DATE	0	0	7 桁の整数
INTEGER-OF-DAY	0	0	7 桁の整数
LENGTH	適用外	0	9 桁の整数
ORD	適用外	0	3 桁の整数
ORD-MAX	適用外	0	9 桁の整数
ORD-MIN	適用外	0	9 桁の整数
YEAR-TO-YYYY	0	0	9 桁の整数

表 9-5 は、関数結果の精度の要約です。

表 9-5. 整数組み込み関数の精度

機能	Inner Dmax	Outer Dmax	関数結果
DATE-OF-INTEGER	0	0	8 桁の整数
DAY-OF-INTEGER	0	0	7 桁の整数
FACTORIAL	0	0	固定小数点、30 桁整数
INTEGER-OF-DATE	0	0	7 桁の整数
INTEGER-OF-DAY	0	0	7 桁の整数
LENGTH	適用外	0	9 桁の整数
MOD	0	0	min(i1 i2) と同じ桁数の整数
ORD	適用外	0	3 桁の整数
INTEGER		0	固定小数点引数を使用すると、結果はその引数より整数桁が 1 つ多い固定小数点整数になります。浮動小数点引数を使用すると、結果は固定小数点の 30 桁の整数になります。
INTEGER-PART		0	固定小数点引数を使用すると、結果はその引数と同じ整数桁の固定小数点整数になります。浮動小数点引数を使用すると、結果は固定小数点の 30 桁の整数になります。

表 9-6 は、関数結果の精度の要約です。

表 9-6. 整数組み込み関数の精度

機能	Inner Dmax	Outer Dmax	関数結果
DATE-OF-INTEGER	0	0	8 桁の整数
DAY-OF-INTEGER	0	0	7 桁の整数
FACTORIAL	0	0	固定小数点、30 桁整数
INTEGER-OF-DATE	0	0	7 桁の整数
INTEGER-OF-DAY	0	0	7 桁の整数
LENGTH	適用外	0	9 桁の整数
MOD	0	0	min(i1 i2) と同じ桁数の整数

表 9-6. 整数組み込み関数の精度 (続き)

機能	Inner Dmax	Outer Dmax	関数結果
ORD	適用外	0	3 桁の整数
ORD-MAX		0	9 桁の整数
ORD-MIN		0	9 桁の整数
INTEGER		0	固定小数点引数を使用すると、結果はその引数より整数桁が 1 つ多い固定小数点整数になります。浮動小数点引数を使用すると、結果は固定小数点の 30 桁の整数になります。
INTEGER-PART		0	固定小数点引数を使用すると、結果はその引数と同じ整数桁の固定小数点整数になります。浮動小数点引数を使用すると、結果は固定小数点の 30 桁の整数になります。

混合関数

コンパイラーが混合関数を固定小数点演算として処理すると、結果は、整数または小数部付きの固定小数点になります (いずれかの引数が浮動小数点である場合、関数は浮動小数点関数となり、浮動小数点の規則に従います)。 MAX、MIN、 RANGE、REM、および SUM に関しては、outer-dmax は常に inner-dmax と等しくなります。これらの関数に戻される結果の精度を判別するためには、関数結果を計算するために使用されるアルゴリズムの各ステップにおいて、固定小数点演算に関する規則を適用してください。

MAX:

1. 関数結果に最初の引数を割り当てます。
2. 残りのそれぞれの引数について、次のことを行います。
   a. 関数結果の代数値を引数と比較します。
   b. 2 つのうち大きい方を関数結果に割り当てます。

MIN:

1. 関数結果に最初の引数を割り当てます。
2. 残りのそれぞれの引数について、次のことを行います。
   a. 関数結果の代数値を引数と比較します。
   b. 2 つのうちの小さい方を関数結果に割り当てます。

RANGE:

1. MAX のステップを使用して、最大引数を選択します。
2. MIN のステップを使用して、最小引数を選択します。
3. 最大引数から最小引数を減算します。
4. 差を関数結果に割り当てます。

REM:

1. 引数-1 を引数-2 で除算します。
2. ステップ 1 の結果からすべての非整数桁を除去します。
3. ステップ 2 の結果に引数-2 を乗算します。
4. 引数-1 からステップ 3 の結果を減算します。

5. 差を関数結果に割り当てます。

SUM:

1. 関数結果に値 0 を割り当てます。
2. それぞれの引数について、次のことを行います。
   a. 関数結果に引数を加算します。
   b. 関数結果に合計を割り当てます。

浮動小数点データと中間結果

IBM Extension

浮動小数点命令は、以下の任意の条件が真の場合、算術式を計算するために使用されます。

- 式の受け入れ項目またはオペランドは、COMP-1、COMP-2、外部浮動小数点データ、または浮動小数点リテラルです。
- 組み込み数字関数は、浮動小数点関数です。
- 式は、浮動小数点関数の引数です。

算術式の任意の演算が浮動小数点で計算された場合、その式全体は、すべてのオペランドが浮動小数点に変換されたかのように、また演算が浮動小数点命令を使用して評価されたかのように、計算されます。

式が浮動小数点で計算される場合は、式の任意の受け入れ項目またはオペランドが COMP-1 でないとき、または乗算または指数演算が式に現れたとき、倍精度浮動小数点が使用されます。算術式の 1 つの演算で倍精度浮動小数点が使用された場合は、式のすべての演算が、倍精度浮動小数点命令が使用されたかのように計算されます。

浮動小数点の指数は、常に倍精度浮動小数点演算を使用して評価されます。

負の数の、小数の累乗の値は未定義になります。例えば、(-2) ** 3 は -8 になりますが、(-2) ** (3.000001) は未定義になります。指数が浮動小数点で評価され、その指数の値が未定義になる (上記の例のように) 可能性がある場合、その指数の値が実際に整数であるかどうかを判別するため、実行時に評価されます。

浮動小数点の数字関数は、常に倍精度浮動小数点値を戻します。浮動小数点関数および固定小数点関数のリストについては、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」の『数字関数のタイプ』を参照してください。

算術式は、算術ステートメント以外のコンテキストの中に出てきてもかまいません。例えば、算術式を IF ステートメントで使用できます。このようなステートメントでは、中間結果、浮動小数点、および倍精度浮動小数点に関する規則は、次の変更とともに適用されます。

- 省略形の IF ステートメントは、そのステートメントが省略されなかったかのように処理されます。
- 必要関係演算子が 2 つのオペランド (ここでは被比較数として参照) 間の比較を定義するために使用されている場合、明示的比較条件が存在します。片方または両方の被比較数が算術式である明示的比較条件において、中間結果に関する規則は、両方の被比較数の属性を考慮して判別されます。このことはつまり、dmax は、除数および指数を除く、いずれかの被比較数の任意のオペランドに対して定義された小数桁の最大数になるように定義されているということです。浮動小数点および倍精度浮動小数点に関する規則は、いずれかの被比較数の任意のオペランドが、COMP-1、COMP-2、外部浮動小数点データ、または浮動小数点リテラルである場合に、適用されます。

例えば、次のステートメントの場合、

```
IF operand-1 = expression-1 THEN . . .
```

オペランド -1 が COMP-2 で定義されているデータ名で、式 -1 が固定小数点オペランドのみを含んでいる場合、浮動小数点演算に関する規則は、式 -1 が浮動小数点オペランドと比較されているため、式 -1 に適用されます。

- 算術式と、データ項目または他の算術式のいずれかとの間の比較が、関係演算子を使用しないで定義されていた場合、明示的比較条件が存在するとは言えません。これらの場合、その比較は、明示的な演算子を持つ 1 つまたは複数の IF ステートメントとして書き直すことができます。したがって、それぞれの IF ステートメントは、上記で概説された明示的な比較条件についての規則に従います。例えば、次のステートメントの場合、

```
EVALUATE expression-1
  WHEN expression-2 THRU expression-3
  WHEN expression-4
  .
  .
  .
END-EVALUATE
```

同等の IF ステートメントは、次のようになります。

```
IF expression-1 >= expression-2 AND
   expression-1 <= expression-3
IF expression-1 = expression-4
   then for each IF statement, the comparands must be looked at
   to determine if all the arithmetic in that IF statement will
   be fixed-point or floating-point.
```

End of IBM Extension

付録 C. EBCDIC および ASCII 照合順序

この付録では、EBCDIC (拡張 2 進化 10 進交換コード) と ASCII (情報交換用米国標準コード) の両方の文字セット用の昇順の照合順序を示します。各文字の記号および意味の他に、順序数 (1 から始まる)、10 進表記、および 16 進表記が示されています。

EBCDIC 照合順序

順序数	記号	意味	10 進表記	16 進表記
65	␣	スペース	64	40
. . .				
75	¢	セント記号	74	4A
76	.	ピリオド、小数点	75	4B
77	<	不等号 (より小)	76	4C
78	(	左括弧	77	4D
79	+	プラス記号	78	4E
80	|	縦線、論理 OR	79	4F
81	&	アンパーサンド	80	50
. . .				
91	!	感嘆符	90	5A

順序数	記号	意味	10 進表記	16 進表記
92	$	円記号	91	5B
93	*	アスタリスク	92	5C
94	)	右括弧	93	5D
95	;	セミコロン	94	5E
96	¬	論理 NOT	95	5F
97	-	マイナス、ハイフン	96	60
98	/	スラッシュ	97	61
. . .				
108	,	コンマ	107	6B
109	%	パーセント記号	108	6C
110	_	下線	109	6D
111	>	不等号 (より大)	110	6E
112	?	疑問符	111	6F
. . .				
123	:	コロン	122	7A
124	#	番号記号、ポンド記号	123	7B
125	@	単価記号、推定記号	124	7C
126	'	アポストロフィ、プライム符号	125	7D
127	=	等号	126	7E
128	"	引用符	127	7F
. . .				
130	a		129	81
131	b		130	82
132	c		131	83
133	d		132	84
134	e		133	85
135	f		134	86
136	g		135	87
137	h		136	88
138	i		137	89
. . .				
146	j		145	91
147	k		146	92
148	l		147	93
149	m		148	94
150	n		149	95
151	o		150	96
152	p		151	97
153	q		152	98
154	r		153	99
. . .				

