IBM COBOL for Linux on x86 1.1

言語解説書

SC43-5386-00

注記

本書および本書で紹介する製品をご使用になる前に、567 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM® COBOL for Linux® on x86 バージョン 1.1 (プログラム番号 5737-L11)、および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。製品のレベルに合った正しい版をご使用ください。

ソフトコピー資料は COBOL for Linux on x86 ライブラリーにおいて無償で表示またはダウンロードできます。

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

原典：
SC28-3117-00
IBM COBOL for Linux on x86 1.1
Language Reference
First edition

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

目次

表

前書き

本書について

本書では、IBM COBOL for Linux on x86 (本書では COBOL for Linux と呼びます) でサポートされる COBOL 言語について説明します。

COBOL プログラムの記述、コンパイル、デバッグに役立つ情報と例については、「*IBM COBOL for Linux on x86 プログラミング・ガイド*」を参照してください。

構文図の見方

本書全体を通して、COBOL for Linux の構文は図で示されています。

本書に示されている構文図は、以下の説明に従って読んでください。

- 構文図は、左から右、上から下へと線をたどって読んでください。

  ▶▶── は、構文図の開始点を示す記号です。

  ──▶ は、構文図が次の線に続くことを示す記号です。

  ▶── は、構文図が前の線から続いていることを示す記号です。

  ──▶◀ は、構文図の終わりを示す記号です。

  完全なステートメント以外の構文単位の図は、▶── 記号で始まり、──▶ 記号で終わっています。
- 必須項目は、横線 (主経路) 上に表示されます。

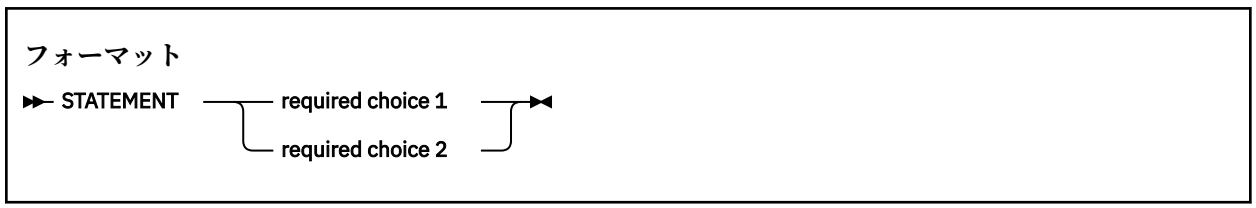
  

- オプション項目は、主経路の下に示されます。

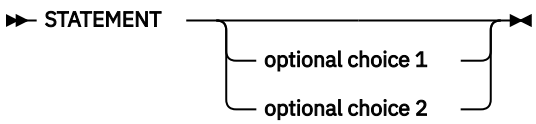
- 複数の項目から選択できる場合には、それらの項目は縦方向に重ねて示されます。

  それらの項目のうち 1 つを選択しなければならない 場合には、スタックのうち 1 つの項目が主経路と同じ高さに示されます。

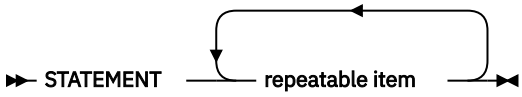
  いずれか 1 つの項目の選択が任意である場合は、重ねられた項目全体が主経路の下に示されます。

フォーマット



- 主経路から分岐して左へ戻る矢印は、反復可能な項目を示しています。

フォーマット

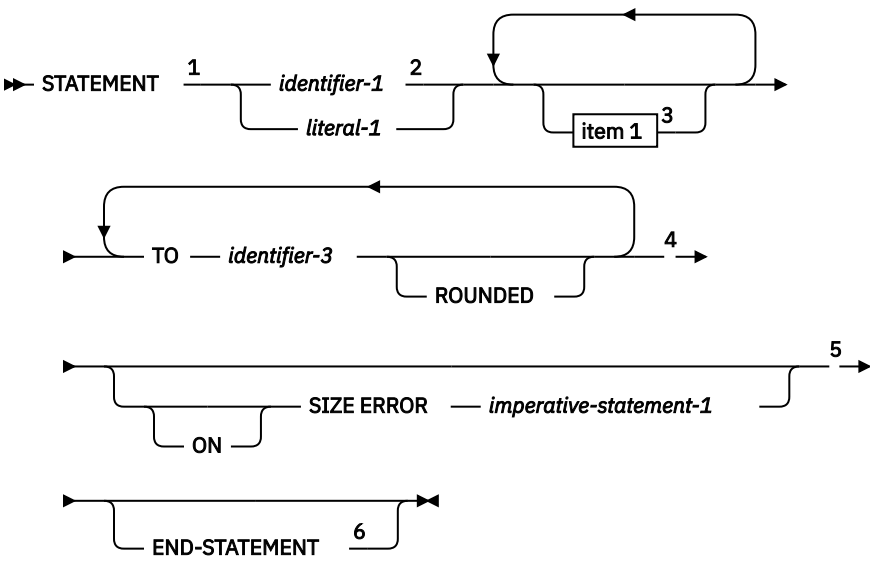


重ねられた項目の上に反復矢印がある場合は、重ねられた項目から 2 つ以上の 項目を選択するか、または 1 つの項目を繰り返すことができます。

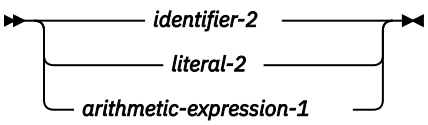
- 変数はイタリックの小文字で示されます (例えば *parmx*)。 それらの変数は、ユーザーが指定する名前や値を表します。
- 句読記号、括弧、算術演算子などの記号が示されている場合には、 これらも構文の一部として入力しなければなりません。

次の例は、構文の使い方を示したものです。

フォーマット



項目 1



注:

[1] STATEMENT キーワードは必ず指定しなければならず、ここに示されているとおり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須です。 *ID-1* またはリテラル*-1* のどちらかをコーディングする必要があります。

[3] 項目 1 のフラグメントはオプションです。アプリケーションの必要に応じてコーディングでき、またコーディングしないことも可能です。 項目 1 をコーディングする場合には、1 つまたは複数の COBOL 分離文字で区切ることによって、各項目ごとに繰り返すことができます。 このフラグメントに使用できる選択項目は、この図の最下部に記述されています。
[4] オペランド *ID-3* および関連付けられた TO キーワードは必須であり、各項目を分ける 1 つまたは複数の COBOL 分離文字によって繰り返すことができます。 各項目にはキーワード ROUNDED を割り当てることができます。
[5] ON SIZE ERROR 句および関連付けられた命令ステートメント-*1* はオプションです。 ON SIZE ERROR 句をコーディングした場合、キーワード ON はオプションです。
[6] END-STATEMENT キーワードをコーディングしてステートメントを 終了することができます。 これは必須区切り文字ではありません。

IBM 拡張

IBM 拡張は、通常、565 ページの『付録 I 業界仕様』 にリストされている ANSI および ISO COBOL 標準では指定されていない、機能、構文、または規則を追加するものです。 本書では、85 COBOL 標準 という用語はこれらの標準を指します。

拡張は規則の緩和など重要度の低いものから XML サポート、Unicode サポート、DBCS 文字の処理などの主要機能まで多岐にわたります。

本書の残りの部分では、拡張機能を区別せずに、言語全体について説明しています。 標準言語エレメントのみを使用する場合は、509 ページの『付録 A IBM 拡張』、および「*COBOL for Linux on x86* プログラミング・ガイド」の『コンパイラー・オプション』 を確認する必要があります。

廃止される言語エレメント

廃止される言語エレメントとは、85 COBOL 標準 で廃止 にカテゴリー化されたエレメントです。これらのエレメントは、2002 COBOL 標準 にはありません。

このことは、IBM では 85 COBOL 標準 で廃止されるエレメントを COBOL for Linux の将来のリリースから除去する意図があることを意味するものではありません。

以下の言語エレメントは、85 COBOL 標準では廃止として分類されています。

- ALTER ステートメント
- AUTHOR 段落
- コメント項目
- DATA RECORDS 節
- DATE-COMPILED 段落
- DATE-WRITTEN 段落
- DEBUG-ITEM 特殊レジスター
- デバッグ・セクション
- ENTER ステートメント
- 指定されたプロシージャー名のない GO TO
- INSTALLATION 段落
- LABEL RECORDS 節
- MEMORY SIZE 節
- MULTIPLE FILE TAPE 節
- RERUN 節

- REVERSED 句
- SECURITY 段落
- 分割モジュール
- STOP ステートメントの STOP リテラル・フォーマット
- USE FOR DEBUGGING 宣言部
- VALUE OF 節
- 形象定数の ALL リテラル (この形象定数が数字項目または数字編集項目と関連付けられているときに、1 より大きい長さを持つ場合)

DBCS の表記

リテラル、コメント、およびユーザー定義語にある 2 バイト文字セット (DBCS) ストリングは、 シフトアウト文字とシフトイン文字によって区切られます。

本書では、DBCS 文字は、D1D2D3 という形式で示されています。 DBCS 表現のローマ字 (英字) は、.A.B.C という形式で示されています。

注 :

- 1 バイト文字と 2 バイト文字が混合している EBCDIC DBCS データでは、2 バイト文字ストリングはシフトアウト文字とシフトイン文字によって区切られます。 本書では、一目で分かるように、シフトアウト区切り文字は記号 < で、 シフトイン区切り文字は記号 > で表しています。 シフトアウトとシフトインの区切り文字に対する 1 バイト EBCDIC コードは、それぞれ X'0E' と X'0F' です。 記号 <> は、 シフトアウト文字とシフトイン文字が連続していることを表します。 記号 >< は、 シフトイン文字とシフトアウト文字が連続していることを表します。
- 1 バイト文字と 2 バイト文字が混合している ASCII DBCS データでは、2 バイト文字ストリングはシフトアウト文字とシフトイン文字によって区切られません。

謝辞

以下は、ユーザーの方々への情報および手引きとして、 米国政府公式文書 (Form No.1965-0795689) から抜粋したものです。

> COBOL 報告書および仕様書の全部または一部を複製して、この報告書に盛られた考え方を教材、またはその他の目的の基礎資料として利用したいと考えているどの組織も、これを自由に行うことができます。 ただし、その作成資料の導入部にこのセクションを複写する必要があります。 書評にあるような短い文章で述べる場合には、 出典について述べる謝辞の中で COBOL について簡単に触れれば、 このセクション全体を掲げる必要はありません。
>
> COBOL は工業言語であり、特定の会社または会社のグループ、 あるいは特定の団体または団体のグループが所有するものではありません。
>
> COBOL 委員会や COBOL の発展に貢献したいかなる個人によって、プログラミング・システムや言語の正確性、 機能に関しての保証が与えられることはありません。 また上記の委員会および個人は、 それらに関してどのような責任も負いません。
>
> COBOL の保守にかかわるプロシージャーは確立されています。 変更の提案についてのプロシージャーに関するお問い合わせは、 CODASYL の理事会 (Executive Committee of the Conference on Data Systems Languages) あてにお願い致します。
>
> 以下の資料の著者および著作権所有者は、
>
> - FLOW-MATIC (Trademark of Sperry Rand Corporation), Programming for the UNIVAC (R) I and II, Data Automation Systems copyrighted 1958, 1959, by Sperry Rand Corporation
> - IBM Commercial Translator, Form No. F28-8013, copyrighted 1959 by IBM
> - FACT, DSI 27A5260-2760, copyrighted 1960 by Minneapolis-Honeywell
>
> この資料の一部または全部を使用することを COBOL 仕様書の中で特別に認可しています。 また、この認可の範囲には、 プログラミング・マニュアルや同種の出版資料において COBOL 仕様書を複写して使用することも含まれます。

注: 現在、上記の CODASYL は存在しません。

アクセシビリティー

アクセシビリティー機能は、運動障害または視覚障害などの障害を持つユーザーが情報技術製品を快適に使用できるように サポートします。

アクセシビリティー機能

IBM COBOL for Linux on x86 では、US Section 508 および Web Content Accessibility Guidelines (WCAG) 2.0 に確実に準拠するために、最新の W3C 標準である WAI-ARIA 1.0 が使用されています。アクセシビリティー機能を利用するには、最新リリースのスクリーン・リーダーを、この製品でサポートされる最新の Web ブラウザーと併用してください。

キーボード・ナビゲーション

この製品では、標準的なナビゲーション・キーを使用します。

インターフェース情報

Text-to-speech (TTS) ツールのような音声認識ソフトウェアを使用して、製品によって生成された出力を表示できます。

オンライン製品資料は、IBM 資料で入手でき、標準の Web ブラウザーで表示できます。

PDF ファイルでのアクセシビリティー・サポートは限定的です。 PDF 資料では、オプションのフォント拡大機能およびハイコントラスト表示設定を使用でき、キーボードのみでナビゲートできます。

スクリーン・リーダーを使用して、構文図、ソース・コード例、およびピリオドやコンマなどの PICTURE の記号を含むテキストを正確に読むには、句読点をすべて読むようにスクリーン・リーダーを設定する必要があります。

関連アクセシビリティー情報

標準の IBM ヘルプ・デスクとサポート Web サイトに加え、IBM は、聴覚が不自由なお客様が営業やサポート・サービスにアクセスするために使用できる TTY 電話サービスを立ち上げました。

TTY service 800-IBM-3383 (800-426-3383) (北アメリカ内)

IBM およびアクセシビリティー

IBM のアクセシビリティーに対する取り組みについて詳しくは、IBM アクセシビリティーを参照してください。

第 1 部 COBOL 言語の構造

第 1 章 文字

COBOL 言語の最も基本的で最小の単位は、文字 です。 基本文字セットには、ラテン・アルファベット、数字、および特殊文字が含まれています。

COBOL 言語では、個々の文字が組み合わされて、文字ストリング および分離文字 が形成されます。 文字ストリングおよび分離文字を使用して、言語を形成するワード、リテラル、句、節、ステートメント、および文が形成されます。

ソース・コードの文字ストリングおよび分離文字の形成に使用する基本文字セットは、3 ページの表 1 に示されています。.

一部の言語エレメントでは、この基本文字セットは コンパイル時に使用されたコード・ページ に応じて次の文字セットで拡張されています。

- ASCII 2 バイト文字セット (DBCS)。 DBCS 文字では、1 つの文字を表わすために隣接する 2 バイトが使用されます。 ソース・コード内の複数バイトで表わされる文字 (DBCS 文字を含む) のことを本書ではマルチバイト文字 と呼びます。 DBCS 文字のみを含む文字ストリングのことは、*DBCS* 文字ストリング または *2* バイト文字ストリング と呼びます。
- UTF-8 は、Unicode 文字セットのエンコード形式です。 UTF-8 文字では、1 文字につき 1 バイトから 4 バイトが使用されます。 2 バイト以上が使用される UTF-8 文字のことを本書ではマルチバイト文字 と呼びます。
- 拡張 UNIX コード (EUC)。EUC 文字の場合、文字ごとに 1 バイトから 4 バイト (またはコード・ページによっては 1 バイトから 3 バイト) が使用されます。 2 バイト以上が使用される EUC 文字のことを本書ではマルチバイト文字 と呼んでいます。

マルチバイト文字は、ユーザー定義語の形成に使用することができます。

英数字リテラル、コメント行、およびコメント項目の内容には、コンピューターのコンパイル時文字セットの任意の文字を 使用することができ、1 バイト文字およびマルチバイト文字をどちらも使用することができます。

実行時データには、コンピューターの実行時文字セットに含まれる任意の文字を使用することができます。コンピューターの実行時文字セットには、英数字、マルチバイト文字、および国別文字を含むことができます。 国別文字は UTF-16 (Unicode の 16 ビット・エンコード形式) で表記されます。

NSYMBOL (NATIONAL) コンパイラー・オプションが有効なときは、開始区切り文字 N" または N' で識別されるリテラルは、国別リテラルであり、コンパイル時のコード・ページに有効な任意の 1 バイト文字またはマルチバイト文字 (あるいは両方) を含むことができます。 国別リテラルに含まれる文字は、実行時に国別文字として表記されます。

詳しくは、10 ページの『マルチバイト 文字を持つユーザー定義語』、29 ページの『DBCS リテラル』、および 31 ページの『国別リテラル』 を参照してください。

表 *1.* **基本 *COBOL* 文字セット**. 下の表で、基本 COBOL 文字セットについて説明します。

文字	**意味**
	スペース
+	正符号
-	負符号またはハイフン
*	アスタリスク
/	スラッシュまたは固相線
=	等号

表 *1.* **基本 *COBOL* 文字セット**. 下の表で、基本 COBOL 文字セットについて説明します。 *(続き)*	
文字	**意味**
$	通貨符号
,	コンマ
;	セミコロン
.	小数点またはピリオド
"	引用符
'	アポストロフィ
(	左括弧
)	右括弧
>	より大きい
<	より小さい
:	コロン
_	下線
A から Z	英字 (大文字)
a から z	英字 (小文字)
0 から 9	数字

第 2 章 文字セットおよびコード・ページ

文字セット は、情報を表現するために使用される文字、数字、特殊文字、およびその他のエレメントのセットです。 コード化文字セットを表すために コード・ページ という用語を使用します。

文字セットはコード化表現に依存しません。 コード化文字セット は文字セットのコード化表現で、各文字にはエンコード・スキームでコード・ポイント と呼ばれる数値位置が割り当てられます。 コード化文字セットの例としては ASCII および EBCDIC があります。 ASCII または EBCDIC の各バリエーションは特定のコード化文字セットです。

IBM が定義する各コード・ページは、コード・ページ 名 (IBM-1252 など) およびコード化文字セット *ID* (CCSID) (1252 など) によって識別されます。

コンパイル時コード・ページ

コンパイル時コード・ページは、ASCII 1 バイトまたは ASCII 2 バイトのコード・ページ、EUC コード・ページ、または UTF-8 とすることができます。特定のコード・ページは、コンパイル時ロケール、または有効な環境変数によって示されます。

ソース・プログラム (ユーザー定義語および英数字リテラル、 DBCS リテラル、国別リテラルの内容を含む) は、コンパイル時に有効なロケールまたは環境変数によって示されるコード・ページでエンコードされます。

ランタイム・コード・ページ

実行時に使用されるコード・ページは、データ項目の USAGE 節、有効なコンパイラー・オプション、および有効なロケール (または環境変数値) の組み合わせによって決定されます。

CHAR(NATIVE) コンパイラー・オプションが有効な場合、USAGE DISPLAY または USAGE DISPLAY-1 で記述されるデータ項目は、ランタイムのロケールで指定される ASCII、EUC、 または UTF-8 コード・ページでエンコードされます。

CHAR(EBCDIC) コンパイラー・オプションが有効な場合、USAGE DISPLAY または USAGE DISPLAY-1 で記述されたデータ項目は、EBCDIC コード・ページでエンコードされます。ただし、項目の USAGE 節で NATIVE 句を指定した場合は除きます。 NATIVE 句を指定した場合、使用されるコード・ページは、ランタイム・ロケールによって示された ASCII 、EUC 、または UTF-8 コード・ページです。

EBCDIC の場合、コード・ページは EBCDIC_CODEPAGE 環境変数から決定されます (設定されている場合)。EBCDIC_CODEPAGE 環境変数が設定されていない場合は、現行ランタイム・ロケールに関連付けられたデフォルト EBCDIC コード・ページが使用されます。 サポートされる各ロケールに関連付けられたデフォルト EBCDIC コード・ページは、「*COBOL for Linux on x86* プログラミング・ガイド」の『サポートされるロケールおよびコード・ページ』に示されています。

DBCS の場合、コード・ページは DBCS_CODEPAGE 環境変数が設定されていればそれによって決定されます。DBCS_CODEPAGE 環境変数が設定されていない場合は、現在のランタイム・ロケールに関連したデフォルトの DBCS コード・ページが使用されます。

USAGE NATIONAL で記述されたデータ項目および国別リテラルのデフォルトのコード・ページは、UTF-16LE (リトル・エンディアン)、CCSID 1200 です。 国別リテラルのソース・テキスト表記は、実行時にコンパイル時コード・ページから UTF-16LE に変換されます。

USAGE NATIONAL で記述されたデータ項目および国別リテラルのエンディアンネス表現を変更するには、UTF16 コンパイラー・オプションを参照してください。

本書で *UTF-16* と表記するときは、UTF-16LE を指します。

文字エンコード・ユニット

文字エンコード・ユニット (またはエンコード・ユニット) は、実行時に COBOL によって 1 文字として処理されるデータの単位です。 本書では、文字 および文字位置 という用語は、単一のエンコード・ユニットを意味します。

データ項目およびリテラルのエンコード・ユニットのサイズは、以下のように、データ項目の USAGE 節またはリテラルのカテゴリーによって異なります。

- USAGE DISPLAY で記述されたデータ項目および英数字リテラルの場合は、使用されるコード・ページや任意の図形文字 を表すのに使用されるバイト数に関係なく、エンコード・ユニットは 1 バイトです。
- USAGE DISPLAY-1 で記述されたデータ項目 (DBCS データ項目) および DBCS リテラルの場合は、エンコード・ユニットは 2 バイトです。
- USAGE NATIONAL で記述されたデータ項目および国別リテラルの場合は、エンコード・ユニットは 2 バイトです。

図形文字とエンコード・ユニットの関係は、データ項目またはリテラルに使用されるコード・ページのタイプによって異なります。 実行時のコード・ページのタイプは、以下のとおりです。

- 1 バイト ASCII または EBCDIC
- マルチバイト ASCII ベース (ASCII DBCS、EUC、または UTF-8) またはマルチバイト EBCDIC DBCS
- Unicode UTF-16

コード・ページの各タイプの詳細については、以下のセクションを参照してください。

「*COBOL for Linux on x86* プログラミング・ガイド」の『文字データのコード・ページの指定』のセクションも参照してください。

1 バイト・コード・ページ

単一バイト ・コード・ページは、USAGE DISPLAY を指定して記述したデータ項目、および英数字カテゴリーのリテラルで使用できます。 エンコード・ユニットは 1 バイトであり、図形文字はそれぞれ 1 バイトで表されます。 これらのデータ項目およびリテラルについては、エンコード・ユニットを気にする必要はありません。

マルチバイト・コード・ページ

USAGE DISPLAY

USAGE DISPLAY で記述されたデータ項目 (英数字カテゴリー) および 英数字カテゴリーのリテラルでは、マルチバイト ASCII ベースまたは EBCDIC コード・ページでエンコードされたデータを使用できます。 エンコード・ユニットは 1 バイトであり、図形文字のサイズは、コード・ページによって 1 バイトから 4 バイトまでです。

英数字データ項目またはリテラルにマルチバイトの データが含まれている場合、プログラマーは、操作によって、1 つの図形文字を形成する複数のエンコード・ユニットが意図しない個所で分離されることのないようにする必要があります。 参照変更は慎重に行い、また、移動中に切り捨てが行われないようにする必要があります。 COBOL ランタイム・システムでは、図形文字を形成するエンコード・ユニットが分割されていないかどうかの検査は行いません。

問題を回避するため、英数字リテラル、および USAGE DISPLAY で記述されたデータ項目は、データ項目またはリテラルを USAGE NATIONAL で記述されたデータ項目へ移動するか、または NATIONAL-OF 組み込み関数を使用することによって、国別データ (UTF-16) へ変換できます。 こうすることで、図形文字が分割されることを気にせずに国別データを操作できます。 このデータは、DISPLAY-OF 組み込み関数を使用して変換して、USAGE DISPLAY に戻すことができます。

変換するソース・データが UTF-8 である場合、USAGE NATIONAL のターゲット・データ項目での図形文字の分割について考慮する必要があります。結果の国別データには UTF-16 サロゲート・ペアもあるからです。また文字シーケンスの結合についても考慮する必要があります。これらについては、7 ページの『Unicode UTF-16』を参照してください。

USAGE DISPLAY-1

USAGE DISPLAY-1 で記述されたデータ項目および DBCS カテゴリーのリテラルでは、マルチバイト ASCII DBCS または EBCDIC DBCS コード・ページの 2 バイト文字を使用できます。 エンコード・ユニットは 2 バイトであり、各図形文字は 2 バイトのエンコード・ユニット 1 つで表されます。これらのデータ項目およびリテラルについては、エンコード・ユニットを気にする必要はありません。

USAGE DISPLAY-1 で記述されるデータ項目では、UTF-8 または EUC は使用できません。

Unicode UTF-16

USAGE NATIONAL で記述されるデータ項目で UTF-16 を使用できます。 ソース・プログラムで使用されるコード・ページに関係なく、国別リテラルは UTF-16 文字として保管されます。 USAGE NATIONAL のデータ項目および国別リテラルのエンコード・ユニットは 2 バイトです。

UTF-16 のほとんどの文字の場合、図形文字は 1 つのエンコード・ユニットです。 EBCDIC、ASCII、または EUC のコード・ページから UTF-16 に変換された文字は、1 つの UTF-16 エンコード・ユニットとして表現されます。 それ以外に UTF-16 には、サロゲート・ペア または文字シーケンスの結合 で表現される 図形文字もあります。 1 組のサロゲート・ペアは 2 つのエンコード・ユニット (4 バイト) から構成されます。文字シーケンスの結合は、1 つの基本文字と 1 つ以上の結合マーク または 1 つ以上の結合マークの シーケンス (4 バイト以上で、2 バイトずつ増分される) で表現されます。 USAGE NATIONAL のデータ項目では、2 バイトのエンコード・ユニットが 1 文字として扱われます。

国別データにサロゲート・ペアまたは文字シーケンスの結合が含まれている場合、プログラマーは、国別文字に対して行う操作に よって、1 つの図形文字を形成する複数のエンコード・ユニットが意図しない個所で分離されることのないようにする必要があります。 参照変更は慎重に行い、また、移動中に切り捨てが行われないようにする必要があります。 COBOL ランタイム・システムでは、1 つの図形文字を形成する複数のエンコード・ユニットが分割されていないかどうかのチェックは行いません。

第 3 章 文字ストリング

文字ストリング とは、COBOL ワード、リテラル、PICTURE 文字ストリング、またはコメント項目を形成する 1 つの文字または一連の連続文字です。 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 とは、文字ストリングを区切るために使用する連続文字のストリングです。 分離文字については、35 ページの『第 4 章 分離文字』で詳しく説明します。

文字ストリングと特定の分離文字は、テキスト・ワード を形成します。 テキスト・ワードとは、ソース・テキスト、ライブラリー・テキスト、または疑似テキスト内の文字位置 8 から 72 まで (固定ソース形式)、または文字位置 8 から 252 まで (拡張ソース形式) (8 と 72 を含む) にある、1 つの文字または連続する文字のシーケンスです (複数行にわたる場合もあります)。 疑似テキストの詳細については、47 ページの『疑似テキスト』を参照してください。

ソース・テキスト、ライブラリー・テキスト、および疑似テキストは、1 バイトの 文字および (一部の文字ストリングの場合) コンパイル時ロケールで示された ASCII、UTF-8、または EUC コード・ページのマルチバイト文字で 記述できます。

1 バイトおよびマルチバイト文字ストリングを使用して以下の項目を形成できます。

- COBOL ワード
- リテラル
- コメント・テキスト

PICTURE 文字ストリングを形成する場合は、使用できるのは 1 バイト文字のみです。

1 バイト文字から構成される COBOL ワード

COBOL ワード は、ユーザー定義語、システム名、または予約語を構成する文字ストリングです。 COBOL ユーザー定義語の最大サイズは 30 バイトです。 指定できる文字数は、コンパイル時のロケールによって示されるコード・ページによって異なります。

算術演算子および比較文字を除いて、COBOL ワードの各文字は以下のセットから選択されます。

- 英大文字 A から Z
- 英小文字 a から Z
- 数字 0 から 9
- - (ハイフン)
- _ (下線)

ハイフンは、 COBOL ワードの最初の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。 下線は、COBOL ワードの最初の文字として使用することはできません。 ほとんどのユーザー定義語 (セクション名、段落名、優先順位番号、およびレベル番号を除くすべて) には、少なくとも 1 つの英字が含まれていなければなりません。 優先順位番号とレベル番号は固有である必要はありません。ある優先順位番号またはレベル番号の指定が、ほかの優先順位番号またはレベル番号と同じであってもかまいません。

COBOL ワード (ただし、英数字、DBCS、および国別 の各リテラルの内容とは異なります) では、1 バイトの英小文字はそれぞれ対応する 1 バイトの英大文字と等価であるとみなされます。

すべての COBOL ワードに次の規則が適用されます。

- 予約語をユーザー定義語またはシステム名として使用することはできません。
- ただし、 同じ COBOL ワードをユーザー定義語とシステム名の両方として使用することはできます。 COBOL ワードの特定のオカレンスの分類は、 その語が現れる節または句の文脈によって判別されます。

マルチバイト 文字を持つユーザー定義語

ユーザー定義語のコンテキストで使用する場合、マルチバイト という用語は、3 種類のワードを指します。

3 種類のワードは以下のとおりです。

- DBCS 文字で構成されたワード (1 バイト文字と組み合わされている場合もある)
- 1 バイト以上からなる UTF-8 文字から構成されるワード
- 1 バイト以上からなる EUC 文字から構成されるワード

マルチバイト文字からユーザー定義語を形成する場合の規則は次のとおりです。

含まれる文字
ユーザー定義語は、1 バイト文字とマルチバイト文字の両方から構成できます。 文字が 1 バイト形式とマルチバイト形式の両方で存在する場合、その 1 バイトとマルチバイトの表記は同じではありません。

ユーザー定義語の 1 バイト文字は、次の文字に制限されています。

- 英大文字 A から Z
- 英小文字 a から z
- 数字 0 から 9
- - (ハイフン)
- _ (下線)

1 バイトのエンコードされたハイフンは、ユーザー定義語の最初の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。

1 バイトのエンコードされた下線は、ユーザー定義語の最初の文字として使用することはできません。

大文字および小文字
COBOL ワードでは、小文字の 1 バイト・エンコード文字「a」から「z」は、それぞれ対応する 1 バイト・エンコードの大文字と同じとみなされます。 マルチバイト・エンコードの大文字と小文字は同じではありません。

値の範囲
マルチバイト文字の有効な値の範囲は、使用される個別のコード・ページによって異なります。

最大長
30 バイト。 30 バイト内で指定できる文字数は、ソース・コード・ページおよびユーザー定義語で使用される文字によって決まります。

継続
マルチバイト文字で形成されるワードは、複数行にまたがって続けることはできません。

シフトアウト文字およびシフトイン文字の使用
ダミーのシフトイン・シフトアウト (SOSI) コンパイラー・オプションが有効である場合のみ適用可能です。 SOSI コンパイラー・オプションの詳細については、*COBOL for Linux on x86* プログラミング・ガイド内の *SOSI* を参照してください。

ユーザー定義語

ユーザー定義語は、節またはステートメントの形式を満たすためにユーザーが指定する必要のある COBOL ワードです。

以下のユーザー定義語のセットはサポートされています。 2 番目の列は、特定のセットのワードで マルチバイト文字を使用できるかどうかを示しています。

ユーザー定義語	マルチバイト文字が使用可能か
英字名	はい
割り当て名	いいえ

ユーザー定義語	マルチバイト文字が使用可能か
クラス名 (データの)	はい
条件名	はい
データ名	はい
ファイル名	はい
指標名	はい
レベル番号: 01–49、66、77、88	いいえ
ライブラリー名	いいえ
簡略名	はい
段落名	はい
優先順位番号: 00–99	いいえ
プログラム名	いいえ
レコード名	はい
レコード・キー名 **制約事項 :** レコード・キー名 は、STL ファイル・システムでのみサポートされます。	はい
セクション名	はい
シンボリック文字	はい
テキスト名	いいえ

ユーザー定義語の最大長は 30 バイトです。ただし、レベル番号と優先順位番号を除きます。 レベル番号と優先順位番号はそれぞれ 1 桁または 2 桁の整数である必要があります。

特定の数値は優先順位番号である場合と、レベル番号である場合がありますが、それ以外の各ユーザー定義語は、 これらのセットのうちの 1 つだけに属することができます。 1 つのセット内の各ユーザー定義語は、優先順位番号とレベル番号以外については、53 ページの『第 8 章 データ名、コピー・ライブラリー、および PROCEDURE DIVISION の名前を参照』 に指定されているものを除き、固有でなければなりません。

次のタイプのユーザー定義語は、 そのユーザー定義語が宣言されているプログラムの中のステートメントや項目によって参照できます。

- 段落名
- セクション名

次のタイプのユーザー定義語は、 コンパイルするシステムが関連のライブラリーまたは他のシステムをサポートしており、参照されるエンティティーがそのシステムに認識されていれば、どの COBOL プログラムでも参照できます。

- ライブラリー名
- テキスト名

次のタイプの名前は、構成セクション内で宣言されているときは、 構成セクションを含んでいるプログラムの中でも、 あるいはそのプログラム内に含まれる任意のプログラムの中でも、 ステートメントと項目によって参照できます。

- 英字名
- 条件名
- 簡略名

- シンボリック文字

それぞれのユーザー定義語の機能は、 それが現れる節またはステートメントの中に記述されます。

関連参照
561 ページの『付録 G コンテキストに依存した語』

システム名

システム名 は、システムにとって特別の意味のある文字ストリングです。

システム名には以下の 3 つのタイプがあります。

- コンピューター名
- 言語名
- インプリメンター名

インプリメンター名には次の つのタイプがあります。

- 環境名
- 外部クラス名
- 割り当て名

それぞれのシステム名の意味は、それが現れるフォーマットで説明しています。

システム名にはマルチバイト文字ストリングを使用できます。

関数名

関数名 は、組み込み関数の値を決定するために提供されたメカニズムを指定します。

1 つのプログラムにおいて、 異なる文脈の中で同一の語をユーザー定義語としてもシステム名としても使用することができます。 関数名とその定義のリストについては、429 ページの表 58 を参照してください。

予約語

予約語 とは、COBOL ソース単位であらかじめ定義された意味を持っている文字ストリングです。

予約語が、545 ページの『付録 F 予約語』にリストされています。 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・ワード
- 形象定数
- 特殊文字ワード
- 特殊レジスター

キーワード
キーワードとは、一定の節、項目、またはステートメントで必要な予約語です。 構文図の各フォーマットでは、これらのキーワードは主経路上に大文字で示されています。

オプショナル・ワード
オプショナル・ワードとは、節、項目、またはステートメントを読みやすくするために、それらのフォーマットの中に 含めることができる予約語です。 プログラムの実行には影響しません。

形象定数
13 ページの『形象定数』を参照してください。

特殊文字ワード
特殊文字ワード は以下に示すように 5 種類あります。これらは 1 バイト文字で表示されるときのみに特殊文字として認識されます。

- **算術演算子:** + - / * **

  234 ページの『算術式』を参照してください。
- **比較演算子**: < > = <= >=

  238 ページの『条件式』を参照してください。
- **浮動コメント標識**: *>

  46 ページの『浮動コメント標識 (*>)』を参照してください。
- **COPY および REPLACE ステートメントにおける疑似テキスト区切り文字**: ==

  476 ページの『COPY ステートメント』および 488 ページの『REPLACE ステートメント』を参照してください。
- **コンパイラー・ディレクティブ標識**: >>

  497 ページの『第 22 章 コンパイラー指示』を参照してください。

特殊レジスター
15 ページの『特殊レジスター』を参照してください。

形象定数

形象定数 とは、特定の定数値に名前を付け、それを参照する場合に使用する予約語です。 このセクションには、形象定数の予約語とその意味が示されています。

ZERO、ZEROS、ZEROES
コンテキストに応じて、数値ゼロ (0)、英数字文字ゼロの 1 つ以上のオカレンス、またはブール値 B"0" を表します。

英数字を指定する必要がある文脈で形象定数 ZERO、ZEROS、または ZEROES を使用すると、英数字ゼロが使用されます。 国別文字ゼロを指定する必要がある文脈では、国別文字ゼロ (値 NX'3000') が使用されます。 文脈が判別できない場合は、英数字ゼロが使用されます。

SPACE、 SPACES
1 つ以上のブランクまたはスペースを表します。 SPACE は、英数字を指定する必要があるコンテキストで使用された場合には英数字リテラルとして扱われ、DBCS 文字を指定する必要があるコンテキストで使用された場合には DBCS リテラルとして扱われ、国別文字を指定する必要があるコンテキストで使用された場合には国別リテラルとして扱われます。

HIGH-VALUE、 HIGH-VALUES
使用されている照合シーケンスにおいて最も高い順位にある文字の 1 つ以上のオカレンスを表します。

HIGH-VALUE は、英数字を必要とする文脈内では英数字リテラルとして扱われます。 EBCDIC 照合シーケンスの英数字データでは、値は X'FF' です。 その他の英数字データでは、値はロケールで示される照合シーケンスによって異なります。 ロケールの詳細については、563 ページの『付録 H ロケールの考慮事項』を参照してください。

HIGH-VALUE は、国別文字を必要とする文脈で使用された場合には、国別リテラルとして扱われます。値は、国別文字の NX'FFFF' です。 HIGH-VALUE は、国別リテラルを必要とするコンテキストで使用できるのは、NCOLLSEQ(BIN) コンパイラー・オプションが有効になっている場合のみです。

文脈を判別できない場合は英数字文脈と見なされ、値の X'FF' が使用されます。

使用上の注意: あるデータ表現と別の表現との間の変換が行われることになる方法では、HIGH-VALUE (または HIGH-VALUE から割り当てられた値) は使用しないでください。 X'FF' は有効な EBCDIC または ASCII 文字を表しません。また、NX'FFFF' は有効な国別文字を表しません。 英数字または国別の HIGH-VALUE 表現をもう一方の表現に変換すると、置換文字が使用されるようになります。 例えば、X'FF' を UTF-16 へ変換すると、NX'FFFF' ではなく、1 つの置換文字が使用されます。

LOW-VALUE、 LOW-VALUES
使用されている照合シーケンスにおいて最も低い順位にある文字の 1 つ以上のオカレンスを表します。

LOW-VALUE は、英数字を必要とする文脈内では英数字リテラルとして扱われます。 EBCDIC 照合シーケンスの英数字データでは、値は X'00' です。 その他の英数字データでは、値はロケールで示される照

合シーケンスによって異なります。 ロケールの詳細については、563 ページの『付録 H ロケールの考慮事項』を参照してください。

LOW-VALUE は、国別文字を必要とする文脈で使用された場合には、国別リテラルとして扱われます。値は、国別文字の NX'0000' です。 LOW-VALUE は、国別リテラルを必要とするコンテキストで使用できるのは、NCOLLSEQ(BIN) コンパイラー・オプションが有効になっている場合のみです。

文脈を判別できない場合は、英数字文脈と見なされ、値の X'00' が使用されます。

QUOTE、 QUOTES
以下の 1 つ以上のオカレンスを表します。

- 引用符文字 (")。ただし QUOTE コンパイラー・オプションが有効である場合
- アポストロフィ文字 (')。ただし APOST コンパイラー・オプションが有効である場合

QUOTE または QUOTES は、英数字を必要とする文脈で使用された場合は英数字を、国別文字を必要とする文脈で使用された場合は国別文字を表します。引用符の国別文字値は NX'2200' です。 アポストロフィの国別文字値は NX'2700' です。

英数字リテラルを囲むために、QUOTE および QUOTES を引用符またはアポストロフィの代わりに使用することはできません。

ALL リテラル
リテラル は、英数字リテラル、DBCS リテラル、国別リテラル、または ALL リテラル以外の形象定数です。

リテラル が形象定数でない場合、ALL リテラル はそのリテラルを構成する文字ストリングの 1 つまたは複数のオカレンスを表します。

リテラル が形象定数の場合、ワード ALL は意味を持たず、読みやすさの目的でのみ使用されます。

CALL、INSPECT、STOP、または STRING の各ステートメントでは、形象定数 ALL リテラル を使用することはできません。

シンボリック文字
SPECIAL-NAMES 段落の SYMBOLIC CHARACTERS 節で、シンボリック文字 の値として指定された 1 つ以上の文字を表します。

シンボリック文字 は常に英数字を表し、英数字から国別文字への暗黙の変換が定義されている場合にのみ、国別文字を指定する必要がある文脈で使用できます。 (例えば、受け取り項目が国別クラスの項目である MOVE ステートメントでは、国別文字または UTF-8 文字を使用できます。これは、送り出し項目が英数字であり、受け取り項目が国別 である場合は暗黙の変換が定義されるからです。)

コンパイル時ロケール設定でマルチバイト・コード・ページが示されている場合は、SYMBOLIC CHARACTERS 節は指定できません。 ロケールの詳細については、563 ページの『付録 H ロケールの考慮事項』を参照してください。

NULL、 NULLS
USAGE POINTER、USAGE PROCEDURE-POINTER、USAGE FUNCTION-POINTER、あるいは有効なアドレスを持たない特殊レジスターの ADDRESS OF で定義されるデータ項目を示すために使用される値を表します。 NULL は、構文フォーマットで明示的に許可されている場所でだけ使用することができます。 NULL の値は 0 です。

NULL、ZERO、SPACE、HIGH-VALUE、LOW-VALUE、 および QUOTE の単数形と複数形は同義で使用できます。 例えば、DATA-NAME-1 が 5 文字のデータ項目の場合は、次の各ステートメントによって 5 つのスペースが DATA-NAME-1 に移動されます。

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

COBOL の規則によって形象定数名の特定のスペルが許可されているときは、その形象定数名について 任意の代替スペルを指定することができます。

リテラル が構文図で現れている場所では、明示的に禁止されているところを除いて、どこでも形象定数を使用することができます。 構文図の中で数字リテラルが現れるときは、 形象定数 ZERO (ZEROS または ZEROES) だけを使用することができます。 形象定数は、式が関数の引数である算術式の中以外では、関数の引数として使用できません。 表意定数 ZERO はブール・リテラルとして使用できます。

形象定数の長さは、使用される文脈によって異なります。 次の規則が適用されます。

- 形象定数が VALUE 節で指定されるか、 またはデータ項目と関連付けられている場合は (例えば、 別の項目に移動されたり、別の項目と比較されたりする場合)、 その形象定数文字ストリングの長さは、 1 かまたはその関連しているデータ項目のサイズの文字位置のうちの大きい方に等しくなります。
- ALL リテラル以外の形象定数が別のデータ項目に関連付けられていない場合 (例えば CALL、 STOP、 STRING、または UNSTRING ステートメント) は、文字ストリングの長さは 1 文字になります。

特殊レジスター

特殊レジスター とは、コンパイラーによって生成されたストレージ域に名前を付ける予約語です。 それらの主な用途は、 特定の COBOL 機能によって作成される情報を保管することにあります。 そのようなストレージ域はそれぞれ決まった名前を持っており、 プログラムの中でさらに定義する必要はありません。

RECURSIVE 属性が指定されたプログラムや では、 以下の特殊レジスター用のストレージは呼び出しごとに割り振られます。

- ADDRESS OF
- RETURN-CODE
- SORT-CONTROL
- SORT-CORE-SIZE
- SORT-FILE-SIZE
- SORT-MESSAGE
- SORT-MODE-SIZE
- SORT-RETURN
- TALLY
- XML-CODE
- XML-EVENT
- XML-NTEXT
- XML-TEXT

プログラムのキャンセル後に最初にそのプログラムを呼び出したときに、コンパイラーは特殊レジスター・フィールドを初期値に初期設定します。

以下の 2 つの場合、

- INITIAL 節が指定されたプログラム
- RECURSIVE 節が指定されたプログラム

以下の特殊レジスターが、各プログラム項目の初期値に 再設定されます。

- RETURN-CODE
- SORT-CONTROL
- SORT-CORE-SIZE
- SORT-FILE-SIZE
- SORT-MESSAGE
- SORT-MODE-SIZE
- SORT-RETURN
- TALLY
- XML-CODE

- XML-EVENT

さらに、前述の 2 つの場合、ADDRESS OF 特殊レジスターに設定された値は、その特定プログラム の起動中にのみ持続します。

それ以外の場合は、特殊レジスターはリセットされません (前の CALL 時に設定されていた値のままです)。

明示的な制限がない限り、特殊レジスターの暗黙の定義と同じ定義を持っているデータ名または ID を使用できる個所では、特殊レジスターを使用することができます。 暗黙の定義を適用できる場合は、各特殊レジスターの仕様に示されます。

特別の禁止がない限り、 英数字引数が使用できる関数ではどこでも英数字特殊レジスターを指定することができます。

修飾を使用できる場合は、必要に応じて特殊レジスターを修飾することによって固有性を持たせることができます。 (詳細については、53 ページの『修飾』を参照。)

ADDRESS OF

ADDRESS OF 特殊レジスターは、LINKAGE SECTION、LOCAL-STORAGE SECTION、または WORKING-STORAGE SECTION のデータ項目のアドレスを参照します。

LINKAGE SECTION の 01 および 77 レベル項目の場合は、ADDRESS OF 特殊レジスターは送り出し項目または受け取り項目のいずれかとして使用できます。 それ以外のすべてのオペランドの場合は、ADDRESS OF 特殊レジスターは送り出し項目としてのみ使用できます。

ADDRESS OF 特殊レジスターは、暗黙的に USAGE POINTER として定義されます。

ADDRESS OF 特殊レジスターのオペランドとして関数 ID を使用することはできません。

DEBUG-ITEM

DEBUG-ITEM 特殊レジスターは、デバッグ・セクションの実行の原因となった条件に関する情報を、デバッグ宣言型プロシージャーに提供します。

DEBUG-ITEM には、次のような暗黙の記述があります。

```
01  DEBUG-ITEM.
    02   DEBUG-LINE     PICTURE IS X(6).
    02   FILLER         PICTURE IS X  VALUE SPACE.
    02   DEBUG-NAME     PICTURE IS X(30).
    02   FILLER         PICTURE IS X  VALUE SPACE.
    02   DEBUG-SUB-1    PICTURE IS S9999  SIGN IS LEADING SEPARATE CHARACTER.
    02   FILLER         PICTURE IS X  VALUE SPACE.
    02   DEBUG-SUB-2    PICTURE IS S9999  SIGN IS LEADING SEPARATE CHARACTER.
    02   FILLER         PICTURE IS X  VALUE SPACE.
    02   DEBUG-SUB-3    PICTURE IS S9999  SIGN IS LEADING SEPARATE CHARACTER.
    02   FILLER         PICTURE IS X  VALUE SPACE.
    02   DEBUG-CONTENTS PICTURE IS X(n).
```

各デバッグ・セクションが実行される前に、DEBUG-ITEM はスペースで埋められます。 DEBUG-ITEM サブフィールドの内容は MOVE ステートメントの規則に従って更新されます。 ただし、1 つの例外があります。それは、 データの内部表示の形式を別の形式に変換しないで、 移動が英数字間の基本移動であるかのように DEBUG-CONTENTS が更新されるということです。

更新後、DEBUG-ITEM サブフィールドの内容は以下のようになります。

DEBUG-LINE
デバッグ・セクションの実行を引き起こしたソース・ステートメントのシーケンス番号 (または指定されたコンパイラー・オプションに 応じたコンパイラー生成のシーケンス番号)。

DEBUG-NAME
デバッグ・セクションの実行を引き起こした名前の最初の 30 文字。 修飾子はいずれも 'OF' という語によって区切られます。

DEBUG-SUB-1、 DEBUG-SUB-2、 DEBUG-SUB-3
必ずスペースに設定されます。 これらのサブフィールドは、以前の COBOL 製品との互換性のために説明されています。

DEBUG-CONTENTS
データは、以下の表に示すように DEBUG-CONTENTS の中に移動されます。

表 *2.* ***DEBUG-ITEM*** **サブフィールドの内容**

デバッグ・セクションの実行の原因	**DEBUG-LINE で参照されるステートメント**	**DEBUG-NAME の内容**	**DEBUG-CONTENTS の内容**
プロシージャー名-*1* ALTER 参照	ALTER ステートメント	プロシージャー名-*1*	TO PROCEED TO 句の中のプロシージャー名-*n*
GO TO プロシージャー名-*n*	GO TO ステートメント	プロシージャー名-*n*	スペース
SORT または MERGE 入出力プロシージャーの中のプロシージャー名-*n*	SORT ステートメントまたは MERGE ステートメント	プロシージャー名-*n*	"SORT INPUT"、 "SORT OUTPUT"、または "MERGE OUTPUT" (適用できる場合)
PERFORM ステートメントの制御移動	この PERFORM ステートメント	プロシージャー名-*n*	"PERFORM LOOP"
USE プロシージャーの中の プロシージャー名-*n*	USE プロシージャーの実行を引き起こすステートメント	プロシージャー名-*n*	"USE PROCEDURE"
1 つ前の順番のプロシージャーからの暗黙の移動	1 つ前の順番のプロシージャーで実行された前のステートメント [1]	プロシージャー名-*n*	"FALL THROUGH"
最初の非宣言型プロシージャーの最初の実行	最初の非宣言型プロシージャー名の行番号	最初の非宣言型プロシージャーの名前	"START PROGRAM"

1. このプロシージャーの前にセクション・ヘッダーがあり、 制御がそのセクション・ヘッダーを介して渡される場合は、 ステートメント番号はセクション・ヘッダーを指します。

FORMAT OF

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。

LENGTH OF

LENGTH OF 特殊レジスターには、データ項目によって使用されたバイト数が入れられます。

LENGTH OF は暗黙の特殊レジスターを作成します。その内容は、ID によって参照されるデータ項目の現在のバイト長に等しくなります。

USAGE DISPLAY-1 で記述されたデータ項目 (DBCS データ項目) および USAGE NATIONAL で記述されたデータ項目では、各文字は 2 バイトのストレージを占めます。

LENGTH OF 特殊レジスターの暗黙の定義と同じ定義の数字データ項目が使用される個所であれば、PROCEDURE DIVISION のどこであっても、LENGTH OF を使用することができます。

ADDR(32) コンパイラー・オプションが指定されている場合、 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).
```

ID によって参照されるデータ項目が GLOBAL 節を含む場合には、 LENGTH OF 特殊レジスターはグローバル・データ項目です。

LENGTH OF 特殊レジスターは、参照変更指定の開始文字位置または長さ式の中で使用することができます。しかし、LENGTH OF 特殊レジスターを、 参照変更されたオペランドに適用することはできません。

LENGTH OF オペランドは関数にすることはできませんが、 LENGTH OF 特殊レジスターは、 整数引数が使用できる関数の中では使用することができます。

LENGTH 関数への引数として LENGTH OF 特殊レジスターが使用された場合、 LENGTH OF で指定された引数とは無関係に、 その結果は必ず 4 になります。

LENGTH 関数への引数として ADDRESS OF 特殊レジスターが使用された場合、ADDRESS OF で指定された引数とは無関係に、その結果は必ず 4 になります。

LENGTH OF は、以下の項目にすることはできません。

- 受け入れ側のデータ項目
- 添え字

LENGTH OF 特殊レジスターが CALL ステートメントに対するパラメーターとして使用さ れる場合には、BY CONTENT または BY VALUE によって渡される必要があります。

テーブル・エレメントが指定される場合、LENGTH OF 特殊レジスターは 1 つのエレメントの長さをバイト数で保持します。 テーブル・エレメントを参照するとき、エレメント名に添え字を付ける必要はありません。

ID によって参照される領域が現在はプログラムに使用可能でない場合であっても、 長さが判別できる ID はすべて値が戻されます。

LENGTH OF 句を使用して参照される ID ごとに、別個の LENGTH OF 特殊レジスターが存在します。 次に例を示します。

```
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
```

組み込み関数 LENGTH を使用してデータ項目の長さを取得することもできます。 USAGE NATIONAL のデータ項目の場合、LENGTH 関数から戻される長さは国別文字位置数であり、バイト数ではありません。したがって、USAGE NATIONAL のデータ項目の場合、LENGTH OF 特殊レジスターと LENGTH 組み込み関数では結果が異なります。 その他のデータ項目については、結果は同じです。

LENGTH 組み込み関数がヌル終了英数字リテラルに適用される場合、ヌル終了文字より前にそのリテラルのバイト数を戻しますが、そのヌル終了文字は含まれません。(LENGTH 特殊レジスターはリテラルのオペランドをサポートしていません。) ヌル終了の英数字リテラルについて詳しくは、28 ページの『ヌル終了英数字リテラル』を参照してください。

LINAGE-COUNTER

LINAGE 節を含む各 FD 項目ごとに、別個の LINAGE-COUNTER 特殊レジスターが生成されます。 複数の特殊レジスターが生成されるときは、 それぞれの LINAGE-COUNTER 参照をそれに関係付けられたファイル名で修飾しなければなりません。

LINAGE-COUNTER 特殊レジスターに関する暗黙の記述は、以下のケースのいずれかです。

- LINAGE 節でデータ名を指定している場合には、 LINAGE-COUNTER はそのデータ名と同じ PICTURE および USAGE になります。
- LINAGE 節で整数を指定している場合には、 LINAGE-COUNTER はその整数と同じ桁数の 2 進数項目です。

詳しくは、155 ページの『LINAGE 節』を参照してください。.

ある時点の LINAGE-COUNTER の値は、 現在のページ内で装置が位置付けられている行の行番号です。 LINAGE-COUNTER は PROCEDURE DIVISION ステートメントで参照することができますが、そのステートメントで修正してはなりません。

LINAGE-COUNTER は、 その関連ファイルの OPEN ステートメントが実行されたときに、 1 に初期設定されます。

LINAGE-COUNTER は、そのファイルに対するいずれの WRITE ステートメントによっても自動的に修正されます。 (398 ページの『WRITE ステートメント』を参照してください。).)

順次ファイルのファイル記述項目が LINAGE 節および EXTERNAL 節を含んでいる場合は、 LINAGE-COUNTER データ項目は外部データ項目です。 順次ファイルのファイル記述項目が LINAGE 節および GLOBAL 節を含んでいる場合は、 LINAGE-COUNTER データ項目はグローバル・データ項目です。

整数引数が使用できる関数であれば、 どこでも LINAGE-COUNTER 特殊レジスターを指定できます。

LOCALE OF

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。

RETURN-CODE

RETURN-CODE 特殊レジスターは、現在の COBOL プログラムの終了時に、呼び出し側プログラムまたはオペレーティング・システムに戻りコードを渡すために使用できます。

COBOL プログラム終了時に、次のようになります。

- 制御がオペレーティング・システムに戻る場合、 RETURN-CODE 特殊レジスターの値はユーザー戻りコードとしてオペレーティング・システムに渡されます。 サポートされているユーザー戻りコード値はオペレーティング・システムごとに違っており、 その中に RETURN-CODE 特殊レジスターの値の範囲全体が含まれるとは限りません。 Linux でのユーザー戻りコード値については、「*COBOL for Linux on x86* プログラミング・ガイド」の『日時呼び出し可能サービスからのフィードバックの取得』 を参照してください。
- 制御が呼び出し側プログラムに戻る場合、 RETURN-CODE 特殊レジスターの値は呼び出し側プログラムに渡されます。 呼び出し側プログラムが COBOL プログラムの場合、 呼び出し側プログラム側の RETURN-CODE 特殊レジスターは、 呼び出されるプログラム側の RETURN-CODE 特殊レジスターの値に設定されます。

RETURN-CODE 特殊レジスターには、次のような暗黙の定義があります。

```
01  RETURN-CODE GLOBAL PICTURE S9(4) USAGE BINARY VALUE ZERO.
```

ネストされたプログラムで使用される場合、この特殊レジスターは最外部プログラムの GLOBAL 節で暗黙的に定義されます。

以下の例は、RETURN-CODE 特殊レジスターの設定方法を示しています。

- `COMPUTE RETURN-CODE = 8`
- `MOVE 8 to RETURN-CODE`

RETURN-CODE 特殊レジスターは、 CALL...RETURNING を使用するプログラムからの値を戻しません。 詳しくは、 284 ページの『CALL ステートメント』を参照してください。

整数引数が使用できる関数であれば、どこでも RETURN-CODE 特殊レジスターを指定できます。

RETURN-CODE 特殊レジスターは、 日付/時刻呼び出し可能サービスからの情報を返しません。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『日時の操作』を参照してください。

SHIFT-OUT と SHIFT-IN

英数字引数が使用できる関数であれば、どこでも SHIFT-OUT 特殊レジスターと SHIFT-IN 特殊レジスターを指定できます。

SHIFT-OUT および SHIFT-IN 特殊レジスターがサポートされるのは、CHAR(EBCDIC) コンパイラー・オプションでコンパイルされる場合のみです。 ただし、その値は、COBOL for Linux 用にサポートされるコード・ページでは 2 バイト文字の区切り文字としては認識されません。

SHIFT-OUT および SHIFT-IN 特殊レジスターは、次のフォーマットの英数字データ項目として暗黙に定義されています。

```
01  SHIFT-OUT GLOBAL PICTURE X(1) USAGE DISPLAY VALUE X"0E".
01  SHIFT-IN  GLOBAL PICTURE X(1) USAGE DISPLAY VALUE X"0F".
```

ネストされたプログラムで使用される場合、これらの特殊レジスターは最外部プログラムの GLOBAL 属性で暗黙的に定義されます。

これらの特殊レジスターは、印刷不能文字である EBCDIC シフトアウトおよびシフトインの各制御文字を表します。

これらの特殊レジスターは受け入れ側の項目になることはできません。 マルチバイト ユーザー定義語を定義しているとき、または EBCDIC DBCS リテラルを指定しているときには、キーボード制御文字として SHIFT-OUT と SHIFT-IN を使用することはできません。

SORT-CONTROL

SORT-CONTROL 特殊レジスターは、英数字データ項目の名前です。

SORT-CONTROL 特殊レジスターには、次のような暗黙の定義があります。

```
01 SORT-CONTROL GLOBAL PICTURE X(160) VALUE "file name".
```

"file name" は、追加のソート/マージ・オプション用のソースとしてソートで使用されるファイル名です。

ネストされたプログラムで使用される場合、 この特殊レジスターは最外部プログラムの GLOBAL 属性で暗黙的に定義されます。

英数字引数が使用できる関数であれば、 どこでも SORT-CONTROL 特殊レジスターを指定できます。

ソートおよびマージ操作が成功した場合には、 SORT-CONTROL 特殊レジスターは必要ありません。

ソート制御ファイルは SORT 特殊レジスターに優先します。

SORT-CORE-SIZE

SORT-CORE-SIZE 特殊レジスターは、2 進データ項目の名前です。これを使用することによって、ソート・ユーティリティー・プログラムで利用できるストレージのバイト数を指定することができます。

SORT-CORE-SIZE 特殊レジスターには、次のような暗黙の定義があります。

```
01  SORT-CORE-SIZE GLOBAL PICTURE S9(8) USAGE BINARY VALUE ZERO.
```

ネストされたプログラムで使用される場合、 この特殊レジスターは最外部プログラムの GLOBAL 属性で暗黙的に定義されます。

SORT-CORE-SIZE 特殊レジスターで示されるストレージの量には、SORT または MERGE 関数に関連しない COBOL ライブラリー関数に必要なメモリー領域は含まれません。 ソートおよびマージの実行に必要なメモリー領域の固定量 (モジュール、制御ブロック、固定サイズ作業域) も含まれません。

整数引数が使用できる関数であれば、 どこでも SORT-CORE-SIZE 特殊レジスターを指定できます。

SORT-FILE-SIZE

SORT-FILE-SIZE 特殊レジスターは、2 進データ項目の名前です。これを使用することによって、ソート入力ファイル (ファイル名-*1*) にある予測レコード数を指定することができます。

SORT-FILE-SIZE 特殊レジスターには、次のような暗黙の定義があります。

```
01  SORT-FILE-SIZE GLOBAL PICTURE S9(8) USAGE BINARY VALUE ZERO.
```

ネストされたプログラムで使用される場合、 この特殊レジスターは最外部プログラムの GLOBAL 属性で暗黙的に定義されます。

SORT-FILE-SIZE 特殊レジスターへの参照は、コンパイラーによって解決されますが、この特殊レジスターの値は、SORT または MERGE ステートメントの実行には何の影響もありません。

整数引数が使用できる関数であれば、 どこでも SORT-FILE-SIZE 特殊レジスターを指定できます。

SORT-MESSAGE

SORT-MESSAGE 特殊レジスターは、ソートとマージの両方のプログラムで利用できる英数字データ項目の名前です。

SORT-MESSAGE 特殊レジスターへの参照は、コンパイラーによって解決されますが、この特殊レジスターの値は、SORT または MERGE ステートメントの実行には何の影響もありません。

SORT-MESSAGE 特殊レジスターには、次のような暗黙の定義があります。

```
01  SORT-MESSAGE GLOBAL PICTURE X(8) USAGE DISPLAY VALUE "SYSOUT".
```

ネストされたプログラムで使用される場合、 この特殊レジスターは最外部プログラムの GLOBAL 属性で暗黙的に定義されます。

英数字引数が使用できる関数であれば、 どこでも SORT-MESSAGE 特殊レジスターを指定できます。

SORT-MODE-SIZE

SORT-MODE-SIZE 特殊レジスターは、最も頻繁に出現する可変長レコードの長さを指定するために使用できる 2 進データ項目の名前です。

SORT-MODE-SIZE 特殊レジスターには、次のような暗黙の定義があります。

```
01  SORT-MODE-SIZE GLOBAL PICTURE S9(5) USAGE BINARY VALUE ZERO.
```

ネストされたプログラムで使用される場合、 この特殊レジスターは最外部プログラムの GLOBAL 属性で暗黙的に定義されます。

SORT-MODE-SIZE 特殊レジスターへの参照は、コンパイラーによって解決されますが、この特殊レジスターの値は、SORT または MERGE ステートメントの実行には何の影響もありません。

整数引数が使用できる関数であれば、 どこでも SORT-MODE-SIZE 特殊レジスターを指定できます。

SORT-RETURN

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利用できます。

SORT-RETURN 特殊レジスターには、次のような暗黙の定義があります。

```
01  SORT-RETURN GLOBAL PICTURE S9(4) USAGE BINARY VALUE ZERO.
```

ネストされたプログラムで使用される場合、 この特殊レジスターは最外部プログラムの GLOBAL 属性で暗黙的に定義されます。

ソートまたはマージ処理の完了時に、SORT-RETURN 特殊レジスターに戻りコード 0 (成功) または 16 (不成功) が入れられます。 ソートまたはマージ操作が正しく実行されず、 プログラムのどこにもこの特殊レジスターの参照が存在しなければ、 メッセージが端末に表示されます。

すべてのレコードが処理される前にソートまたはマージ操作を終了させるには、 エラー宣言部分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 に設定します。 ソートまたはマージ操作の次の入出力機能を実行するときに、操作は終了します。

整数引数が使用できる関数であれば、 どこでも SORT-RETURN 特殊レジスターを指定できます。

TALLY

TALLY 特殊レジスターは、バイナリー・データ項目の名前です。

次のバイナリー・データ項目の定義を参照してください。

```
01  TALLY GLOBAL PICTURE 9(5) USAGE BINARY VALUE ZERO.
```

ネストされたプログラムで使用される場合、 この特殊レジスターは最外部プログラムの GLOBAL 属性で暗黙的に定義されます。

TALLY の内容を参照したり変更したりすることができます。

整数引数が使用できる関数であれば、 どこでも TALLY 特殊レジスターを指定できます。

WHEN-COMPILED

WHEN-COMPILED 特殊レジスターには、コンパイル開始時の日付が入れられます。

WHEN-COMPILED は、次のような暗黙の定義の英数字データ項目です。

```
01  WHEN-COMPILED GLOBAL PICTURE X(16) USAGE DISPLAY.
```

ネストされたプログラムで使用される場合、 この特殊レジスターは最外部プログラムの GLOBAL 属性で暗黙的に定義されます。

WHEN-COMPILED 特殊レジスターのフォーマットは、次のとおりです。

```
MM/DD/YYhh.mm.ss (MONTH/DAY/YEARhour.minute.second)
```

例えば、コンパイルが 2007 年 10 月 15 日午後 2 時 4 分に始まった場合、 WHEN-COMPILED には値として 10/15/0714.04.00 が入れられます。

WHEN-COMPILED は、 MOVE ステートメントで送り出しフィールドとしてのみ使用することができます。

WHEN-COMPILED 特殊レジスターに入れられたデータは参照変更できません。

この特殊レジスターに納められたコンパイル日時は、組み込み関数 WHEN-COMPILED を使用すればアクセス可能です (468 ページの『WHEN-COMPILED』 を参照)。 この関数は 4 桁の年値をサポートしており、他にも情報を提供します。

XML-CODE

XML-CODE 特殊レジスターは、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との間で状況をやり取りし、XML GENERATE ステートメントが正常に実行されたこと、または XML の生成中に例外が発生したことを示すために使用されます。

XML-CODE 特殊レジスターには、次のような暗黙の定義があります。

```
01  XML-CODE PICTURE S9(9) USAGE BINARY VALUE 0.
```

ネストされたプログラムで使用される場合、 この特殊レジスターは最外部プログラムの GLOBAL 属性で暗黙的に定義されます。

XML パーサーは、XML イベントを検出すると、XML-CODE を設定し、処理プロシージャーに制御を渡します。 EXCEPTION イベント以外のすべてのイベントについて、処理プロシージャーが制御を受け取ったとき、XML-CODE はゼロ (0) に設定されています。

EXCEPTION イベントの場合、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。 XML PARSE 例外コードについては、「*COBOL for Linux on x86* プログラミング・ガイド」の『*XML PARSE* の例外処理』 を参照してください。

パーサーに制御を戻す前に、次のように XML-CODE を設定することができます。

- -1 に設定。通常イベント後に、パーサーに対して、残りの XML 文書テキストを処理せず、また EXCEPTION イベントを発生させずに即時に終了するよう指示します。
- ゼロに設定。継続可能な EXCEPTION イベント後に、パーサーに対して処理の継続を指示します。 パーサーは XML 文書の処理の継続を試みますが、結果は保証されません。
- 場合によっては、エンコードの矛盾の例外後に、XML-CODE をコード・ページ ID に設定できます。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『*XML PARSE* の例外処理』を参照してください。

パーサーに制御を戻す前に XML-CODE を上記以外の値に設定した場合、結果は保証されません。

パーサーから XML PARSE ステートメントに制御が戻ったとき、XML-CODE には、処理プロシージャーまたはパーサーによって設定された最新の値が入っています。 パーサーは、処理プロシージャーによって設定された値をオーバーライドする場合があります。

XML GENERATE ステートメントの終了時、XML-CODE には XML 生成が正常に完了したことを示すゼロ、または XML 生成中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 例外コードの詳細については、「*COBOL for Linux on x86* プログラミング・ガイド」の『*XML GENERATE* 例外』 を参照してください。

関連概念
XML-CODE (*COBOL for Linux on x86* プログラミング・ガイド)

関連タスク
XML PARSE の例外処理 (*COBOL for Linux on x86* プログラミング・ガイド)

関連参照
XML GENERATE 例外 (*COBOL for Linux on x86* プログラミング・ガイド)

XML-EVENT

XML-EVENT 特殊レジスターは、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との間のイベント情報のやり取りに使用されます。

XML パーサーは、処理プロシージャーに制御を渡す前に、XML-EVENT 特殊レジスターを XML イベントの名前に設定します (24 ページの表 3 を参照)。.

XML-EVENT には、次のような暗黙の定義があります。

```
01  XML-EVENT USAGE DISPLAY PICTURE X(30) VALUE SPACE.
```

ネストされたプログラムで使用される場合、 この特殊レジスターは最外部プログラムの GLOBAL 属性で暗黙的に定義されます。

XML-EVENT の内容は、CHAR コンパイラー・オプション (EBCDIC、NATIVE、または S390) の設定によって EBCDIC または ASCII コード・ページ内でランタイムにエンコードされます。

XML-EVENT を受け取りデータ項目として使用することはできません。

表 *3.* ***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	属性値内の事前定義のエンティティー参照に対応する 1 文字
ATTRIBUTE-CHARACTERS	引用符またはアポストロフィで囲まれた値。 値にエンティティー参照が含まれているときは、この値は属性値のサブストリングである場合があります。
ATTRIBUTE-NAME	属性名。等号の左方にあるストリング
ATTRIBUTE-NATIONAL-CHARACTER	XML PARSE ステートメント内の *identifier-1* によって指定される XML 文書のタイプに関係なく、XML-TEXT は空で、長さがゼロになり、XML-NTEXT には数字参照に対応する 1 文字の国別文字が入ります。
COMMENT	開始文字シーケンス「<!--」と終了文字シーケンス「-->」で囲まれたコメントのテキスト
CONTENT-CHARACTER	要素コンテント内の事前定義のエンティティー参照に対応する 1 文字

表 3. ***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-CHARACTERS	開始タグと終了タグに囲まれたエレメントの文字内容。 内容にエンティティー参照または別のエレメントが含まれているときは、この値は文字内容のサブストリングである場合があります。
CONTENT-NATIONAL-CHARACTER	XML PARSE ステートメント内の *identifier-1* によって指定される XML 文書のタイプに関係なく、XML-TEXT は空で、長さがゼロになり、XML-NTEXT には数字参照に対応する 1 文字の国別文字が入ります。[1]
DOCUMENT-TYPE-DECLARATION	開始文字シーケンス「<!DOCTYPE」と終了文字シーケンス「>」で囲まれた文書タイプ宣言全体
ENCODING-DECLARATION	引用符またはアポストロフィで囲まれた、XML 宣言内のエンコード宣言の値
END-OF-CDATA-SECTION	ストリング "]]>"
END-OF-DOCUMENT	長さゼロで空
END-OF-ELEMENT	終了エレメント・タグの名前または空のエレメント・タグ
EXCEPTION	例外が検出された時点までの正常にスキャンされた文書部分。[2] 特殊レジスター XML-CODE には、例外を示す固有のエラー・コードが入ります。
PROCESSING-INSTRUCTION-DATA	処理命令の残りの部分 (ターゲット名の後)、終了シーケンス「?>」は含まれませんが、末尾の空白文字と先頭以外の空白文字は含まれます
PROCESSING-INSTRUCTION-TARGET	処理命令の開始シーケンス「<?」の直後に出現する、処理命令のターゲット名
STANDALONE-DECLARATION	XML 宣言内のスタンドアロン宣言の引用符またはアポストロフィで囲まれた値 ("yes" または "no")
START-OF-CDATA-SECTION	ストリング「<![CDATA[」
START-OF-DOCUMENT	文書全体
START-OF-ELEMENT	開始エレメント・タグまたは空エレメント・タグの名前 (エレメント・タイプともいいます)
UNKNOWN-REFERENCE-IN-CONTENT	エンティティー参照名 (区切り文字の「&」と「;」を除く)
UNKNOWN-REFERENCE-IN-ATTRIBUTE	エンティティー参照名 (区切り文字の「&」と「;」を除く)
VERSION-INFORMATION	引用符またはアポストロフィで囲まれた XML 宣言内のバージョン情報の値

1. 65,535 (NX"FFFF") より大きいスカラー値を持つ国別文字は、2 つのエンコード・ユニット (「サロゲート・ペア」) を使用して表現されます。XML-NTEXT の内容に対する操作によって図形文字を構成するエンコード・ユニットが分割されると、 無効データが形成されるので、プログラマーはこのような分割が発生しないように考慮する必要があります。
2. エンコード矛盾の例外は、構文解析が開始される前にシグナル通知されます。 このような例外の場合、XML-TEXT または XML-NTEXT は長さゼロになるか、文書のエンコード宣言値のみを含みます。 XML 例外コードの詳細については、*COBOL for Linux on x86* プログラミング・ガイド内の *XML GENERATE* 例外を参照してください。

XML-NTEXT

XML-NTEXT 特殊レジスターは XML 構文解析中に定義され、USAGE NATIONAL で表される文書フラグメントを含みます。

XML-NTEXT は、国別カテゴリーの基本データ項目であり、長さはその中に含まれている XML 文書フラグメントの長さになります。 XML-NTEXT の長さは 0 から 2,000,000 国別文字数とすることができます。最大バイト長は 4,000,000 です。

対応する COBOL データ記述項目はありません。

ネストされたプログラムで使用される場合、 この特殊レジスターは最外部プログラムの GLOBAL 属性で暗黙的に定義されます。

以下の場合には、パーサーは XML-NTEXT をイベントに関連付けられた文書フラグメントに設定してから、処理プロシージャーに制御権を渡します。

- XML PARSE ステートメントのオペランドが国別カテゴリーのデータ項目である場合
- ATTRIBUTE-NATIONAL-CHARACTER イベントの場合
- CONTENT-NATIONAL-CHARACTER イベントの場合

XML-NTEXT が設定されると、XML-TEXT 特殊レジスターは長さゼロになります。 XML-NTEXT 特殊レジスターと XML-TEXT 特殊レジスターは、両方同時にゼロでない長さを持つことはできません。

LENGTH 関数を使用すると、XML-NTEXT に含まれている国別文字の数を判別することができます。 LENGTH OF 特殊レジスターを使用して、XML-NTEXT に含まれている国別文字の数ではなく、バイト数を判別します。

XML-NTEXT を受け取り項目として使用することはできません。

XML-TEXT

XML-TEXT 特殊レジスターは、XML 構文解析時に定義され、USAGE DISPLAY で表される文書フラグメントを含みます。

XML-TEXT は、英数字カテゴリーの基本データ項目であり、長さはその中に含まれている XML 文書フラグメントの長さになります。 XML-TEXT の長さは 0 から 2,147,483,646 バイトとすることができます。

対応する COBOL データ記述項目はありません。

ネストされたプログラムで使用される場合、 この特殊レジスターは最外部プログラムの GLOBAL 属性で暗黙的に定義されます。

XML-TEXT の内容には、ソース XML 文書のエンコードがあります。エンコードは、CHAR(NATIVE) コンパイラー・オプションが有効な場合は ASCII または UTF-8 で、CHAR(EBCDIC) コンパイラー・オプションが有効な場合は EBCDIC になります。

ATTRIBUTE-NATIONAL-CHARACTER イベントと CONTENT-NATIONAL-CHARACTER イベントの場合を 除き、XML PARSE ステートメントのオペランドが英数字データ項目であり、パーサーは XML-TEXT をイベントに関連付けられた文書フラグメントに設定してから、制御権を処理プロシージャーに転送します。

XML-TEXT が設定されると、XML-NTEXT 特殊レジスターは長さゼロになります。 XML-NTEXT 特殊レジスターと XML-TEXT 特殊レジスターは、両方同時にゼロでない長さを持つことはできません。

LENGTH 関数または XML-TEXT 用の LENGTH OF 特殊レジスターを使用すると、XML-TEXT に含まれているバイト数を判別することができます。

XML-TEXT を受け取り項目として使用することはできません。

リテラル

リテラル は、文字ストリングを構成する文字によって、または形象定数の使用によって、その値が指定される文字ストリングです。

形象定数について詳しくは、13 ページの『形象定数』を参照してください。

各種のリテラルの説明については、以下のトピックを参照してください。

- 27 ページの『英数字リテラル』
- 29 ページの『ブール・リテラル』
- 29 ページの『DBCS リテラル』
- 31 ページの『国別リテラル』
- 31 ページの『数字リテラル』

英数字リテラル

COBOL for Linux では、次のフォーマットの英数字リテラルをサポートしています。

英数字リテラルの形式は以下のとおりです。

- フォーマット 1: 27 ページの『基本英数字リテラル』
- フォーマット 2: 28 ページの『英数字リテラルの 16 進表記』
- フォーマット 3: 28 ページの『ヌル終了英数字リテラル』

基本英数字リテラル

基本英数字リテラルには、1 バイト文字またはマルチバイト文字のみを使用することができます。

基本英数字リテラルのフォーマット:

フォーマット 1: 基本英数字リテラル