順序数	記号	意味	10 進表記	16 進表記
163	s		162	A2
164	t		163	A3
165	u		164	A4
166	v		165	A5
167	w		166	A6
168	x		167	A7
169	y		168	A8
170	z		169	A9
. . .				
194			193	C1
195	B		194	C2
196	C		195	C3
197	D		196	C4
198	E		197	C5
199	F		198	C6
200	G		199	C7
201	H		200	C8
202	I		201	C9
. . .				
210	J		209	D1
211	K		210	D2
212	L		211	D3
213	M		212	D4
214	N		213	D5
215	O		214	D6
216	P		215	D7
217	Q		216	D8
218	R		217	D9
. . .				
227	S		226	E2
228	T		227	E3
229	U		228	E4
230	V		229	E5
231	W		230	E6
232	X		231	E7
233	Y		232	E8
234	Z		233	E9
. . .				
241	0		240	F0
242	1		241	F1
243	2		242	F2

順序数	記号	意味	10 進表記	16 進表記
244	3		243	F3
245	4		244	F4
246	5		245	F5
247	6		246	F6
248	7		247	F7
249	8		248	F8
250	9		249	F9

ASCII 照合順序

順序数	記号	意味	10 進表記	16 進表記
1		ヌル	0	0
. . .				
33	␣	スペース	32	20
34	!	感嘆符	33	21
35	"	引用符	34	22
36	#	番号記号	35	23
37	$	円記号	36	24
38	%	パーセント記号	37	25
39	&	アンパーサンド	38	26
40	'	アポストロフィ、プライム符号	39	27
41	(	左括弧	40	28
42	)	右括弧	41	29
43	*	アスタリスク	42	2A
44	+	プラス記号	43	2B
45	,	コンマ	44	2C
46	-	ハイフン、マイナス	45	2D
47	.	ピリオド、小数点	46	2E
48	/	スラッシュ	47	2F
49	0		48	30
50	1		49	31
51	2		50	32
52	3		51	33
53	4		52	34
54	5		53	35
55	6		54	36
56	7		55	37
57	8		56	38
58	9		57	39
59	:	コロン	58	3A
60	;	セミコロン	59	3B

順序数	記号	意味	10 進表記	16 進表記
61	<	不等号 (より小)	60	3C
62	=	等号	61	3D
63	>	不等号 (より大)	62	3E
64	?	疑問符	63	3F
65	@	単価記号、推定記号	64	40
66			65	41
67	B		66	42
68	C		67	43
69	D		68	44
70	E		69	45
71	F		70	46
72	G		71	47
73	H		72	48
74	I		73	49
75	J		74	4A
76	K		75	4B
77	L		76	4C
78	M		77	4D
79	N		78	4E
80	O		79	4F
81	P		80	50
82	Q		81	51
83	R		82	52
84	S		83	53
85	T		84	54
86	U		85	55
87	V		86	56
88	W		87	57
89	X		88	58
90	Y		89	59
91	Z		90	5A
92	[	左大括弧	91	5B
93	\	逆スラッシュ	92	5C
94	]	右大括弧	93	5D
95	^	曲折アクセント記号、脱字記号	94	5E
96	_	下線	95	5F
97		アクサングラーブ、右プライム記号	96	60
98	a		97	61
99	b		98	62
100	c		99	63
101	d		100	64

順序数	記号	意味	10 進表記	16 進表記
102	e		101	65
103	f		102	66
104	g		103	67
105	h		104	68
106	i		105	69
107	j		106	6A
108	k		107	6B
109	l		108	6C
110	m		109	6D
111	n		110	6E
112	o		111	6F
113	p		112	70
114	q		113	71
115	r		114	72
116	s		115	73
117	t		116	74
118	u		117	75
119	v		118	76
120	w		119	77
121	x		120	78
122	y		121	79
123	z		122	7A
124	{	左中括弧	123	7B
125	ª	縦の分割線	124	7C
126	}	右中括弧	125	7D
127		波形記号	126	7E

付録 D. ILE COBOL 関数名およびコンテキストに依存した語のリスト

この節は、ILE COBOL のコンテキストに依存した語および関数名をすべてリストしたものです。

表示キー

以下のキーは、ILE COBOL 言語の関数名およびコンテキストに依存した語を識別します。

ブランク
標準 COBOL からの、ILE COBOL の関数名またはコンテキストに依存した語。

(1) 標準 COBOL に対する IBM 拡張である ILE COBOL の関数名またはコンテキストに依存した語。

(2) ILE COBOL コンパイラーで使用されていない、1985 (1989 年に改訂) ANSI 規格からの COBOL 関数名。

関数名

関数名	関数名	関数名
ACOS	ADD-DURATION (1)	ANNUITY
ASIN	ATAN	CHAR
CONVERT-DATE-TIME (1)	COS	CURRENT-DATE
DATE-OF-INTEGER	DATE-TO-YYYYMMDD (1)	DAY-OF-INTEGER
DAY-TO-YYYYDDD (1)	DISPLAY-OF	EXTRACT-DATE-TIME (1)
FACTORIAL	FIND-DURATION (1)	INTEGER
INTEGER-OF-DATE	INTEGER-OF-DAY	INTEGER-PART
LENGTH	LOCALE-DATE (1)	LOCALE-TIME (1)
LOG	LOG10	LOWER-CASE
MAX	MEAN	MEDIAN
MIDRANGE	MIN	MOD
NATIONAL-OF	NUMVAL	NUMVAL-C
ORD	ORD-MAX	ORD-MIN
PRESENT-VALUE	RANDOM	RANGE
REM	REVERSE	SIN
SQRT	STANDARD-DEVIATION	SUBTRACT-DURATION (1)
SUM	TAN	TEST-DATE-TIME
TRIM (1)	TRIML (1)	TRIMR (1)
UPPER-CASE	UTF8STRING (1)	VARIANCE
WHEN-COMPILED	YEAR-TO-YYYY (1)	

コンテキストに依存した語

IBM Extension

コンテキストに依存した語	コンテキスト
APPEND	`XML GENERATE FILE-STREAM APPEND data-1 FROM data-2`
DAYS	`MOVE FUNCTION ADD-DURATION(date-1 DAYS 90)` (SUBTRACT-DURATION、FIND-DURATION、および EXTRACT-DATE-TIME においても使用可能。)
DEFAULT	`SET LOCALE LC_ALL FROM DEFAULT`
HOURS	`MOVE FUNCTION ADD-DURATION(time-1 HOURS 90)` (SUBTRACT-DURATION、FIND-DURATION、および EXTRACT-DATE-TIME においても使用可能。)
LC_ALL	`SET LOCALE LC_ALL FROM DEFAULT`
LC_COLLATE	`SET LOCALE LC_COLLATE FROM DEFAULT`
LC_CURRENCY	`SET LOCALE LC_CURRENCY FROM DEFAULT`
LC_MESSAGES	`SET LOCALE LC_MESSAGES FROM DEFAULT`
LC_MONETARY	`SET LOCALE LC_MONETARY FROM DEFAULT`
LC_NUMERIC	`SET LOCALE LC_NUMERIC FROM DEFAULT`
LC_TIME	`SET LOCALE LC_TIME FROM DEFAULT`
LC_TYPE	`SET LOCALE LC_TYPE FROM DEFAULT`

コンテキストに依存した語	コンテキスト
MICROSECONDS	MOVE FUNCTION ADD-DURATION(time-1 MICROSECONDS 30) (SUBTRACT-DURATION、FIND-DURATION、および EXTRACT-DATE-TIME においても使用可能。)
MINUTES	MOVE FUNCTION ADD-DURATION(time-1 MINUTES 35) (SUBTRACT-DURATION、FIND-DURATION、および EXTRACT-DATE-TIME においても使用可能。)
MONTHS	MOVE FUNCTION ADD-DURATION(date-1 MONTHS 12) (SUBTRACT-DURATION、FIND-DURATION、および EXTRACT-DATE-TIME においても使用可能。)
OVERWRITE	XML GENERATE FILE-STREAM OVERWRITE data-1 FROM data-2
SECONDS	MOVE FUNCTION ADD-DURATION(time-1 SECONDS 30) (SUBTRACT-DURATION、FIND-DURATION、および EXTRACT-DATE-TIME においても使用可能。)
SYMBOL	CURRENCY IS "EUR" PICTURE SYMBOL "$"
TIMESTAMP	05 date-1 FORMAT TIMESTAMP (SPECIAL-NAMES 段落の組み込み関数 TEST-DATE-TIME および CONVERT-DATE-TIME でも検出される。)
YEARS	MOVE FUNCTION ADD-DURATION(date-1 YEARS 2) (SUBTRACT-DURATION、FIND-DURATION、および EXTRACT-DATE-TIME においても使用可能。)
YYYYDDD	ACCEPT id-1 FROM DATE YYYYDDD
YYYYMMDD	ACCEPT id-1 FROM DATE YYYYMMDD

End of IBM Extension

付録 E. ILE COBOL 予約語リスト

この節は、ILE COBOL の予約語をすべてリストしたものです。

表示キー

以下のキーは、ILE COBOL 言語の予約語を識別します。

ブランク
標準 COBOL からの ILE COBOL 予約語。

(1) 標準 COBOL に対する IBM 拡張である ILE COBOL 予約語。

(2) ILE COBOL コンパイラーによって使用されない標準 COBOL からの COBOL 予約語。インストール・システムで互換性を重視する場合は、これらの語を使用すべきではありません。これを使用すると、診断メッセージが出されます。

(3) 標準 COBOL の中になく、ILE COBOL コンパイラーによってサポートされていない COBOL 予約語。これを使用すると、診断メッセージが出されます。

予約語

予約語	予約語
ACCEPT	ACCESS
ACQUIRE (1)	ADD
ADDRESS (1)	ADVANCING
AFTER	ALIAS (1)
ALL	ALPHABET
ALPHABETIC	ALPHABETIC-LOWER
ALPHABETIC-UPPER	ALPHANUMERIC
ALPHANUMERIC-EDITED	ALSO
ALTER	ALTERNATE
AND	ANY (2)
ARE	AREA
AREAS	ARITHMETIC (3)
ASCENDING	ASSIGN
AT	ATTRIBUTE (1)
AUTHOR	AUTO (1)
AUTO-SKIP (1)	AUTOMATIC (3)
BACKGROUND-COLOR (1)	BACKGROUND-COLOUR (1)
B-AND (3)	BEEP (1)
BEFORE	BELL (1)
B-EXOR (3)	BINARY
BIT (3)	BITS (3)
BLANK	B-LESS (3)
BLINK (1)	BLOCK
B-NOT (3)	BOOLEAN (3)
B-OR (3)	BOTTOM
BY	CALL
CANCEL	CD (2)
CF (2)	CH (2)
CHARACTER	CHARACTERS
CLASS	CLOCK-UNITS
CLOSE	COBOL (2)
CODE	CODE-SET
COL (1)	COLLATING
COLUMN	COMMA
COMMIT (1)	COMMITMENT (1)
COMMON	COMMUNICATION (2)
COMP	COMP-0 (3)
COMP-1 (1)	COMP-2 (1)
COMP-3 (1)	COMP-4 (1)
# COMP-5 (1)	COMP-6 (3)
COMP-7 (3)	COMP-8 (3)
COMP-9 (3)	COMPUTATIONAL
COMPUTATIONAL-0 (3)	COMPUTATIONAL-1 (1)
COMPUTATIONAL-2 (1)	COMPUTATIONAL-3 (1)
# COMPUTATIONAL-4 (1)	COMPUTATIONAL-5 (1)
COMPUTATIONAL-6 (3)	COMPUTATIONAL-7 (3)
COMPUTATIONAL-8 (3)	COMPUTATIONAL-9 (3)
COMPUTE	CONFIGURATION
CONNECT (3)	CONSOLE (1)
CONTAINED (3)	CONTAINS
CONTENT	CONTINUE
CONTROL	CONTROL-AREA (1)

予約語
CONTROLS
COPY
CORRESPONDING
CRT (1)
CURRENCY
CURSOR (1)
DATE
DATE-WRITTEN
DAY-OF-WEEK
DB-ACCESS-CONTROL-KEY (3)
DB-EXCEPTION (3)
DB-RECORD-NAME (3)
DB-STATUS (3)
DBCS-EDITED (1)
DEBUG-CONTENTS
DEBUG-LINE
DEBUG-SUB-1
DEBUG-SUB-3
DECIMAL-POINT
DEFAULT (3)
DELIMITED
DEPENDING
DESCRIBED (1)
DETAIL (2)
DISCONNECT (3)
DISPLAY-1 (1)
DISPLAY-3 (3)
DISPLAY-5 (3)
DISPLAY-7 (3)
DISPLAY-9 (3)
DIVISION
DROP (1)
DUPLICATES
EBCDIC (1)
EJECT (1)
EMI (2)
EMPTY-CHECK (1)
END
END-ADD
END-COMPUTE
END-DISPLAY (1)
END-EVALUATE
END-INVOKE (1)
END-OF-PAGE
END-READ
END-RETURN
END-SEARCH
END-STRING
END-UNSTRING
END-XML (1)
ENTRY (1)
EOP
EQUALS (3)
ERROR

予約語
CONVERTING
CORR
COUNT
CRT-UNDER (1)
CURRENT (3)
DATA
DATE-COMPILED
DAY
DB (3)
DB-DATA-NAME (3)
DB-FORMAT-NAME (1)
DB-SET-NAME (3)
DBCS (1)
DE (2)
DEBUG-ITEM
DEBUG-NAME
DEBUG-SUB-2
DEBUGGING
DECLARATIVES
DELETE
DELIMITER
DESCENDING
DESTINATION (2)
DISABLE (2)
DISPLAY
DISPLAY-2 (3)
DISPLAY-4 (3)
DISPLAY-6 (3)
DISPLAY-8 (3)
DIVIDE
DOWN
DUPLICATE (3)
DYNAMIC
EGI (2)
ELSE
EMPTY (3)
ENABLE (2)
END-ACCEPT (1)
END-CALL
END-DELETE
END-DIVIDE
END-IF
END-MULTIPLY
END-PERFORM
END-RECEIVE (2)
END-REWRITE
END-START
END-SUBTRACT
END-WRITE
ENTER
ENVIRONMENT
EQUAL
ERASE (3)
ESI (2)

予約語
EVALUATE
EXCEEDS (3)
EXCLUSIVE (3)
EXTEND
EXTERNALLY-DESCRIBED-KEY (1)
FD
FILE
FILE-STREAM (1)
FILLER
FIND (3)
FIRST
FOR
FOREGROUND-COLOUR (1)
FREE (3)
FULL (1)
GENERATE
GIVING
GO
GREATER
HEADING (2)
HIGH-VALUE
I-O
ID (1)
IF
INDEX
INDEX-1 (3)
INDEX-3 (3)
INDEX-5 (3)
INDEX-7 (3)
INDEX-9 (3)
INDICATE
INDICATORS (1)
INITIALIZE
INPUT
INSPECT
INTO
INVOKE (1)
JUST
KANJI (1)
KEY
LAST
LEADING
LEFT-JUSTIFY (1)
LENGTH-CHECK (1)
LIBRARY (1)
LIMIT (2)
LINAGE
LINE
LINES
LOCALE (1)
LOCAL-STORAGE (1)
LOW-VALUE
MEMBER (3)
MERGE

予約語
EVERY
EXCEPTION
EXIT
EXTERNAL
FALSE
FETCH (3)
FILE-CONTROL
FILES (3)
FINAL (2)
FINISH (3)
FOOTING
FOREGROUND-COLOR (1)
FORMAT (1)
FROM
FUNCTION
GET (3)
GLOBAL
GOBACK (1)
GROUP (2)
HIGHLIGHT (1)
HIGH-VALUES
I-O-CONTROL
IDENTIFICATION
IN
INDEXED
INDEX-2 (3)
INDEX-4 (3)
INDEX-6 (3)
INDEX-8 (3)
INDIC (1)
INDICATOR (1)
INITIAL
INITIATE
INPUT-OUTPUT
INSTALLATION
INVALID
IS
JUSTIFIED
KEEP (3)
LABEL
LD (3)
LEFT
LENGTH
LESS
LIKE (1)
LIMITS (2)
LINAGE-COUNTER
LINE-COUNTER (2)
LINKAGE
LOCALLY (3)
LOCK
LOW-VALUES
MEMORY
METACLASS (1)

予約語
MODE
MODIFY (3)
MOVE
MULTIPLY
NATIONAL
NEGATIVE
NO
NONE (3)
NULL-KEY-MAP (1)
NULL (1)
NUMBER
NUMERIC-EDITED
OBJECT-COMPUTER
OF
OMITTED
ONLY (3)
OPTIONAL
ORDER
OTHER
OVERFLOW
PACKED-DECIMAL
PAGE
PARSE (1)
PF (2)
PICTURE
PIC
POSITION
PREFIX (1)
PRINTING
PROCEDURE
PROCEDURES
PROCESS (1)
PROGRAM-ID
PROGRAM
PURGE (2)
QUOTE
RANDOM
READ
REALM (3)
RECURSIVE (1)
RECORD
RECORDS
REEL
REFERENCE-MONITOR (3)
RELATION (3)
RELEASE
REMOVAL
REPEATED (3)
REPLACING
REPORTING (2)
REPOSITORY (1)
RERUN
RESET
RETRIEVAL (3)

予約語
MODIFIED (1)
MODULES
MULTIPLE
MESSAGE (2)
NATIVE
NEXT
NO-ECHO (1)
NOT
NULL-MAP (1)
NULLS (1)
NUMERIC
OBJECT (1)
OCCURS
オフ
ON
OPEN
OR
ORGANIZATION
OUTPUT
OWNER (3)
PADDING
PAGE-COUNTER (2)
PERFORM
PH (2)
PLUS (2)
POINTER
POSITIVE
PRESENT (3)
PRIOR (1)
PROCEDURE-POINTER (1)
PROCEED
PROCESSING (1)
PROMPT (1)
PROTECTED (3)
QUEUE (2)
QUOTES
RD (2)
READY (3)
RECEIVE (2)
RECONNECT (3)
RECORD-NAME (3)
REDEFINES
REFERENCE
REFERENCES
RELATIVE
REMAINDER
RENAMES
REPLACE
REPORT (2)
REPORTS (2)
REQUIRED (1)
RESERVE
RETAINING (3)
RETURN

予約語
RETURNING (1)
REVERSED
REWIND
RF (2)
RIGHT
ROLLBACK (1)
ROUNDED
SAME
SD
SECTION
SECURITY
SEGMENT-LIMIT
SEND (2)
SEPARATE
SEQUENTIAL
SHARED (3)
SIZE
SKIP2 (1)
SORT
SORT-RETURN (1)
SOURCE-COMPUTER
SPACE-FILL (1)
SPECIAL-NAMES
STANDARD-1
START
STATUS
STORE (3)
SUB-QUEUE-1 (2)
SUB-QUEUE-3 (2)
SUBFILE (1)
SUBTRACT
SUPPRESS
SYNC
SYSIN (1)
TABLE (2)
TAPE
TERMINAL
TEST
THAN
THROUGH
TIME
TITLE (1)
TOP
TRAILING-SIGN (1)
TRUE
TYPEDEF (1)
UNEQUAL (3)
UNSTRING
UP
UPON
USAGE-MODE (3)
USING
VALIDATE (3)
値

予約語
RETURN-CODE (1)
REVERSE-VIDEO (1)
REWRITE
RH (2)
RIGHT-JUSTIFY (1)
ROLLING (1)
RUN
SCREEN (1)
SEARCH
SECURE (1)
SEGMENT (2)
SELECT
SENTENCE
SEQUENCE
SET
SIGN
SKIP1 (1)
SKIP3 (1)
SORT-MERGE
SOURCE (2)
SPACE
SPACES
STANDARD
STANDARD-2
STARTING (1)
STOP
STRING
SUB-QUEUE-2 (2)
SUB-SCHEMA (3)
SUBSTITUTE (1)
SUM (2)
SYMBOLIC
SYNCHRONIZED
SYSOUT (1)
TALLYING
TENANT (3)
TERMINATE (2)
TEXT (2)
THEN
THRU
TIMES
TO
TRAILING
TRANSACTION (1)
TYPE
UNDERLINE (1)
UNIT
UNTIL
UPDATE (1)
USAGE
USE
VALID (3)
VALUE
VARYING

予約語	予約語
VLR (1)	WAIT (3)
WHEN	WHEN-COMPILED (1)
WITH	WITHIN (3)
WORDS	WORKING-STORAGE
WRITE	XML (1)
XML-CODE (1)	XML-EVENT (1)
XML-NTEXT (1)	XML-TEXT (1)
ZERO	
ZEROES	ZERO-FILL (1)
ZEROS	<
<=	+
*	**
-	/
>	>=
=	

付録 F. ファイル構造サポートの要約およびファイル状況キーの値

ファイル構造サポート・テーブル

9-24 ページの表 9-7 は、サポートされている各種のファイル構造についての必要およびオプショナルの記入項目をリストしたものです。装置タイプがディスクであるファイルは、データベースまたは非データベース補助記憶装置ファイルに割り当てることができます。使用されているコードは、以下のとおりです。

. 該当せず

B サブファイルをサポートするワークステーションの場合はオプショナル

C オプショナルの記入項目 (コメントとしてだけ扱われる)

D DATABASE- に割り当てられるファイルの場合はオプショナル。データベース・ファイルに割り当てられない場合は許可されない

I 入力または入出力用にオープンされたファイルの場合はオプショナル

J 入出力用にオープンされたファイルの場合はオプショナル

O オプショナル

R 必要

S サブファイルをサポートするワークステーションの場合は必要

X 必要 (構文検査されるが、文書として扱われる)

9-27 ページの表 9-8 および 9-28 ページの表 9-9 は、状況キーの値とその意味を示します。

ファイル構造サポートの要約

表 9-7. ファイル構造サポート

装置タイプ	プリンター	テープ	ディスク (順次アクセス)	ディスク (相対順次)	ディスク (相対ランダム)	ディスク (相対動的)	ディスク (索引順次)	ディスク (索引ランダム)	ディスク (索引動的)	ワークステーション	ディスケット	形式ファイル
環境部												
RERUN...RECORDS	C	C	C	C	C	C	C	C	C	C	C	C
SAME	O	O	O	O	O	O	O	O	O	O	O	O
AREA	C	C	C	C	C	C	C	C	C	C	C	C
RECORD AREA	O	O	O	O	O	O	O	O	O	O	O	O
SORT AREA	.	C	C	.	.	.	.	.	.	.	.	.
SORT MERGE AREA	.	C	C	.	.	.	.	.	.	.	.	.
MULTIPLE FILE TAPE	.	C	.	.	.	.	.	.	.	.	.	.
COMMITMENT CONTROL	.	.	D	D	D	D	D	D	D	.	.	.
SELECT	R	R	R	R	R	R	R	R	R	R	R	R
ASSIGN	R	R	R	R	R	R	R	R	R	R	R	R
OPTIONAL	.	.	I	I	I	I	.	.	.	.	.	.
ORGANIZATION	O	O	O	R	R	R	R	R	R	R	O	O
SEQUENTIAL	O	O	O	.	.	.	.	.	.	.	O	O
RELATIVE	.	.	.	R	R	R	.	.	.	.	.	.
INDEXED	.	.	.	.	.	.	R	R	R	.	.	.
TRANSACTION	.	.	.	.	.	.	.	.	.	R	.	.
ACCESS	O	O	O	O	R	R	O	R	R	O	O	O
SEQUENTIAL	O	O	O	O	.	.	O	.	.	O	O	O
RANDOM	.	.	.	.	R	.	.	R	.	.	.	.
DYNAMIC	.	.	.	.	.	R	.	.	R	S	.	.
RESERVE	C	C	C	C	C	C	C	C	C	.	C	C
RELATIVE KEY	.	.	.	O	R	R	.	.	.	S	.	.
RECORD KEY	.	.	.	.	.	.	R	R	R	.	.	.
DUPLICATES	.	.	.	.	.	.	D	D	D	.	.	.
FILE STATUS	O	O	O	O	O	O	O	O	O	O	O	O
CONTROL-AREA	.	.	.	.	.	.	.	.	.	O	.	.
データ部												
LABEL RECORDS	X	R	X	X	X	X	X	X	X	X	X	X
STANDARD	.	O	R	R	R	R	R	R	R	O	R	R
OMITTED	R	O	.	.	.	.	.	.	.	O	.	.
VALUE OF	C	C	C	C	C	C	C	C	C	C	C	C
BLOCK CONTAINS	O	O	O	O	O	O	O	O	O	O	O	O
RECORD CONTAINS	O	O	O	O	O	O	O	O	O	O	O	O
DATA RECORDS	O	O	O	O	O	O	O	O	O	O	O	O
CODE-SET	.	O	.	.	.	.	.	.	.	.	O	.
LINAGE	O	.	.	.	.	.	.	.	.	.	.	.
手続き部												
OPEN	R	R	R	R	R	R	R	R	R	R	R	R