```
"single-byte or multibyte characters"
'single-byte or multibyte characters'
```

リテラルを囲んでいる引用符やアポストロフィは、プログラムのコンパイル時にリテラルから取り除かれます。

組み込みの引用符やアポストロフィを使用する場合、その文字が開始の区切り文字として使用されている文字であるときは、2 つの引用符 (" ") または 2 つのアポストロフィ (' ') で表現する必要があります。 次に例を示します。

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

リテラルの開始の区切り文字として使用する区切り文字は、 そのリテラルの終了の区切り文字としても使用しなければなりません。 次に例を示します。

```
'THIS IS RIGHT'
"THIS IS RIGHT"
'THIS IS WRONG"
```

リテラル区切り文字としてアポストロフィまたは引用符を使用できます (APOST/QUOTE コンパイラー・オプションとは 無関係)。

英数字リテラルの中に含まれている他の句読文字はすべて、 そのリテラルの値の一部になります。

"*single-byte or multibyte characters*" には、 コンパイル時に有効なロケールによって示されたコード・ページで表現される任意の文字を指定できます。

英数字リテラルの最大長は 160 バイトです。 最小長は 1 バイトです。

英数字リテラルは、英数字データ・クラスおよびカテゴリーに属します (データ・クラスおよびカテゴリーについては、137 ページの『データのクラスとカテゴリー』で解説しています)。

COBOL for Linux on x86 プログラミング・ガイド 内の *SOSI* を参照してください。 英数字リテラルでの値 X'1E' と X'1F' の特殊処理の説明、および SOSI コンパイラー・オプションが有効な場合にこれらの文字が含まれる英数字リテラルについて、プログラミングの追加制約事項の説明があります。

使用上の注意: 16 進表記を使用すると、英数字リテラル内の制御文字 X'00' から X'1F' を表わすことができます。 基本英数字リテラル内でこれらの制御文字を使用すると、予測不能な結果が生じます。

英数字リテラルの 16 進表記

英数字リテラルでは 16 進表記を使用することができます。

16 進表記のフォーマットは以下のとおりです。

フォーマット 2: 英数字リテラルの 16 進表記

```
X"hexadecimal-digits"
X'hexadecimal-digits'
```

X" または X'
英数字リテラルの 16 進表記の開始の区切り文字

" または '
英数字リテラルの 16 進表記の終了の区切り文字。 開始の区切り文字に引用符を使用した場合は、終了の区切り文字にも引用符を使用する必要があります。同様に、開始の区切り文字にアポストロフィを使用した場合は、終了の区切り文字にもアポストロフィを使用する必要があります。

16 進数字は '0' から '9'、'a' から 'f'、および 'A' から 'F' の範囲に含まれる文字です。 2 つの 16 進数字で 1 バイト文字セット (EBCDIC または ASCII) の 1 つの文字を表します。 4 つの 16 進数字で DBCS 文字セットに含まれる 1 つの文字を表現します。 UTF-8 または EUC の場合、文字を表わす 16 進数字の数は、文字のバイト長によって決まります。 16 進数は、偶数桁で指定しなければなりません。 16 進リテラルの最大長は、320 桁までです。

継続に関する規則は、他の英数字リテラルのための規則と同じです。 開始の区切り文字 (X" または X') は、行にまたがって分割できません。

16 進表記の英数字リテラルは、英数字のデータ・クラスおよびカテゴリーに属します。 コンパイラーは、16 進表記を英数字リテラルの通常文字に変換します。 英数字リテラルの 16 進表記は、英数字リテラルを使用できるのであればどこでも使用できます。

ただし、英数字リテラルの 16 進表記は、CHAR(EBCDIC) コンパイラー・オプションが有効な場合には EBCDIC として解釈されます。 これに対し、基本英数字リテラルは、常に現行ロケールの (ASCII ベース) コード・ページで解釈されます。 CHAR(EBCDIC) コンパイラー・オプションの詳細については、*COBOL for Linux on x86* プログラミング・ガイド内の *CHAR* を参照してください。

使用上の注意: 16 進表記を使用すると、英数字リテラル内の制御文字 X'00' から X'1F' を表わすことができます。 基本英数字リテラル内でこれらの制御文字を使用すると、予測不能な結果が生じます。

32 ページの『国別リテラルの 16 進表記』も参照してください。

ヌル終了英数字リテラル

英数字リテラルは、ヌル終了にすることができます。

ヌル終了英数字リテラルの形式は以下のとおりです。

形式 3: ヌル終了英数字リテラル

```
Z"mixed-characters"
Z'mixed-characters'
```

Z" または Z'
ヌル終了英数字リテラルの開始の区切り文字 開始の区切り文字の両方の文字 (Z" または Z') は、同じソース線になければなりません。

" または '
ヌル終了英数字リテラルの終了の区切り文字

開始の区切り文字に引用符を使用した場合は、終了の区切り文字にも引用符を使用する必要があります。同様に、開始の区切り文字にアポストロフィを使用した場合は、終了の区切り文字にもアポストロフィを使用する必要があります。

混合文字
これは以下の文字のいずれかにすることができます。

- 1 バイト文字のみ
- 1 バイト文字とマルチバイト文字の混合
- マルチバイト文字のみ
- UTF-8 文字
- EUC 文字
- DBCS 文字

ただし、値 X'00' を含む 1 バイト文字は指定できません。 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。 それ以外については、リテラルの内容には、マルチバイト文字が含まれる英数字リテラル (フォーマット 1) と同じ規則および制約事項が適用されます。

リテラル内容に含まれる文字ストリングの長さは、0 から 159 バイトになります。 リテラルの実際の長さにはヌル終了文字が含まれるので、最大長は 160 バイトです。

ヌル終了英数字リテラルは、「英数字」データ・クラスおよびカテゴリーに属します。 ヌル終了英数字リテラルは、英数字リテラルを使用できる任意の場所で使用できますが、ALL リテラル 形象定数ではサポートされていません。

LENGTH 組み込み関数がヌル終了英数字リテラルに適用される場合、ヌル終了文字より前にそのリテラルのバイト数を戻しますが、そのヌル終了文字は含まれません。(LENGTH 特殊レジスターはリテラルのオペランドをサポートしていません。)

ブール・リテラル

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。

DBCS リテラル

このセクションでは、DBCS リテラルの形式および規則を示します。

DBCS リテラルのフォーマット

```
G"DBCS-characters"
G'DBCS-characters'
N"DBCS-characters"
N'DBCS-characters'
```

G"、 G'、 N"、または N'
開始の区切り文字。

NSYMBOL(DBCS) コンパイラー・オプションが有効な場合、N" と N' は DBCS リテラルを識別します。これらの区切り文字は、NSYMBOL(NATIONAL) コンパイラー・オプションが有効な場合は国別リテラルを識別します。その場合は、31 ページの『国別リテラル』で解説している規則が適用されます。

" または '
終了の区切り文字。 開始の区切り文字に引用符を使用した場合は、終了の区切り文字にも引用符を使用する必要があります。 同様に、開始の区切り文字にアポストロフィを使用した場合は、終了の区切り文字にもアポストロフィを使用する必要があります。

DBCS-characters
任意の DBCS 文字。

最大長
最大長は、1 つのソース行で使用可能なスペースによって制限されています。

継続規則
複数行にまたがって続けることはできません。

SOSI コンパイラー・オプションを指定した DBCS リテラル

SOSI コンパイラー・オプションが有効な場合、ワークステーションのシフトアウト (SO) 制御文字とシフトイン (SI) 制御文字によって、ソース・テキスト内の DBCS 文字が区切られます。 以下のセクションに、シフトインとシフトアウトの区切り文字を持つ DBCS リテラルを示します。

DBCS リテラルのフォーマット

```
G"<DBCS-characters>"
G'<DBCS-characters>'
N"<DBCS-characters>"
N'<DBCS-characters>'
```

<
シフトアウト制御文字 (X'1E') を表します。

>
シフトイン制御文字 (X'1F') を表します。

DBCS 文字、リテラル区切り文字、最大長、および継続に関する規則は、SOSI コンパイラー・オプションのない DBCS リテラルと同じです。 SOSI コンパイラー・オプションの詳細については、*COBOL for Linux on x86* プログラミング・ガイド内の *SOSI* を参照してください。

DBCS リテラルを使用できる場合

DBCS リテラルは、次のような個所で使用できます。

- データ部
  - DBCS クラスのデータ項目を定義するデータ記述項目の VALUE 節の中
  - ファイル記述項目の VALUE OF 節の中
- 手続き部
  - 被比較数が DBCS データ項目、国別クラスの基本データ項目、国別グループ項目、または英数字グループ項目の場合の比較条件の中
  - CALL ステートメントの BY CONTENT を渡される引数として
  - DISPLAY ステートメントおよび EVALUATE ステートメントの中
  - 以下のステートメントの中
    - INITIALIZE。詳細については、312 ページの『INITIALIZE ステートメント』を参照してください。
    - INSPECT。詳細については、315 ページの『INSPECT ステートメント』を参照してください。
    - MOVE。詳細については、327 ページの『MOVE ステートメント』を参照してください。
    - STRING。詳細については、384 ページの『STRING ステートメント』を参照してください。
    - UNSTRING。詳細については、392 ページの『UNSTRING ステートメント』を参照してください。
  - 形象定数 ALL の中

  - NATIONAL-OF 組み込み関数への引数として
- コンパイラー指示ステートメント COPY、REPLACE、および TITLE

数字リテラル

数字リテラル とは、0 から 9 の数字、符号文字 (+ または -)、および小数点で構成される文字ストリングです。

リテラルが小数点を含まない場合、そのリテラルは整数です。 (本書では、フォーマットの中に現れる整数という語は、符号と小数点を含まない非ゼロ値の数字リテラルを表しています。ただし、その他の規則がフォーマットの説明に含まれている場合は、その説明に従います。) 次の規則が適用されます。

- ARITH(COMPAT) コンパイラー・オプションが有効な場合は、1 から 18 桁が使用できます。
  ARITH(EXTEND) コンパイラー・オプションが有効な場合は、1 から 31 桁が使用できます。
- 符号文字は 1 つだけ使用できます。 符号文字を付ける場合、 それはリテラルの左端の文字でなければなりません。 リテラルが符号なしの場合、 そのリテラルは正の値です。
- 小数点は 1 つだけ使用できます。 小数点を入れる場合は、 想定小数点として扱われます (つまり、 リテラル中の 1 桁をとりません)。 小数点は、 右端の文字として以外であればリテラル中のどこでも置けます。

数字リテラルの値は、 リテラルの中の数字によって表現される代数的な量です。 数字リテラルのサイズは、ユーザーが指定した数字の桁数と同じです。

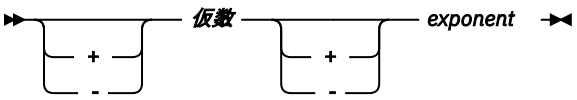
数字リテラルは、固定小数点数または浮動小数点数です。

数字リテラルは、数値のデータ・クラスおよびカテゴリーに属します。 (データ・クラスおよびカテゴリーについては、137 ページの『データのクラスとカテゴリー』で解説しています)。

浮動小数点リテラルの値に関する規則

浮動小数点リテラルのフォーマットおよび規則を以下に示します。

フォーマット



- 仮数および指数の前の符号はオプショナルです。 符号を省略すると、コンパイラーは正の値を想定します。
- 仮数は、1 から 16 桁の数字を含めることができます。 小数点は、仮数に含めなければなりません。
- 指数は、E の文字とそれに続くオプションの符号および 1 桁か 2 桁の数字によって表現されます。
- 浮動小数点リテラルの値の大きさの範囲は次のとおりです。
  - 32 ビット表記: $1.175(10^{-38})$ から $3.403(10^{38})$

国別リテラル

COBOL for Linux で提供される国別リテラルの形式は、基本国別リテラルと、国別リテラルの 16 進表記です。

形式について詳しくは、31 ページの『基本国別リテラル』および 32 ページの『国別リテラルの 16 進表記』を参照してください。

基本国別リテラル

このセクションでは、基本国別リテラルの形式および規則を示します。

フォーマット 1: 基本国別リテラル

```
N"character-data"
N'character-data'
```

NSYMBOL(NATIONAL) コンパイラー・オプションが有効な場合、開始の区切り文字 N" または N' によって国別リテラルが識別されます。 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。

NSYMBOL(DBCS) コンパイラー・オプションが有効な場合は、開始の区切り文字 N" または N' によって DBCS リテラルが識別されます。その場合は、29 ページの『DBCS リテラル』で解説している規則が適用されます。

N" または N'
開始の区切り文字。 開始の区切り文字は、1 バイト文字としてコーディングする必要があります。 開始の区切り文字を複数の行にまたがって継続することはできません。

" または '
終了の区切り文字。 終了の区切り文字は、1 バイト文字としてコーディングする必要があります。 開始の区切り文字に引用符を使用した場合は、終了の区切り文字にも引用符を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。

開始の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。 例えば、次のように指定します。

```
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"
```

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。 文字データには、ソース・コードに有効なコード・ページで表現される 1 バイト文字およびマルチバイト文字の任意の組み合わせを使用することができます。

リテラルの内容の DBCS 文字は、SOSI コンパイラー・オプションで説明しているようにワークステーションのシフトアウト制御文字とシフトイン制御文字で区切ることができます。 SOSI コンパイラー・オプションの詳細については、*COBOL for Linux on x86* プログラミング・ガイド内の *SOSI* を参照してください。

最大長
国別リテラルの最大長は 80 文字位置 (開始と終了の区切り文字を除く) です。 リテラルのソース内容に 1 つ以上のマルチバイト文字が含まれている場合は、最大長は単一のソース行の領域 B で使用可能なスペースまでです。

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。 リテラルに含まれる各 1 バイト文字は 1 つの文字位置としてカウントされ、 リテラルに含まれる各マルチバイト文字は 1 つの文字位置としてカウントされます。 DBCS 文字のワークステーション・シフトイン区切り文字とシフトアウト区切り文字は、カウントされません。

継続規則
リテラルの内容にマルチバイト文字が含まれている場合は、リテラルを継続できません。 リテラルの内容にマルチバイト文字が含まれていない場合は、標準の継続規則が適用されます。

文字データ のソース・テキスト表現は、実行時の使用のために自動的に UTF-16 へ変換されます (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たときや、国別カテゴリーのデータ項目と比較されたとき)。

国別リテラルの 16 進表記

このセクションでは、国別リテラルの 16 進表記の形式および規則を示します。

フォーマット 2: 国別リテラルの 16 進表記

```
NX"hexadecimal-digits"
NX'hexadecimal-digits'
```

国別リテラルの 16 進表記形式は、NSYMBOL コンパイラー・オプションによる影響を受けません。

NX" または NX'
開始の区切り文字。 開始の区切り文字は、1 バイト文字で表現する必要があります。 開始の区切り文字を複数の行にまたがって継続することはできません。

" または '
終了の区切り文字。 終了の区切り文字は、1 バイト文字で表現する必要があります。

開始の区切り文字に引用符を使用した場合は、終了の区切り文字にも引用符を使用する必要があります。同様に、開始の区切り文字にアポストロフィを使用した場合は、終了の区切り文字にもアポストロフィを使用する必要があります。

***16* 進数字**
'0' から '9'、'a' から 'f'、および 'A' から 'F' の範囲に含まれる 16 進数字。 4 つの 16 進数字からなるグループで 1 つの国別文字を表現します。 各グループは UTF-16 に含まれる有効なコード・ポイントを表現している必要があります。 16 進数字の数は、4 の倍数でなければなりません。

最大長
16 進表記の国別リテラルの長さは 4 から 320 文字の 16 進文字 (開始と終了の区切り文字を除く) でなければなりません。 長さは 4 の倍数でなければなりません。

継続規則
標準の継続規則が適用されます。

16 進表記の国別リテラルの内容は、国別文字として保管されます。 結果としての内容が意味するものは、同じ国別文字を指定する基本国別文字が意味するものと同じです。

16 進表記の国別リテラルは、「国別」データ・クラスおよびカテゴリーに属し、基本国別リテラルを 使用できる場所であれば、どこでも使用できます。

国別リテラルを使用できる場合

国別リテラルはさまざまな方法で使用できます。

国別リテラルは、次のような個所に使用できます。

- 国別クラスのデータ項目に関連付けられた VALUE 節、または USAGE NATIONAL で定義された条件変数の条件名に関連付けられた VALUE 節の中
- 形象定数 ALL の中
- 比較条件の中
- フォーマット 2 の SEARCH ステートメントの WHEN 句の中 (二分探索)
- INSPECT ステートメントの ALL 句、LEADING 句、または FIRST 句の中
- INSPECT ステートメントの BEFORE 句または AFTER 句の中
- STRING ステートメントの DELIMITED BY 句の中
- UNSTRING ステートメントの DELIMITED BY 句の中
- CALL ステートメントの BY CONTENT で渡される引数として
- CALL ステートメントの BY VALUE で渡される引数として
- DISPLAY ステートメントおよび EVALUATE ステートメントの中
- 以下のプロシージャー・ステートメントの送り出し項目として
  - INITIALIZE
  - INSPECT

  - MOVE
  - STRING
  - UNSTRING
- 以下の組み込み関数に対する引数リストの中

  DISPLAY-OF、LENGTH、LOWER-CASE、MAX、MIN、ORD-MAX、ORD-MIN、REVERSE、UPPER-CASE、USUPPLEMENTARY、および UVALID

  注 : DBCS リテラルを USUPPLEMENTARY および UVALID 関数で使用することはできません。
- コンパイラー指示ステートメント COPY、REPLACE、および TITLE の中

国別リテラルは、本書の詳細規則に従って使用する必要があります。

PICTURE 文字ストリング

PICTURE 文字ストリング は、通貨記号と COBOL 文字セットの中の特定の組み合わせから構成されます。 PICTURE 文字ストリングは、分離文字のスペース、コンマ、セミコロン、 またはピリオドによってのみ区切られます。

PICTURE 節記号の図は、181 ページの表 17 に示されています。

コメント

コメント は、コンピューターの文字セットの文字を任意に組み合わせたものからなる文字ストリングです。

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 3 つの形式があります。

コメント項目 (IDENTIFICATION DIVISION)
この形式については、85 ページの『オプションの段落』 に説明があります。

コメント行 (任意の部)
この形式については、46 ページの『コメント行』 に説明があります。

インライン・コメント (任意の部)
インライン・コメントは、プログラムのテキスト域にある、前に 1 つ以上の文字ストリングが付いた浮動コメント標識 (*>) で示され、コンパイル・グループの任意の行に書き込むことができます。 浮動コメント標識に続く、領域 B の終わりまでの文字はすべて、コメント・テキストです。

コメントを構成する文字ストリングには、1 バイト文字またはコンパイル用に組み合わせることができるコード・ページのマルチバイト文字を使用できます。

マルチバイト・ストリングを含む複数のコメント行も使用できます。 コメント行へのマルチバイト文字の埋め込みは、行単位で行う必要があります。 これらの文字を含むワードを複数行にまたがって継続することはできません。 コメント行のストリングの有効性に関する構文検査は行われません。

第 4 章 分離文字

分離文字 は、文字ストリングを区切る 1 つの文字、または複数の連続した文字です。

分離文字を以下の表に示します。

表 *4.* **分離文字**	
分離文字	**意味**
b[1]	スペース
,*b*[1]	コンマ
.*b*[1]	ピリオド
;*b*[1]	セミコロン
(	左括弧
)	右括弧
:	コロン
"*b*[1]	引用符
'*b*[1]	アポストロフィ
B"	ブール・リテラルの開始分離文字
X"	16 進形式英数字リテラルの開始の区切り文字
X'	16 進形式英数字リテラルの開始の区切り文字
Z"	ヌル終了英数字リテラルの開始の区切り文字
Z'	ヌル終了英数字リテラルの開始の区切り文字
N"	国別リテラルの開始の区切り文字 [2]
N'	国別リテラルの開始の区切り文字 [2]
NX"	16 進形式国別リテラルの開始の区切り文字
NX'	16 進形式国別リテラルの開始の区切り文字
G"	DBCS リテラルの開始の区切り文字
G'	DBCS リテラルの開始の区切り文字
==	疑似テキスト区切り文字

1. *b* はブランクを表します。
2. NSYMBOL(DBCS) コンパイラー・オプションが有効な場合、N" および N' は DBCS リテラルの開始区切り文字です。

分離文字の規則

分離文字は、1 つ以上の句読文字のストリングです。

以下の説明では、{} (中括弧) はそれぞれの分離文字を囲み、*b* はスペースを表しています。 スペースが分離文字または分離文字の一部として使用されるところでは、複数のスペースを使用できます。

スペース {*b*}
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。

- 開始の疑似テキスト区切り文字 (先行スペースが必要なところ)。
- 引用符号で囲まれた内部。 引用符と引用符の間にあるスペースは、英数字リテラルの一部とみなされ、 分離文字とはみなされません。

ピリオド {*.b*}、 コンマ {*,b*}、 セミコロン {*;b*}
分離文字コンマは 1 つのコンマとその後の 1 つのスペースで構成されます。 分離文字ピリオドは 1 つのピリオドとその後の 1 つのスペースで構成されます。 分離文字セミコロンは 1 つのセミコロンとその後の 1 つのスペースで構成されます。

分離文字ピリオドは、ある文の終わりを示す場合にだけ使用するか、 またはフォーマットに示されているとおりに使用しなければなりません。 分離文字コンマと分離文字セミコロンは、分離文字スペースが使用される場合は、どこでも使用できます。

- IDENTIFICATION DIVISION では、それぞれの段落が分離文字ピリオドで終わっていなければなりません。
- ENVIRONMENT DIVISION では、 SOURCE-COMPUTER、 OBJECT-COMPUTER、 SPECIAL-NAMES、および I-O-CONTROL 段落は分離文字ピリオドで終わっていなければなりません。 FILE-CONTROL 段落では、それぞれのファイル制御項目が分離文字ピリオドで終わっていなければなりません。
- DATA DIVISION では、ファイル (FD)、ソート/マージ・ファイル (SD)、およびデータ記述項目がそれぞれ分離文字ピリオドで終わっていなければなりません。
- PROCEDURE DIVISION では、分離文字コンマまたは分離文字セミコロンで、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。

括弧 {(} ... {) }
疑似テキストの中を除き、括弧は左右の括弧が対応した形で使用しなければなりません。 括弧は、添え字、関数の引数のリスト、参照修飾子、算術式、または条件を区切ります。

コロン {: }
コロンは分離文字の 1 つで、一般フォーマットの中に示されているときは必須です。

引用符 {"} ... {"}
開始の引用符は、 直前にスペースまたは左括弧がなければなりません。 終了の引用符は、直後に分離文字 (スペース、コンマ、セミコロン、 ピリオド、右括弧、または疑似テキスト区切り文字) が必要です。 引用符は、対で使用しなければなりません。 これらは英数字リテラルを区切ります。ただし、そのリテラルが継続している場合は別です (44 ページの『継続行』を参照)。

アポストロフィ {'} ... {'}
開始のアポストロフィは、直前にスペースまたは左括弧がなければなりません。 終了のアポストロフィは、直後に分離文字 (スペース、コンマ、セミコロン、ピリオド、右括弧、または疑似テキスト区切り文字) が必要です。 アポストロフィは、対になって使わなければなりません。 これらは英数字リテラルを区切ります。ただし、そのリテラルが継続している場合は別です (44 ページの『継続行』を参照)。

ヌル終了リテラル区切り文字 {Z"} ... {"}, {Z'} ... {'}
開始の区切り文字は、直前にスペースまたは左括弧がなければなりません。 終了の区切り文字は、直後に分離文字 (スペース、コンマ、セミコロン、ピリオド、右括弧、または疑似テキスト区切り文字) が必要です。

DBCS リテラル区切り文字 {G"} ... {"}、{G'} ... {'}、{N"} ... {"}, {N'} ... {'}
開始の区切り文字は、直前にスペースまたは左括弧がなければなりません。 終了の区切り文字は、直後に分離文字 (スペース、コンマ、セミコロン、ピリオド、右括弧、または疑似テキスト区切り文字) が必要です。 NSYMBOL(DBCS) コンパイラー・オプションが有効な場合、N" および N' は DBCS リテラル区切り文字です。

国別リテラル区切り文字 {N"} ... {"}, {N'} ... {'}、{NX"} ... {"}、{NX'} ... {'}
開始の区切り文字は、直前にスペースまたは左括弧がなければなりません。 終了の区切り文字は、直後に分離文字 (スペース、コンマ、セミコロン、ピリオド、右括弧、または疑似テキスト区切り文字) が

必要です。 NSYMBOL(DBCS) コンパイラー・オプションが有効な場合、N" および N' は DBCS リテラル区切り文字です。

疑似テキスト区切り文字 {*b*==} ... {==*b*}
開始の疑似テキスト区切り文字は、 直前にスペースがなければなりません。 終了の疑似テキスト区切り文字は、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) が必要です。 疑似テキスト区切り文字は、対で使用しなければなりません。 これらは疑似テキストを区切ります。 (476 ページの『COPY ステートメント』を参照。)

PICTURE 文字ストリング、コメント文字ストリング、 または英数字リテラルの中に含まれる句読記号は、句読記号とみなされず、文字ストリングまたはリテラルの一部とみなされます。

第 5 章 セクションと段落

セクションと段落は、プログラムを定義するものです。 セクションと段落は、文、ステートメント、および項目に細分されます。

文はステートメントに細分され、ステートメントはさらに句に細分されます。 項目は、節に細分されます。

詳しくは、以下を参照してください。

- 39 ページの『文、ステートメント、および項目』
- 40 ページの『ステートメント』
- 40 ページの『句』
- 40 ページの『節』

セクション、段落、およびステートメントに関する詳細については、 233 ページの『プロシージャー』 を参照してください。

文、ステートメント、および項目

関連する規則が他に特に明記していない限り、 それぞれに必須の節やステートメントは、 そのフォーマットに示されたシーケンスで記述しなければなりません。 オプションの節やステートメントを使用する場合には、 それらのフォーマットに示されているシーケンスで記述しなければなりません。 これらの規則は、コメントとして扱われる節やステートメントに関しても同様に適用されます。

構文の階層は、次のとおりです。

- IDENTIFICATION DIVISION
  - 段落
    - 項目
      - 節
- ENVIRONMENT DIVISION
  - セクション
    - 段落
      - 項目
        - 節
          - 句
- DATA DIVISION
  - セクション
    - 項目
      - 節
        - 句
- PROCEDURE DIVISION
  - セクション
    - 段落
      - 文
        - ステートメント
          - 句

項目

項目 とは、分離文字ピリオドで終わる一連の節のことです。 項目は、見出し部、環境部、およびデータ部において指定できます。

節

節 とは、項目の属性を指定するために順番に並べられた、連続する COBOL 文字ストリングの集合のことです。 節は、見出し部、環境部、およびデータ部において指定できます。

文

文 とは、分離文字ピリオドで終わる 1 つ以上のステートメントの列です。 文は、PROCEDURE DIVISION で作成できます。

ステートメント

ステートメント は、プログラムによって取られる処置を指定します。 ステートメントは、PROCEDURE DIVISION で作成できます。

各種のステートメントの説明については、次の個所を参照してください。

- 260 ページの『命令ステートメント』
- 261 ページの『条件ステートメント』
- 49 ページの『第 7 章 名前のスコープ』
- 473 ページの『第 21 章 コンパイラー指示ステートメント』

句

プログラムの中のそれぞれの節やステートメントは、句 と呼ばれるさらに小さな単位に細分化されることがあります。

第 6 章 参照形式

COBOL ソース・テキストは COBOL 参照形式 で作成する必要があります (これは、固定ソース・フォーマット または拡張ソース・フォーマット のいずれかになります)。

固定ソース形式は、72 文字を 1 行として、以下の領域から構成されます。 拡張ソース形式 (Extended Source Format) は、252 文字を 1 行として以下の領域から構成されます。固定ソース・フォーマットでも拡張ソース・フォーマットでも、プログラム・テキスト域は文字位置 8 から始まり、領域 B の終わりまで続きます。

シーケンス番号域
桁 1 から 6

標識域
桁 7

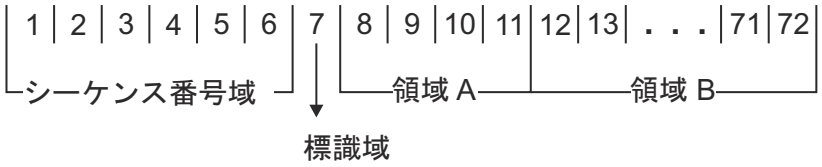
領域 A
桁 8 から 11

領域 B
桁 12 から 72 の固定ソース形式

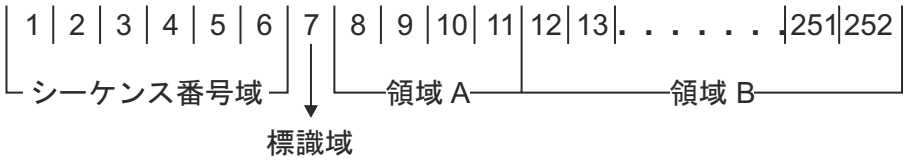
桁 12 から 252 の拡張ソース形式 (Extended Source Format)

要確認 : 固定ソース形式および拡張ソース形式 (Extended Source Format)では、最大長よりも短い行はスペースで最大長まで拡張されます。

この図は、COBOL ソース行の固定ソース形式を示しています。



この図は、COBOL ソース行の拡張ソース形式 (Extended Source Format) を示しています。



ソース形式を示す方法の詳細については、*COBOL for Linux on x86* プログラミング・ガイド内の *SRCFORMAT* を参照してください。

以下のトピックには上記の領域に関する詳細が記載されています。

- 41 ページの『シーケンス番号域』
- 42 ページの『標識域』
- 42 ページの『領域 A』
- 43 ページの『領域 B』
- 45 ページの『領域 A または領域 B』
- 47 ページの『ソース変換ユーティリティー (scu)』

シーケンス番号域

シーケンス番号域は、ソース・ステートメント行にラベルを付けるために使用されます。 この領域の内容には、コンピューターの文字セットの中のどの文字でも使用することができます。

標識域

ワードまたは英数字リテラルが前の行から現在行に継続していることを指定し、テキストを文書として扱うことを指定し、またデバッグ行を指定するには、標識域を使用します。

44 ページの『継続行』、46 ページの『コメント行』、および 47 ページの『デバッグ行』を参照してください。

標識域は、ソース・リスト・フォーマット設定に使用することができます。 標識列に置かれたスラッシュ (/) は、そのソース・リストの新しいページを開始するようにコンパイラーに指示し、対応するソース・レコードをコメントとして扱うようにします。 その効果は、LINECOUNT コンパイラー・オプションによって決まります。 LINECOUNT コンパイラー・オプションについては、「*COBOL for Linux on x86* プログラミング・ガイド」内の 『*LINECOUNT*』 を 参照してください。

領域 A

ある項目は、領域 A から開始しなければなりません。

このような項目には以下があります。

- 部のヘッダー
- 42 ページの『セクション・ヘッダー』
- 段落ヘッダーまたは段落名
- レベル標識またはレベル番号 (01 および 77)
- DECLARATIVES および END DECLARATIVES
- END PROGRAM のマーカー

部のヘッダー

部のヘッダーは、ワードの組み合わせと、その直後に置かれ、部の始まりを示す分離文字ピリオドで構成されます。

以下の部のヘッダーを参照してください。

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

PROCEDURE DIVISION のヘッダーで USING 句を指定する場合を除き、部のヘッダーのすぐ後には分離文字ピリオドが続く必要があります。 USING 句を除いて、同じ行にテキストがあってはなりません。

セクション・ヘッダー

環境部と手続き部の中で、セクション・ヘッダーは一連の段落の開始を示します。

次に例を示します。

```
INPUT-OUTPUT SECTION.
```

DATA DIVISION では、セクション・ヘッダーは項目の開始を示します。 以下に例を示します。

```
FILE SECTION.
LINKAGE SECTION.
LOCAL-STORAGE SECTION.
WORKING-STORAGE SECTION.
```

セクション・ヘッダーは、そのすぐ後に分離文字ピリオドが必要です。

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、段落の開始を示します。

ENVIRONMENT DIVISION では、段落は段落ヘッダーとそれに続く 1 つ以上の項目から構成されます。 次に例を示します。

```
OBJECT-COMPUTER. コンピューター名.
```

PROCEDURE DIVISION では、段落は段落名とそれに続く 1 つ以上の文から構成されます。

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

レベル標識は、FD か SD のいずれかにすることができます。

レベル標識は領域 A 内から始め、その後にスペースがなければなりません。 (150 ページの『FILE SECTION』 を参照してください。) 領域 A で開始されなければならないレベル番号は、01 または 77 の値の 1 桁か 2 桁の整数です。 これはその後にスペースまたは分離文字ピリオドが続く必要があります。

DECLARATIVES および END DECLARATIVES

DECLARATIVES と END DECLARATIVES は、ソース単位の宣言部分の始まりと終わりを示すキーワードです。

PROCEDURE DIVISION では、キーワード DECLARATIVES および END DECLARATIVES はそれぞれ領域 A で開始し、その後にすぐ分離文字ピリオドを付けなければなりません。それ以外のテキストは同じ行にはありません。 キーワード END DECLARATIVES の後には、 次のセクション・ヘッダーより前に何かテキストを置くことはできません。 (232 ページの『宣言部分』 を参照してください。)

END PROGRAM マーカー

END PROGRAM マーカーは、COBOL プログラムの終わりを示します。

次に例を示します。

```
END PROGRAM program-name.
```

PROGRAM の場合
プログラム名 は、 対応している PROGRAM-ID 段落のプログラム名 と一致している必要があります。 COBOL プログラムは、ネストされたプログラムがなく、 その後に別のバッチ・プログラムが続かない最外部のプログラムを除き、 いずれも END PROGRAM マーカーで終わっていなければなりません。

領域 B

ある項目は、領域 B から開始しなければなりません。

このような項目には以下があります。

- 項目、文、ステートメント、および節
- 継続行

項目、文、ステートメント、節

最初の項目、文、ステートメント、または節は、 前にあるヘッダーまたは段落名と同じ行から始めるか、あるいは、コメント行でもなくかつブランク行でもない次の行の領域 B から始めます。 連続する文または項目は、 その前の文または項目と同じ行の領域 B から始めるか、あるいは、 コメント行でもなくかつブランク行でもない次の行の領域 B から始めます。

1 つの項目や文の中では、領域 B にある連続する行は、同じフォーマットにすることも、プログラム・ロジックを見やすくするために字下げすることもできます。 入力されたステートメントが字下げされている場合のみ、出力リストが字下げされて印刷できます。 字下げしてもプログラムの意味は変わりません。 領域 B の幅に関する制約に従えば、 プログラマーはどれだけ字下げするかを自由に決めることができます。 39 ページの『第 5 章 セクションと段落』も参照してください。

継続行

複数の行を必要とする文、項目、節、または句は、 次の行 (コメント行や意図的なブランク行以外の行) の領域 B に続けることができます。

継続前の行は 継続される行であり、継続後の行は継続行です。 継続行の領域 A は、ブランクでなければなりません。

標識域 (7 桁目) がハイフン (-) でない場合、先行する行の最後の文字の後にスペースがあるとみなされます。

以下の項目は継続できません。

- マルチバイト・ユーザー定義語
- DBCS リテラル
- マルチバイト 文字を含む英数字リテラル
- マルチバイト 文字を含む国別リテラル

ただし、16 進表記の英数字リテラルおよび国別リテラルは、16 進表記での文字表現の種類に関係なく、継続することができます。

開始のリテラル区切り文字を構成する文字は、すべて同じ行になければなりません。 例えば、Z"、G"、N"、NX"、または X"。

疑似テキスト区切り文字 =、浮動コメント標識 *>、またはコンパイラー・ディレクティブ標識 >> を構成する文字は、両方とも同じ行に置かれている必要があります。

コンパイラー・ディレクティブまたはコンパイラー・ディレクティブ句 (>> で始まるもの) は同じ行に指定する必要があります。

ある行の標識域にハイフンがある場合、 その継続行の最初のブランク以外の文字は、 間にスペースを置かずに、 その継続される行の最後のブランク以外の文字のすぐ後に続きます。

英数字リテラルおよび国別リテラルの継続

英数字リテラルおよび国別リテラルは、そのリテラルの内容に マルチバイト文字が含まれていない場合にのみ、 継続することができます。

次の規則は、以下のように マルチバイト文字を含まない英数字リテラル、および国別リテラルに適 用されます。

- 継続される行に、英数字リテラルまたは国別リテラルが含まれているが、終了の引用符がない場合、その行の末尾 (72 桁まで (固定ソース形式)、または 252 桁まで (拡張ソース形式)) にあるスペースはすべてリテラルの一部とみなされます。 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、その引用符のすぐ後の文字から始まります。
- 次の行に継続される英数字リテラルまたは国別リテラルが、72 桁目 (固定ソース形式) または 252 桁目 (拡張ソース形式) に最後の文字として引用符を 1 つ持つ場合、その継続行は 2 つの連続した引用符で開始されなければなりません。 これによって、引用符がリテラルの値の一部となります。

  英数字リテラルまたは国別リテラルの継続される行にある最後の文字が、領域 B にある引用符である場合、継続行は引用符で始めることができます。これは、1 つのリテラルが行にまたがって継続しているとみなされず、 2 つの連続したリテラルとみなされます。

区切り文字で引用符の代わりにアポストロフィが使用されている場合にも、同じ規則が適用されます。

リテラルを継続して、ある行とその継続行が 1 つのリテラルを構成するようにするには、次のようにします。

- それぞれの継続行の標識域でハイフンをコーディングします。
- 継続される行のすべての桁 (72 桁目まで (固定ソース形式)、または 252 桁目まで (拡張ソース形式)) を使用して、 リテラルの値をコーディングします。 (継続される行を、スペースが続く引用符で終了してはなりません)。
- それぞれの継続行のリテラルの先頭文字の前で引用符をコーディングします。
- 最後の継続行を、引用符にスペースを続けて終了します。

以下の固定ソース形式の例では、作成されるリテラルの数およびサイズを示しています。

```
|...+.*..1....+....2....+....3....+....4....+....5....+....6....+....7..
000001               "AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDDEEEEEEEEEE
      -              "GGGGGGGGGGHHHHHHHHHHIIIIIIIIIIJJJJJJJJJJKKKKKKKKKK
      -    "LLLLLLLLLLMMMMMMMMMM"
```

- リテラル 000001 は、長さ 120 バイトの 1 つの英数字リテラルと解釈されます。 継続される行の開始の引用符と最高 72 桁までの間の文字は、 リテラルの一部としてカウントされます。

```
|...+.*..1....+....2....+....3....+....4....+....5....+....6....+....7..
000003             N"AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDDEEEEEEEEEE
      -              "GGGGGGGGGG"
```

- リテラル 000003 は、国別文字位置 60、長さ 120 バイトの 1 つの国別リテラルと解釈されます。 継続される行の開始の引用符マークと終了の引用符マークとの間にあるすべての文字は、 リテラルの一部としてカウントされます。 1 バイト文字が入力されていますが、リテラルの値は国別文字として保管されます。

```
|...+.*..1....+....2....+....3....+....4....+....5....+....6....+....7..
000005     "AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDDEEEEEEEEEE
      -    "GGGGGGGGGGHHHHHHHHHHIIIIIIIIIIJJJJJJJJJJKKKKKKKKKK
      -    "LLLLLLLLLLMMMMMMMMMM"
```

- リテラル 000005 は長さが 140 バイトの 1 つのリテラルと解釈されます。 継続される行は引用符で終了しないため、 それぞれの継続される行の最後のブランクはリテラルの一部としてカウントされます。

```
|...+.*..1....+....2....+....3....+....4....+....5....+....6....+....7..
000010     "AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDDEEEEEEEEEE"
      -    "GGGGGGGGGGHHHHHHHHHHIIIIIIIIIIJJJJJJJJJJKKKKKKKKKK"
      -    "LLLLLLLLLLMMMMMMMMMM"
```

- リテラル 000010 は 3 つの別個のリテラルとして解釈されます。 それぞれの長さは 50、50、および 20 バイトです。 引用符の後にスペースが続くと、継続される行を終了します。 引用符の中の文字だけが、リテラルの一部としてカウントされます。 リテラル 000010 は、非レベル 88 データ項目の VALUE 節として有効ではありません。

リテラルのそれぞれの連続部分の長さが領域 B の長さより短い連続リテラルをコーディングするには、連続部分の最後の文字が 72 桁 (固定ソース形式)、または 252 桁 (拡張ソース形式) になるように始まりの桁を調整してください。

領域 A または領域 B

ある項目は、領域 A または領域 B のどちらからでも始めることができます。

このような項目には以下があります。

- レベル番号
- コメント行
- 浮動コメント標識 (*>)
- コンパイラー指示ステートメント
- コンパイラー・ディレクティブ

- デバッグ行
- 疑似テキスト
- ブランク行

レベル番号

領域 A または領域 B で始まるレベル番号は、1 桁または 2 桁の整数で、その値は 02 から 49、66、または 88 です。

領域 A で開始されなければならないレベル番号は、01 または 77 の値の 1 桁か 2 桁の整数です。 レベル番号は後にスペースまたは分離文字ピリオドが続かなければなりません。 詳しくは、160 ページの『レベル番号』を参照してください。.

コメント行

コメント行とは、行の標識域 (7 桁目) にアスタリスク (*) またはスラッシュ (/) がある行、またはプログラムのテキスト域 (領域 A および領域 B) に最初の文字ストリングとして浮動コメント標識 (*>) がある行のことです。

コメントは、その行のプログラム・テキスト領域 のどこにでも書くことができ、 コンピューターの文字セットの文字を任意に組み合わせて書くことができます。

コメント行は、プログラムのどの場所ででも使用できます。 IDENTIFICATION DIVISION ヘッダーの前にコメント行を置くこともできますが、その場合には何らかの制御カード (例えば、PROCESS または CBL) を前に置く必要があります。

注: 制御カードとコメントが混在していると、制御カードによっては無効になるものがあ り、それらがエラーと診断される場合があります。

複数コメント行も可能です。 各コメント行は、標識域のアスタリスク (*) またはスラッシュ (/) 、または浮動コメント標識 (*>)で開始する必要があります。

浮動コメント標識 (floating comment indicator) の詳細については、46 ページの『浮動コメント標識 (*>)』を参照してください。

アスタリスク (*) が付いたコメント行は、出力リストでは、次に利用可能な行に出力されます。 その効果は、LINECOUNT コンパイラー・オプションによって決まります。 LINECOUNT コンパイラー・オプションについては、「*COBOL for Linux on x86* プログラミング・ガイド」内の 『*LINECOUNT*』 を 参照してください。 スラッシュ (/) が付いたコメント行の場合、出力リストの現行ページが送られ、次のページの最初の行に出力されます。

コンパイラーはコメント行を文書として扱い、 構文的なチェックをしません。

浮動コメント標識 (*>)

ソース参照形式の標識域に指定できる固定標識に加えて、コメント行またはインライン・コメントを示すために、浮動コメント標識 (*>) をプログラムのテキスト域のどこにでも指定できます。

浮動コメント標識がプログラムのテキスト域 (領域 A プラス領域 B) 内の最初の文字ストリングである場合は、この行がコメント行であることを示します。 また、浮動コメント標識がプログラムのテキスト領域内の 1 つ以上の文字ストリングの後にある場合は、インライン・コメントを示します。

浮動コメント標識 (floating comment indicator)の規則は次のとおりです。

- 複数文字の浮動標識を形成する両方の文字 (* および >) は、連続していて同じ行になければなりません。
- インライン・コメント用の浮動コメント標識 (floating comment indicator)は、セパレーター・スペースが先に置かれなければならず、セパレーター・スペースが指定可能な場所ならどこでも指定できます。
- 浮動コメント標識に続く、領域 B の終わりまでの文字はすべて、コメント・テキストです。

コンパイラー指示ステートメント

大部分のコンパイラー指示ステートメントは、COPY および REPLACE を含め、領域 A からでも領域 B からでも開始することができます。

BASIS、PROCESS (CBL)、*CBL (*CONTROL)、DELETE、EJECT、INSERT、 SKIP1、SKIP2、SKIP3、および TITLE ステートメントも、領域 A および領域 B のどちらでも開始できます。

コンパイラー指示

コンパイラー・ディレクティブは領域 B から開始しなければなりません。

詳しくは、497 ページの『第 22 章 コンパイラー指示』を参照してください。.

デバッグ行

デバッグ行 とは、行の標識域に D (または d) がある行です。

デバッグ行は、ENVIRONMENT DIVISION (OBJECT-COMPUTER 段落の後)、DATA DIVISION、 および PROCEDURE DIVISION に記述することができます。 デバッグ行で領域 A および領域 B にスペースしかない場合には、 それはブランク行とみなされます。

89 ページの『SOURCE-COMPUTER 段落』の『WITH DEBUGGING MODE』を参照してください。

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区切り文字は、領域 A または領域 B のどちらからでも始めることができます。

ただし、疑似テキスト開始区切り文字の次の行の標識領域 (7 桁目) にハイフンがある場合、その行の領域 A はブランクにしなければならず、継続行の規則がテキスト・ワードの形成に適用されます。 詳しくは、44 ページの『継続行』を参照してください。

ブランク行

ブランク行とは、7 から 72 桁 (固定ソース形式のとき)、または 7 から 252 桁 (拡張ソース形式 (Extended Source Format) のとき) まで、スペース以外何も含まない行のことです。 ブランク行は、プログラム内のどこにでも入れることができます。

ソース変換ユーティリティー (scu)

ソース変換ユーティリティー (scu) は、スタンドアロン Linux プログラムであり、COBOL ソース・プログラムを IBM 以外のまたは自由形式のソース形式から COBOL for Linux によってコンパイル可能な形式に変換するときの支援となります。

scu について詳しくは、529 ページの『付録 C ソース変換ユーティリティー (scu)』を参照してください。

第 7 章 名前のスコープ

ユーザー定義語は、データ・リソースまたは COBOL プログラミング・エレメントに名前を割り当てます。名前付きデータ・リソースの例としては、ファイル、データ項目、またはレコードがあります。 名前付きプログラミング・エレメントの例としては、プログラムや段落があります。

以下のセクションでは、COBOL での名前のタイプの定義、および名前の参照先を示します。

- 49 ページの『名前のタイプ』
- 51 ページの『外部および内部リソース』
- 51 ページの『名前の解決』

名前のタイプ

リソースの識別に加えて、名前の属性にはグローバル属性とローカル属性があります。 名前の一部はいつでもグローバルで、一部はいつでもローカルです。また、プログラムで定義される名前の仕様によっては、ローカルになったりグローバルになったりする名前もあります。

PROGRAM の場合

グローバル名 を使用して、 その名前が関連するリソースを次のプログラムから参照することができます。

- グローバル名が定義されるプログラムの中
- グローバル名を定義するプログラムに入っているその他のプログラムの中

名前がグローバルであることを示すには、データ記述項目で GLOBAL 節を使用します。 GLOBAL 節の使用の詳細については、151 ページの『GLOBAL 節』を参照してください。

ローカル名 を使用して、それが定義されたプログラムの中からローカル名に関連するリソースを参照することができます。

デフォルトでは、データ記述項目にあるデータ名、ファイル名、レコード名、レコード・キー名、または条件名の定義に GLOBAL 節が含まれない場合、その名前はローカル名です。

制約事項: 特定の規則によって、あるデータ記述、ファイル記述、 またはレコード記述項目に GLOBAL 節を指定できない場合があります。

以下のリストでは、使用できる名前と、 その名前がローカル名とグローバル名のどちらであるかを示しています。

データ名

データ名 はデータ項目に名前を割り当てます。

GLOBAL 節が、データ名を定義するデータ記述項目か、そのデータ記述項目が従属している別の項目のどちらかに指定されている場合、そのデータ名はグローバル名です。

ファイル名

ファイル名 はファイル結合子に名前を割り当てます。

ファイル名は、そのファイル名に対するファイル記述項目の中で GLOBAL 節が指定されている場合、グローバル名になります。

レコード名

レコード名 はレコードに名前を割り当てます。

GLOBAL 節がそのレコード名を定義するレコード記述で指定されている場合、あるいは FILE SECTION 内のレコード記述項目の場合、レコード記述項目と関連付けられているファイル名のファイル記述項目の中で GLOBAL 節が指定されている場合は、レコード名はグローバル名になります。

レコード・キー名

レコード・キー名は、索引付きファイルに関連付けられているキーに名前を割り当てます。

ALTERNATE RECORD KEY 節の SOURCE 句、または索引付きファイルのファイル制御項目の RECORD KEY 節を使用して、レコード・キー名を定義することができます。 そのファイルのファイル記述項目で GLOBAL 節が指定されている場合、レコード・キー名はグローバル名です。

制約事項 : レコード・キー名 は、STL ファイル・システムでのみサポートされます。

条件名
条件名 は条件変数に値を関連付けます。

データ記述項目が、GLOBAL 節を指定する別の項目に従属している場合は、条件名はそのデータ記述項目で定義されます。

構成セクションの中で定義される条件名は、常にグローバル名になります。

プログラム名
プログラム名 は、外部プログラムか内部 (ネストされた) プログラムのどちらかに名前を割り当てます。 詳しくは、78 ページの『プログラム名の命名規約』を参照してください。

プログラム名はローカル名でもグローバル名でもありません。 詳しくは、78 ページの『プログラム名の命名規約』を参照してください。

セクション名
セクション名 は PROCEDURE DIVISION のセクションに名前を割り当てます。

セクション名はいつでもローカル名です。

段落名
段落名 は PROCEDURE DIVISION の段落に名前を割り当てます。

段落名はいつでもローカル名です。

基本名
基本名 は、コンパイラーがソース単位に組み込むソース・テキストの名前を指定します。 詳細については、473 ページの『BASIS ステートメント』を参照してください。

ライブラリー名
ライブラリー名 は、コンパイラーが COPY テキストを組み込むために使用する COBOL ライブラリーを指定します。 詳細については、476 ページの『COPY ステートメント』を参照してください。

テキスト名
テキスト名 は、コンパイラーがソース単位に組み込む COPY テキストの名前を指定します。 詳細については、476 ページの『COPY ステートメント』を参照してください。

英字名
英字名 は、 ENVIRONMENT DIVISION の SPECIAL-NAMES 段落で特定の文字セットまたは 照合シーケンス、あるいはその両方に名前を割り当てます。

英字名はいつでもグローバル名です。

クラス名 (データの)
クラス名 は、 ENVIRONMENT DIVISION の SPECIAL-NAMES 段落で真の値を定義できる提案されたクラスに名前を割り当てます。

クラス名はいつでもグローバル名です。

簡略名
簡略名 はインプリメンター名にユーザー定義語を割り当てます。

簡略名はいつでもグローバル名です。

シンボリック文字
シンボリック文字 はユーザー定義形象定数を表します。

シンボリック文字はいつでもグローバル名です。

指標名
指標名 は特定のテーブルに関連する指標に名前を割り当てます。

グローバル属性のデータ項目が指標によってアクセスされるテーブルを含んでいる場合は、その指標もグローバル属性を持ちます。 さらに、指標名のスコープはテーブルが含まれるデータ名のスコープと同じです。

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。

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。

外部および内部リソース

データ項目またはファイル結合子に関連するストレージは、リソースが宣言 されるプログラムの外部 または内部 になることができます。

あるリソースに関連したストレージが実行単位の中の特定のプログラムではなく、その実行単位に 関連付けられている場合、データ項目またはファイル結合子は外部です。 外部リソースは、そのリソースについて記述する実行単位の中のどのプログラムからでも参照できます。 リソースの別個の記述を使用して異なるプログラムから外部リソースを参照することは、いつでも同じリソースを 参照することです。 1 つの実行単位の中では、外部リソースを代表するものは 1 つしかありません。

あるリソースに関連したストレージが、そのリソースについて記述するプログラムだけに関連付けられている場合、そのリソースは内部です。

外部リソースも内部リソースもグローバル名かローカル名のどちらかを持つことができます。

WORKING-STORAGE SECTION で記述されたデータ・レコードには、そのレコードのデータ記述項目に EXTERNAL 節があると、それによって外部属性が与えられます。 あるデータ記述項目によって記述されているデータ項目があり、そのデータ記述項目が、外部レコードを記述している データ記述項目に従属している場合は、そのようなデータ項目にも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は、それが記述され るプログラムの内部データの一部になります。

実行単位内の 2 つのプログラムは、次のような場合には同じファイル結合子を参照することができます。

- 外部ファイル結合子は、そのファイル結合子を記述しているどのプログラムからでも参照することができます。
- あるプログラムとそれを含むプログラムは、 含むプログラムの中で、 あるいは含むプログラムを直接的または間接的に含むプログラムの中で、 関連付けられたグローバル・ファイル名を参照することによって、 グローバル・ファイル結合子を参照できます。

実行単位内の 2 つのプログラムは、次のような場合には共通データを参照することができます。

- 外部データ・レコードのデータ内容が、どのようなプログラムからでも参照できるとき。ただし、 そのプログラムがそのデータ・レコードを記述していた場合。
- あるプログラムが別のプログラム内に含まれている場合、両方のプログラムは、次のどちらのデータも参照できる。すなわち、 プログラムの中にあるグローバル属性データ、またはその含んでいるプログラムを直接的または間接的に 含んでいるいずれかのプログラムの中にあるグローバル属性データ。

EXTERNAL 節が含まれていないファイル記述項目またはソート・マージ・ファイル記述項目に従属するものとして記述される データ・レコードは、そのようなレコードのデータ記述項目に従属するものとして記述されたデータ項目と同様に、いつでも ファイル名を記述するプログラムに対して内部です。 EXTERNAL 節にファイル記述項目が含まれていると、データ・レコードおよびデータ項目にも外部属性が与えられます。

名前の解決

あるプログラム (プログラム B) が別のプログラム (プログラム A) の中に含まれていると、それら 2 つのプログラムは、同じユーザー定義語で条件名、データ名、ファイル名、レコード・キー名、またはレコード

名を定義することができます。 そのような重複名がプログラム B で参照されると、 次のステップに従って、参照されるリソースが判別されます。。

1. 参照されたリソースは、 プログラム B で定義されたすべての名前のセットから、 およびプログラム A とそれを直接または間接に含んでいる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。1 つ以上のリソースが識別されるまで、この名前のセットに対して参照の修飾に関する標準規則および参照の固有性に関するその他の規則が適用されます。
2. リソースが 1 つしか識別されない場合は、それが参照されたリソースです。
3. 複数のリソースが識別される場合は、それらのリソースのうちの 1 つだけがプログラム 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っているリソースが 1 つしかない場合、 またはまったくない場合は、次の規則が当てはまります。
   - プログラム B で名前が宣言される場合は、 プログラム B のリソースは参照されたリソースである。
   - プログラム B で名前が宣言されない場合は、 参照されたリソースは次のようなものである。
     - プログラム A で名前が宣言されている場合は、プログラム A のリソース
     - プログラム A を含むプログラムで名前が宣言されている場合は、含んでいる方のプログラムのリソース。

     この規則は、有効なリソースが見つかるまで、 含んでいる方のプログラムに適用されます。

第 8 章 データ名、コピー・ライブラリー、および PROCEDURE DIVISION の名前を参照

参照は外部リソースおよび内部リソースに対して行うことができます。 データおよびプロシージャーは、明示的にも暗黙的にも参照することができます。

修飾に関する規則と、データを明示的および暗黙的に参照する場合の規則について詳しくは、以下のトピックを参照してください。

- 53 ページの『参照の固有性』
- 66 ページの『データ属性の指定』

参照の固有性

COBOL プログラムのユーザー定義名は、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にユーザーによって割り当てられるものです。 あるリソースを利用するには、COBOL プログラムのステートメントは、 そのリソースを固有なものとして識別するための参照名を含む必要があります。

参照の固有性を確保するために、ユーザー定義名の修飾を行うことができます。 テーブル・エレメントへの固有の参照のために 1 つの添え字が必要です。ただし、59 ページの『添え字付け』で指定されている場合を除きます。 データ名または関数名、添え字 (複数可)、特定の参照修飾子は、参照変更によって定義されたデータ項目を一意的に参照します。

別々のプログラムで、あるタイプのリソースの 2 つ以上のオカレンスに対して同じ名前が割り当てられているときに、修飾するのみではこれらのプログラムのうちのいずれかの参照で同じ名前のリソースを区別できない場合、名前の有効範囲を制限する特定の規約が適用されます。 この規約により、識別されるリソースが、参照を含んでいるプログラムで記述されたリソースになることが確実になります。 プログラム名の解決に関する詳細については、51 ページの『名前の解決』を参照してください。

ステートメントに関する規則によってそれ以外のことが指定されない場合は、 そのステートメントの実行の最初のステップとして、 添え字および参照変更が 1 回だけ評価されます。

修飾

ある名前が複数の名前の階層で存在している場合、 その階層の上位のレベルの名前を 1 つまたは複数指定することによって、名前を固有にすることができます。 高位レベルの名前は修飾子 と呼ばれ、このような名前を固有にするプロセスは修飾 と呼ばれます。

修飾を指定するには、ユーザー指定名の後に 1 つ以上の句を指定します。 それぞれの句は修飾子の前に IN または OF という語を 付けたものです。 (IN と OF は論理的に等価です。)

どのような階層においても、データ名を参照する場合、最高レベルに関連付けられたデータ名は固有でなければならず、これを修飾することはできません。

修飾は名前を固有にできるものでなければなりませんが、 常に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、EMPLOYEE-NO というフィールドを含むレコードを持つファイルが複数あり、そのうちの 1 つのファイルだけに MAIN-RECORD という名前のレコードがあるとします。この場合、次のことが言えます。

- EMPLOYEE-NO OF MAIN-RECORD という指定は、EMPLOYEE-NO を修飾するために十分な指定です。
- EMPLOYEE-NO OF MAIN-RECORD OF MAIN-FILE は有効ですが、不必要です。

修飾の規則

名前を修飾する際の規則は、次のとおりです。

- 名前を修飾する必要がない場合でも、それを修飾できます。 ただし、REDEFINES 節では修飾してはなりません。

- それぞれの修飾子は、それが修飾する名前より高いレベルになければならず、また同じ階層構造に属していなければなりません。
- 固有なものにすることができる修飾子の組み合わせが複数ある場合には、 そのような組み合わせのいずれを使用することもできます。

同一の名前

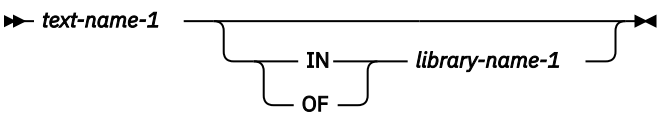
プログラムが直接または間接に他のプログラムの中に含まれている場合、 それぞれのプログラムは同一のユーザー定義語を使用してリソースに名前を付けることができます。

プログラムは、ユーザー定義語のタイプが異なる名前であっても他のプログラムで記述された同じ名前のリソースではなく、そのプログラム自体が記述するリソースを参照します。

COPY ライブラリーの参照

ライブラリー名-*1* が指定されていない場合、SYSLIB がライブラリー名とみなされます。

フォーマット

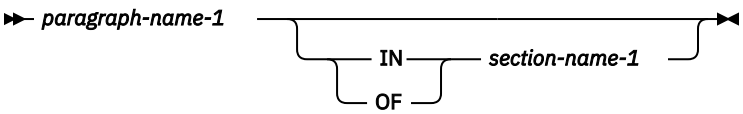


COPY ライブラリーの参照に関する規則については、476 ページの『COPY ステートメント』を参照してください。

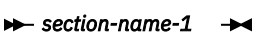
PROCEDURE DIVISION の名前の参照

プログラムで明示的に参照される PROCEDURE DIVISION の名前は、セクション内で固有でなければなりません。

フォーマット 1



フォーマット 2



セクション名は、段落名に対して使用できる最高でしかも唯一の修飾子であり、 参照される場合は固有でなければなりません。 (セクション名については、233 ページの『プロシージャー』で解説しています。)

段落名が明示的に参照される場合は、その段落名はセクションの中で重複することはできません。 段落名がセクション名によって修飾される場合、SECTION という語を含めてはなりません。 段落名は、それが属するセクション内で参照される場合は、修飾する必要はありません。 あるプログラムの中の段落名またはセクション名は、 他のどのプログラムからも参照できません。

DATA DIVISION の名前の参照

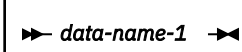
このセクションでは、以下のタイプの参照について説明します。

- 55 ページの『単純なデータ参照』
- 55 ページの『ID』

単純なデータ参照

COBOL プログラムのデータ項目を参照する最も基本的な方法は、単純なデータ参照 です。 これは、修飾、添え字付け、または参照変更を行わないデータ名-*1* のことです。 単純なデータ参照は、単一の基本項目またはグループ項目の参照に使用されます。

フォーマット



データ名-*1*
任意のデータ記述項目にすることができます。

データ名-*1* はプログラム中で固有でなければなりません。

ID

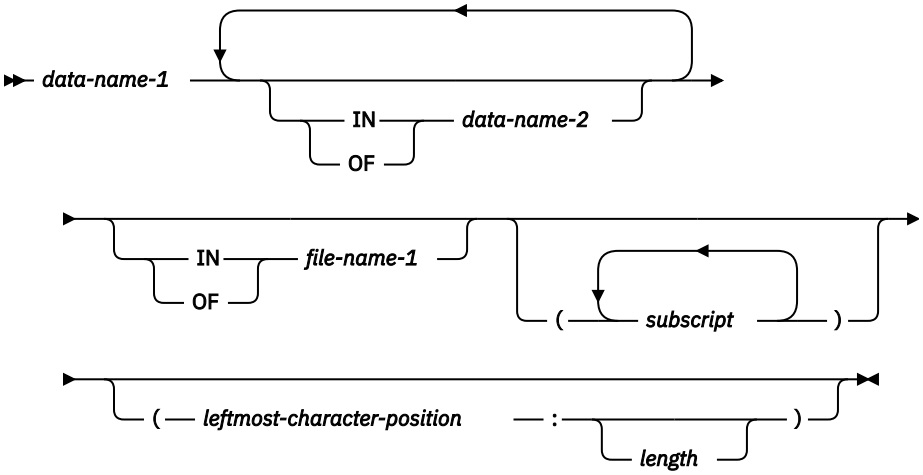
本書の構文図で使用される場合、*ID* という語は、参照の固有性の必要に応じて修飾子、添え字、および参照修飾子が付けられた、データ名または関数 ID の有効な組み合わせのことを言います。

ただし、あるフォーマットに関連する ID の規則によっては、 修飾、添え字付け、または参照変更を伴う参照を、特に禁止している場合があります。

データ名 とは、そのフォーマットの規則が特に認めている場合を除いて、 修飾、添え字付け、または参照変更をしてはならない名前を指します。

- 修飾に関する説明は、53 ページの『修飾』を参照してください。
- 添え字付けに関する説明は、59 ページの『添え字付け』を参照してください。
- 参照変更に関する説明は、62 ページの『参照変更』を参照してください。

フォーマット 1

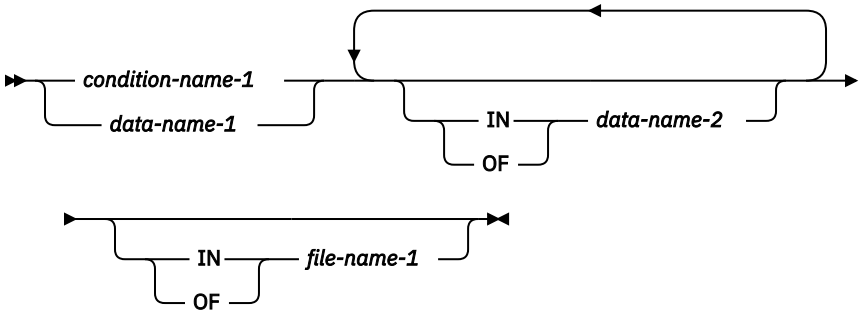


データ名-*1*、 データ名-*2*
レコード名を指定できます。

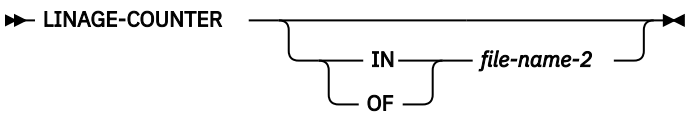
ファイル名-*1*
DATA DIVISION の項目 FD または SD と一致している必要があります。

このプログラムの中でファイル名-*1* は固有でなければなりません。

フォーマット 2



フォーマット 3



データ名-1 、 データ名-2
レコード名を指定できます。

条件名-1
構成セクションを含んでいるプログラムの中か、 またはそのプログラムに含まれるプログラムの中で、ステートメントおよび項目によって参照できます。

ファイル名-1
DATA DIVISION の項目 FD または SD と一致している必要があります。

このプログラムの中で固有でなければなりません。

LINAGE-COUNTER
LINAGE 節を含んでいるファイル記述項目がソース単位で複数指定されている場合は、それを参照するたびに修飾する必要があります。

ファイル名-2
DATA DIVISION の項目 FD または SD と一致している必要があります。 このプログラムの中でファイル名-*2* は固有でなければなりません。

修飾によってデータ名を固有なものにできない場合には、データ名が重複しないようにしてください。

ある同じプログラムの中で、レベル番号が 01 で、 EXTERNAL 節を含む項目のサブジェクトとして指定されたデータ名は、 EXTERNAL 節を含む 他のデータ記述項目に指定されたデータ名と同じにしてはなりません。

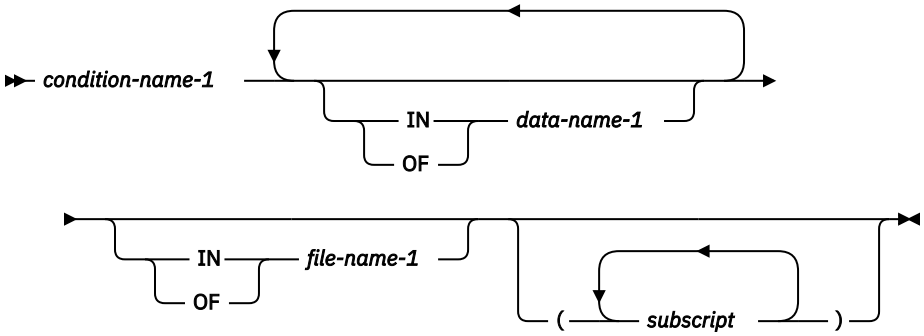
同じ DATA DIVISION の中では、同じデータ名が指定されている任意の 2 つのデータ項目に関するデータ記述項目に GLOBAL 節を含めてはなりません。

明示的に参照される DATA DIVISION の名前は、固有名に定義するか、または修飾によって固有な名前にしなければなりません。 参照されないデータ項目は、固有なものである必要はありません。 データ階層の最高レベル (レベル標識 (FILE SECTION の FD または SD)、またはレベル番号 01 に関連するデータ項目) は 参照される場合は固有な名前にする必要があります。 02 から 49 のレベル番号に関連するデータ項目は、階層内の連続するより低いレベルになります。

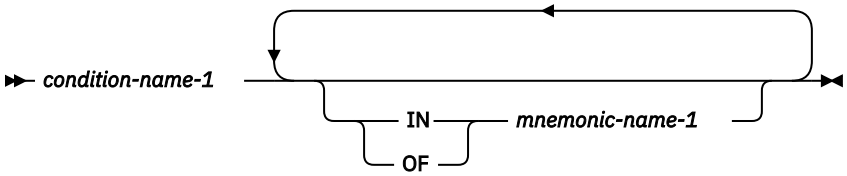
条件名

詳しくは、構文および説明を参照してください。

フォーマット 1: データ部の条件名



フォーマット 2: SPECIAL-NAMES 段落の条件名



条件名-*1*

条件名-*1* の定義を含んでいるプログラムの中か、またはそのプログラム内に含まれるプログラムの中で、 ステートメントおよび項目によって参照できます。

明示的に参照される場合は、名前の有効範囲自体が参照の固有性を確認する場合を除いて、条件名は固有にするか、または 修飾や添え字付けによって固有にしなければなりません。

条件名を固有にするために修飾を使用する場合、関連する条件変数が最初の修飾子として使用されます。 修飾が使用される場合、条件名を固有にするために、 条件変数自体に関連する名前の階層を使用しなければなりません。

条件変数の参照で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、 同じ添え字付けの組み合わせも必要です。

本書では、条件名 は必要に応じて修飾または添え字付けされる条件名を指します。

データ名-*1*

レコード名を指定できます。

ファイル名-*1*

DATA DIVISION の項目 FD または SD と一致している必要があります。

このプログラムの中でファイル名-*1* は固有でなければなりません。

簡略名-*1*

簡略名 として使用できる値の詳細については、91 ページの『SPECIAL-NAMES 段落』を参照してください。

指標名

指標名は指標を示します。 指標は、コンパイラーがテーブルでの作業に使用するために生成する専用特殊レジスターとみなされます。 指標に名前を付けるには、テーブルを定義する OCCURS 節で INDEXED BY 句を指定します。

指標名は以下の言語エレメントでのみ使用できます。

- SET ステートメント
- PERFORM ステートメント
- SEARCH ステートメント
- 添え字
- 比較条件

指標名は指標データ項目の名前とは異なります。また、指標名をデータ名と同様に使用することはできません。

指標データ項目

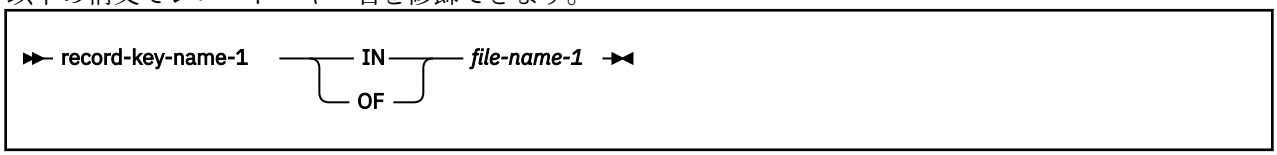
指標データ項目は、指標の値を保持できるデータ項目です。

指標データ項目を定義するには、データ記述項目で USAGE IS INDEX 節を指定します。 指標データ項目の名前はデータ名です。 指標データ項目は、特定のステートメントの規則で特に指示がない限り、データ名または ID を使用可能な場所であればどこでも使用できます。 SET ステートメントを使用して、指標データ項目に指標 (指標名によって参照される) の値を保管できます。

レコード・キー名

レコード・キー名は、索引付きファイルに関連付けられるキーに名前を付けるユーザー定義語です。

以下の構文でレコード・キー名を修飾できます。

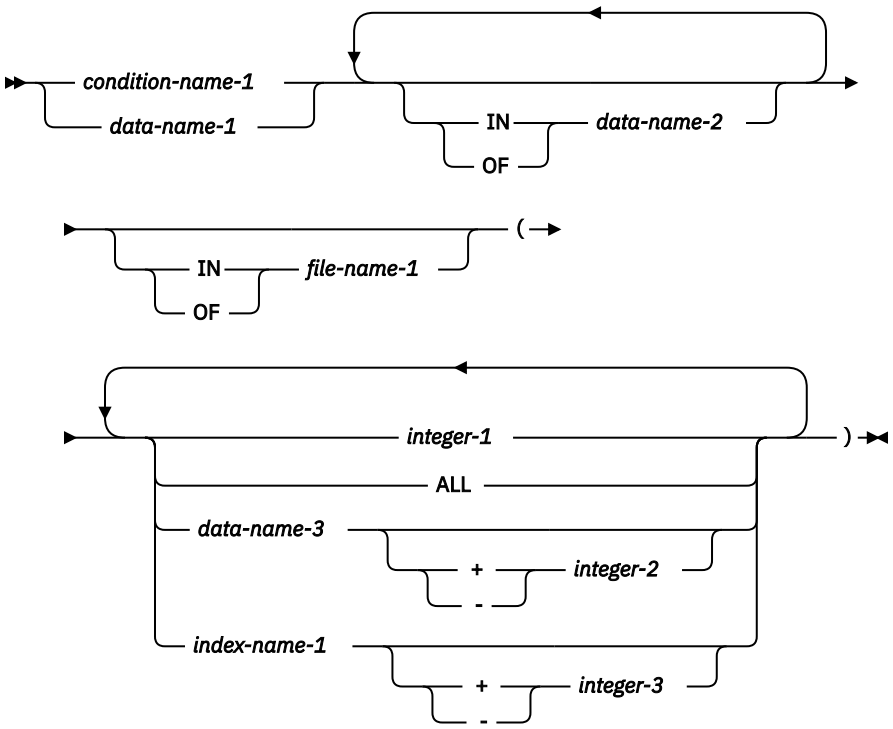


制約事項 : レコード・キー名は、STL ファイル・システムでのみサポートされます。

添え字付け

添え字付け とは、 添え字を利用してテーブル参照を行う手法のことです。 添え字 は正の整数で、その値はテーブル・エレメントのオカレンス番号を指定します。

フォーマット



条件名-*1*
条件名-*1* の条件変数は、OCCURS 節を含んでいるか、OCCURS 節を含むデータ記述項目に従属していなければなりません。

データ名-*1*
OCCURS 節を含んでいるか、OCCURS 節を含むデータ記述項目に従属していなければなりません。

データ名-*2* 、 ファイル名-*1*
データ名-*1* が含まれているデータ項目またはレコードでなければなりません。

整数-*1*
符号を付けることができます。 符号を付ける場合、その符号は正でなければなりません。

ALL
これは、*condition-name-1* が指定されている場合は指定してはなりません。

これは、添え字付きの ID が組み込み関数の引数として使用されている場合にのみ、またはフォーマット 2 SORT ステートメント (テーブル SORT ステートメント) でテーブルを識別するためにのみ使用する必要があります。

ALL が指定された場合は、ALL 添え字の許可対象となる関数の規則において指定されているように、添え字は関連テーブルの添え字のすべての使用可能な値です。

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* は修飾することができます。 データ名-*3* をウィンドウ表示日付フィールドにすることはできません。

指標名-*1*
参照されるテーブルの階層内にあって、 その名前を指定している INDEXED BY 句が含まれているデータ記述項目に対応します。

整数-*2*、整数-*3*
符号を付けることはできません。

添え字は括弧に囲んで、テーブル・エレメントの名前の修飾のすぐ後に続けて記述します。 このような参照における添え字の個数は、参照されるエレメントを含むテーブルの次元数と同じでなければなりません。つまり、該当のデータ名自体も含めて、 そのデータ名を含む階層の各 OCCURS 節ごとに対応する添え字がなければなりません。

複数の添え字が必要な場合には、データ編成の次元が低い順から指定します。 多次元テーブルが一連のネストされたテーブルとしてみなされ、 ネストの中で最も包括的または最外部のテーブルがメジャー・テーブルで、 最も内側または最も包括的でないテーブルがマイナー・テーブルとみなされる場合は、 添え字は左から右へ、メジャー、中間、マイナーの順に指定されます。

例えば、TABLE-THREE が次のように定義されているとします。