表 9-7. ファイル構造サポート (続き)

装置タイプ	プリンター	テープ	ディスク (順次アクセス)	ディスク (相対順次)	ディスク (相対ランダム)	ディスク (相対動的)	ディスク (索引順次)	ディスク (索引ランダム)	ディスク (索引動的)	ワークステーション	ディスケット	形式ファイル
INPUT	.	O	O	O	O	O	O	O	O	.	O	.
OUTPUT	R	O	O	O	O	O	O	O	O	.	O	O
I-O	.	.	O	O	O	O	O	O	O	R	.	.
NO REWIND	.	I	.	.	.	.	.	.	.	.	.	.
REVERSED	.	I	.	.	.	.	.	.	.	.	.	.
EXTEND	.	O	O	.	.	.	.	.	.	.	.	O
CLOSE	R	R	R	R	R	R	R	R	R	R	R	R
REEL/UNIT	.	O	.	.	.	.	.	.	.	.	.	.
REMOVAL	.	O	.	.	.	.	.	.	.	.	.	.
NO REWIND	.	O	.	.	.	.	.	.	.	.	.	.
NO REWIND	.	O	.	.	.	.	.	.	.	.	.	.
WITH LOCK	O	O	O	O	O	O	O	O	O	O	O	O
READ	.	I	I	I	I	I	I	I	I	I	I	.
NEXT	.	.	.	.	.	I	.	.	I	.	.	.
FIRST	.	.	.	.	.	.	.	.	D	.	.	.
LAST	.	.	.	.	.	.	.	.	D	.	.	.
PRIOR	.	.	.	.	.	.	.	.	D	.	.	.
INTO	.	I	I	I	I	I	I	I	I	I	I	.
WITH NO LOCK	.	.	J	J	J	J	J	J	J	.	.	.
KEY IS	.	.	.	.	.	.	.	I	I	.	.	.
AT END	.	I	I	I	.	I	I	.	I	I	I	.
NOT AT END	.	I	I	I	.	I	I	.	I	I	I	.
INVALID KEY	.	.	.	.	I	I	.	I	I	B	.	.
NOT INVALID KEY	.	.	.	.	I	I	.	I	I	B	.	.
FORMAT	.	.	D	.	.	.	D	D	D	J	.	R
NULL-KEY-MAP	.	.	.	.	.	.	D	D	D	.	.	.
NULL-MAP	.	.	D	D	D	D	D	D	D	.	.	.
NEXT MODIFIED	.	.	.	.	.	.	.	.	.	B	.	.
SUBFILE	.	.	.	.	.	.	.	.	.	B	.	.
INDICATORS	.	.	.	.	.	.	.	.	.	J	.	.
TERMINAL	.	.	.	.	.	.	.	.	.	O	.	.
NO DATA	.	.	.	.	.	.	.	.	.	O	.	.
WRITE	O	O	O	O	O	O	O	O	O	O	O	O
FROM	O	O	O	O	O	O	O	O	O	O	O	O
INVALID KEY	.	.	.	O	O	O	O	O	O	B	.	.
NOT INVALID KEY	.	.	.	O	O	O	O	O	O	B	.	.
ADVANCING	O	.	.	.	.	.	.	.	.	.	.	.
AT END-OF-PAGE	O	.	.	.	.	.	.	.	.	.	.	.
NOT AT END-OF-PAGE	O	.	.	.	.	.	.	.	.	.	.	.

表 9-7. ファイル構造サポート (続き)

装置タイプ	プリンター	テープ	ディスク (順次アクセス)	ディスク (相対順次)	ディスク (相対ランダム)	ディスク (相対動的)	ディスク (索引順次)	ディスク (索引ランダム)	ディスク (索引動的)	ワークステーション	ディスケット	形式ファイル
FORMAT	.	.	D	.	.	.	D	D	D	R	.	R
NULL-KEY-MAP	.	.	.	.	.	.	D	D	D	.	.	.
NULL-MAP	.	.	D	D	D	D	D	D	D	.	.	.
STARTING	.	.	.	.	.	.	.	.	.	O	.	.
ROLLING	.	.	.	.	.	.	.	.	.	O	.	.
INDICATORS	.	.	.	.	.	.	.	.	.	O	.	.
SUBFILE	.	.	.	.	.	.	.	.	.	B	.	.
TERMINAL	.	.	.	.	.	.	.	.	.	O	.	.
START	.	.	.	O	.	O	O	.	O	.	.	.
KEY	.	.	.	O	.	O	O	.	O	.	.	.
INVALID KEY	.	.	.	O	.	O	O	.	O	.	.	.
NOT INVALID KEY	.	.	.	O	.	O	O	.	O	.	.	.
FORMAT	.	.	.	.	.	.	D	D	D	.	.	.
NULL-KEY-MAP	.	.	.	.	.	.	D	D	D	.	.	.
REWRITE	.	.	O	O	O	O	O	O	O	B	.	.
FROM	.	.	O	O	O	O	O	O	O	B	.	.
INVALID KEY	.	.	.	.	O	O	.	O	O	B	.	.
NOT INVALID KEY	.	.	.	.	O	O	.	O	O	B	.	.
FORMAT	.	.	.	.	.	.	.	D	D	B	.	.
NULL-KEY-MAP	.	.	.	.	.	.	D	D	D	.	.	.
NULL-MAP	.	.	D	D	D	D	D	D	D	.	.	.
INDICATORS	.	.	.	.	.	.	.	.	.	B	.	.
SUBFILE	.	.	.	.	.	.	.	.	.	S	.	.
TERMINAL	.	.	.	.	.	.	.	.	.	O	.	.
DELETE	.	.	.	O	O	O	O	O	O	.	.	.
NULL-KEY-MAP	.	.	.	.	.	.	D	D	D	.	.	.
INVALID KEY	.	.	.	.	O	O	.	O	O	.	.	.
NOT INVALID KEY	.	.	.	.	O	O	.	O	O	.	.	.
FORMAT	.	.	.	.	.	.	.	D	D	.	.	.
USE	O	O	O	O	O	O	O	O	O	O	O	O
EXCEPTION/ERROR	O	O	O	O	O	O	O	O	O	O	O	O
FOR DEBUGGING	O	O	O	O	O	O	O	O	O	O	O	O
COMMIT	.	.	D	D	D	D	D	D	D	.	.	.
ROLLBACK	.	.	D	D	D	D	D	D	D	.	.	.
ACQUIRE	.	.	.	.	.	.	.	.	.	O	.	.
DROP	.	.	.	.	.	.	.	.	.	O	.	.

戻りコードは、トランザクション入出力の後にシステムによって設定されますが、これには ICF ファイルまたは DISPLAY ファイルが必要とされます。

戻りコードについて詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」を参照してください。

表 9-8. ファイル状況キーおよび対応する戻りコード

ファイル状況キー	メジャー戻りコード	マイナー戻りコード	説明
00	00 03 08 09	xx xx (09 を除く) 00 00	正常終了 (操作が成功した)。 データが受信されなかった。 すでに活動状態のセッションまたは装置を 獲得しようとする操作が試みられた。 ファイルが OPEN OUTPUT 用に動的に作成された。 (動的ファイル作成の詳細は、 「*ILE COBOL プログラマーズ・ガイド*」に記載されている CRTCBLMOD コマンドの OPTION(*CRTF) パラメーターの 説明を参照。)
0A	02 03	xx 09	ジョブが取り消し中 (制御付き)。
10	11	00	送信勧誘されたプログラム装置からの読み取りが拒否された。 未処理の送信勧誘はない。
30	80	xx	永続的なシステム・エラー。セッションは終了した。
92	81	xx	永続的な装置エラーまたはセッション・エラー。
9C	82	xx	オープンまたは獲得が失敗した。セッションは開始されなかった。
9G	34	xx	装置またはセッションに対する出力例外。
9I	04	xx	装置またはセッションに対する出力例外。
9K	83	E0	形式が見つからない。
9N	83	xx (E0 を除く)	セッション・エラー。セッションはまだ活動状態。

ファイル状況キーの値および意味

エラー処理については、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」のセクション『エラーおよび例外処理』を参照してください。

ファイル状況キーの値

表 9-9. ファイル状況キーの値

高位桁	意味	低位桁	意味
0	正常終了	0	これ以上情報はありません。
		2	READ ステートメントが正常に実行されましたが、重複キーが検出されました。つまり、参照の現行キーのキー値が次のレコードのキー値と同じです。ファイル状況 02 を使用可能にする方法については、READ ステートメントの注意事項を参照してください。
		4	関連するファイル名の RECORD IS VARYING 文節によって認められている最大レコードよりも大きいか、または最小レコードよりも小さいレコードを読み取る試みがありました。
		5	OPEN ステートメントが正常に実行されましたが、参照されたオプショナル・ファイルが OPEN ステートメントの実行時に存在しませんでした。オープン・モードが I-O または EXTEND の場合は、ファイルが作成されています。CPF4101、CPF4102、CPF4103、CPF4207、CPF9812
		7	NO REWIND、 REEL/UNIT、または FOR REMOVAL 句が指定された CLOSE ステートメント、または NO REWIND 句が指定された OPEN ステートメントの場合に、参照されたファイルが非リール/装置メディア上にあります。
			ジョブが CL コマンド ENDJOB、PWRDWNSYS、 ENDSYS、または ENDSBS CPF4741 によって制御付きで終了されました。入力受け入れ操作、つまり送信勧誘されたプログラム装置からの READ 中に、エスケープ・メッセージが送信されました (複数装置リストだけ)。
		M	最後のレコードがサブファイルに書き込まれました。 CPF5003
		P	ファイルが正常にオープンされましたが、ヌル可能フィールドを含んでおり、ASSIGN 文節が ALWNULL および装置タイプ DATABASE を指定していません。
		Q	順次処理相対ファイルに対する CLOSE ステートメントが正常に実行されました。ファイルは *INZDLT および *NOMAX オプションを指定して作成されたため、それの境界は書き込まれたレコードの数に設定されています。
1	AT END 条件	0	順次 READ ステートメントが試みられましたが、ファイルの終わりに達しているため、ファイル内に次の論理レコードが存在しませんでした (未処理の送信勧誘はありません)。 CPF4740、CPF5001、CPF5025
		2	**IBM Extension** 変更サブファイル・レコードが見つかりません。 CPF5037 **End of IBM Extension**
		4	相対ファイルに対して順次 READ ステートメントが試みられましたが、相対レコード番号内の有効数字の数が、そのファイルについて記述された相対キー・データ項目のサイズを超えていました。

表 9-9. ファイル状況キーの値 (続き)

高位桁	意味	低位桁	意味
2	無効キー	1	順次アクセスの索引付きファイルにシーケンス・エラーが存在します。ある READ ステートメントが正常に実行されてから、そのファイルに対する次の REWRITE ステートメントが実行されるまでの間に、基本レコード・キー値がプログラムによって変更されているか、あるいは連続するレコード・キー値の昇順要件に対する違反がありました。 あるいは、重複キーを持つランダムまたは動的アクセス・ファイルに対する正常な READ 操作と後続の REWRITE または DELETE 操作との間に、レコード・キー値がプログラムによって変更されています。
		2	相対ファイルの中で重複キーを作成することになる、レコードの書き込みが試みられました。あるいは、索引付きファイルの中で重複基本レコード・キーを作成することになる、レコードの書き込みまたは再書き込みが試みられました。 CPF4759、CPF5008、CPF5026、CPF5034、CPF5084、CPF5085
		3	ファイル内に存在しないレコードにランダムにアクセスする試みがありました。 CPF5001、CPF5006、 CPF5013、CPF5020、CPF5025
		4	相対または索引付きファイルの外部定義境界を超えて書き込む試みがありました。あるいは、相対ファイルに対して順次 WRITE ステートメントが試みられましたが、相対レコード番号内の有効数字の数が、そのファイルについて記述された相対レコード・キー・データ項目のサイズを超えていました。 CPF5006、CPF5018、 CPF5021、CPF5043、CPF5272

表 9-9. ファイル状況キーの値 (続き)

高位桁	意味	低位桁	意味
3	永続エラー条件	0	これ以上情報はありません。 CPF4192、CPF5101、CPF5102、CPF5129、CPF5030、CPF5143
		4	境界違反のために永続エラーが存在します。順次ファイルの外部定義境界を超えて書き込む試みがありました。 CPF5116、CPF5018、CPF5272 (編成が順次の場合)
		5	INPUT、I-O、または EXTEND 句を指定した OPEN ステートメントが、存在しない非オプショナル・ファイルに対して試みられました。CPF4101、CPF4102、CPF4103、CPF4207、CPF9812
		7	ファイルに対して OPEN ステートメントが試みられましたが、その OPEN ステートメントで指定されたオープン・モードはそのファイルでサポートされません。次に示す違反の可能性があります。 • EXTEND または OUTPUT 句が指定されたが、ファイルは書き込み操作をサポートしない。 • I-O 句が指定されたが、ファイルは許可された入力および出力操作をサポートしない。 • INPUT 句が指定されたが、ファイルは読み取り操作をサポートしない。 CPF4194
		8	以前にロック付きでクローズされたファイルに対して OPEN ステートメントが試みられました。
		9	固定ファイル属性と、プログラム内でそのファイルについて指定された属性との間に矛盾が検出されたため、OPEN ステートメントが失敗しました。考えられる原因は次のとおりです。 • プログラムによって指定された最小レコード長が、そのファイルに必要な最小レコード長よりも小さいものでした。レベル検査エラー。CPF4131 • ファイルが、ALTERNATE RECORD KEY 文節を指定し、次のエラーのうち 1 つが検出されました。 1. 代替レコード・キーとして使用されるデータベース・ファイルのフィールド (複数のフィールドの場合もあり) が無効です。 2. データベース・ファイルが、分散データ管理機能 (DDM) ファイルです。 3. データベース・ファイルは、そのオープン・データ・パスが共有されることを許可しました。 4. プログラムの各キーに対して指定された DUPLICATES 文節が、データベース・ファイルの重複属性とは一致しませんでした。これには、基本キーも含まれます。

表 9-9. ファイル状況キーの値 (続き)

高位桁	意味	低位桁	意味
4	論理エラー条件	1	オープン・モードのファイルに対して OPEN ステートメントが試みられました。
		2	すでにクローズされたファイルに対して CLOSE ステートメントが試みられました。
		3	順次アクセス・モードの順次ファイルの場合、REWRITE ステートメントの前にファイルに対して実行された最後の入出力ステートメントが、正常に実行された READ ステートメントではありませんでした。順次アクセス・モードの相対および索引付きファイルの場合、 DELETE または REWRITE ステートメントの前にファイルに対して実行された最後の入出力ステートメントが、正常に実行された READ ステートメントではありませんでした。
		4	レコードのファイルへの再書き込みが試みられましたが、そのレコードが、置き換えられるレコードと同じサイズではなかったため、境界違反が存在します。関連するファイル名の RECORD IS VARYING 文節によって認められている最大レコードよりも大きいか、または最小レコードよりも小さいレコードを書き込みまたは再書き込みする試みがありました。
		6	入力または入出力モードでオープンされたファイルに対して順次 READ、READ NEXT、または READ PRIOR ステートメントが試みられましたが、前の START ステートメントが失敗したか、あるいは前の READ ステートメントが失敗または AT END 条件を起こしたため、有効な次のレコードが確立されていません。 CPF5001、CPF5025、CPF5183
		7	入力または入出力モードでオープンされていないファイルに対して READ または START ステートメントの実行が試みられました。
		8	出力または拡張モードでオープンされていない順次ファイルに対して WRITE ステートメントの実行が試みられました。入出力、出力、拡張モードでオープンされていない索引付きまたは相対ファイルに対して WRITE ステートメントの実行が試みられました。
		9	入出力モードでオープンされていないファイルに対して DELETE または REWRITE ステートメントの実行が試みられました。

表 9-9. ファイル状況キーの値 (続き)

高位桁	意味	低位桁	意味
9	その他のエラー	0	その他のエラー • ファイルが見つからない • メンバーが見つからない • ファイルが、ALTERNATE RECORD KEY 文節を指定し、次のエラーのうち 1 つが検出されました。 1. 代替レコード・キー・オープン ID と既存のものとの間で、矛盾が検出されました。 2. 永続索引が検出されず、CRTARKIDX オプションが指定されていませんでした。 3. 代替レコード・キーを形成するために使用される隣接する DDS フィールドの最大数 (156) を超過しました。 • 予期しない入出力例外 ファイル (OPEN OUTPUT の非オプショナル・ファイル) について USE が適用可能な場合は、CPF4101、 CPF4102、CPF4103。一般に、次の例外がモニターされます。 • CPF4101 ～ CPF4399 • CPF4501 ～ CPF4699 • CPF4701 ～ CPF4899 • CPF5001 ～ CPF5099 • CPF5101 ～ CPF5399 • CPF5501 ～ CPF5699 これらの例外が見つかると、FILE STATUS が 90 に設定されます。
		1	未定義または無許可のアクセス・タイプ。CPF2207、 CPF4104、CPF4236、CPF4238、CPF5057、CPF5109、 CPF5134、CPF5279
		2	論理エラー • ファイルがロックされている • ファイルがすでにオープンされている • クローズされたファイルに対する入出力 • ファイルの終わりの後の READ • オープンされていないファイルに対する CLOSE CPF4106、CPF4132、CPF4740、CPF5067、CPF5070、 CPF5119、CPF5145、CPF5146、CPF5149、CPF5176、 CPF5209
		4	順次アクセスで**なく**、最後の操作が成功した READ でなかった場合に、ファイル位置標識のない REWRITE/DELETE が試みられました。

表 9-9. ファイル状況キーの値 (続き)

高位桁	意味	低位桁	意味
9	その他のエラー	5	無効または不完全なファイル情報 (1) COBOL プログラム内に重複キーが指定されています。ファイルは正常にオープンされていますが、固有のキーを持つ索引付きデータベース・ファイルが作成されました。または (2) COBOL プログラム内に重複キーが指定されておらず、重複キーを認めるデータベース・ファイルが作成されました。
		9	未定義 (表示または ICF)。
		C	獲得が失敗しました。セッションは開始されませんでした。
		D	レコードがロックされています。CPF5027、CPF5032
		G	装置またはセッションに対する出力例外。
		H	ACQUIRE 操作が失敗しました。リソースが別のプログラムに所有されているか、または使用不可でした。 (9H は、ACQUIRE 操作によって、90 についてモニターされるオペレーティング・システムの例外、または 9N が発生した場合の結果です。)
		I	WRITE 操作が失敗しました。 CPF4702、CPF4737、 CPF5052、CPF5076
		K	無効な形式名。形式が見つかりません。CPF5022、 CPF5023、CPF5053、CPF5054、CPF5121、CPF5152、 CPF5153、CPF5186、CPF5187

表 9-9. ファイル状況キーの値 (続き)

高位桁	意味	低位桁	意味
9	その他のエラー	N	一時的な (回復の可能性がある) ハードウェア入出力エラー。 (通信セッション中のエラー。) CPF4145, CPF4146, CPF4193, CPF4229, CPF4291, CPF4299, CPF4354, CPF4526, CPF4542, CPF4577, CPF4592, CPF4602, CPF4603, CPF4611, CPF4612, CPF4616, CPF4617, CPF4622, CPF4623, CPF4624, CPF4625, CPF4628, CPF4629, CPF4630, CPF4631, CPF4632, CPF4705, CPF5013, CPF5107, CPF5128, CPF5166, CPF5198, CPF5280, CPF5282, CPF5287, CPF5293, CPF5352, CPF5353, CPF5517, CPF5524, CPF5529, CPF5530, CPF5532, CPF5533, CPF5257.
		P	ファイルをコミットメント制御下に置くことができないために、OPEN が失敗しました。 CPF4293、CPF4326、 CPF4327、CPF4328、CPF4329
		Q	ランダムまたは動的アクセスの相対ファイルに対する OPEN ステートメントが、ファイルのサイズが *NOMAX であるために失敗しました。ファイルのサイズを必要なサイズに変更し (例えば、CHGPF を使用して)、プログラムを再実行依頼してください。
		R	参照保全エラー。CPF502D、CPF502E、CPF503A
		S	最後の READ 操作で NO LOCK が指定されたため、REWRITE または DELETE が失敗しました。
		T	トリガー・プログラム例外。CPF502B
		U	READ NEXT からのブロックにレコードが残っているために READ PRIOR が完了できないか、またはその逆です。 CPF5184 ファイルをいったんクローズしてから、もう一度オープンしてください。
		W	制約の例外をチェックしてください。CPF502F
		X	マルチスレッド・ジョブではそのファイル・タイプをサポートしていないため、OPEN が失敗しました。ファイル・タイプを DATABASE、PRINTER (スプール・ファイルのみ) に変更するか、タイプ *IP の DDM ファイルに変更して、プログラムを再実行依頼してください。CPF4380
		Y	ファイルが置かれている補助ストレージ・プール (ASP) 装置が使用できないため、OPEN が失敗しました。CPF980B.