```
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 の有効な添え字付き参照は次のようになります。

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

添え字付き参照も参照変更することができます。 63 ページの『参照変更の例』の 3 番目の例を参照してください。 . ある項目に対する参照には、その項目がテーブル・エレメント、あるいはテーブル・エレメントに関連付けられた項目または条件名でない限り、添え字を付けることはできません。

各テーブル・エレメント参照は、次のような参照がある場合を除いて、 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 節の中
- OCCURS 節の KEY IS 句の中
- フォーマット 2 SORT ステートメント (テーブル SORT ステートメント) の中
- LIKE 節の中

フォーマット 2 SORT ステートメントでは、右端の添え字をワード ALL にして添え字付けを指定できます。

添え字で表すことが可能な最小のオカレンス番号は 1 です。個々の場合に許される最大のオ カレンス番号は、OCCURS 節で指定されている項目の最大オカレンス項目数です。

データ名を使用した添え字付け

データ名を使用して添え字を表す場合、 そのデータ名は別のテーブルの中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。 同じデータ名は、1 つの項目を持つ 1 つだけの添え字として、 および別の項目を持つ 2 つ以上の添え字の 1 つとして指定することができます。 データ名の添え字は修飾できます。しかし、添え字や指標を付けることはできません。 例えば、TABLE-THREE に対する有効な添え字付き参照 (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) には以下が含まれます。

```
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)

指標付けを行うことによって、テーブルの検索や特定の項目の処理などの操作ができるようになります。指標付けを使用するには、1 つ以上の指標名をデータ記述項目に OCCURS 節を含む項目と関連付けます。

指標名に関連付けられる指標は添え字の働きをし、 その値は指標名が関連付けられている項目のオカレンス番号に対応しています。

INDEXED BY 句は、指標名を識別し、特定のテーブルに関連付ける場合に使用されるものですが、OCCURS 節のオプショナル部分です。 指標名に関連付けられる指標を記述するための別個の項目はありません。実行時には、指標の内容が、 それが関連付けられるテーブルの特定の次元のオカレンス番号に対応します。

実行時の指標の初期値は不定であり、 添え字として使用する前に初期設定しなければなりません。 指標の初期値の割り当ては、次のステートメントのいずれかによって行います。

- 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

整数またはデータ名を、 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使用しても、 そのテーブルに関連付けられた指標が変更されることはありません。

テーブルを参照するために指標名を使用することができます。 しかし、 参照されているテーブルと索引名が関連付けられているテーブルの各エレメントの長さがそれぞれ一致している必要があります。 一致していない場合は、 参照はそれぞれのテーブルの同じテーブル・エレメントに対するものにならず、 実行時エラーが発生する可能性があります。

テーブル形式で配列されているデータは、頻繁に検索されることになります。 SEARCH ステートメントは、逐次検索や非逐次検索が行える機能を備えています。 このステートメントは、テーブルを検索し、特定の条件を満たすテーブル・エレメントをテーブルから検索し、関連付けられた指標の値を そのテーブル・エレメントを指し示すように調整する場合に使います。

実行中に有効であるためには、指標の値は 1 以上、かつ最大許容オカレンス番号以下のテーブル・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。

指標名の詳細については、57 ページの『指標名』と 176 ページの『INDEXED BY 句』を参照してください。

相対添え字付け

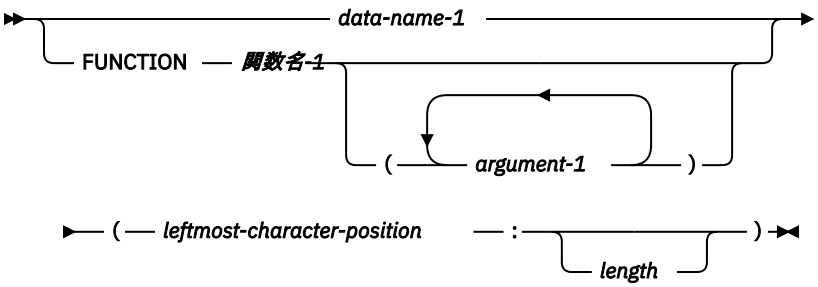
相対添え字付け では、テーブル・エレメント名の後に添え字が付けられます。添え字は、データ名または指標名の後に + または - が付き、正の整数リテラルまたは符号なし整数リテラルが続く形式です。

演算子の + と - の前後にはスペースが必要です。 使用される添え字の値は、 指標名またはデータ名が整数の値によって上下に設定されているかのように、 それらの値は同じになります。 相対指標付けを使用しても、 プログラムが指標の値を変更することはありません。

参照変更

参照変更 は、データ項目の左端の文字位置 (開始桁) とそのデータ項目の長さ (オプション) を指定することによってデータ項目を定義するものです。

フォーマット: 参照変更



データ名-*1*

USAGE DISPLAY、DISPLAY-1、または NATIONAL により明示的または暗黙的に記述されているデータ項目を参照する必要があります。 国別グループ項目は、国別カテゴリーの基本データ項目として処理されます。

データ名-*1* は修飾したり添え字を付けたりすることができます。 データ名-*1* をウィンドウ表示日付フィールドにすることはできません。

データ名-*1* は、ブール・データ項目を参照してはなりません。

データ名-*1* は、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。

左端の文字の開始位置

算術式でなければなりません。 左端の文字の開始位置 の評価結果は、データ名-*1* によって参照されるデータ項目の桁数以下の、ゼロ以外の正の整数でなければなりません。

左端の文字の開始位置 の評価結果が、ウィンドウ表示日付フィールドであってはなりません。

長さ

算術式でなければなりません。

長さ の評価結果は、ゼロ以外の正の整数でなければなりません。

長さ の評価結果が、ウィンドウ表示日付フィールドであってはなりません。

左端の文字の開始位置 と長さ の合計から 1 を引いた値は、データ名-*1* の桁数以下でなければなりません。 長さ を省略した場合、使用する長さは、データ名-*1* の文字位置に 1 を足した値から左端の文字の開始位置 を引いた値に等しくなります。

関数名-*1*

英数字または国別関数を参照しなければなりません。

USAGE DISPLAY-1 および NATIONAL の場合、それぞれの文字位置は各文字の 2 バイトを占有します。 参照変更は文字位置全体に対して機能するのであり、USAGE DISPLAY-1 および NATIONAL の文字の個々のバイトに対して機能するのではありません。 USAGE DISPLAY の場合、参照変更はそれぞれの文字が 1 バイト文字であるものとして機能します。

特に断りがない限り、参照変更は、参照変更データ項目と同じ USAGE のデータ項目または関 数を参照する ID または関数 ID が使用できる個所であればどこでも使用できます。

データ名-*1* または 関数名-*1* によって参照されるそれぞれの文字位置には、 左端の位置から右端の位置にかけて 1 ずつ大きくなる序数が割り当てられます。左端の位置には序数として 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 節がある場合は、 符号の位置にもそのデータ項目における序数が割り当てられます。

データ名-*1* が USAGE DISPLAY で記述され、数字、数字編集、英字、英数字編集、または外部浮動小数のカテゴリーに属する場合、データ名-*1* は、データ名-*1* で参照されるデータ項目と同じサイズの英数字カテゴリーのデータ項目として再定義されたものとして、参照変更のために処理されます。

データ名-*1* が USAGE NATIONAL で記述され、数字、数字編集、国別編集、または外部浮動小数点のカテゴリーに属する場合、データ名-*1* は、データ名-*1* で参照されるデータ項目と同じサイズの国別カテゴリーのデータ項目として再定義されたものとして、参照変更の対象となります。

データ名-*1* が国別グループ項目の場合、データ名-*1* は国別カテゴリーの基本データ項目として処理されます。

データ名-*1* が拡張日付フィールドである場合、参照変更の結果は非日付データとなります。

参照変更は、データ名-*1* のサブセットまたは関数名-*1* とその引数 (引数がある場合) によって参照されるデータ項目のサブセットである固有のデータ項目を作成します。 この固有のデータ項目は、JUSTIFIED 節を伴わない基本データ項目とみなされます。

関数が参照変更の場合、この固有のデータ項目は、クラスおよびカテゴリーを持ち、関数の タイプが国別の場合は、USAGE NATIONAL になります。国別の関数でない場合は、英数字 のクラスおよびカテゴリーに属し、USAGE DISPLAY になります。

データ名-*1* が参照変更される場合、以下の場合を除き、この固有のデータ項目は、データ名-*1* で参照されるデータ項目に定義されたものと同じクラス、カテゴリー、および USAGE になります。

- データ名-*1* が国別編集のカテゴリーである場合、この固有のデータ項目は国別カテゴリーになります。
- データ名-*1* が USAGE NATIONAL で、カテゴリーが数字編集、数字、または外部浮動小数点の場合は、この固有のデータ項目は国別カテゴリーになります。
- データ名-*1* が USAGE DISPLAY で、カテゴリーが数字編集、英数字編集、数字、または外部浮動小数点の場合は、この固有のデータ項目は英数字カテゴリーになります。
- データ名-*1* が英数字グループ項目を参照する場合、この固有のデータ項目は USAGE DISPLAY および英数字カテゴリーであるとみなされます。
- データ名-*1* が国別グループ項目を参照する場合、この固有のデータ項目は USAGE NATIONAL および国別カテゴリーであるとみなされます。

長さ の指定がない場合、得られる固有のデータ項目の位置は、左端の文字位置 によって指定される文字位置 (その文字位置を含む) から、データ名-*1* によって参照されたデータ項目の右端の文字位置 (その文字位置を含む) までになります。

オペランドの評価

オペランドに対する参照変更は、次のように評価されます。

- そのオペランドに添え字付けの指定があれば、 参照変更は、添え字の評価の直後に評価されます。
- そのオペランドに添え字付けの指定がなければ、参照変更は、 添え字の指定がある場合に添え字付けが評価されるのと同じ時点で評価されます。

参照変更の例

例にあるステートメントは、WHOLE-NAME によって参照されるデータ項目の最初の 10 文字を、FIRST-NAME によって参照されるデータ項目に転送します。

```
77  WHOLE-NAME  PIC X(25).
77  FIRST-NAME  PIC X(10).

77  START-P     PIC 9(4) BINARY VALUE 1.
77  STR-LENGTH  PIC 9(4) BINARY VALUE 10.

...
    MOVE WHOLE-NAME(1:10) TO FIRST-NAME.
    MOVE WHOLE-NAME(START-P:STR-LENGTH) TO FIRST-NAME.
```

次の例では、 WHOLE-NAME によって参照されたデータ項目の最後の 15 文字を、 LAST-NAME によって参照されたデータ項目に転送します。

```
77  WHOLE-NAME  PIC X(25).
77  LAST-NAME   PIC X(15).
...
    MOVE WHOLE-NAME(11:) TO LAST-NAME.
```

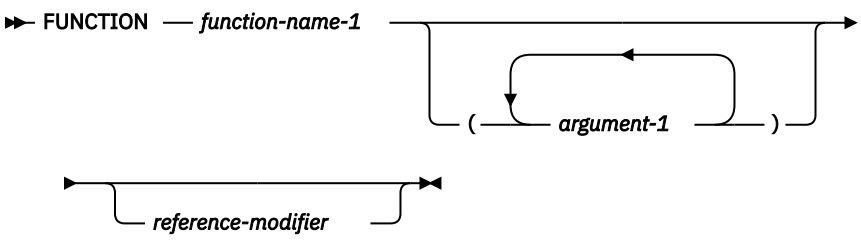
次の例では、 TAB の 3 番目のオカレンスの 4 番目と 5 番目の文字を、 SUFFIX という変数に転送します。

```
01  TABLE-1.
    02  TAB  OCCURS 10 TIMES  PICTURE X(5).
77  SUFFIX                   PICTURE X(2).
...
    MOVE TAB OF TABLE-1 (3) (4:2) TO SUFFIX.
```

関数 ID

関数 *ID* とは、関数の評価からの結果であるデータ項目を一意的に参照する、文字ストリングおよび分離文字のシーケンスです。

フォーマット



引数-*1*
引数-1 は、ID、リテラル (形象定数を除く)、または算術式でなければなりません。

詳しくは、421 ページの『第 20 章 組み込み関数』を参照してください。.

関数名-*1*
関数名-*1* は、組み込み関数の名前のうちの 1 つでなければなりません。

参照修飾子
タイプが英数字または国別の関数についてのみ指定できます。

英数字または国別の関数を参照する関数 ID はそれぞれ、英数字カテゴリーまたは国別カテゴリーのデータ項目が参照可能であって、関数への参照が特に禁止されていないところであれば、どこでも指定することができます。ただし、以下のような例外があります。

- 任意のステートメントの受け入れ側のオペランドとする場合。
- データ項目が特別の特性 (クラスとカテゴリー、サイズ、 符号および暗黙的値など) を持つように要求され、 その定義と指定された特定の引数に応じた関数の評価がこれらの特性を持たないようなところ。

整数関数または数字関数を参照する関数 ID は、算術式を使用できるところであればどこでも使用できます。

日時項目の参照

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側のオペランドとする場合。

- 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。

ブール項目の参照

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側のオペランドとする場合。
- 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。

ユーザー定義データ・タイプ

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 節
- GLOBAL 節
- OCCURS 節
- TYPEDEF 文節
- VALUE 文節

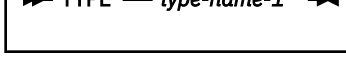
タイプ名の有効範囲に関する規則は、データ名の場合の規則と同じです。

TYPE 文節と TYPEDEF 文節の詳細については 210 ページの『TYPE 節』および 211 ページの『TYPEDEF 文節』を参照してください。

TYPE 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。

フォーマット

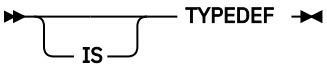


type-name-1
対象とするデータ名を定義するために使用するタイプ名

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。

フォーマット



データ属性の指定

明示的なデータ属性 とは、COBOL コーディングに指定したデータ属性です。 暗黙のデータ属性 とは、デフォルト値のことです。 プログラマーがデータ属性を明示的に指定しない場合には、 コンパイラーがデフォルト値を想定します。

たとえば、データ項目の USAGE は指定する必要はありません。 USAGE を省略し、PICTURE 節に記号 N を指定しない場合、デフォルトは USAGE DISPLAY で、暗黙のデータ属性になります。 PICTURE 記号 N が使用され、NSYMBOL(DBCS) コンパイラー・オプションが有効であるときは、USAGE DISPLAY-1 がデフォルトです。NSYMBOL(NATIONAL) コンパイラー・オプションが有効であるときは、USAGE NATIONAL がデフォルトです。 これは、暗黙のデータ属性です。

第 9 章 制御の移動

PROCEDURE DIVISION では、制御が明示的に 移される場合や次の実行可能ステートメントがない場合を除けば、 プログラムの流れは、ステートメントが書かれている順序に従って、 あるステートメントから次のステートメントへと制御が移ります。 このような通常のプログラムの流れを、暗黙の 制御の移動と呼びます。

暗黙に行われる制御の移動は、 あるステートメントから次のステートメントへという場合の他に、 プロシージャーのブランチ・ステートメントを実行せずに、 通常のプログラムの流れが変更される場合にも生じることがあります。 次に示す例では、暗黙の 制御の移動で、ステートメントからステートメントへの制御の移動が変更されます。

- 別の COBOL ステートメントの制御のもとで実行されているプロシージャーの最後のステートメントの実行が終わると、 制御は暗黙のうちに移されます。 (プロシージャーの実行を制御する COBOL ステートメントは、例として、MERGE、PERFORM、SORT、および USE です。) さらに、 繰り返し実行を起こさせる PERFORM ステートメントの制御の下である段落が実行される場合で、 しかもその段落がその PERFORM ステートメントの範囲内の最初の段落である場合には、 その段落が繰り返し実行されるたびに、その PERFORM ステートメントに関連する制御メカニズムとその段落の最初のステートメントとの間で暗黙の制御の移動が行われます。
- SORT ステートメントまたは MERGE ステートメントを実行する間は、入力または出力プロシージャーに制御が暗黙のうちに移されます。
- XML PARSE ステートメントの実行中は、制御が暗黙的に処理プロシージャーに移動します。
- 宣言型プロシージャーの実行を起こさせる COBOL ステートメントのいずれかを実行する間は、その宣言型プロシージャーに制御が暗黙のうちに移されます。
- いずれかの宣言型プロシージャーの実行が終わると、 その宣言型プロシージャーの実行を引き起こしたステートメントに関連する制御メカニズムに制御が暗黙のうちに戻されます。

COBOL では、プロシージャー・ブランチ・ステートメント、プログラムの呼び出し、 あるいは条件ステートメントを実行することにより、 制御を 明示的に 移すこともできます。 (260 ページの『ステートメントのカテゴリー』に、 プロシージャー・ブランチ・ステートメントおよび条件ステートメントのリストがあります。)

定義: 次の実行可能ステートメント という言葉は、 上述の規則に従って制御が移されるという点で次の 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。

- そのプログラムに PROCEDURE DIVISION がない場合。
- 宣言セクションの最後のステートメントの後であり、 そのステートメントの含まれている段落が、 他のいずれの COBOL ステートメントの制御の下でも実行されない場合。
- プログラムの最後のステートメントの後であり、 そのステートメントの含まれている段落が、 他のいずれの COBOL ステートメントの制御の下でも実行されない場合。
- 別のセクションで実行されるアクティブな PERFORM ステートメントの 範囲内にある、宣言セクションの ステートメントの後であり、宣言セクションの最後のステートメントが、 アクティブな 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。
- COBOL プログラムの外に制御を移す STOP RUN ステートメント または EXIT PROGRAM ステートメントの後である場合。
- COBOL プログラムの外に制御を移す GOBACK ステートメントの後である場合。
- END PROGRAM マーカー。

次の実行可能ステートメントがなく、 しかも制御がその COBOL プログラムの外に移されないときは、 プログラムの実行が CALL ステートメントの制御の下にあるプログラムの非宣言型プロシージャー部分で行われており、 暗黙の EXIT PROGRAM ステートメントが実行される場合を除いて、 プログラムの制御のフローは予測できません。

第 10 章 2000 年言語拡張および日付フィールド

2000 年言語拡張は、2000 年問題 (一部のアプリケーションの日付フィールドで年を表すために 2 桁を使用することで発生) を解決し、日付フィールドで最も一般的な操作をサポートするために設計されたものです。

多くのアプリケーションでは、日付フィールドで年号を表すのに 4 桁ではなく 2 桁を使用しており、これらの値は 1900 から 1999 の年を表すと想定しています。 この短縮した日付フォーマットは 1900 年代の間は正しく機能しますが、2000 年を超えると機能しません。これらのアプリケーションでは「00」を 2000 ではなく 1900 と解釈するため、間違った結果が生成されるためです。

2000 年言語拡張は、既存のコードに最小の変更を加えるだけで、2 桁の年号を使用するアプリケーションが 2000 年を超えても正しく実行し続けることができるように設計されています。 これは、2 桁の年号フィールドがすべて 1900 から 1999 の年を表すという想定を取り除く、ウィンドウ表示 と呼ばれる手法を使用して行われます。 ウィンドウ表示では 2 桁の年号フィールドを使用して、任意の 100 年範囲内の年号を表すことができます。これを世紀ウィンドウ と呼びます。

例えば、2 桁の年号フィールドに値 15 が含まれている場合、 ほとんどのアプリケーションはそれを 1915 と解釈します。 しかし、1960 から 2059 の世紀ウィンドウを使用すれば、その年は 2015 と解釈されます。

2000 年言語拡張は、日付フィールドに対する一般的なほとんどの操作 (比較、 移動および保管、増加および減少) をサポートします。 このサポートは、特定のフォーマットの日付フィールドに限定されます。詳細については、162 ページの『DATE FORMAT 節』を参照してください。

日付フィールドを使用するときにサポートされる操作および制約事項については、163 ページの『日付フィールドの使用に関する制約事項』を参照してください。

2000 年言語拡張の構文

2000 年言語拡張は、DATE FORMAT 節の言語エレメントといくつかの組み込み関数を導入します。

- データ記述項目における DATE FORMAT 節。これはデータ項目を日付フィールドとして定義します。
- 以下の組み込み関数。

  DATEVAL
  非日付データを日付フィールドに変換します。

  UNDATE
  日付フィールドを非日付データに変換します。

  YEARWINDOW
  YEARWINDOW コンパイラー・オプションによって指定された世紀ウィンドウの最初の年を戻します。

アプリケーションでの 2000 年言語拡張の使用について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」内の 『*2000* 年言語拡張 *(MLE)*』 を 参照してください。

ご使用のプログラムが DATEPROC コンパイラー・オプションを使用し、YEARWINDOW コンパイラー・オプションで世紀ウィンドウを指定してコンパイルされていない限り、2000 年言語拡張は効力を持ちません。

用語と概念

本書で 2000 年言語拡張を取り上げている場合は、以下の用語を確認してください。

- 70 ページの『日付フィールド』
- 71 ページの『非日付データ』
- 71 ページの『世紀ウィンドウ』

日付フィールド

日付フィールド は、いくつかの項目のいずれかにすることができます。

使用できる日付フィールドは以下のとおりです。

- データ記述項目が DATE FORMAT 節を含むデータ項目。
- 次の組み込み関数の 1 つで戻される値。
  - DATE-OF-INTEGER
  - DATE-TO-YYYYMMDD
  - DATEVAL
  - DAY-OF-INTEGER
  - DAY-TO-YYYYDDD
  - YEAR-TO-YYYY
  - YEARWINDOW
- ACCEPT ステートメントの概念上のデータ項目である DATE、DATE YYYYMMDD、DAY、または DAY YYYYDDD。
- 特定の算術演算の結果 (詳細については、236 ページの『日付フィールドを使用する算術計算』を参照)。

日付フィールド という用語は、拡張日付フィールド およびウィンドウ表示日付フィールド の両方を指します。

ウィンドウ表示日付フィールド

ウィンドウ表示日付フィールド とは、ウィンドウ表示西暦年を含む日付フィールドです。 ウィンドウ表示西暦年 は、世紀ウィンドウ内の年を表す 2 桁から構成されます。

拡張日付フィールド

拡張日付フィールド とは、拡張西暦年を含む日付フィールドです。 拡張西暦年 は 4 桁から構成されます。

拡張日付フィールドの主な用途は、ウィンドウ表示日付フィールドと組み合わせて使用し たときに正しい結果をもたらすことです。例えば、4 桁年号の日付への移行が不完全な場合などです。 アプリケーションの中のすべての日付が 4 桁の年号を使用する場合には、2000 年言語拡張を使用する必要はありません。

年末尾型日付フィールド

年末尾型日付フィールド とは、DATE FORMAT 節において YY または YYYY の前に 1 つ以上の X が 指定されている日付フィールドのことです。 年末尾型日付フィールドがサポートされるのは、 同じデータ (年末尾型) 日付フォーマットの別の日付が関係している場合や、非日付データが関係している場合など、一部の操作においてです。

日付形式

日付フォーマット とは日付フィールドの日付パターンであり、 次のいずれかの方法で指定されます。

- DATE FORMAT 節または DATEVAL 組み込み関数 引数-*2* によって明示的に指定される。
- 日付フィールドを戻すステートメントおよび組み込み関数によって暗黙的に指定される。

互換日付フィールド

互換 という用語の意味は、日付フィールドに適用される場合、その日付フィールドが COBOL のどの部で使用されるのかによって異なります。

データ部

2 つの日付フィールドの USAGE が同じであり、以下の少なくとも 1 つの条件を満たす場合、その 2 つの日付フィールドには互換性がある。

- 日付フォーマットが同じである。
- 両方ともウィンドウ表示日付フィールドである (一方がウィンドウ表示西暦年であ る DATE FORMAT YY だけで構成されている)。
- 両方とも拡張日付フィールドである (一方が拡張西暦年である DATE FORMAT YYYY だけで構成されている)。
- 一方が DATE FORMAT YYXXXX で、他方が YYXX である。
- 一方が DATE FORMAT YYYYXXXX で、他方が YYYYXX である。

ウィンドウ表示日付フィールドを拡張日付グループ・データ項目の従属とすることもできる。2 つの日付フィールドに互換性があるのは、従属のほうの日付フィ ールドに USAGE DISPLAY が指定されており、グループ拡張日付フィールドの 開始より 2 バイト後で開始していて、かつ 2 つのフィールドが以下の条件の少な くとも 1 つを満たしている場合です。

- 従属日付フィールドに指定されている DATE FORMAT パターンの中の X の数が、グループ日付フィールドの DATE FORMAT パターンと同じであ る。
- 従属日付フィールドに DATE FORMAT YY が指定されている。
- グループ日付フィールドに DATE FORMAT YYYYXXXX が指定されてお り、従属日付フィールドに DATE FORMAT YYXX が指定されている。

手続き部

2 つの日付フィールドの日付フォーマットが、年部分を除いて同じ場合、それらには互換性がある。これらは、ウィンドウ表示または拡張が可能。例えば、DATE FORMAT YYXXX のウィンドウ表示日付フィールド は、以下と互換性がある。

- DATE FORMAT YYXXX の別のウィンドウ表示日付フィールド
- DATE FORMAT YYYYXXX の拡張日付フィールド

非日付データ

非日付 は、いくつかの項目のいずれかにすることができます。

使用できる非日付は以下のとおりです。

- 日付記述項目が DATE FORMAT 節を含んでいないデータ項目
- UNDATE 関数を使用して変換された日付フィールド
- リテラル
- 参照変更された日付フィールド
- 日付フィールド・オペランドを含む特定の算術演算の結果。例えば、2 つの互換日付 フィールドの差

世紀ウィンドウ

世紀ウィンドウ とは、2 桁年号が固有に決まる 100 年間のことです。

COBOL プログラムで使用できる世紀ウィンドウには、次のものがあります。

- ウィンドウ表示日付フィールドの場合、世紀ウィンドウは YEARWINDOW コンパイラー・オプションで指定されます。
- ウィンドウ表示組み込み関数 DATE-TO-YYYYMMDD、DAY-TO-YYYYDDD、および YEAR-TO-YYYY の場合、世紀ウィンドウは引数-*2* によって指定されます。

第 2 部 COBOL ソース単位の構造

第 11 章 COBOL プログラムの構造

COBOL ソース・プログラムは、構文上、正確な COBOL ステートメントの集合です。

ネストされたプログラム
ネストされたプログラム は、別のプログラムに含まれるプログラムです。 中に含まれるこれらのプログラムは、中に含むプログラムのリソースの一部を参照することができます。 プログラム B が プログラム A に含まれているとします。同じくプログラム B を含むプログラムがプログラム A に他に存在しなければ、 プログラム B は直接的に プログラム A に含まれています。あるプログラムがプログラム A に含まれており、そのプログラムがプログラム B を含む場合は、 プログラム B は間接的に プログラム A に含まれていることになります。 ネストされたプログラムについて詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」内の 『ネストされたプログラム』 を 参照してください。

オブジェクト・プログラム
オブジェクト・プログラム は、実行可能な機械語命令と、 問題解決のために提供されるデータと対話するために設計された他のエンティティーからなる集合またはグループです。 オブジェクト・プログラムは、一般にソース・プログラムに対して COBOL コンパイルを操作した結果生じたマシン言語です。

実行単位
実行単位 は、相互に作用し合い、実行時に問題解決を提供するエンティティーとして機能する 1 つ以上のオブジェクト・プログラムです。

兄弟プログラム
兄弟プログラム は、同じプログラムに直接的に含まれるプログラムです。

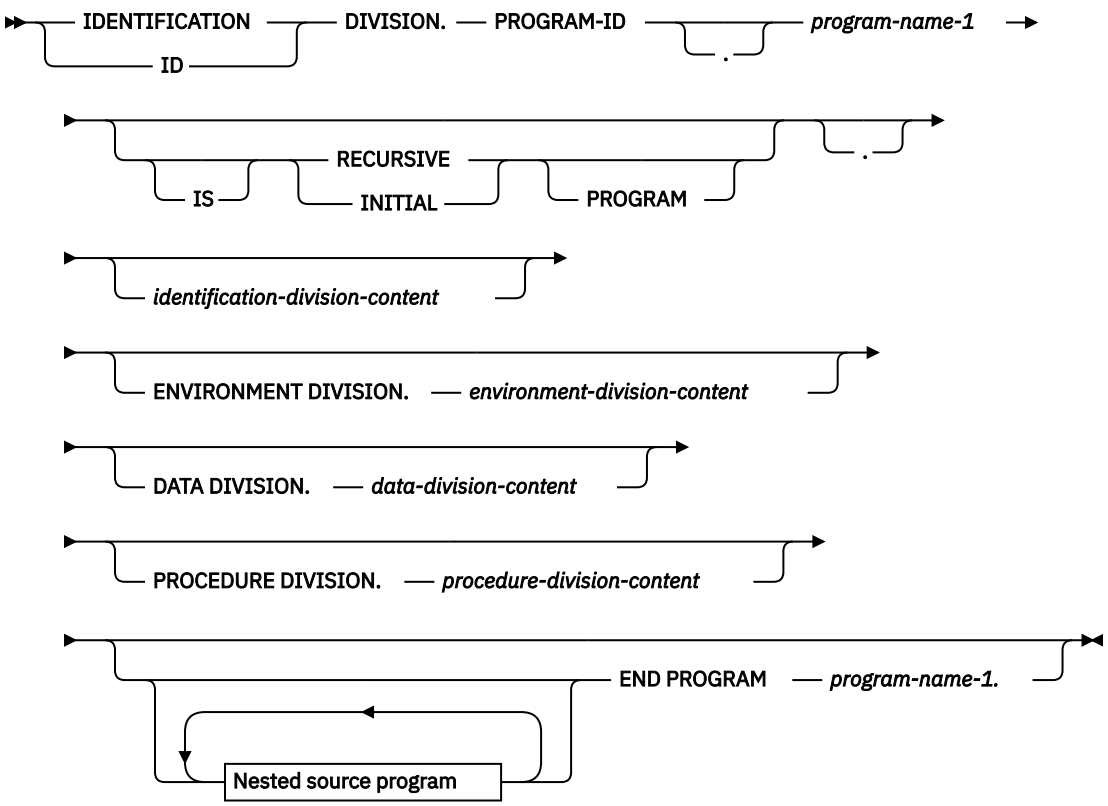
COPY および REPLACE ステートメントと END PROGRAM マーカーを除き、 COBOL ソース・プログラムのステートメント、項目、段落、およびセクションは、 以下の 4 つの部に分類されます。

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

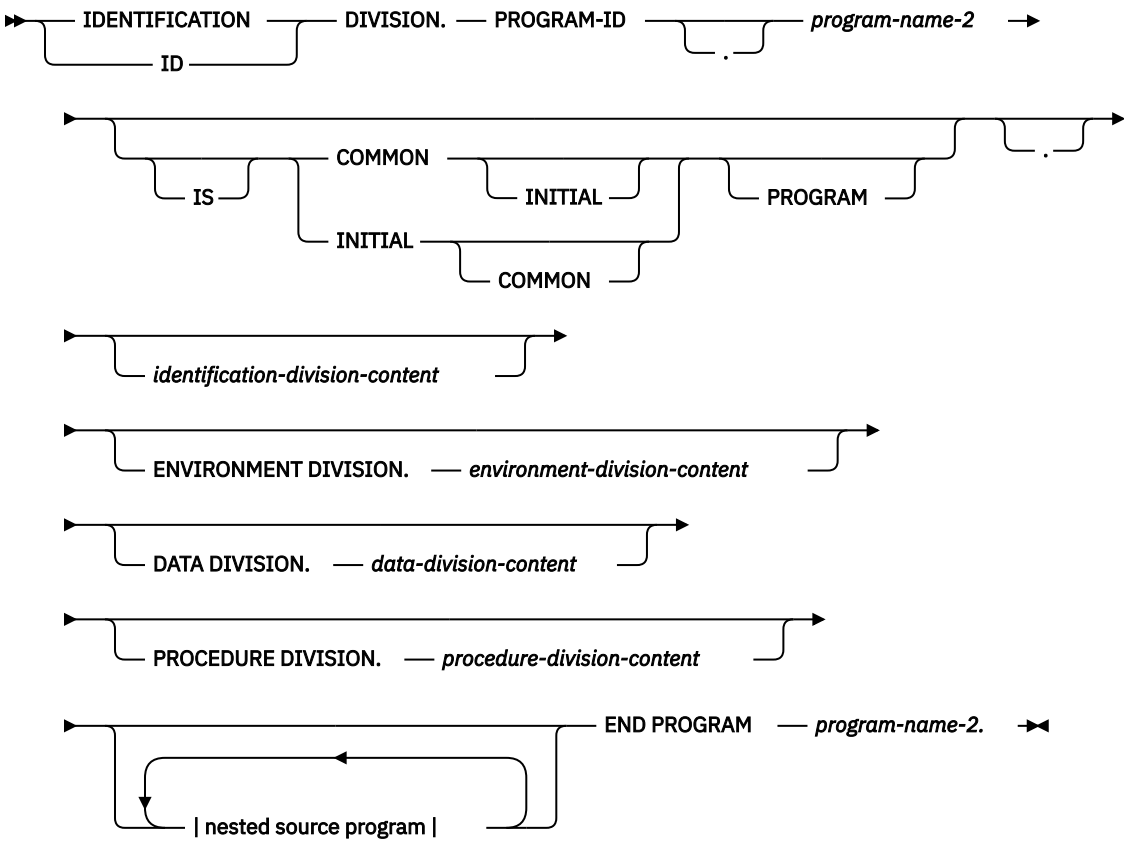
COBOL ソース・プログラムの終了は、END PROGRAM マーカーによって示されます。 ネストされたプログラムがない場合には、 ソース・プログラム行の終わりによっても COBOL プログラムの終了になります。

独立してコンパイルされる COBOL ソース・プログラムを構成する項目とステートメントのフォーマットは次のとおりです。

フォーマット: **COBOL** ソース・プログラム

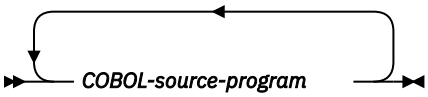


ネストされたソース・プログラム



一連の別々の COBOL プログラムもコンパイラーへの入力として使用できます。 バッチ・コンパイルの場合の一連のソース・プログラムを構成する項目とステートメントのフォーマットは次のとおりです。

フォーマット: 一連の COBOL ソース・プログラム



END PROGRAM プログラム名

END PROGRAM マーカーは、一連のプログラムの 1 つ 1 つを区切ります。 プログラム名 は、先行する PROGRAM-ID 段落で宣言したプログラム名と一致する必要がありま す。

プログラム名 は、ユーザー定義語として、または英数字リテラルで、 指定することができます。 いずれにしても、プログラム名 は、プログラム名の形成規則に従う必要があり ます。プログラム名 は、形象定数にすることはできません。 リテラルに英小文字が含まれていれば、それは大文字に変換されます。

一連のプログラムの最後のプログラムでは、そのプログラムがネストされたソース・プログラムを何も含まない場合に限り、END PROGRAM マーカーの指定は任意です。

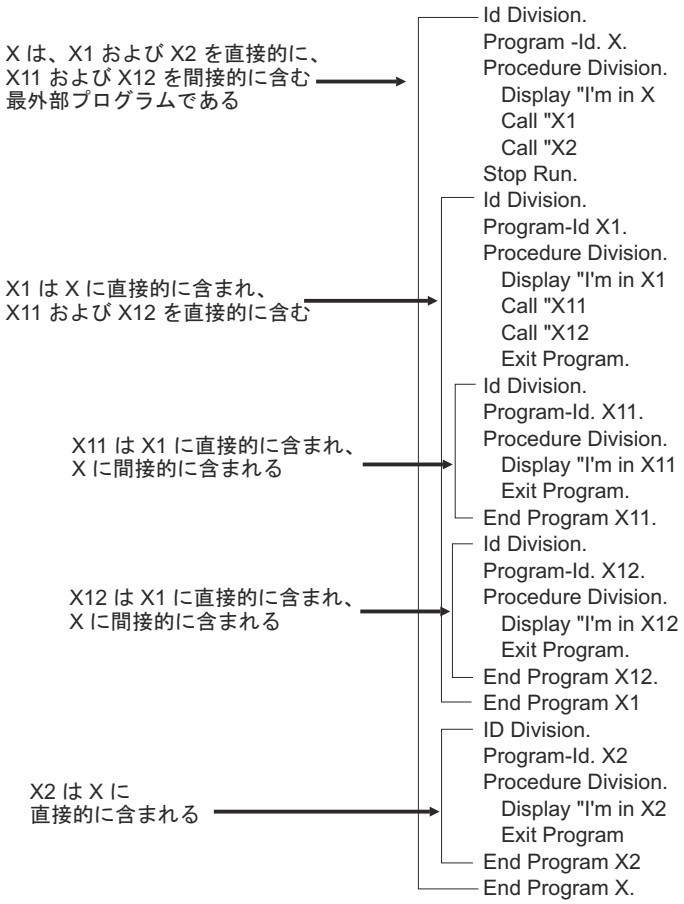
ネストされたプログラム

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、 さらにその含まれたプログラムの中に別のプログラムを含めることができます。 これらの中に含まれたプログラムは、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、 それを含むプログラムの中に直接的に または間接的に 含めることができます。

以下のコード・フラグメントでは、プログラム Outer-program は直接的に プログラム Inner-1 を含みます。 プログラム Inner-1 は直接的 にプログラム Inner-1a を含み、 Outer-program は間接的 に Inner-1a を含みます。