属性データ形式

属性データのレイアウトおよび値は、システムによって異なります。以下の形式は ILE COBOL 言語で使用されます。

装置タイプおよびレイアウトの完全なリストについては、「**IBM i Information Center**」(Web サイト http://www.ibm.com/systems/i/infocenter/) 内のカテゴリー『データベースおよびファイル・システム』のセクション『*DB2 for i*』を参照してください。

```
 01  DISPLAY-ICF-ATTRIBUTES.
     02  PROGRAM-DEVICE-NAME        PIC X(10).
     02  DEVICE-DESCRIPTION-NAME    PIC X(10).
     02  USER-ID                    PIC X(10).
     02  DEVICE-CLASS               PIC X.
*          D - DISPLAY
*          I - ICF
*          U - UNKNOWN
     02  DEVICE-TYPE                PIC X(6).
*          '      ' - UNKNOWN
```

```
*          '3179  ' - 3179 DISPLAY
*          '317902' - 3179 MOD 2 DISPLAY
*          '3180  ' - 3180 DISPLAY
*          '3196A ' - 3196 MOD A1/A2 DISPLAY
*          '3196B ' - 3196 MOD B1/B2 DISPLAY
*          '3197C1' - 3197 MOD C1 DISPLAY
*          '3197C2' - 3197 MOD C2 DISPLAY
*          '3197D1' - 3197 MOD D1 DISPLAY
*          '3197D2' - 3197 MOD D2 DISPLAY
*          '3197W1' - 3197 MOD W1 DISPLAY
*          '3197W2' - 3197 MOD W2 DISPLAY
*          '3270  ' - 3270 DISPLAY
*          '3476EA' - 3476 MOD EA DISPLAY
*          '3476EC' - 3476 MOD EC DISPLAY
*          '3477FG' - 3477 MOD FG DISPLAY
*          '3477FA' - 3477 MOD FA DISPLAY
*          '3477FC' - 3477 MOD FC DISPLAY
*          '3477FD' - 3477 MOD FD DISPLAY
*          '3477FW' - 3477 MOD FW DISPLAY
*          '3477FE' - 3477 MOD FE DISPLAY
*          '525111' - 5251 DISPLAY
*          '5291  ' - 5291 DISPLAY
*          '5292  ' - 5292 DISPLAY
*          '529202' - 5292 MOD 2 DISPLAY
*          '5555B1' - 5555 MOD B01 DISPLAY
*          '5555C1' - 5555 MOD C01 DISPLAY
*          '5555E1' - 5555 MOD E01 DISPLAY
*          '5555F1' - 5555 MOD F01 DISPLAY
*          '5555G1' - 5555 MOD G01 DISPLAY
*          '5555G2' - 5555 MOD G02 DISPLAY
*          '3486BA' - 3486 MOD BA DISPLAY
*          '3487HA' - 3487 MOD HA DISPLAY
*          '3487HG' - 3487 MOD HG DISPLAY
*          '3487HW' - 3487 MOD HW DISPLAY
*          '3487HC' - 3487 MOD HC DISPLAY
*          'DHCF77' - 3277 DHCF DISPLAY
*          'DHCF78' - 3278 DHCF DISPLAY
*          'DHCF79' - 3279 DHCF DISPLAY
*          'APPC  ' - ADVANCED-PROGRAM-TO-PROGRAM COMMUNICATIONS DEVICE
*          'ASYNC ' - ASYNCHRONOUS COMMUNICATION DEVICE
*          'BSC   ' - BISYNCHRONOUS COMMUNICATION
*          'BSCEL ' - BSCEL COMMUNICATION DEVICE
*          'FINANC' - ICF FINANCE COMMUNICATION DEVICE
*          'INTRA ' - INTRA SYSTEMS COMMUNICATION
*          'LU1   ' - LU1 COMMUNICATION DEVICE
*          'RETAIL' - RETAIL COMMUNICATION DEVICE
*          'SNUF  ' - SNA UPLINE FACILITY COMMUNICATION DEVICE
*          'NVT   ' - NETWORK VIRTUAL TERMINAL (NVT)
     02  REQUESTOR-DEVICE           PIC X.
*          N - NOT A REQUESTOR DEVICE
*          Y - A REQUESTOR DEVICE
     02  ACQUIRE-STATUS             PIC X.
*          N - DEVICE NOT ACQUIRED
*          Y - DEVICE ACQUIRED
     02  INVITE-STATUS              PIC X.
*          N - DEVICE NOT INVITED
*          Y - DEVICE INVITED
     02  DATA-AVAILABLE-STATUS      PIC X.
*          N - NO DATA IS AVAILABLE
*          Y - INVITED DATA AVAILABLE
     02  DISPLAY-DIMENSIONS.
         03  NUMBER-OF-ROWS         PIC S9(4)  COMP-4.
         03  NUMBER-OF-COLUMNS      PIC S9(4)  COMP-4.
     02  DISPLAY-ALLOW-BLINK        PIC X.
*          N - NOT BLINK CAPABLE
*          Y - BLINK CAPABLE
     02  ONLINE-OFFLINE-STATUS      PIC X.
```

```
*          O - DISPLAY IS ONLINE
*          F - DISPLAY IS OFFLINE
     02  DISPLAY-LOCATION           PIC X.
*          L - LOCAL DISPLAY
*          R - REMOTE DISPLAY
     02  DISPLAY-TYPE               PIC X.
*          A - ALPHANUMERIC OR KATAKANA
*          I - IDEOGRAPHIC
*          G - GRAPHIC DBCS
     02  KEYBOARD-TYPE              PIC X.
*          A - ALPHANUMERIC OR KATAKANA KEYBOARD
*          I - IDEOGRAPHIC KEYBOARD
     02  CONVERSATION-STATUS        PIC X.
*          N - CONVERSATION NOT INITIATED
*          Y - CONVERSATION INITIATED
*          (VALID FOR ALL COMMUNICATION TYPES).
     02  SYNCHRONIZATION-LEVEL      PIC X.
*          0 - SYNCHRONIZATION LEVEL 0  (SYNLVL(*NONE))
*          1 - SYNCHRONIZATION LEVEL 1  (SYNLVL(*CONFIRM))
*          (APPC APPLICATIONS ONLY)
*          2 - SYNCHRONIZATION LEVEL 2  (SYNLVL(*COMMIT))
     02  CONVERSATION-USED          PIC X.
*          M - MAPPED CONVERSATION
*          B - BASIC CONVERSATION
*          (APPC APPLICATIONS ONLY)
     02  REMOTE-LOCATION-NAME       PIC X(8).
*          (ALL COMMUNICATION TYPES)
     02  LOCAL-LU-NAME              PIC X(8).
*          (APPC APPLICATIONS ONLY)
     02  LOCAL-NETWORK-ID           PIC X(8).
*          (APPC APPLICATIONS ONLY)
     02  REMOTE-LU-NAME             PIC X(8)
*          (APPC APPLICATIONS ONLY)
     02  REMOTE-NETWORK-ID          PIC X(8).
*          (APPC APPLICATIONS ONLY)
     02  MODE                       PIC X(8).
*          (APPC APPLICATIONS ONLY)
     02  WORKSTATION-CONTROLLER     PIC X.
*          N - NOT ATTACHED
*          1 - ATTACHED TO CONTROLLER 1
*          2 - ATTACHED TO CONTROLLER 2
*          3 - ATTACHED TO CONTROLLER 3
     02  DISPLAY-IS-COLOR           PIC X.
*          Y - YES
*          N - NO
     02  DISPLAY-ALLOWS-GRID-LINES  PIC X.
*          N - NO
*          1 - YES
     02  LU6-CONVERSATION-STATE     PIC X.
*          '00'X - RESET
*          '01'X - SEND
*          '02'X - DEFER RECEIVED
*          '03'X - DEFER DEALLOCATE
*          '04'X - RECEIVE
*          '05'X - CONFIRM
*          '06'X - CONFIRM SEND
*          '07'X - CONFIRM DEALLOCATE
*          '08'X - COMMIT
*          '09'X - COMMIT SEND
*          '0A'X - COMMIT DEALLOCATE
*          '0B'X - DEALLOCATE
*          '0C'X - ROLLBACK REQUIRED
     02  LU6-CONVERSATION-CORRELATE PIC X(8).
     02  FILLER                     PIC X(31).
*          RESERVED
     02  CALLING-PARTY-ID.
         03  REMOTE-NUMBER-LENGTH       PIC S9(4)    COMP-4.
```

```
         03  REMOTE-NUMBERING-TYPE      PIC X(2).
*              00 - UNKNOWN
*              01 - INTERNATIONAL
*              02 - NATIONAL
*              03 - NETWORK-SPECIFIC
*              04 - SUBSCRIBER
*              06 - ABBREVIATED
*              07 - RESERVED
         03  REMOTE-NUMBERING-PLAN      PIC X(2).
*              00 - UNKNOWN
*              01 - ISDN/TELEPHONY
*              03 - DATA
*              04 - TELEX
*              08 - NATIONAL STANDARD
*              09 - PRIVATE
*              15 - RESERVED
        03  REMOTE-NUMBER              PIC X(40).
        03  FILLER                     PIC X(4).
*              RESERVED
        03  REMOTE-SUBADDR-LENGTH      PIC S9(4)    COMP-4.
        03  REMOTE-SUBADDR-TYPE        PIC X(2).
*              00 - NSAP
*              02 - USER SPECIFIED
        03  REMOTE-SUBADDRESS          PIC X(40).
        03  FILLER                     PIC X.
*              RESERVED
        03  CALL-TYPE                  PIC X.
*              0 - CALL IN
*              1 - CALL OUT
*              2 - NON-ISDN
        03  REMOTE-NETADDR-LENGTH      PIC S9(4)    COMP-4.
        03  REMOTE-NETADDRESS          PIC X(32).
        03  FILLER                     PIC X(4).
*              RESERVED
        03  REMOTE-ADDREXT-LENGTH      PIC S9(4)    COMP-4.
        03  REMOTE-ADDREXT-TYPE        PIC X.
*              0 - ISO 8348/AD2
*              2 - NOT ISO 8348/AD2
        03  REMOTE-ADDRESS-EXTENSION   PIC X(40).
        03  FILLER                     PIC X(4).
*              RESERVED
        03  X25-CALL-TYPE              PIC X.
*              0 - INCOMING SVC
*              1 - OUTGOING SVC
*              2 - NOT X25 SVC
     02  TRANSACTION-PROGRAM-NAME       PIC X(64).
     02  LU6-PROTECTED-LUWID.
        03  LENGTH-OF-PROT-LUWID-FIELDS       PIC S9(4)    COMP-4.
        03  FILLER REDEFINES LENGTH-OF-PROT-LUWID-FIELDS.
            05  LENGTH-OF-PROT-LUWID-FIELD    PIC X.
            05  LENGTH-OF-PROT-LU-NAME        PIC X.
        03  NETWORK-QUAL-PROT-LU-NAME         PIC X(17).
        03  PROTECTED-INST-SEQ-NUMBERS.
            05  PROT-INSTANCE-NUMBER          PIC X(6).
            05  PROT-SEQUENCE-NUMBER          PIC S9(4)    COMP-4.
     02  LU6-UNPROTECTED-LUWID.
        03  LENGTH-OF-UNPROT-LUWID-FIELDS     PIC S9(4)    COMP-4.
        03  FILLER REDEFINES LENGTH-OF-UNPROT-LUWID-FIELDS.
            05  LENGTH-OF-UNPROT-LUWID-FIELD  PIC X.
            05  LENGTH-OF-UNPROT-LU-NAME      PIC X.
        03  NETWORK-QUAL-UNPROT-LU-NAME       PIC X(17).
        03  UNPROTECTED-INST-SEQ-NUMBERS.
            05  UNPROT-INSTANCE-NUMBER        PIC X(6).
            05  UNPROT-SEQUENCE-NUMBER        PIC S9(4)    COMP-4.
```

付録 G. PROCESS ステートメント

PROCESS ステートメントは、COBOL ソース・プログラムのオプショナルの部分です。これより、通常はコンパイル時に指定するオプションを指定できます。

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。

PROCESS ステートメントの形式は、次のとおりです。

PROCESS ステートメント - 形式

対応する作成コマンドのオプション

以下の表では、使用できる PROCESS ステートメントのオプションと、同等の CRTCBLMOD および CRTBNDCBL コマンドのパラメーターおよびオプションを示します。デフォルトには、下線が付けられています。

PROCESS ステートメントのオプションの説明は、CRTCBLMOD および CRTBNDCBL パラメーターのパラメーターおよびオプションの説明と一致します。詳しくは、「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」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
NOXREF XREF	***NOXREF** *XREF
GEN NOGEN	***GEN** *NOGEN

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
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

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
STDINZ NOSTDINZ STDINZHEX00	***STDINZ** *NOSTDINZ *STDINZHEX00
NODDSFILLER DDSFILLER	***NODDSFILLER** *DDSFILLER
該当せず	***NOIMBEDERR** *IMBEDERR
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

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
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/sort-seq-table-name" "CURLIB/sort-seq-table-name" "library-name/sort-seq-table-name" "sort-seq-table-name"	***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 "language-identifier-name"	***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 1 バイト・データの CCSID	

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
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 1 バイト・データの CCSID	
CCSID (上記の「a」に指定された CCSID を使用) 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・データの CCSID	
CCSID (上記の「a」に指定された CCSID を使用) 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(上記の「a」に指定された CCSID を使用) 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 桁の基本世紀 (デフォルト 1900) *2 桁の基本年度* (デフォルト 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 EXTEND31FULL EXTEND63	***NOEXTEND** *EXTEND31 *EXTEND31FULL *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 NATIONALPICNLIT	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 = 単一バイト・データから国別データ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・データから国別データ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 = 国別データから国別データ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・インターフェースの情報	
NOPGMINFO PCML	***NO** *PCML
b = 生成されるプログラム情報の位置	
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**
INHERITTERASPACE	*INHERIT *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	*CALLER
'活動化グループ名'	活動化グループ名

付録 H. 複合 OCCURS DEPENDING ON

複合 OCCURS DEPENDING ON (ODO) は COBOL 85 標準の拡張としてサポートされます。

コンパイラーによって許可される複合 ODO の基本形式は、以下のとおりです。

- 可変位置の項目またはグループ: DEPENDING ON オプションが指定された OCCURS 文節によって記述され、非従属データ項目またはグループが後に続いているデータ項目。
- 可変位置のテーブル: DEPENDING ON オプションが指定された OCCURS 文節によって記述され、OCCURS 文節によって記述された非従属データ項目が後に続いているデータ項目。
- 可変長エレメントを持つテーブル: DEPENDING ON オプションが指定された OCCURS 文節によって記述された従属データ項目を含んでいる OCCURS 文節によって記述されたデータ項目。
- 可変長エレメントを持つテーブルの指標名。
- 可変長エレメントを持つテーブルのエレメント。

複合 ODO によって、ディスク・スペースを節約できますが、使用が容易でなく、コードの保守を困難にする可能性があります。

以下の例は、考えられる複合 ODO のオカレンス・タイプを示しています。

```
01  FIELD-A.
    02 COUNTER-1                          PIC S99.
    02 COUNTER-2                          PIC S99.
    02 TABLE-1.
       03 RECORD-1 OCCURS 1 TO 5 TIMES
                   DEPENDING ON COUNTER-1    PIC X(3).
    02 EMPLOYEE-NUMBER                    PIC X(5).  1
    02 TABLE-2 OCCURS 5 TIMES                        2 3
              INDEXED BY INDX.                       4
       03 TABLE-ITEM                      PIC 99.    5
       03 RECORD-2 OCCURS 1 TO 3 TIMES
                   DEPENDING ON COUNTER-2.
          04 DATA-NUM                     PIC S99.
```

この例で、COUNTER-1 は、RECORD-1 の DEPENDING ON 文節のオブジェクトなので、ODO オブジェクトです。 RECORD-1 は、ODO サブジェクトと呼ばれます。同様に、COUNTER-2 は、対応する ODO サブジェクトである RECORD-2 の ODO オブジェクトです。

上記の例に示した複合 ODO のオカレンス・タイプは、次のとおりです。

1 可変位置の項目: EMPLOYEE-NUMBER は、同じレベル 01 レコード内の可変長テーブルに続くデータ項目ですが、この可変長テーブルに従属していません。

2 可変位置のテーブル: TABLE-2 は、同じレベル 01 レコード内の可変長テーブルに続くテーブルですが、この可変長テーブルに従属していません。

3 可変長エレメントを持つテーブル: TABLE-2 は、オカレンス数が ODO オブジェクトの内容によって異なる従属データ項目 RECORD-2 を含むテーブルです。

4 可変長エレメントを持つテーブルの指標名 INDX。

5 可変長エレメントを持つテーブルのエレメント TABLE-ITEM。

各レコードの可変部分の長さは、その ODO オブジェクトと、その ODO サブジェクトの長さの積です。例えば、上記の複合 ODO 項目の 1 つへの参照を行う場合、実際の長さ (使用する場合) は以下のように計算されます。

- TABLE-1 の長さは、COUNTER-1 の内容 (RECORD-1 の数) と 3 (RECORD-1 の長さ) を乗算した結果です。
- TABLE-2 の長さは、COUNTER-2 の内容 (RECORD-2 の数) と 2 (RECORD-2 の長さ) を乗算した結果です。
- FIELD-A の長さは、COUNTER-1、COUNTER-2、TABLE-1、 EMPLOYEE-NUMBER、および TABLE-2 x 5 の長さの合計です。

グループ内の複合 ODO 項目を参照する前に、グループ内のすべての ODO オブジェクトを設定する必要があります。例えば、上記のコードの EMPLOYEE-NUMBER を参照する前に、EMPLOYEE-NUMBER がその値を入手するために COUNTER-1 および COUNTER-2 に直接依存していない場合でも、両方の ODO オブジェクトを設定する必要があります。

ODO 値の変更の影響

DEPENDING ON オプションを指定した OCCURS 文節で記述したデータ項目の後に、同じグループ内で 1 つまたは複数の非従属データ項目が続く場合には (複合 ODO 形式)、ODO オブジェクトの値の変更によって、そのレコード内の複合 ODO 項目への以後の参照が影響を受けます。

- 関係のある ODO 文節を含むすべてのグループのサイズが、ODO オブジェクトの新しい値に反映されます。
- ODO サブジェクトを含んでいるグループへの MOVE は、ODO オブジェクトの新しい値に基づいて実行されます。
- ODO 文節によって記述された項目に続くすべての非従属項目の位置は、ODO オブジェクトの新しい値による影響を受けます。 (非従属項目の内容を保存するには、ODO オブジェクトの値を変更する前に、内容を作業域に移動し、後から戻します。)

ODO オブジェクトの値は、データを ODO オブジェクトに移動するか、またはこのオブジェクトが含まれているグループに移動すると、変更される可能性があります。この値は、ODO オブジェクトが、READ ステートメントのターゲットであるレコードに含まれている場合も、変更される可能性があります。

ODO オブジェクトの値を変更する場合のエラーの防止

テーブル内の従属データ項目に関する ODO オブジェクトの値を変更した後に、複合 ODO 指標名 (可変長エレメントのテーブルの指標名) を参照する場合には、注意が必要です。 ODO オブジェクトの値を変更した場合、関連する複合 ODO 指標内のバイト・オフセットは、テーブルの長さが変更されているので、

もはや有効ではありません。この後に、以下の方法で指標名を参照するようにコーディングする場合、注意しないと予期しない結果が生じる可能性があります。

- テーブルのエレメントへの参照
- SET *整数データ項目* TO *指標名* という形式の SET ステートメント (形式 -1)
- SET *指標名* UP|DOWN BY *整数* という形式の SET ステートメント (形式 -2)

このタイプのエラーを回避するには、以下のステップを行います。

1. 指標項目を整数データ項目に保管する。 (この実行によって、暗黙の変換が行われます。整数項目は、指標内のオフセットに対応する、テーブル・エレメントのオカレンス番号を受け取ります。)
2. ODO オブジェクトの値を変更する。
3. 即時に、整数データ項目から指標項目を復元する。 (この実行によって、暗黙の変換が行われます。指標項目は、整数項目内のテーブル・エレメントのオカレンス番号に対応するオフセットを受け取ります。このオフセットは、その時点で有効であるテーブル長に従って計算されます。)

以下のコードは、ODO オブジェクト COUNTER-2 を変更する場合に 9-46 ページの『付録 H. 複合 OCCURS DEPENDING ON』に示した指標名を保管および復元する方法を示しています。

```
 77  INTEGER-DATA-ITEM-1      PIC 99.
 ...
     SET INDX TO 5.
*         INDX is valid at this point.
     SET INTEGER-DATA-ITEM-1 TO INDX.
*         INTEGER-DATA-ITEM-1 now has the
*         occurrence number corresponding to INDX.
     MOVE NEW-VALUE TO COUNTER-2.
*         INDX is not valid at this point.
     SET INDX TO INTEGER-DATA-ITEM-1.
*         INDX is now valid, containing the offset
*         corresponding to INTEGER-DATA-ITEM-1, and
*         can be used with the expected results.
```

可変テーブルへのエレメントの追加時のオーバーレイの防止

同じグループ内で 1 つまたは複数の非従属データ項目が後に続く可変オカレンス・テーブルのエレメント数を増加する場合には、注意が必要です。 ODO オブジェクトの値を増加し、テーブルにエレメントを追加する場合、テーブルの後に続く可変位置データ項目に意図せずにオーバーレイしてしまう可能性があります。

このタイプのエラーを回避するには、以下のステップを行います。

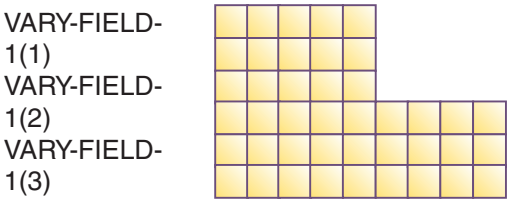
1. テーブルの後に続く可変位置データ項目を、別のデータ域に保管する。
2. ODO オブジェクトの値を増加する。
3. 新しいテーブル・エレメントにデータを移動する (必要な場合)。
4. 可変位置データ項目を保管したデータ域から復元する。