```
Id division.
Program-id. Outer-program.
  Procedure division.
    Call "Inner-1".
    Stop run.
Id division.
  Program-id. Inner-1
  ...
    Call Inner-1a.
    Stop run.
  Id division.
  Program-id. Inner-1a.
  ...
  End Inner-1a.
  End Inner-1.
End Outer-program.
```

以下の図は、直接または間接に含まれたプログラムのある、より複雑なネストされたプログラム構造を示しています。



プログラム名の命名規約

プログラム名は、プログラムの IDENTIFICATION DIVISION の PROGRAM-ID 段落で指定されます。 プログラム名が参照されるのは、CALL ステートメント、CANCEL ステートメント、 SET ステートメント、または END PROGRAM マーカーに限られます。

実行単位を構成するプログラムの名前は必ずしも固有とは限りませんが、 同じ実行単位内の 2 つのプログラムが同じ名前の場合は、 それらのうちの少なくとも 1 つが、それら 2 つのプログラムを含まない、 別々にコンパイルされる他のプログラムに直接的または間接的に含まれている必要があります。

独立してコンパイルされるプログラムと、 その中に直接的および間接的に含まれるプログラムすべては、その独立してコンパイルされるプログラム内で、 固有のプログラム名を持っている必要があります。

プログラム名の規則

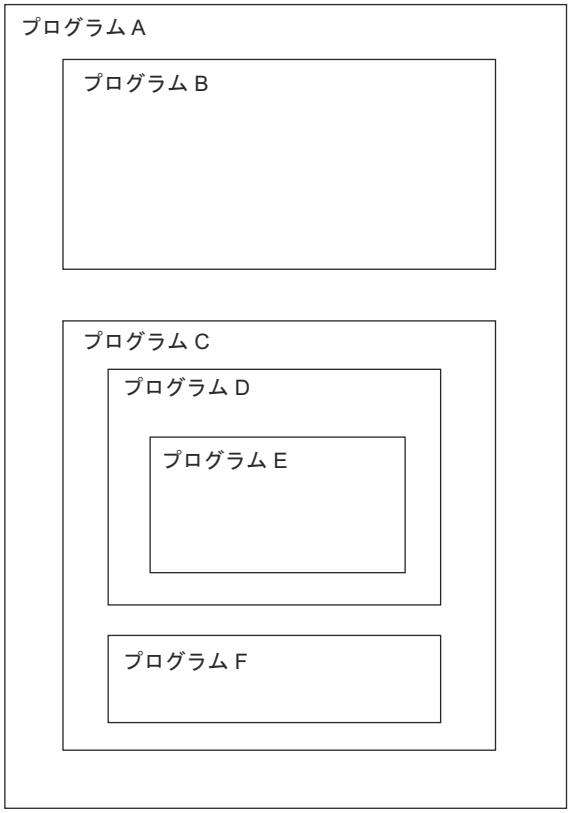
プログラム名のスコープは、次に示す規則によって定義されます。

- プログラム名が COMMON 属性を持たないプログラムのプログラム名であり、 そのプログラムが別のプログラム内に直接的に含まれている場合、 そのプログラム名は、 プログラムを含んでいる側のプログラムに記述されているステートメントによってのみ参照できます。
- プログラム名が COMMON 属性を持つプログラムのプログラム名であり、 そのプログラムが別のプログラム内に直接的に含まれている場合、 そのプログラム名は、プログラムを含んでいる側のプログラム、およびその含んでいる側のプログラムに直接的または間接的に含まれているすべてのプログラムに記述されているステートメントによってのみ参照できます。 ただし、COMMON 属性を持つプログラムとそのプログラムに含まれるすべてのプログラムを除きます。
- プログラム名が、別にコンパイルされたプログラムのプログラム名である場合、 そのプログラム名は、実行単位の中の他のどのプログラムに含まれたステートメントによっても参照できます。 ただし、そのプログラム名を持つプログラムが直接的または間接的に含むプログラムは除きます。

どのプログラムを呼び出すかを判別するために使用するメカニズムは以下のとおりです。

- CALL ステートメントで指定された名前と同じ名前を持つ 2 つのプログラムのうちの 1 つが、 CALL ステートメントを含むプログラム内に直接的に含まれる場合は、 そのプログラムが呼び出されます。
- CALL ステートメントで指定された名前と同じ名前を持つ 2 つのプログラムのうちの 1 つが COMMON 属性を持ち、CALL ステートメントを含むプログラムを直接的または間接的に含む別のプログラムに直接的に含まれる場合、 呼び出し側プログラムがその共通プログラム内に含まれるのでない限り、 その共通プログラムが呼び出されます。
- 上記以外の場合は、別々にコンパイルされたプログラムが呼び出されます。

別のプログラム内に含まれるプログラムのプログラム名を参照する際には、次に示す規則が適用されます。この説明では、プログラム A はプログラム B とプログラム C を含んでおり、プログラム C はプログラム D とプログラム F、プログラム D はプログラム E を含んでいます。



プログラム D に COMMON 属性が指定されていない場合は、 プログラム D はそれを直接的に含むプログラム、 すなわちプログラム C によってしか参照できません。

プログラム D に COMMON 属性が指定されている場合は、プログラム C はプログラム D を参照できます (プログラム C にプログラム D が含まれているため) し、プログラム C に含まれるプログラムもすべてプログラム D を参照できます。 ただし、プログラム D に含まれるプログラムはプログラム D を参照できません。 言い換えれば、プログラム D に COMMON 属性が指定されている場合、プログラム D はプログラム C およびプログラム F において参照できますが、プログラム E、プログラム A、またはプログラム B においてはステートメントで参照できません。

第 3 部 見出し部

第 12 章 IDENTIFICATION DIVISION

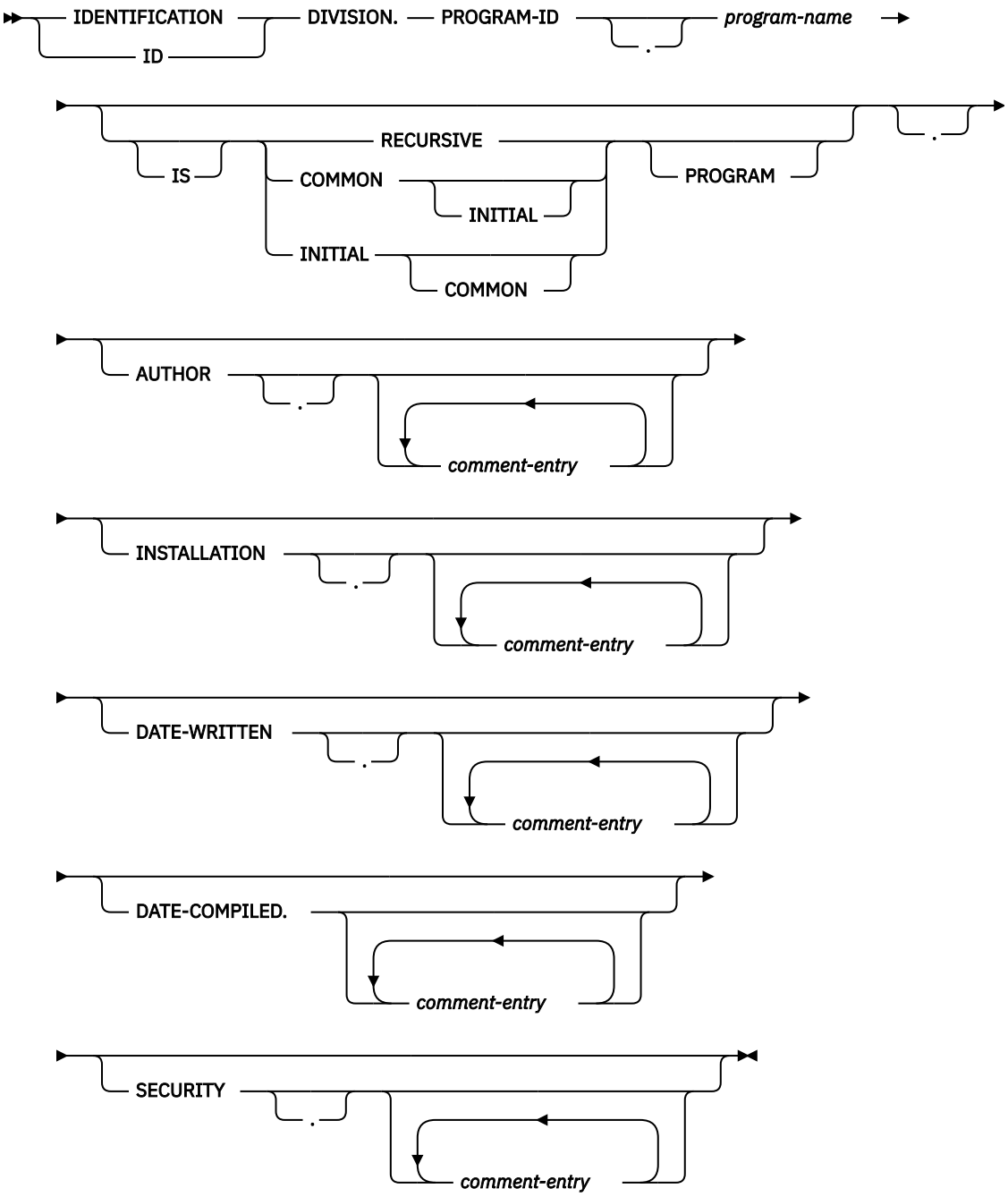
IDENTIFICATION DIVISION は、それぞれの COBOL ソース・プログラム。IDENTIFICATION DIVISION には、プログラムが書かれた日付やコンパイルの日付、およびその他の文書情報を入れることができます。

プログラム IDENTIFICATION DIVISION

プログラムの場合、IDENTIFICATION DIVISION 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ンであり、任意の順序で指定することができます。

プログラム IDENTIFICATION DIVISION のフォーマットは次のとおりです。

フォーマット: プログラム見出し部



PROGRAM-ID 段落

PROGRAM-ID 段落は、プログラムの名前を指定し、選択されたプログラム属性をそのプログラムに割り当てます。 PROGRAM-ID 段落は必須であり、IDENTIFICATION DIVISION の最初の段落でなければなりません。

プログラム名

プログラムを指定するユーザー定義語または英数字リテラル (形象定数ではない)。 これは、PGMNAME コンパイラー・オプションの設定値に応じて、 次の形成規則に従う必要があります。

PGMNAME (LONGUPPER)

プログラム名 がユーザー定義語の場合、その長さは最大 30 文字です。

プログラム名 が英数字リテラルの場合、リテラルの長さは最大 160 文字です。 リテラルは、形象定数にすることはできません。

名前をユーザー定義として指定するときは、ハイフン、下線、0 から 9 の数字、および英字だけが名前に使用できます。

少なくとも 1 文字は英字でなければなりません。

ハイフンは最初または最後の文字に使用することはできません。

プログラム名が英数字リテラルの場合、下線文字を先頭にすることができます。

外部プログラム名は、大文字変換された英字で処理されます。

PGMNAME (LONGMIXED)

プログラム名 は、英数字リテラルとして指定する必要があります。長さは 160 文字までです。 リテラルは、形象定数にすることはできません。

英字を使用できる場合、常にマルチバイト文字を使用できます。

PGMNAME コンパイラー・オプションおよびコンパイラーによる名前の処理方法については、「*COBOL for Linux on x86* プログラミング・ガイド」を参照してください。 「*COBOL for Linux on x86* プログラミング・ガイド」内の『*PGMNAME*』を参照してください。

RECURSIVE

COBOL プログラムが再帰的に再入するのを認める、オプションの節。

RECURSIVE 節は、コンパイル単位の最外部のプログラムに対してのみ指定することができます。 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はできません。

RECURSIVE 節が指定された場合、それ以前の呼び出しがまだアクティブであっても、プログラム名 に再帰的に再入させることができます。RECURSIVE 節が指定されない場合は、 アクティブ・プログラムは再帰的に再入されることはできません。

再帰的プログラムの WORKING-STORAGE SECTION は、プログラムに対して最初の項目で静的に割り振られて初期設定され、 任意の再帰的呼び出しに対して最後に使用された状態で使用できるストレージを定義します。

非再帰的プログラムと同じく、再帰的プログラムの LOCAL-STORAGE SECTION は、 呼び出しのたびに自動的に割り振り、初期設定、および割り振り解除が行われるストレージを定義します。

再帰的プログラムの FILE SECTION の FD に対応する内部ファイル結合子は、静的に割り振られます。内部ファイル結合子の状況は、 呼び出しを超えて持続するプログラムの最後に使用された状態の一部です。

次の言語エレメントは、再帰的プログラムではサポートされません。

- ALTER
- 指定されたプロシージャー名のない GO TO
- RERUN

- SEGMENT-LIMIT
- USE FOR DEBUGGING

COMMON
プログラム名 によって指定されたプログラムが別のプログラム内に含まれ (つまり、ネストされ)、それを共通プログラムの兄弟プログラム、およびそれらに含まれたプログラムから呼び出すことができることを指定します。 COMMON 節は、ネストされたプログラムでのみ使用できます。 プログラム名の規約についての詳細は、78 ページの『プログラム名の命名規約』を参照してください。

INITIAL
プログラム名が呼び出されたときに、プログラム名 とその中に含まれる (ネストされる) プログラムが初期状態に置かれることを指定します。

プログラムは、次のとき初期状態になります。

- プログラムが実行単位に初めて呼び出されたとき。
- プログラムが初期属性を持っている場合には、それが呼び出されるたびに。
- プログラムを参照している CANCEL ステートメントの実行後、 またはそのプログラムを直接的または間接的に含んでいるプログラムを参照している CANCEL ステートメントの実行後、 そのプログラムが最初に呼び出されたとき。
- そのプログラムを直接的または間接的に含み、 初期属性を持っているプログラムを参照している CALL ステートメントの実行後、そのプログラムが最初に呼び出されたとき。

プログラムが初期状態である場合:

- プログラムの WORKING-STORAGE SECTION にある内部データが初期設定されます。 VALUE 節がデータ項目の記述の中で使用されている場合、そのデータ項目は定義された値に初期設定されます。 VALUE 節がデータ項目と関連していない場合、そのデータ項目の初期値は未定義です。
- プログラムと関連した内部ファイル結合子を持つファイルは、 オープン・モードになっていません。
- 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、それぞれ初期状態に設定されます。
- そのプログラムに含まれ変更された GO TO ステートメントは、 初期状態に設定されます。

固有でないプログラム名に適用される規則については、78 ページの『プログラム名の規則』を参照してください。

オプションの段落

IDENTIFICATION DIVISION にある一部のオプションの段落は、省略可能です。

オプションの段落は以下のとおりです。

AUTHOR
プログラムの作成者名。

INSTALLATION
会社や場所の名前。

DATE-WRITTEN
プログラムの作成期日。

DATE-COMPILED
DATE-COMPILED パラグラフは、ソース・リスト表示にコンパイル日を示します。 コメント項目が指定されている場合は、項目が複数行にわたっていても、項目全体が現在日付と置き換えられます。 コメント項目が省略されている場合は、コンパイラーは DATE-COMPILED が印刷されている行に現在日付を追加します。 以下に例を示します。

```
DATE-COMPILED. 06/30/10.
```

SECURITY
プログラムのセキュリティー・レベル。

オプションの段落の中のコメント項目には、コンピューターの文字セットの文字であればどのような組み合わせでも使用できます。 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。

コメント項目は情報としてのみ役立つものです。 プログラムの意味に影響を与えることはありません。 コメント項目では、 標識域 (第 7 桁) にハイフンを入れることはできません。

マルチバイト文字のほか、EUC、UTF-8、または DBCS コード・ページの 1 バイト文字を、プログラムの IDENTIFICATION DIVISION のコメント項目に含めることができます。 マルチバイト文字ストリングを含むコメント項目は、複数行にすることができます。

第 4 部 環境部

第 13 章 構成セクション

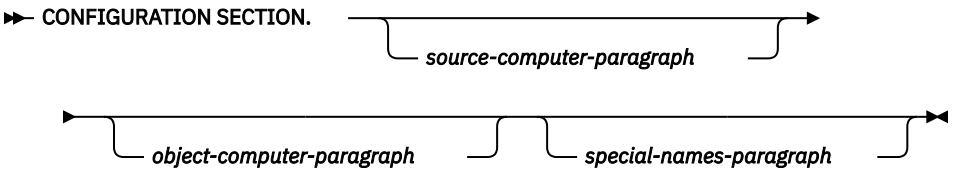
構成セクションは、プログラムのオプショナルのセクションであり、そこではプログラムまたはクラスがコンパイルされ実行されるコンピューター環境を記述することができます。

プログラム構成セクション

構成セクションは、COBOL ソース・プログラムの最外部のプログラムの ENVIRONMENT DIVISION でのみ指定することができます。

別のプログラムに含まれているプログラムでは構成セクションを指定すべきではありません。 あるプログラムの構成セクションの中で指定された項目は、そのプログラムに含まれるどのプログラムに対しても適用されます。

フォーマット:



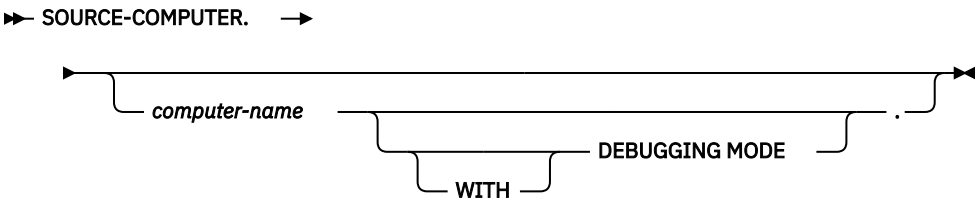
構成セクションでは、次のことが可能です。

- IBM 定義の環境名をユーザー定義の簡略名に関係付ける。
- 照合シーケンスを指定する。
- 通貨符号値、 および PICTURE 節でその通貨符号値を表すために使用する通貨記号を指定する。
- PICTURE 節と数字リテラル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シーケンスに関係付ける。
- シンボリック文字を指定する。
- 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。

SOURCE-COMPUTER 段落

SOURCE-COMPUTER 段落は、ソース・テキストがコンパイルされるコンピューターを記述します。

フォーマット



コンピューター名

システム名。以下に例を示します。

```
IBM-system
```

WITH DEBUGGING MODE

ソース・テキストに書かれた行をデバッグするために、コンパイル時スイッチを活動化します。

デバッグ行 は、コンパイル時スイッチが活動化しているときに限りコンパイルされるステートメントです。 デバッグ行を使用すると、例えば、プロシージャー内のある位置においてデータ名の値を検査することができます。

プログラムの中にデバッグ行を指定するには、第 7 桁 (標識域) に D とコーディングします。 デバッグ行は連続して含めることができますが、それぞれのデバッグ行の第 7 桁が D でなければなりません。このとき、複数行にまたがって文字ストリングを分割することはできません。

すべてのデバッグ行は、 そのデバッグ行がコンパイルされるかコメントとして扱われるかに関係なく、プログラムが構文上正しくなるように記述しなければなりません。

DEBUGGING MODE 節が存在するかどうかは、すべての COPY ステートメントと REPLACE ステートメントが処理された後で、 論理的に判断されます。

デバッグ行は、ENVIRONMENT DIVISION (OBJECT-COMPUTER 段落の後)、データ部、または手続き部にコーディングできます。

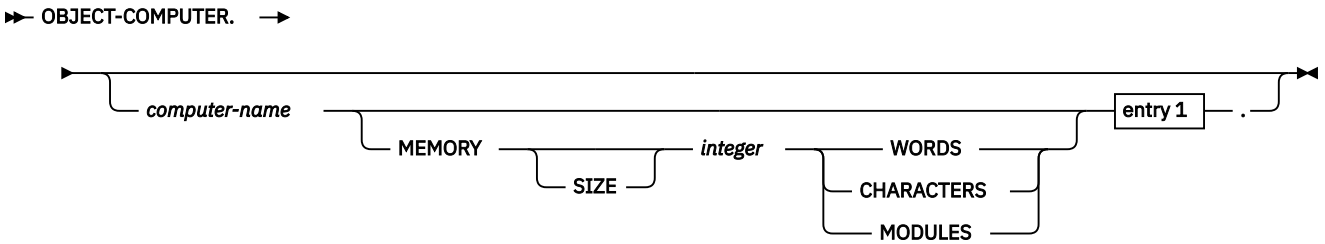
デバッグ行で領域 A および領域 B にスペースしかない場合、デバッグ行はブランク行とみなされます。

SOURCE-COMPUTER 段落はすべて構文チェックされますが、WITH DEBUGGING MODE 節のみがプログラムの実行に影響します。

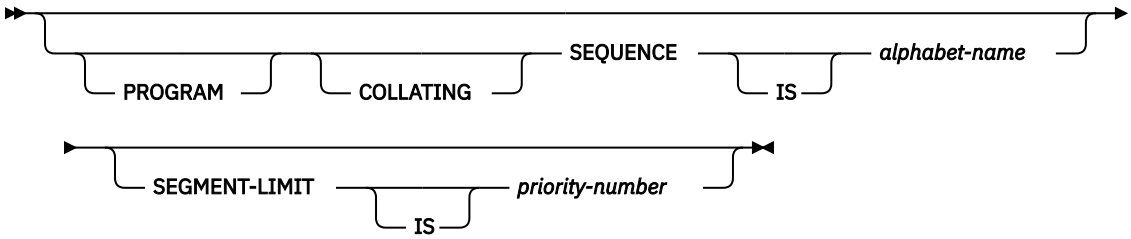
OBJECT-COMPUTER 段落

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。

フォーマット



項目 1



コンピューター名
システム名。以下に例を示します。

```
IBM-system
```

MEMORY SIZE 整数
integer は、オブジェクト・プログラムの実行に必要な主記憶域の容量をワード数、文字数、または モジュール数で指定します。 MEMORY SIZE 節は構文チェックされますが、プログラムの実行には何も影響しません。

PROGRAM COLLATING SEQUENCE IS 英字名
このプログラムで使用される照合シーケンスは、 ここで指定する英字名 と関連付けられた照合シーケンスになります。

この照合シーケンスは、このプログラムとそれが含んでいるすべてのプログラムで使用されます。

PROGRAM COLLATING SEQUENCE は、 次のような英数字比較の真の値を判別する際に使用されます。

- 比較条件において明示的に指定されたもの。
- 条件名条件の中で明示的に指定されたもの。

COLLATING SEQUENCE 句が MERGE または SORT ステートメントで指定されていなければ、PROGRAM COLLATING SEQUENCE 節は、USAGE DISPLAY で記述されているマージ・キーまたはソート・キーにも適用されます。

PROGRAM COLLATING SEQUENCE 節は、USAGE NATIONAL のデータ項目には適用されません。

ソース・コード・ページがマルチバイト・コード・ページである場合、PROGRAM COLLATING SEQUENCE 節を指定できません。

PROGRAM COLLATING SEQUENCE 節を省略した場合は、 COLLSEQ コンパイラー・オプションは、使用される照合シーケンスを示します。 例えば、COLLSEQ(EBCDIC) を指定し、PROGRAM COLLATING SEQUENCE 節を指定しなかった (または、NATIVE である) 場合、EBCDIC 照合シーケンスが使用されます。

SEGMENT-LIMIT IS
SEGMENT-LIMIT 節は構文チェックされますが、プログラムの実行には何も影響しません。

優先順位番号
1 から 49 の範囲の整数。 優先順位番号として 0 から 49 の番号のすべてのセクションが固定永続セグメントです。優先順位番号とセグメンテーション・サポートについて詳しくは、233 ページの『プロシージャー』を参照してください。

OBJECT-COMPUTER 段落はすべて構文チェックされますが、PROGRAM COLLATING SEQUENCE 節のみがプログラムの実行に影響します。

SPECIAL-NAMES 段落

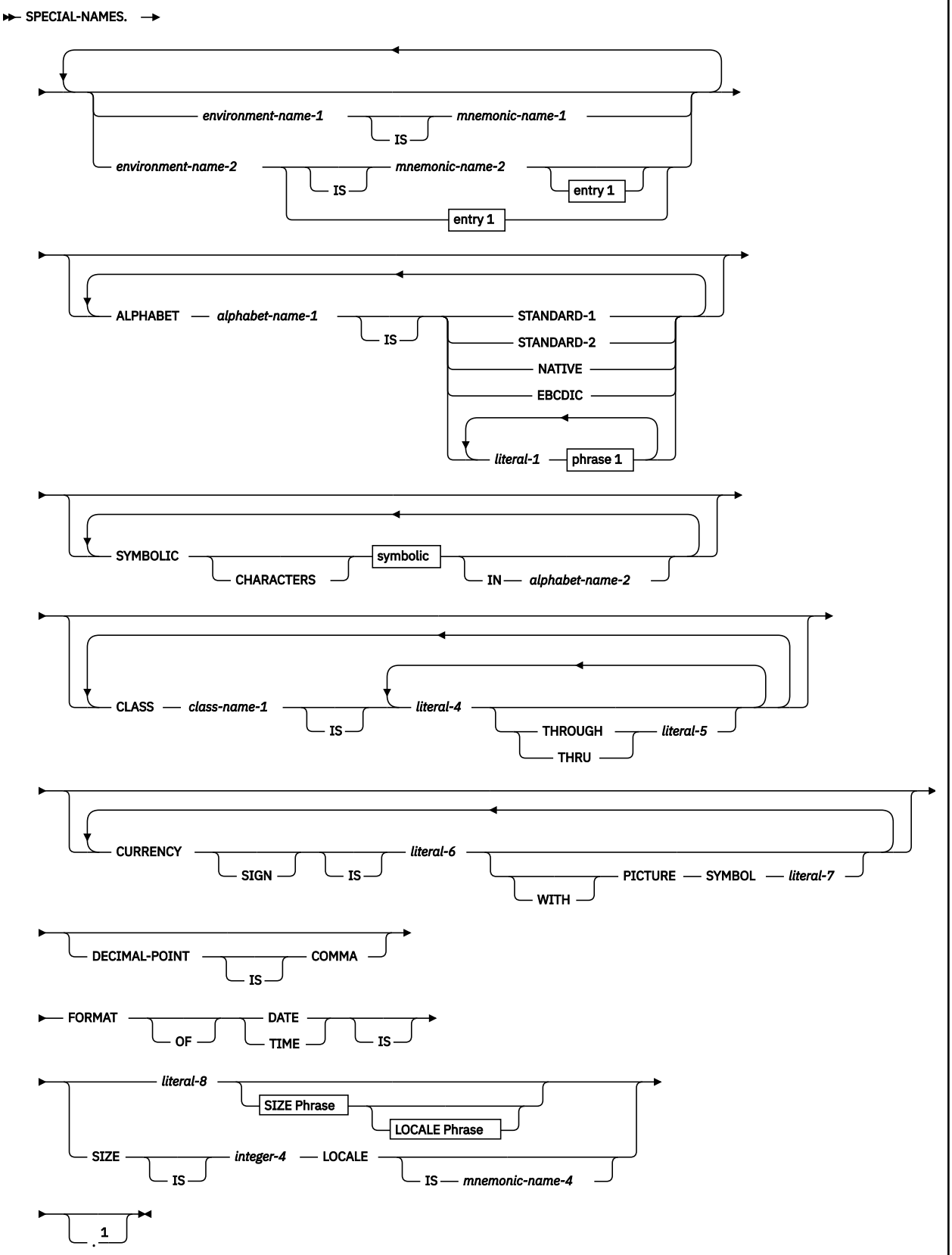
SPECIAL-NAMES 段落は ENVIRONMENT DIVISION の段落の名前で、ここで環境名がユーザー指定の簡略名と関連付けられます。

SPECIAL-NAMES 段落:

- IBM 指定の環境名をユーザー定義の簡略名に関係付ける。
- 英字名を文字セットまたは照合シーケンスに関係付ける。
- シンボリック文字を指定する。
- 文字の集合にクラス名を関係付ける。
- 1 つ以上の通貨符号値を指定して、それぞれの通貨符号値を表すピクチャー記号を PICTURE 節に定義する。
- PICTURE 節と数字リテラルでやり取りされるコンマと小数点の機能を指定する。
- 日時データ・タイプのデフォルト形式を指定する。
- 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。

SPECIAL-NAMES 段落の節は、任意の順序で指定できます。

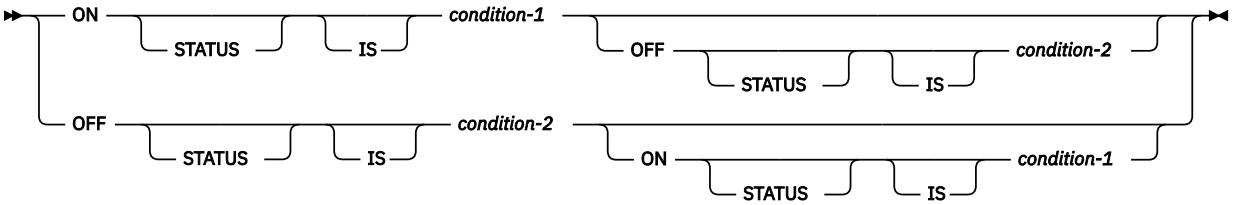
フォーマット: SPECIAL-NAMES 段落



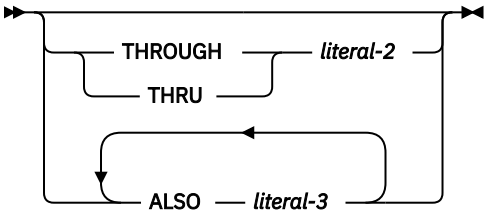
注:

[1] 節が選択されていない場合は、この分離文字ピリオドはオプショナルです。 節を使用する場合は、最後の節の後にピリオドをコーディングする必要があります。

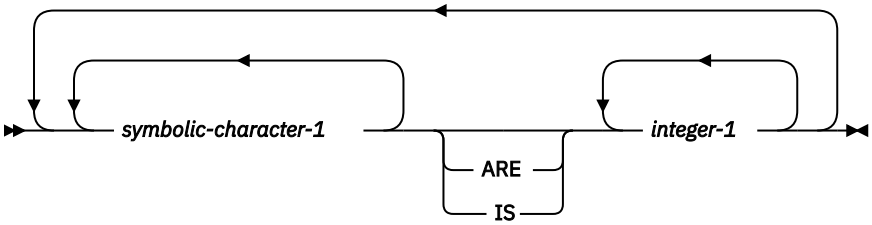
フラグメント
項目 1



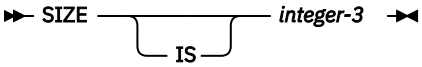
句 1



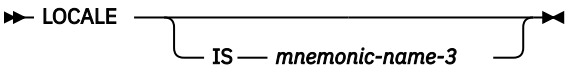
シンボリック



SIZE Phrase



LOCALE Phrase



ソース・コード・ページがマルチバイト・コード・ページである場合、次の節は指定できません。

- ALPHABET 節
- CLASS 節
- SYMBOLIC CHARACTERS 節

環境名-*1*
システム装置、またはコンパイラーの標準システム動作。

環境名-*1* に対して有効な指定を、次の表に示します。

表 5. **環境名の意味**		
環境名-*1*	**意味**	**指定できる句**
SYSIN SYSIPT	システム論理入力装置	ACCEPT
SYSOUT SYSLIST SYSLST	システム論理出力装置	DISPLAY
SYSPUNCH SYSPCH	システムせん孔装置	DISPLAY
CONSOLE	コンソール	ACCEPT および DISPLAY
C01 から C12	チャネル 1 からチャネル 12 の対応するチャネルをスキップします。	WRITE ADVANCING C01 から C12 では 1 行前進します。
CSP	行送りの抑止	WRITE ADVANCING
S01 から S05	せん孔装置のポケット選択 1 から 5	WRITE ADVANCING S01 から S05 では 1 行前進します。
AFP-5A	高機能印刷	WRITE ADVANCING

環境名-*2*
1 バイトのユーザー・プログラマブル状況標識 (UPSI) スイッチ。 環境名-*2* に指定できるのは UPSI-0 から UPSI-7 です。

簡略名-*1* 、 簡略名-*2*
簡略名-*1* および簡略名-*2* は、ユーザー定義名形成の規則に従います。 簡略名-*1* は、ACCEPT、DISPLAY、および WRITE の各ステートメントで使用することができます。 簡略名-*2* は、SET ステートメントの中でのみ参照できます。 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、固有の名前である必要はありません。 簡略名に環境名と同じ名前を選択した場合には、 簡略名としての定義が環境名としての定義に優先します。

ON STATUS IS、OFF STATUS IS
UPSI スイッチは、 年末や年始の処理のようなプログラム内の特別の条件を処理します。 例えば、PROCEDURE DIVISION の冒頭で UPSI スイッチをテストし、それが ON であれば特殊ブランチを実行することができます。(254 ページの『スイッチ状況条件』 を参照。)

条件-*1*、条件-*2*
条件名はユーザー定義名形成の規則に従います。 少なくとも 1 文字は英字でなければなりません。 条件名に関係付けられた値は英数字とみなされます。 条件名は、指定された各 UPSI スイッチのオン状況またはオフ状況に関連付けることができます。

PROCEDURE DIVISION では、UPSI スイッチ状況は、関連付けられた条件名を使用してテストされます。それぞれの条件名は、レベル 88 項目と等価です。 関連した簡略名が指定されれば、それは条件変数とみなされ、 修飾のために使用することができます。

プログラムを含んでいるプログラムの SPECIAL-NAMES 段落に指定された条件名は、含まれている任意のプログラムで参照できます。

ALPHABET 節

ALPHABET 節は、英字名を指定の文字コード・セットまたは照合シーケンスに関連付ける方法を提供します。

関連する文字コード・セットまたは照合シーケンスは、英数字データに適用できますが、国別データには適用できません。

ALPHABET 英字名-*1* IS

英字名-*1* は、以下を使用した場合に、照合シーケンス を指定します。

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

英字名-*1* は、以下で使用される場合に、文字コード・セットを指定します。

- FD 項目 CODE-SET 節
- SYMBOLIC CHARACTERS 節

マルチバイト・コード・ページでソース・コード・ページが有効になっている場合、ALPHABET 節は指定できません。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『照合シーケンスの指定』を参照してください。

STANDARD-1
照合シーケンスが文字のバイナリー・コード値に基づいていて、ロケール設定が無視されることを指定します。

STANDARD-2
照合シーケンスが文字のバイナリー・コード値に基づいていて、ロケール設定が無視されることを指定します。

NATIVE
固有文字コード・セットを指定します。 ALPHABET 節が省略された場合、英字名は、有効なロケールで示された ASCII、UTF-8、または EUC 文字セットに関連付けられます。

EBCDIC
EBCDIC 文字セットを指定します。

リテラル-1* 、 *リテラル-2* 、 *リテラル-3
プログラムが、以下に示す規則に従って、英数字データの照合シーケンスを 決定する指定をします。

- リテラルが現れる順序は、この照合シーケンスにおける文字の序数 (昇順) を指定します。
- 指定する各数字リテラルは、 符号なしの整数でなければなりません。
- 各数字リテラルの値は、有効な照合シーケンス内の有効な 順序位置に対応する値である必要があります。

  1 バイトの EBCDIC 照合シーケンスおよび ASCII 照合シーケンスにおける各文字の序数については、535 ページの『付録 D EBCDIC および ASCII の照合シーケンス』を参照してください。
- 英数字リテラルの中のそれぞれの文字は、文字セット内の実際の文字を表します。 (英数字リテラルに複数の文字が含まれる場合、 それぞれの文字は左端から始めて、 この照合シーケンス内の連続した昇順位置を割り当てられます。)
- 明示的に指定されていない文字はすべて、この照合シーケンスにおいて、 明示的に指定されたどの文字よりも高いと想定されます。 こうした未指定の文字の照合シーケンスにおける相対順序は、COLLSEQ コンパイラー・オプションによって示された照合シーケンスにおけるその相対順序です。
- 1 つの英字名節の中では、 ある 1 つの文字を 2 回以上指定することはできません。
- THROUGH 句または ALSO 句に関係付けられた英数字リテラルは、それぞれ 1 文字の長さでなければなりません。

- THROUGH 句を指定する場合、リテラル-*1* で指定された文字で開始し、リテラル-*2* で指定された文字で終了する固有文字セット内の連続する文字が、この照合シーケンス内の昇順位置に順々に割り当てられます。

  このシーケンスは、当初の固有文字セットの中では昇順でも降順でも可能です。 つまり、"Z" THROUGH "A" と指定した場合、大文字の英字に関する昇順の値は次の左から右のようになります。