以下の例で、そのエレメント数が ODO オブジェクト CONTROL-1 に依存する、テーブル VARY-FIELD-1 にエレメントを追加する必要があるとします。 VARY-FIELD-1 の後には、非従属可変位置データ項目 GROUP-ITEM-1 が続いていて、このエレメントがオーバーレイされる可能性があります。

```
WORKING-STORAGE SECTION.
01  VARIABLE-REC.
    05  FIELD-1                            PIC X(10).
    05  CONTROL-1                          PIC S99.
    05  CONTROL-2                          PIC S99.
    05  VARY-FIELD-1 OCCURS 1 TO 10 TIMES
          DEPENDING ON CONTROL-1            PIC X(5).
```

```
    05  GROUP-ITEM-1.
        10  VARY-FIELD-2
              OCCURS 1 TO 10 TIMES
              DEPENDING ON CONTROL-2         PIC X(9).
01  STORE-VARY-FIELD-2.
    05  GROUP-ITEM-2.
        10  VARY-FLD-2
              OCCURS 1 TO 10 TIMES
              DEPENDING ON CONTROL-2         PIC X(9).
```

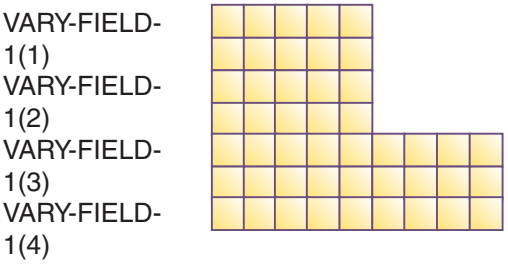
VARY-FIELD-1 の各エレメントは 5 バイトであり、VARY-FIELD-2 の各エレメントは 9 バイトです。CONTROL-1 と CONTROL-2 の両方に値 3 が入っている場合、VARY-FIELD-1 と VARY-FIELD-2 のストレージを以下のような図で示すことができます。



VARY-FIELD-1 に 4 番目のエレメントを追加するには、VARY-FIELD-2 の最初の 5 バイトへのオーバーレイを防止するために、以下のようにコーディングしてください。 (GROUP-ITEM-2 は、可変位置 GROUP-ITEM-1 の一時記憶域として使用します。)

```
MOVE GROUP-ITEM-1 TO GROUP-ITEM-2.
ADD 1 TO CONTROL-1.
MOVE five-byte-field TO
  VARY-FIELD-1 (CONTROL-1).
MOVE GROUP-ITEM-2 TO GROUP-ITEM-1.
```

VARY-FIELD-1 と VARY-FIELD-2 の変更済みのストレージを、以下のような図で示すことができます。



VARY-FIELD-1 の 4 番目のエレメントは、VARY-FIELD-2 の最初のエレメントにオーバーレイしていないことに注意してください。

付録 I. ACCEPT/DISPLAY および COBOL/2 に関する考慮事項

ILE COBOL の拡張 ACCEPT および DISPLAY ステートメントは、IBM COBOL/2™ の ACCEPT および DISPLAY ステートメント (形式 2) に類似していますが、以下の例外があります。

- 一部の WITH 句については、構文検査だけが行われます (拡張 ACCEPT および DISPLAY の構文図に示されています)。
- ON ESCAPE は、ON EXCEPTION 句の代わりには使用されません。

- 表示されるかまたは受け入れられるデータ項目内で句が重複している場合、ILE COBOL コンパイラーは重大エラー・メッセージを出します。 COBOL/2 コンパイラーでは、UPON や BELL など、一部の句の重複が許可されます。
- ILE COBOL の拡張 ACCEPT ステートメントではグループ項目について AUTO-SKIP を指定できますが、COBOL/2 コンパイラーでは重大エラー・メッセージが生成されます。
- ILE COBOL の拡張 ACCEPT ステートメントではグループ項目について BELL を指定できますが、COBOL/2 コンパイラーでは重大エラー・メッセージが生成されます。
- ILE COBOL コンパイラーは、以下のものがグループ項目について指定されている場合には、それを受け入れて該当するフィールドに適用します。 COBOL/2 コンパイラーでは重大エラー・メッセージが出されます。
  - REQUIRED
  - SECURE
  - LEFT-JUSTIFY
  - RIGHT-JUSTIFY
  - SPACE-FILL
  - TRAILING-SIGN
  - UPDATE
  - ZERO-FILL
- COBOL/2 コンパイラーは符号付き数字データ (表示されるかまたは受け入れられる) を左そろえしますが、ILE COBOL コンパイラーはそれらのデータ項目を右そろえします。
- COBOL/2 コンパイラーでは、DISPLAY ステートメントの中で検出された表意定数による特殊な効果を処理します (例えば、DISPLAY SPACE は DISPLAY WITH BLANK SCREEN と同じ効果をもちます)。一方、ILE COBOL コンパイラーでは、拡張 DISPLAY ステートメントの中で表示されるデータ項目として検出された表意定数に特殊な効果を適用しません。
- COBOL/2 コンパイラーでは、表示可能なデータ項目のためにすべての画面位置を使用しますが、ILE COBOL コンパイラーでは、必ず、それぞれの表示可能なデータ項目の前に属性バイト用の 1 バイトが必要とされます。このため、行 1、桁 1 は、ILE COBOL の拡張 ACCEPT または DISPLAY ステートメントでは使用できません。 (コンパイル時にエラー・メッセージ LNC1263 が出され、実行時に LNR7054 が出されます。)
- 1 つの ACCEPT または DISPLAY ステートメントで、1 つの WITH 句に UNDERLINE、HIGHLIGHT、および REVERSE-VIDEO 句が含まれている場合には、HIGHLIGHT 句が無視されます。この組み合わせがコーディングされると、コンパイル時に警告メッセージ (LNC0265) が生成されます。拡張 DISPLAY ステートメントの中では、UPON CRT-UNDER 句は UNDERLINE 句と同等です。フィールドが画面上に表示されないようにするには、SECURE オプションを使用します。
- CRTCBLMOD または CRTBNDCBL コマンドで EXTDSPOPT(*NODFRWRT) パラメーターを指定しない場合、ILE COBOL コンパイラーは、次の ACCEPT ステートメントが発生するまで、すべての拡張 DISPLAY ステートメントをバッファーに入れます。
- *NOUNDSPCHR コンパイラー・オプションが指定されている場合、16 進数 20 よりも小さい値は、拡張 ACCEPT および拡張 DISPLAY 操作において望ましくない結果をもたらします。このハードウェアの制限を無効にするには、(デフォルト) *UNDSPCHR オプションを使用してください。
- ILE COBOL のコンパイラーでは、実行時構成オプションを提供しません。
- COBOL/2 コンパイラーでは CRT STATUS 文節内のデータ名の長さは 3 バイトであり、ILE COBOL のコンパイラーでは 6 バイトです。

参考文献

IBM i 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 i 資料を参照してください。

- 「*Communications Management*」*(SC41-5406-02)*。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: 原始ステートメント入力ユーティリティー*」*(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) を使用してソース・メンバーを作成、編集する方法についての情報が記載されています。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。
- 「*アプリケーション表示プログラミング*」*(SC88-4031-01)*。以下に関する情報を提供します。
  - DDS を使用したアプリケーション用の表示画面の作成および保守
  - システムでのディスプレイ・ファイルの作成および処理
  - オンライン・ヘルプ情報の作成
  - UIM を使用したアプリケーション用のパネルおよびダイアログの定義
  - パネル・グループ、レコード、および文書の使用
- 「*バックアップおよび回復*」*(SD88-5008-10)*。以下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)*。CICS® for IBM i のアプリケーション・プログラミングに関する情報を提供します。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、IBM i 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*」*(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、学習終了時には解説書として役立ちます。 ILE RPG のコード例は、チュートリアルとともに配布され、それらはシステム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ています。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ています。GDDM で使用可能なすべてのグラフィックス・ルーチンについて詳しく説明します。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)*。IBM i 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) キーワードに関する情報、システム提供の形式、戻りコード、ファイル転送サポート、およびプログラム例も記載されています。
- 「*IDDU Use*」*(SC41-5704-00)*。IBM i 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ています。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
  - 照会および文書で使用されるデータを記述するための IDDU の使用の紹介
  - 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と、エラー・リカバリーおよび問題の防止に関する情報
- 「*IBM Rational Development Studio for i: ILE C/C++ プログラマーの手引き*」*(SC88-4024-02)*。ILE C 言語を使用して、アプリケーションを開発する方法を説明しています。プログラムの作成、実行、およびデバッグに関する情報が記載されています。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、EPM C/400 またはシステム C/400 から ILE C へのソース・コードのマイグレーションに関する情報が記載されています。
- 「*IBM Rational Development Studio for i: ILE C/C++ 解説書*」*(SC88-4026-02)*。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 実装について説明しています。
- 「*IBM Rational Development Studio for i: ILE COBOL プログラマーの手引き*」*(SD88-5045-07)*。IBM i システムで ILE COBOL プログラムを作成、コンパイル、バインド、実行、デバッグ、および保守する方法について説明しています。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。

- *「ILE 概念」(SD88-5033-09)*。IBM i 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ています。扱われているトピックとしては、モジュールの作成、プログラムのバインディング、プログラムの実行、プログラムのデバッグ、および例外の処理があります。
- *「IBM Rational Development Studio for i: ILE RPG プログラマーの手引き」(SD88-5042-07)*。IBM i システム上Integrated Language Environment (ILE) での RPG IV 言語の実装である ILE RPG プログラム言語について説明しています。プログラムの作成および実行に関する情報と、プロシージャー呼び出しおよび言語間プログラミングに関する考慮事項が記載されています。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 IBM i 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およびサンプル・コンパイラー・リストが記載されています。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 プログラミング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。
- *「装置構成」(SD88-5003-00)*。IBM i システムでのローカル装置の構成について説明しています。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 IBM i システムの概念、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 (ビジネス・グラフィックス・ユーティリティー (BGU) など)、高機能印刷 (AFP™)、および IBM i システム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キューから別の出力キューに移動する方法など) が記載されています。さらに、付録には、印刷の作業負荷を管理するために使用される制御言語 (CL) コマンドが記載されています。 IBM i システム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。
- *「適用業務開発ツールセット AS/400 用: 画面設計機能 (SDA)」(SD88-5046-00)*。適用業務開発ツールセットの画面設計機能 (SDA) を使用して、表示画面、メニュー、オンライン・ヘルプ情報を設計、作成、保守するための情報を提供します。IBM i システムと、IBM i システムのシステム/38 環境で SDA を使用する方法を理解するのに役立つ例および情報が記載されています。
- *「機密保護解説書」(SD88-5027-11)*。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ています。
- *「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、IBM i システムのインストール済みリリースをすでに所有しているユーザーで、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。

- 「*Systems Application Architecture Common Programming Interface COBOL Reference*」(SC26-4354)。

注

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

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。

〒103-8510
東京都中央区日本橋箱崎町19番21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 Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

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、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。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

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。これらのサンプル・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、お客様の当該サンプル・プログラム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。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。

© Copyright IBM Corp. _年を入れる_.

プログラミング・インターフェース情報

この ILE COBOL 言語解説書の資料には、プログラムを作成するユーザーが IBM i のサービスを使用するためのプログラミング・インターフェースが記述されています。

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 Corporation 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、『www.ibm.com/legal/copytrade.shtml』 をご覧ください。

Adobe、Adobe ロゴ、PostScript、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

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 Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

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。

使用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。 ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、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なしで提供されます。

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

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

[特殊文字]

プログラム番号: 5770-WDS

Printed in Japan

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