  ```
  ZYXWVUTSRQPONMLKJIHGFEDCBA
  ```

- ALSO 句を指定する場合、リテラル-*1*、リテラル-*3* などとして指定された文字は、この照合シーケンスにおいて同じ位置に割り当てられます。 例えば、次のように指定したとします。

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

  文字 D、N、% は、すべてこの照合シーケンスの中で同じ位置にあるとみなされます。
- ALSO 句が指定され、英字名-*1* が SYMBOLIC CHARACTERS 節の中で参照されるときは、 文字セットの中の文字を表すためにリテラル-*1* だけが使用されます。
- この照合シーケンスで最高値の順序位置の文字は、形象定数 HIGH-VALUE に関連しています。 ALSO 句を指定したために複数の文字が最高になる場合は、指定された最後の文字 (または文字が明示的に指定されていない場合はデフォルトの文字) が、 プロシージャー・ステートメント (DISPLAY ステートメントや MOVE ステートメントの 送り出しフィールドなど) の HIGH-VALUE 文字であるとみなされます。 (この照合シーケンスの最上位文字として、 上記の例の ALSO 句が指定されている場合は、HIGH-VALUE 文字は % になります。)
- この照合シーケンスで最低値の順序位置の文字は、形象定数 LOW-VALUE に関連しています。 ALSO 句を指定したために複数の文字が最低になる場合は、指定された最初の文字が LOW-VALUE 文字になります。 (照合シーケンスの最下位文字として上記の例の ALSO 句が指定されていれば、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* を指定する場合、CODE-SET 節の中でその英字名を参照することはできません (156 ページの『CODE-SET 節』を参照)。

リテラル-*1*、リテラル-*2*、およびリテラル-*3* は、英数字リテラルまたは数字リテラルでなければなりません。 すべて、 カテゴリーが同じでなければなりません。 浮動小数点リテラル、国別リテラル、DBCS リテラル、またはシンボリック文字の形象定数を指定してはなりません。

CLASS 節

CLASS 節は、ある名前を、その節内に示されている特定の文字の集合に関係付ける手段を提供します。

マルチバイト・コード・ページでソース・コード・ページが有効になっている場合、CLASS 節を指定できません。

CLASS クラス名-*1* IS
ある名前を、その節の中に示されている特定の文字の集合に関係付ける手段を提供しています。 クラス名-*1* は、クラス条件の中でのみ参照できます。 この節でそのリテラルの値によって指定された文字は、 このクラスを構成する文字の排他的集合を定義します。

リテラル-4、リテラル-5
カテゴリーは数字または英数字であり、または両方とも同じカテゴリーにする必要があります。

数字の場合は、リテラル-*4* およびリテラル-*5* は、符号なしの 1 以上の値の整数で、指定された英字の中の文字数以下の値でなければなりません。 それぞれの数字は、1 バイト EBCDIC または ASCII 照合シーケンス内の各文字の順序位置に対応しています。 浮動小数点リテラルまたは DBCS リテラルとして指定することはできません。

英数字の場合は、リテラル-*4* およびリテラル-*5* は、実際の 1 バイト EBCDIC 文字または 1 バイトの ASCII 文字でなければなりません。

リテラル-*4* およびリテラル-*5* には、シンボリック文字の形象定数を指定しないでください。 英数字リテラルの値に複数の文字が含まれる場合、 リテラル内の各文字は、クラス名によって識別される文字の集合に含まれます。

英数字リテラルが THROUGH 句と関連付けられている場合、そのリテラルは 1 文字の長さでなければなりません。

THROUGH、 THRU
THROUGH と THRU は同じ意味です。 THROUGH が指定されている場合、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含まれることになります。 さらに、THROUGH 句によって指定された文字は、 文字を昇順でも降順でも指定することができます。

CURRENCY SIGN 節

CURRENCY SIGN 節は、PICTURE 文字ストリングに通貨記号 が含まれている数字編集データ項目に影響します。

通貨記号は通貨符号値 を表します。これは、以下のようになります。

- 前述のようなデータ項目が受け取り項目として使用された場合には、そのデータ項目に挿入されます。
- 受け取り側の数字または数字編集の送り出し項目として使用された場合には、そのデータ項目から除去されます。

一般に通貨符号値は、データ項目に保管される通貨単位を示します。 例えば、「$」、「EUR」、「CHF」、「JPY」、「HK$」、「HKD」、または X'9F' (€ (ユーロ通貨記号) 用の一部の EBCDIC コード・ページの中の 16 進コード・ポイント)。 ユーロ通貨の処理に関する プログラミング・テクニックについては、「*COBOL for Linux on x86* プログラミング・ガイド」内の『通貨記号の使用』 を参照してください。

CURRENCY SIGN 節は、通貨符号値、および PICTURE 節でその通貨符号値を表すために使用する通貨記号を指定します。

SPECIAL-NAMES 段落には複数の CURRENCY SIGN 節を含めることができます。 各 CURRENCY SIGN 節ごとに違う通貨記号を指定することが必要です。 他のすべての PICTURE 節記号とは異なり、通貨記号は大/小文字が区別されます。 例えば、'D' と 'd' は異なる通貨記号を指定します。

CURRENCY SIGN IS リテラル-*6*
リテラル-*6* は英数字リテラルでなければなりません。 リテラル-*6* は、形象定数またはヌル終了リテラルにすることはできません。 リテラル-*6* には、マルチバイト文字を含む必要はありません。

PICTURE SYMBOL 句が指定されていない場合、リテラル-*6* は、以下のようになります。

- 通貨符号値と、その通貨符号値を表すための通貨記号を指定します。
- 単一文字でなければなりません。
- 以下の数字および文字は使用できません。
  - 数字 0 から 9
  - 英字 A、a、B、b、C、c、D、d、E、e、G、g、N、n、P、p、R、r、 S、s、V、v、X、x、Z、z、またはスペース
  - 特殊文字 + - , . * / ; ( ) " = ' (正符号、負符号、コンマ、ピリオド、アスタリスク、スラッシュ、セミコロン、左括弧、右括弧、引用符、等号、アポストロフィ)
- 英小文字 f、h、i、j、k、l、m、o、q、t、u、w、y のいずれか。

PICTURE SYMBOL 句が指定されている場合、リテラル-*6* は、以下のようになります。

- 通貨符号値を指定します。 PICTURE SYMBOL 句の中のリテラル-*7* は、この通貨符号値を表すための通貨記号を指定します。
- 1 つ以上の文字で構成できます。
- 以下の数字および文字は使用できません。
  - 数字 0 から 9

– 特殊文字 + - . ,

PICTURE SYMBOL リテラル-7
PICTURE 節でリテラル-6 によって指定される通貨符号値を表すために使用できる通貨記号を指定します。

リテラル-7 は 1 つの 1 バイト文字で構成される英数字リテラルでなければなりません。 リテラル-7 には、以下の数字および文字を使用しないでください。

- 形象定数
- 数字 0 から 9
- 英字 A、a、B、b、C、c、D、d、E、e、G、g、N、n、P、p、R、r、 S、s、V、v、X、x、Z、z、またはスペース
- 特殊文字 + - , . * / ; ( ) " = '

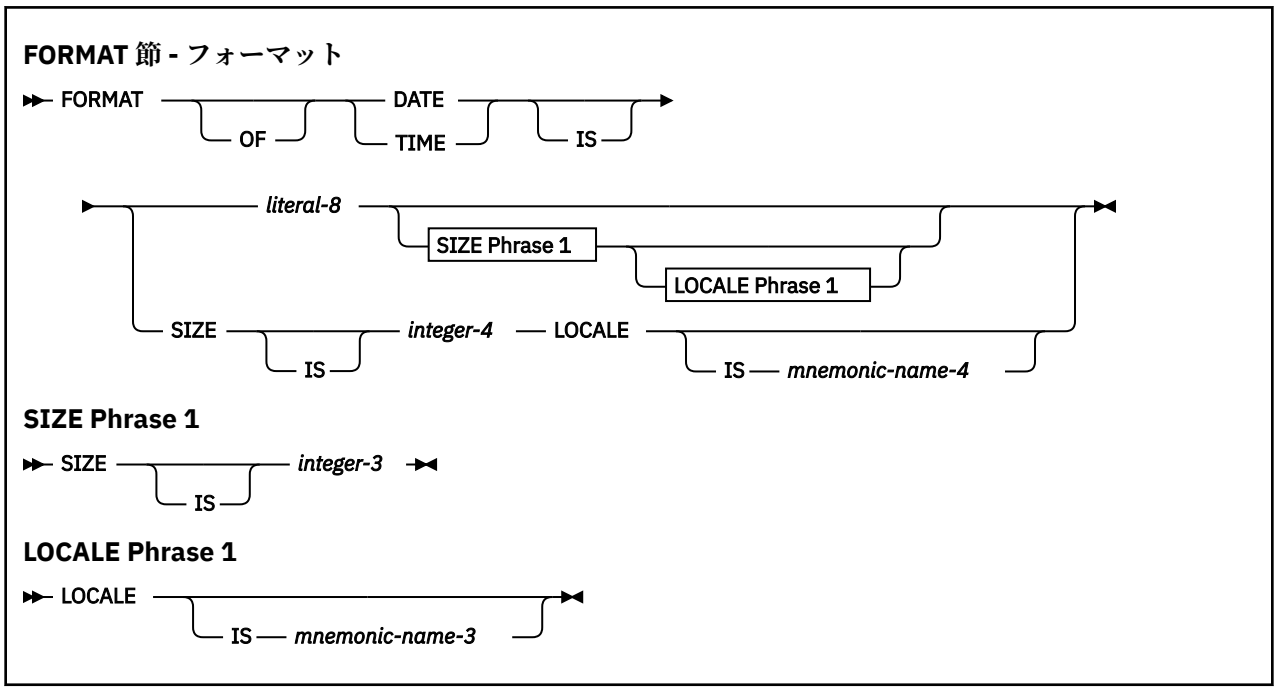
CURRENCY SIGN 節が指定されている場合、CURRENCY および NOCURRENCY コンパイラー・オプションは無視されます。 CURRENCY SIGN 節が指定されない場合に NOCURRENCY コンパイラー・オプションが有効であれば、デフォルトの通貨符号値および 通貨記号としてドル記号 ($) が使用されます。 CURRENCY および NOCURRENCY コンパイラー・オプションについて詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」内の『*CURRENCY*』 を 参照してください。

DECIMAL-POINT IS COMMA 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。

FORMAT 節

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。



literal-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 文字の英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 101 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 99 ページの表 6 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 ロケールの詳細については、「*COBOL for Linux on x86* プログラミング・ガイド」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。
- 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。
- 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。

以下の表に *literal-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 *6.* リテラル-*8* で使用することができる変換指定子

指定子	**説明**	**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
%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

表 6. リテラル-8 で使用することができる変換指定子 (続き)

指定子	説明	長さ	用途
@Sp	1 兆分の 1 秒単位の秒を表す 1 つの整数 [000000000000, 999999999999] に置き換えられる	12 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

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 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。

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。

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。

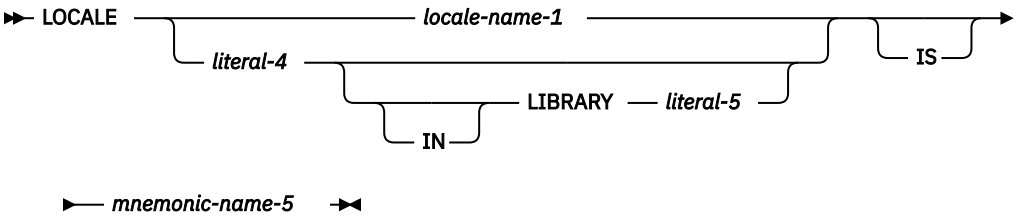
簡略名-*3*、簡略名-*4*
簡略名-*3* または簡略名-*4* が指定されている 場合は、日付項目または時刻項目のために使用されるロケールは SPECIAL-NAMES 段落内の簡略名-*3* または簡略名-*4* に関連付けられているロケールとなります。簡略名-*3* または簡略名-*4* が指定されていない 場合は、現行ロケールが使用されます。現行ロケールを判別するには、「*COBOL for Linux on x86* プログラミング・ガイド」 の『*CEELOCT:* 現在の現地日時の取得』 を参照してください。

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。101 ページの『LOCALE 文節』 を参照してください。

LOCALE 文節

LOCALE 文節は、ロケール簡略名と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



locale-name-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。 COBOL for Linux の場合、サポートされているロケール名-*1* は POSIX だけです。

literal-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。

literal-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。 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。

mnemonic-name-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 9、または数種の組み込み関数の引数リストの中だけです。

SYMBOLIC CHARACTERS 節

1 バイト文字セットを適用できるのは、 SYMBOLIC CHARACTERS 節だけです。 表されるそれぞれの文字は、英数字です。

SYMBOLIC CHARACTERS シンボリック文字-*1*
これは、1 つ以上のシンボリック文字を指定できる手段を提供します。 シンボリック文字-*1* はユーザー定義語であり、少なくとも 1 つの英字を含んでいる必要があります。 同じシンボリック文字は、SYMBOLIC CHARACTERS 節の中では一度しか使用できません。

ソース・コード・ページがマルチバイト・テキスト・ページである場合、SYMBOLIC CHARACTERS 節は使用できません。

シンボリック文字-*1* の内部表現は、 指定された文字セットで表された文字の内部表現になります。 次の規則が適用されます。

- 各シンボリック文字-*1* とそれに対応した整数-*1* との関係は、SYMBOLIC CHARACTERS 節の中のそれらの位置によります。 最初のシンボリック文字-*1* は最初の整数-*1* と対になり、第 2 のシンボリック文字-*1* は第 2 の整数-*1* と対になる、といった具合です。
- シンボリック文字-*1* のオカレンスと整数-*1* のオカレンスは、SYMBOLIC CHARACTERS 節の中で 1 対 1 の対応関係になければなりません。
- IN 句が指定されている場合、整数-*1* には英字名-*2* で指定した文字セットに表されている文字の順序位置を指定します。 この順序位置は実際に存在するものでなければなりません。
- IN 句が指定されていない場合、シンボリック文字-*1* は、その文字の順序位置が固有文字セットの中で整数-*1* によって指定されている文字です。

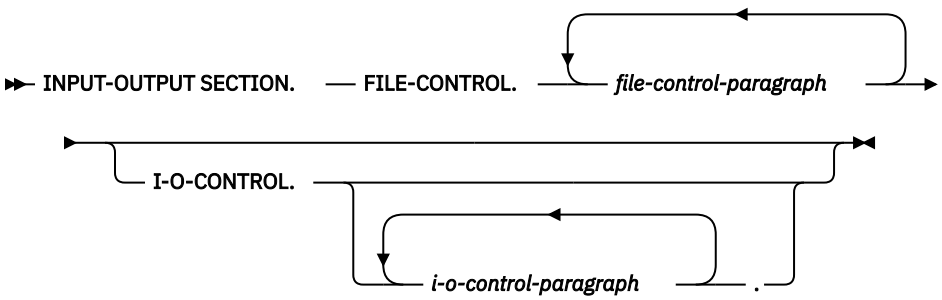
  順序位置には、1 から始まる番号が付けられます。

第 14 章 入出力セクション

ENVIRONMENT DIVISION の入出力セクションには、FILE-CONTROL 段落および I-O-CONTROL 段落があります。

入出力セクションの正確な内容は、使用されているファイル編成とアクセス方式により決まります。 114 ページの『ORGANIZATION 節』および 117 ページの『ACCESS MODE 節』を参照してください。

フォーマット: 入出力セクション



FILE-CONTROL
キーワード FILE-CONTROL は、ファイル制御段落を指定します。 このキーワードは、FILE-CONTROL 段落の先頭に一度だけ指定することができます。 このキーワードは領域 A で開始し、その後に分離文字ピリオドを付けなければなりません。

ファイル制御段落 が指定されておらず、プログラム内でファイルが定義されていない場合、キーワード FILE-CONTROL およびピリオドは省略できます。

ファイル制御段落
ファイルの名前を指定し、それらのファイルを外部ファイルに関連付けます。

SELECT 節を使用し、領域 B から開始する必要があります。 後に分離文字ピリオドを付けなければなりません。 103 ページの『FILE-CONTROL 段落』を参照してください。

プログラム内でファイルが定義されていない場合は、FILE-CONTROL キーワードが指定されていても、ファイル制御段落 を省略できます。

I-O-CONTROL
キーワード I-O-CONTROL は I-O-CONTROL 段落を指定します。

入出力制御段落
データを効率的に、外部ファイルと COBOL プログラムとの間で伝送するために 必要とされる情報を指定します。 一連の項目は、分離文字ピリオドで終了しなければなりません。 124 ページの『I-O-CONTROL 段落』を参照してください。

FILE-CONTROL 段落

FILE-CONTROL 段落は、COBOL プログラム内の各ファイルを外部ファイルに関連 付け、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。

FILE-CONTROL 段落のフォーマットは次のとおりです。

- 順次ファイル項目
- 索引付きファイル項目
- 相対ファイル項目
- 行順次ファイル項目

以下の表には、プログラムで使用可能な各種のファイルのリストを示します。

表 7. **ファイルのタイプ**	
ファイル編成	**ファイル・システム**
順次	SFS, STL, RSD[1], Db2®, QSAM[2]
相対	SFS, STL, Db2
索引付き	SFS, STL, Db2
行順次	LSQ

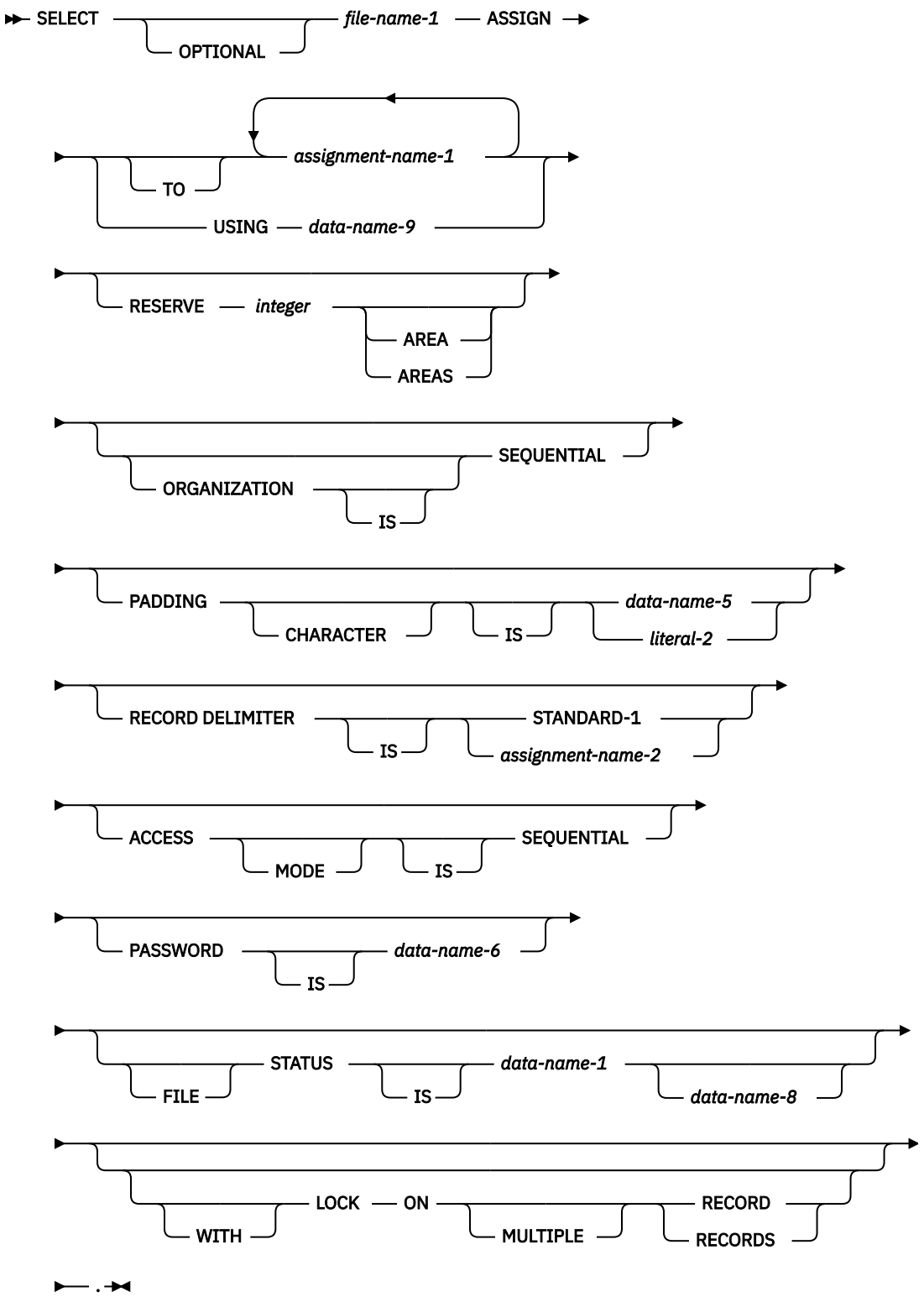
1. RSD ファイル・システムでは、固定長レコードまたは可変長レコードの順次ファイルのみがサポートされます。
2. QSAM ファイル・システムでは、固定長レコード、可変長レコード、およびスパン・レコードがサポートされます。
3. SdU ファイル・タイプを指定すると、STL ファイルを指定した場合と同様に処理されます。

FILE-CONTROL 段落は、FILE-CONTROL という語で開始し、後に分離文字ピリオドが続きます。 ここには、 DATA DIVISION の FD 項目または SD 項目で記述されるそれぞれのファイルに対応して、1 つの (ただ 1 つの) 項目を記述する必要があります。

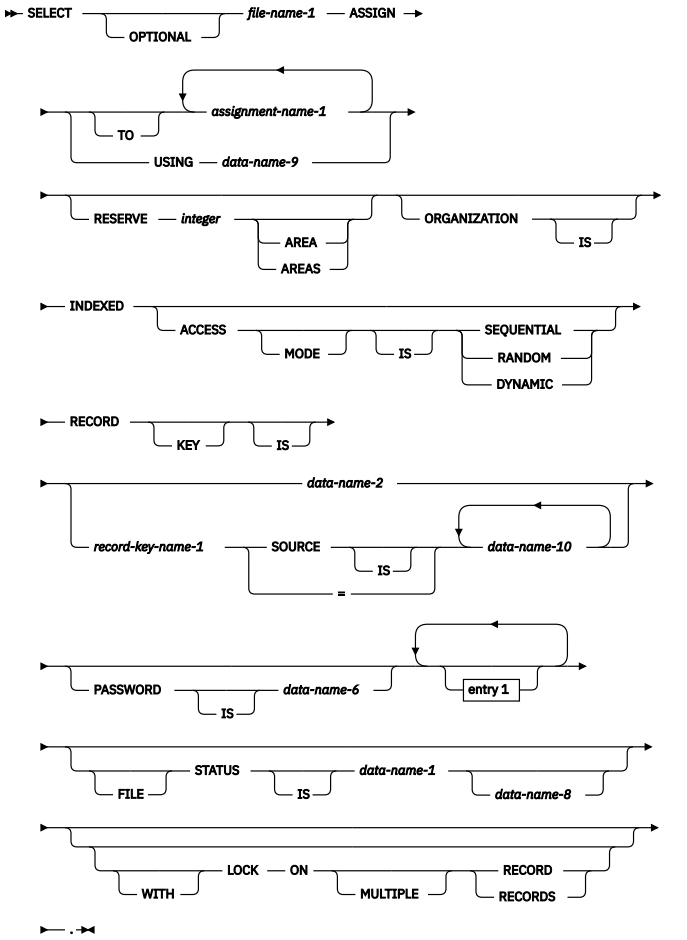
各項目内では、SELECT 節が最初でなければなりません。 その他の節の順序は任意です。

下線は、割り当て名 *1* の外部ファイル名コンポーネントで使用できます。

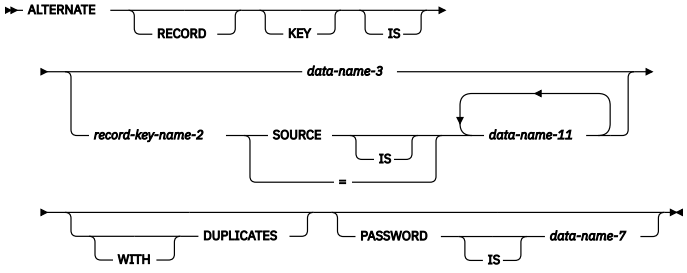
フォーマット 1: 順次ファイル制御項目



フォーマット 2: 索引付きファイル制御項目

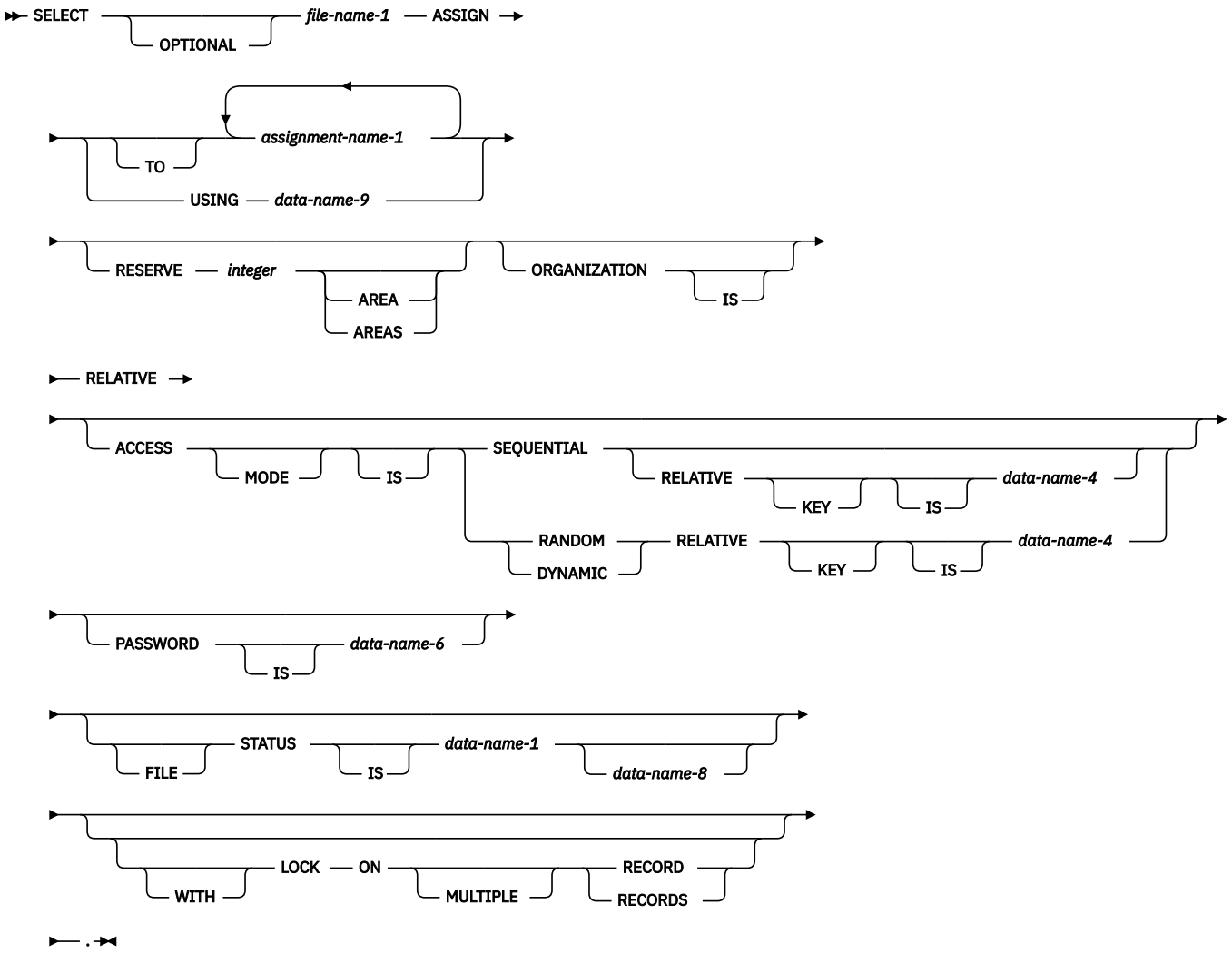


項目 1

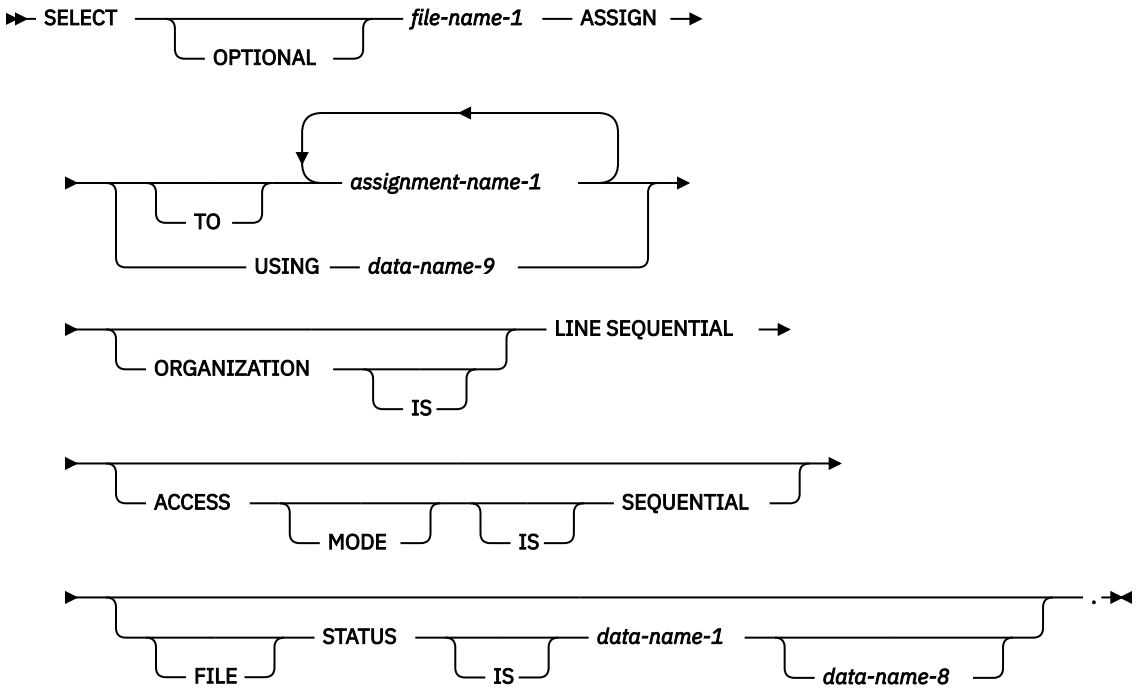


制約事項 : レコード・キー名 は、STL ファイル・システムでのみサポートされます。

フォーマット 3: 相対ファイル制御項目



フォーマット 4: 行順次ファイル制御項目



SELECT 節

SELECT 節は、COBOL プログラムの中で外部ファイルと関連付けるファイルを識別します。

SELECT OPTIONAL
入力、入出力、 または拡張モードでオープンされるファイルに対してのみ指定できます。 オブジェクト・プログラムの実行ごとに、必ずしも使用可能である必要がない入力ファイルについては、SELECT OPTIONAL を指定する必要があります。 詳しくは、338 ページの『OPEN ステートメントに関する注意事項』を参照してください。

ファイル名-*1*
DATA DIVISION の項目 FD または SD と一致している必要があります。 ファイル名は、COBOL ユーザー定義名に関する規則に合致し、 英字を少なくとも 1 文字含み、 このプログラム内で固有である必要があります。

ファイル名-*1* にソート・ファイルまたはマージ・ファイルを指定する場合、 SELECT 節の後には ASSIGN 節だけを記述できます。

ファイル名-*1* により参照されるファイル結合子が外部ファイル結合子である場合は、 このファイル結合子を参照する実行単位内のすべてのファイル制御項目の指定は、OPTIONAL 句と同じでなければなりません。

ASSIGN 節

ASSIGN 節は、COBOL プログラム内の内部ファイル名をシステム・ファイル名と関連付けます。

割り当て名-*1*

ユーザー定義語または英数字リテラルとして指定できます。 どちらの形式も最大 3 つのコンポーネントで構成されます。コンポーネントはハイフンで分離します。 左から右:

1. オプションのコメント・ストリング
2. オプションのファイル・システム ID

3. 割り当て名-*1* がユーザー定義語として指定された場合は、必須の外部ファイル名、割り当て名-*1* がリテラルとして指定された場合は、必須のシステム・ファイル名

割り当て名-*1* の外部ファイル名コンポーネントの決定方法の詳細については、109 ページの『ユーザー定義語およびリテラルの割り当て名』を参照してください。

ユーザー定義語
割り当て名-*1* は、COBOL ワードの規則に従っている必要があり、最大で 1 バイト文字を 30 個分の長さにすることができます。 ユーザー定義語は、予約語と同じにすることはできませんが、プログラム内のその他のユーザー定義語と同じにすることができます。 データ項目の名前を含みます。 外部ファイル名コンポーネントは、実行時に最初は環境変数の名前として扱われ、次に、環境変数が設定されていない場合、または空ストリングとして設定されている場合に、コンポーネントが直接システム・ファイル名として扱われます。

- **外部ファイル名:** COBOL ファイルが開かれる度に、外部ファイル名が環境変数の名前として使用されます。 環境変数が空以外の値として設定されている場合は、値はシステム・ファイル名として扱われるか、またはファイル・システム ID が先頭に付くシステム・ファイル名として扱われます。 詳しくは、112 ページの『データ名および環境変数の割り当て名』を参照してください。 ファイル・システム ID がユーザー定義語内に指定されていて、環境変数内にも指定されている場合は、環境変数が優先されます。 ファイル・システムの優先順位の詳細については、*COBOL for Linux on x86* プログラミング・ガイド内のファイル・システム決定の優先順位を参照してください。
- **システム・ファイル名:** 外部ファイル名によって示される環境変数を設定しない場合、ユーザー定義語は次のいずれかとして扱われます。
  - システム・ファイル名
  - ファイル・システム ID が前に付くシステム・ファイル名
  - コメントに続いてファイル・システム ID が前に付くシステム・ファイル名

  詳細については、109 ページの『ユーザー定義語およびリテラルの割り当て名』を参照してください。

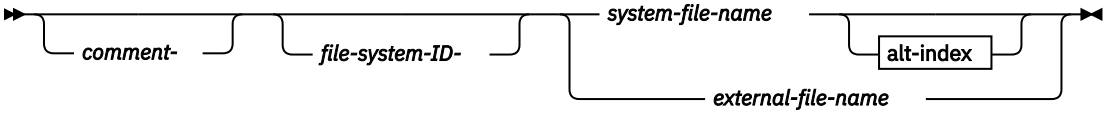
リテラル
割り当て名-*1* は、COBOL 英数字リテラルの規則。 リテラル区切り文字内に指定するすべての文字は、マッピングなしに使用されます。 環境変数の存在に対する確認はランタイムで行われません。 詳しくは、109 ページの『ユーザー定義語およびリテラルの割り当て名』を参照してください。

USING データ名-9
作業用ストレージ・セクションに英数字カテゴリーのデータ項目として定義する必要がありますが、ファイル名-*1* のファイル記述に従属しないようにしてください。 データ名-*9* の内容は、ファイルが開かれる度に割り当て情報を決定するために評価されます。 環境変数の値については、割り当て情報はシステム・ファイル名として扱われるか、またはファイル・システム ID が先頭に付くシステム・ファイル名として扱われます。 詳しくは、112 ページの『データ名および環境変数の割り当て名』を参照してください。

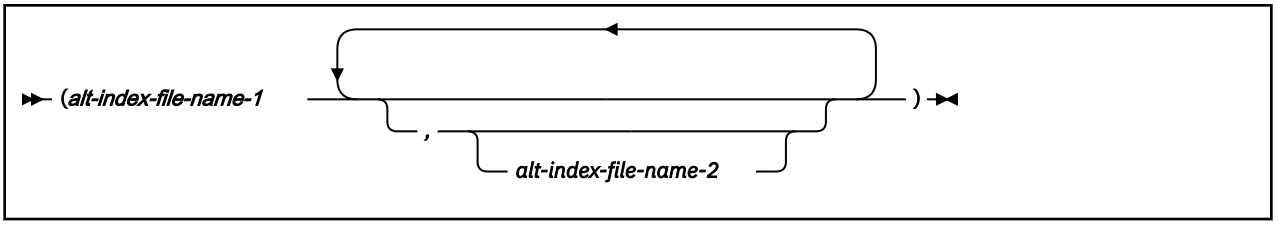
ユーザー定義語およびリテラルの割り当て名

USING が指定されていない場合、割り当て名に指定されたユーザー定義語またはリテラルは、次に説明されているように処理されます。

フォーマット: 割り当て名



代替索引



ユーザー定義語がハイフンを含んでいない場合、ストリング全体が外部ファイル名として扱われます。 リテラルがハイフンを含んでいない場合、ストリング全体がファイル名情報として扱われます。 ファイル名情報は、単一のシステム・ファイル名、代替索引ファイル名のリストが続く単一のシステム・ファイル名、またはコロンで分離されたシステム・ファイル名の連結にすることができます。 いずれを選んだ場合も、各システム・ファイル名はパス名で完全修飾できます。 それ以外の場合は、ユーザー定義語またはリテラルがハイフンで分離された最大 3 つのコンポーネントに分割されます。

コメント
ファイル・システム ID が指定されている場合はその左側にある文字が、また有効なファイル・システム ID がない場合は外部ファイル名またはシステム・ファイル名の左側にある文字が、コメントとして解釈されます。

ファイル・システム *ID*

ファイルが保管され、ファイルへのアクセス時に経由するファイル・システムを指定する 3 文字のストリング。

右端から 2 番目のコンポーネントが次のすべての条件を満たす場合は、ストリングの最初の 3 文字が大文字に変換され、ファイル・システム ID として解釈されます。

- ストリングは 3 つ以上の文字で構成されます。
- 最初 (左端) の 3 文字は、英数字 (A から Z、a から z、または 0 から 9) です。
- 先頭文字は、英字 (A から Z または a から z) です。

ストリングがこれらの基準のすべてを満たさない場合は、コメントの一部として扱われます。

外部ファイル名またはファイル名情報
右端コンポーネントは外部ファイル名か、またはファイル名情報の 3 つの形式のうちの 1 つです。

ハイフンが分離文字として使用されているため、1 つ以上のハイフンを含む外部ファイル名またはシステム・ファイル名はユーザー定義語またはリテラルで直接指定することはできません。1 つ以上のハイフンを含む外部ファイル名またはシステム・ファイル名を指定するには、次のいずれかの方法を使用します。

- 外部ファイル名で示される環境変数をランタイムに適切な値に設定します。
- ASSIGN 節の USING データ名-*9* 形式を指定し、ファイルを開く前に適切な値をデータ名-*9* に移動します。

割り当て名-*1* がユーザー定義語として指定されている場合、外部ファイル名はランタイムに空以外の値に設定された環境変数の名前として解釈され、そのような環境変数がない場合は、直接システム・ファイル名として解釈されます。

割り当て名-*1* がリテラルとして指定されている場合、右端コンポーネントは常にファイル名情報として直接解釈されます。

例えば、次のように指定します。

表 *8.* **ユーザー定義語の割り当て名**

ユーザー定義語	コメント	ファイル・システム ID	外部ファイル名
Read-Only-STL-Orders	Read-Only	STL	ORDERS
This-is-my-file	This-is-my	VSAM (デフォルト = STL)	FILE
Comment-STL--file	Comment-STL-	VSAM (デフォルト = STL)	FILE

表 8. **ユーザー定義語の割り当て名** (続き)			
ユーザー定義語	**コメント**	**ファイル・システム ID**	**外部ファイル名**
Watch-for-this-file	Watch-for	THI (無効)	FILE

表 9. **リテラルの割り当て名**			
リテラル	**コメント**	**ファイル・システム ID**	**システム・ファイル名**
Read-Only-STL-Orders	Read-Only	STL	Orders
Eh?-What's this?	Eh?	VSAM (デフォルト = STL)	What's this?
This-is-a-Db2-CICS.FILE	This-is-a	Db2	FILE (スキーマは CICS)
I-Like-STL!-	I-Like	STL	(NULL)
vsa-/.:/cics/sfs/svr/W123	(なし)	SFS	W123 (SFS サーバー svr の)

代替索引の指定: コンパイラーは通常、適切な代替索引ファイル名を割り当てます。ただし、次に対して代替索引ファイル名を提供する必要があります。

- 代替索引を持つ索引付き SFS ファイル。 各索引ファイル名は、次の形式です。

```
/.:/cics/sfs/sfsServer/base-file-name;index-file-name
```

- COBOL コンパイラー・デフォルト以外の名前の代替索引ファイルを持つ索引付き SdU ファイル。例えば、ファイルが PL/I など異なる言語で作成されている場合など。

代替索引ファイル名を指定する場合、ソース・プログラムで代替レコード・キーが指定されるのと同じ順序で指定する必要があります。代替索引ファイル名は省略できますが、その他の代替索引ファイル名はファイル定義での位置に対応している必要があります。 以下の例は、最初と 3 番目の代替索引ファイル名を指定する方法を示しています。

```
base-file-name(first-index-file-name,,third-index-file-name)
```

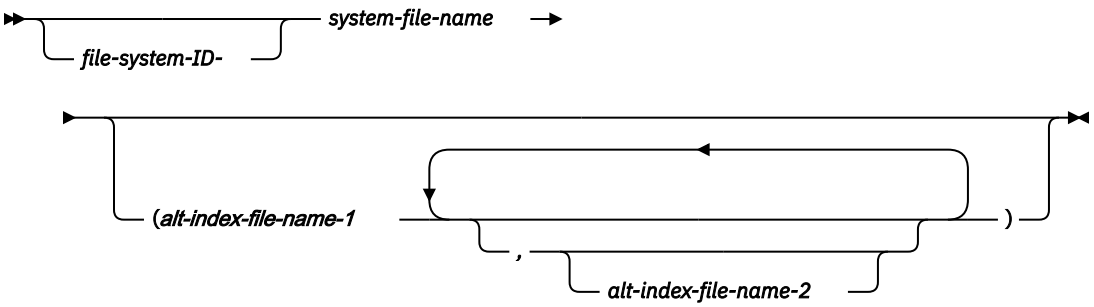
この例では、コンパイラーは 2 番目の代替索引ファイルにデフォルトのファイル名を割り当てます。

SdU または SFS 以外のファイル・システムの場合、代替索引ファイル名は無視されます。

データ名および環境変数の割り当て名

割り当て名に環境変数またはデータ名を指定した場合、データ名または環境変数の値は次のように処理されます。

フォーマット: 環境変数およびデータ名値



データ名-*9* または環境変数内の値は、対応する COBOL ファイルの OPEN ステートメントが実行されるたびに評価されます。 値がハイフンを含んでいない場合、ストリング全体がファイル名情報として扱われます。 ファイル名情報は、単一のシステム・ファイル名、代替索引ファイル名のリストが続く単一のシステム・ファイル名、またはコロンで分離されたシステム・ファイル名の連結にすることができます。いずれを選んだ場合も、各システム・ファイル名はパス名で完全修飾できます。値が 1 つ以上のハイフンと左端のハイフンの後に少なくとも 1 つの文字を含んでいて、左端ハイフンの左にあるストリングが次の条件のすべてを満たしている場合、値はファイル・システム-*ID* の後にファイル名情報が続くと解釈されます。

- ストリングは 3 つ以上の文字で構成されます。
- 最初 (左端) の 3 文字は、英数字 (A から Z、a から z、または 0 から 9) です。
- 先頭文字は、英字 (A から Z または a から z) です。

ファイル・システム *ID*
ファイル・システム *ID* の最初の 3 文字が大文字に変換され、ファイル・システム ID として解釈されて、実行時の妥当性検査の対象となります。 ランタイムに指定されるファイル・システム ID の値は、割り当て名-*1* のユーザー定義語内で指定される任意のファイル・システム ID をオーバーライドします。

ファイル名情報
ファイル名情報の 3 つの形式のうちの 1 つ。 代替索引の指定については、109 ページの『ユーザー定義語およびリテラルの割り当て名』の『代替索引の指定』を参照してください。.

次の表の例は、実行時オプション FILESYS が STL に設定されていて、ファイル・システム ID が STL にデフォルト設定されているとみなします。

表 *10.* **データ名および環境変数の割り当て名**

環境変数値またはデータ名値	ファイル・システム ID	システム・ファイル名
my-fyle	STL	my-fyle
abc/my-fyle	ABC (無効)	fyle
payroll-fyle	PAY (無効)	fyle
rsd-payroll-file	RSD	payroll-file
¥¥Strange (...) name!	STL	¥Strange (...) name!
Db2-File1	Db2	File1
./Db2-File2	STL	./Db2-File2
sfs-/.:/cics/sfs/svr/F1	SFS	F1 (SFS サーバー svr の)

表 *10.* **データ名および環境変数の割り当て名** *(続き)*		
環境変数値またはデータ名値	**ファイル・システム ID**	**システム・ファイル名**
`STL-/.:/cics/sfs/svr/F1`	SFS	(無効なパス)
`stl-file1:file2`	STL	`file1` および `file2` (連結)

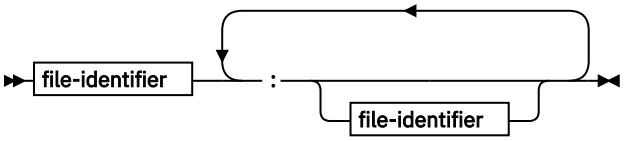
ファイルの識別の詳細については、*COBOL for Linux on x86* プログラミング・ガイド内のファイルの識別を参照してください。

ファイルの概念および用語の詳細については、*COBOL for Linux on x86* プログラミング・ガイド内のファイルの概念と用語を参照してください。

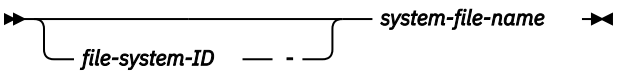
ファイル連結

COBOL for Linux は、内部ファイルへのファイルの連結の割り当てをサポートします (SELECT 節内のファイル名-*1*)。 ファイル連結は、コロン (:) で区切られた複数のファイル ID によって指定されます。

フォーマット: 連結ファイル ID



ファイル ID



ファイル・システム *ID*
ファイル・システム *ID* が少なくとも 3 つの英数字で構成され、最初の文字が英字の場合は、システム・ファイル名が存在するファイル・システム。 それ以外の場合は、ファイル ID 全体がデフォルトのファイル・システム内のファイルの名前として解釈されます。

システム・ファイル名
ファイル・システム *ID* が指定されている場合は、指定されたファイル・システム内のファイルの名前です。それ以外の場合は、ファイル ID 全体がデフォルトのファイル・システム内のファイルの名前として解釈されます。 システム・ファイル名は、そのファイルのドライブおよびパス情報を含むことができます。

次に例を示します。

```
export MYFILE='STL-/home/user1/file1:STL-/home/user1/file2'
```

この例では、`MYFILE` が `/home/user1/file1` および `/home/user1/file2` の (両方とも STL ファイル・システム内にある) ファイル連結の結果です。

ファイル ID は 2 つの分離したファイルの連結として解釈されるため、ファイル ID は、コロン (:) を含むことはできません。

最大 256 個のファイル ID を 1 つの連結に指定することができます。 ファイル ID は固有である必要はありません。 例えば、連結は、同じファイル ID の 256 個のオカレンスで構成することが可能です。 隣接するコロンは、単一のコロンとして扱われます。

ファイル連結は、環境変数値 ASSIGN USING データ項目内容、および割り当て名のリテラル形式に対してサポートされています。

ファイルの連結に割り当てられた COBOL 内部ファイルは、以下の基準を満たす必要があります。

- FILE ORGANIZATION が SEQUENTIAL または LINE SEQUENTIAL である。
- ACCESS MODE が SEQUENTIAL である。
- 任意の OPEN ステートメントのモードが INPUT である。

これらの基準は、連結が割り当て名のリテラル形式で指定されている場合でも、ランタイムで検査されます。

コロンが単一ファイル ID の前または後に付く場合は、ファイルは検証なし連結のメンバーになり、ファイルの権限に関わらず、ファイルを使用するどのプログラム内でも読み取り専用となります。

連結は、すべてのファイル・システムでサポートされています。 ファイル・システム ID は、指定した連結内の任意のまたはすべてのファイル ID に指定することができます。 ただし、連結内のすべてのファイル ID は、ランタイムで同じファイル・システムに指定またはデフォルト設定されている必要があり、ファイルは整合性のある属性を持っていなければなりません。

連結は、個々の世代別ファイルまたは世代別データ・グループ (GDG) 全体を含むことができます。 GDG が連結のメンバーとして指定されている場合は、グループ内の個別のファイルが世代の順序、つまり、最も現行の世代から最も古い世代へ順に読み取られます。 世代別データ・グループまたは世代別ファイルの詳細については、*COBOL for Linux on x86* プログラミング・ガイド内の世代別データ・グループを参照してください。

連結は、空のファイル、つまりレコードを含まないファイルを含めることができます。 空のファイルが読み取られると、ただちに EOF を内部で返すため、連結ファイルを読み取るプログラムには不可視です。 連結が空のファイルのみを含んでいる場合は、最初の READ 操作が EOF を返し、AT END 条件が存在します。

COBOL ファイルが SELECT 節内の OPTIONAL 句で定義されている場合は、連結は使用不可のファイルを含むことができます。 ファイルは、存在しないかまたはファイル権限が要求された操作を許可しない場合に使用不可になります。 使用不可ファイルは、空のファイルのように扱われ、プログラムには不可視です。

ファイルの可用性の詳細については、338 ページの表 51 を参照してください。

ファイルの連結の詳細については、*COBOL for Linux on x86* プログラミング・ガイド内のファイルの連結を参照してください。

RESERVE 節

RESERVE 節は、構文チェックされますが、プログラムの実行には何も影響しません。

ORGANIZATION 節

ORGANIZATION 節は、ファイルの論理構造を指定します。 論理構造は、ファイルが作成された時点に確定され、それ以降は変更できません。

データのいろいろな編成方法や、データ検索の際のいろいろなアクセス方式については、118 ページの『ファイル編成とアクセス・モード』で説明します。.

ORGANIZATION IS SEQUENTIAL (フォーマット 1)
ファイル内のレコード間の先行後続関係は、 レコードが作成または拡張される時点でそれがファイルに入れられる順番によって決まります。

ORGANIZATION IS INDEXED (フォーマット 2)
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更新される索引によって決まります。 索引は、ファイルの各レコード内の埋め込みキーに基づいています。

ORGANIZATION IS RELATIVE (フォーマット 3)
ファイル内の各論理レコードの位置は、 その相対レコード番号によって決まります。

ORGANIZATION IS LINE SEQUENTIAL (フォーマット 4)
ファイル内のレコード間の先行後続関係は、レコードが作成または拡張される時点でそれがファイルに入れられる順番によって決まります。 LINE SEQUENTIAL ファイル内のレコードは、印刷可能文字からのみ構成することができます。

ORGANIZATION 節を省略すると、 コンパイラーは ORGANIZATION IS SEQUENTIAL とみなします。

SELECT 節の中でファイル名-*1* により参照されるファイル結合子が外部ファイル結合子である場合、このファイル結合子を参照する実行単位の中のすべてのファイル制御項目には、 同じ編成を指定しなければなりません。

ファイル編成

ファイルの作成時にデータの編成を決めます。 一度ファイルを作成すると、ファイルを拡張することはできますが、その編成を変更することはできません。

順次編成

ファイルの中にレコードが配置される物理的な順番が、レコードのシーケンスを決定します。 ファイルの拡張はできますが、ファイルの中のレコードの相互関係は変更されません。 レコードは固定長または可変長のどちらも可能であり、キーはありません。

ファイルの中で、最初のレコード以外のレコードには、その先行レコードがそれぞれ一意に決まります。 また最後のレコード以外のレコードには、その後続レコードがそれぞれ一意に決まります。

索引編成

ファイル中の各レコードには、1 つ以上の組み込みキー (キー・データ項目 として参照される) があり、 それぞれのキーは索引に関連しています。 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。 索引付きファイルは、 直接アクセス・ストレージのファイルでなければなりません。 レコードは固定長でも可変長でも可能です。

索引付きファイルの各レコードには、 基本キー・データ項目が埋め込まれていなければなりません。 レコードの挿入、更新、または削除が実行される場合、 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。 したがって、基本キー・データ項目の値はそれぞれ固有な値でなければならず、 レコードの更新時にその値を変更してはなりません。 この基本キー・データ項目の名前は、ファイル制御段落の RECORD KEY 節によって COBOL プログラムに知らせます。

さらに索引付きファイルの各レコードには、 1 つ以上の埋め込み代替キー・データ項目を指定できます。 各代替キーは、検索するレコードを識別する別の方法を提供します。 この代替キー・データ項目の名前は、ファイル制御段落の ALTERNATE RECORD KEY 節によって COBOL プログラムに知らせます。

特定の入出力要求に使用されるキーは、参照キー と呼ばれます。

相対編成

これは、ファイルをそれぞれに 1 つのレコードが含まれているレコード域のストリングとみなします。 それぞれのレコード域は、相対レコード番号で識別されます。 その相対レコード番号に基づいて、アクセス方式によりレコードが格納され検索されます。 例えば、最初のレコード域は相対レコード番号 1 でアドレスが指定され、10 番目のレコードは相対レコード番号 10 でアドレスが指定されます。 ファイルの中でレコードが配置される物理的なシーケンスは、各レコードが入れられるレコード域とは関係がなく、したがって、各レコードの相対レコード番号とも関係がありません。 相対ファイルは、直接アクセス・ファイルでなければなりません。 レコードは固定長でも可変長でも可能です。

行順次編成

行順次ファイルでは、各レコードには、レコード区切り文字で終わる一連の文字が入っています。 区切り文字はレコードの長さには数えられません。

レコードの書き込み時には、レコード区切り文字を追加する前に末尾ブランクがあれば 除去されます。 レコード域に入っている最初の文字から追加された レコード区切り文字までを含む文字が 1 つのレコードとしてファイルに書き込まれます。

レコードの読み取り時には、以下の条件が発生するまで一度に 1 文字ずつ文字がレコード域に読み取られます。

- 最初のレコード区切り文字が検出される。 レコード区切り文字は破棄され、レコードの残りにはスペースが埋められます。
- レコード域全体が文字で満たされる。 最初の未読文字がレコード区切り文字の場合は、 その文字は破棄されます。 レコード区切り文字でない場合、最初の未読文字は、 次の READ ステートメントによって読み取られる最初の文字となります。
- ファイルの終わりが検出されました。 レコード領域の残りにはスペースが埋められます。

レコード長は、以下のいずれかによって決まります。

- 要求された読み取りの長さ
- 改行文字 (¥*n*) の位置

行順次ファイル内の唯一の特殊文字は改行文字 (¥*n*) です。 NULL (0x00) を含め、その他の文字はすべて有効であり、読み取りまたは書き込みエラーの原因にはなりません。

行順次ファイルに書き込まれるレコードは、USAGE DISPLAY や DISPLAY-1 または DISPLAY と DISPLAY-1 項目の 組み合わせとして記述するデータ項目から構成されている必要があります。 CHAR(EBCDIC) コンパイラー・オプションが有効である場合、データ項目の USAGE 節に NATIVE 句が存在するか、しないかによって、DISPLAY または DISPLAY-1 の項目を ASCII か EBCDIC のいずれかでエンコードできます。 ゾーン 10 進データ項目は、符号なしであるか、符号付きの場合は SEPARATE CHARACTER 句で宣言する必要があります。

行順次ファイルには、印刷可能文字と、制御文字を入れることができます。 ただし、ファイルに改行 (LF) 文字 (X'0A') が含まれている場合、その改行文字は、レコード区切り文字として機能します。

行順次ファイルにおいては、次に示す節ではサポートされません。

- APPLY WRITE-ONLY 節
- CODE-SET 節
- DATA RECORDS 節
- LABEL RECORDS 節
- LINAGE 節
- OPEN ステートメントの I-O 句
- PADDING CHARACTER 節
- RECORD CONTAINS 0 節
- RECORD CONTAINS 節フォーマット 2 (例えば、RECORD CONTAINS 100 to 200 CHARACTERS)
- RECORD DELIMITER 節
- RECORDING MODE 節
- RERUN 節
- RESERVE 節
- OPEN ステートメントの REVERSED 句
- REWRITE ステートメント
- ファイル記述項目の VALUE OF 節
- WRITE ... AFTER ADVANCING 簡略名
- WRITE ... AT END-OF-PAGE
- WRITE ... BEFORE ADVANCING

コメントとして扱われる言語エレメント

その他のファイル (順次、相対、および索引付き) の場合、以下の言語エレメントは構文チェックされますが、プログラムの実行には何も影響しません。

- APPLY WRITE-ONLY 節
- CLOSE ... FOR REMOVAL

- CLOSE ... WITH NO REWIND
- CODE-SET 節
- DATA RECORDS 節
- LABEL RECORDS 節
- MULTIPLE FILE TAPE 節
- OPEN ... REVERSE
- PADDING CHARACTER 節
- PASSWORD 節
- RECORD CONTAINS 0 節
- RECORD DELIMITER 節
- RECORDING MODE 節 (相対ファイルおよび索引付きファイルの場合)
- RERUN 節
- RESERVE 節
- SAME AREA 節
- SAME SORT AREA 節
- SAME SORT-MERGE AREA 節
- ファイル記述項目の VALUE OF 節

エラー・メッセージは生成されません (LABEL RECORDS、USE ... AFTER ... LABEL PROCEDURE、および GO TO MORE-LABELS 節のデータ名オプションを除きます)。

PADDING CHARACTER 節

PADDING CHARACTER 節は、順次ファイルでブロック埋め込みのために使用される文字を指定します。

データ名-*5*
英字、英数字、または国別カテゴリーの 1 文字のデータ項目として、DATA DIVISION に定義しなければなりません。また FILE SECTION に定義してはなりません。 データ名-*5* は修飾することができます。

リテラル-*2*
1 文字の英数字リテラルまたは国別リテラルでなければなりません。

外部ファイルで、データ名-*5* が指定されている場合、これは外部のデータ項目を参照しなければなりません。

PADDING CHARACTER 節は構文チェックされますが、プログラムの実行には何も影響しません。

RECORD DELIMITER 節

RECORD DELIMITER 節は、外部メディア上にある可変長レコードの長さを決定する方法を指定します。 これは可変長レコードの場合にのみ指定できます。

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 ワードを指定できます。

RECORD DELIMITER 節は構文チェックされますが、プログラムの実行には何も影響しません。

ACCESS MODE 節

ACCESS MODE 節は、ファイル内のレコード処理できるようにする際の方法を定義します。 ACCESS MODE 文節を指定しない場合には、順次アクセスとみなされます。

相対ファイルの順次アクセスの場合、 ACCESS MODE 節を RELATIVE KEY 節の前に置く必要はありません。

ACCESS MODE IS SEQUENTIAL
すべてのフォーマットで指定できます。

フォーマット 1: 順次
ファイルの中のレコードは、ファイルが作成または拡張された時点で確立されたシーケンスでアクセスされます。 フォーマット 1 は、順次アクセスのみをサポートします。

フォーマット 2: 索引付き
ファイルの中のレコードは、 ファイルの照合シーケンスに従ってレコード・キー値の昇順にアクセスされます。

フォーマット 3: 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。

フォーマット 4: 行順次
ファイルの中のレコードは、ファイルが作成または拡張された時点で確立されたシーケンスでアクセスされます。 フォーマット 4 は、順次アクセスのみをサポートします。

ACCESS MODE IS RANDOM
フォーマット 2 とフォーマット 3 でのみ指定できます。

フォーマット 2: 索引付き
レコード・キー・データ項目の中に入っている値が、 アクセスするレコードを指定します。

フォーマット 3: 相対
相対キー・データ項目の中に入っている値が、 アクセスするレコードを指定します。

ACCESS MODE IS DYNAMIC
フォーマット 2 とフォーマット 3 でのみ指定できます。

フォーマット 2: 索引付き
使用された特定の入出力ステートメントのフォーマットに応じて、ファイルの中のレコードを順次にまたはランダムにアクセスすることができます。

フォーマット 3: 相対
特定の入出力要求のフォーマットに応じて、ファイルの中のレコードを順次にまたはランダムにアクセスすることができます。

ファイル編成とアクセス・モード

ファイル編成 は、ファイルの永続的な論理構造です。 アクセス・モード (順次、ランダム、または動的) を指定することによって、 ファイルからレコードを取り出す方法をコンピューターに指示します。

アクセス方式とデータ編成の詳細については、104 ページの表 7 を参照してください。

順次編成のデータは、順次でのみアクセスできます。 ただし、索引編成または 相対編成のデータは、3 つのアクセス・モードの いずれでもアクセスできます。

アクセス・モード

以下のタイプのアクセス・モードについての説明を確認します。

順次アクセス・モード
このモードでは、 ファイルのレコードを順次に読み取ったり書き込んだりすることができます。 参照される順序は、ファイル内でのレコードの位置によって暗黙に規定されます。

ランダム・アクセス・モード
このモードでは、 プログラマーの指定した方法でレコードの読み書きを実行できます。 ファイルの参照を連続して実行する場合の制御は、 ユーザーがそのために定義したキーにより指定されます。

動的アクセス・モード
このモードでは、 特定の入出力ステートメントによってアクセス・モードを決めることができます。 そのため、レコードは順次またはランダム、あるいはその両方で処理できます。

外部ファイルの場合、外部ファイルと関連付けられている実行単位の中のすべてのファイル制御項目は、同じアクセス・モードを 指定していなければなりません。 さらに、相対ファイル項目では、データ名-*4* は

外部データ項目を参照しなければならず、関連付けられた 各ファイル制御項目内の RELATIVE KEY 句は、同じ外部データ項目を参照しなければなりません。

データ編成とアクセス・モードの関係

このセクションでは、データ編成の各タイプごとに有効なアクセス・モードについて説明します。

順次ファイル
順次編成のファイルは、順次的な方法でのみアクセスできます。 レコードがアクセスされる順序は、最初にレコードが作成された順序になります。

行順次ファイル
順次ファイル (上記) の場合と同じです。

索引付きファイル
3 種類のアクセス・モードがすべて使用できます。

順次アクセス・モードでのレコードのアクセス順は、レコードのキー値の昇順 (または必要によっては降順) です。 代替レコード・キーの値が重複しているレコードの集合における検索順序は、 レコードがその集合に書き込まれた順序になります。

ランダム・アクセス・モードでは、レコードにアクセスする順番を制御できます。 特定のレコードには、RECORD KEY データ項目 (および ALTERNATE RECORD KEY データ項目) の中でそのレコードのキーの値を 指定することによってアクセスします。 あるレコードの集合の代替レコード・キー値が重複している場合は、最初に書き込まれたレコードだけにアクセスできます。

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用して、順次アクセスからランダム・アクセスへと必要に応じて 変更することができます。

相対ファイル
3 種類のアクセス・モードがすべて使用できます。

順次アクセス・モードでのレコードのアクセス順は、ファイル内に存在するすべてのレコードの相対レコード番号の昇順 (または必要によっては降順) です。

ランダム・アクセス・モードでは、レコードにアクセスする順番を制御できます。 特定のレコードが、RELATIVE KEY データ項目に相対レコード番号を入れることによってアクセスされます。 ファイルのレコード記述項目内で RELATIVE KEY を定義することはできません。

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用して、順次アクセスからランダム・アクセスへと必要に応じて 変更することができます。

RECORD KEY 節

RECORD KEY 節 (フォーマット 2) は、索引付きファイルの基本 RECORD KEY であるレコード内のデータ項目を指定します。 基本 RECORD KEY データ項目に含まれる値は、そのファイル内のレコード間で固有でなければなりません。

データ名-2
基本 RECORD KEY データ項目。

データ名-2 は、そのファイルに関連付けられているレコード記述項目の中で記述する必要があります。キーは、以下のデータ・カテゴリーのいずれかにすることができます。

- 英数字
- 数字
- 数字編集 (USAGE DISPLAY または NATIONAL)
- 英数字編集
- 英字
- 外部浮動小数点 (USAGE DISPLAY または NATIONAL)
- 内部浮動小数点
- DBCS

- 国別
- 国別編集

キー・データ項目のカテゴリーには関係なく、キーは英数字項目として扱われます。 キーの照合順序は、レコードの位置決め、または そのファイルに関連付けられているファイル位置標識の設定にキーが使用されたときの項目の 2 進値の順序によって決まります。

データ名-*2* をウィンドウ表示日付フィールドにすることはできません。

データ名-*2* は、可変長データ項目を指していてはなりません。データ名-*2* は修飾することができます。

索引ファイルに可変長レコードが含まれる場合、データ名-*2* は、そのファイルで指定されている最小レコード・サイズ内に含まれている必要はありません。 すなわち、データ名-*2* はレコードの最小レコード・サイズを 超えることも可能ですが、これはお勧めできません。 データ名-*2* は、そのレコードの最初の *n* バイト (*n* はファイルに指定された最小レコード・サイズと等価) に入っていなければなりません。 データ名-*2* はレコードの最小レコード・サイズを超えることも可能ですが、これはお勧めできません。詳しくは、152 ページの『RECORD 節』を参照してください。

データ名-*2* のデータ記述とそのレコード内での相対位置は、ファイルが定義されたときに使用されたものと同じでなければなりません。

ファイルが 2 つ以上のレコード記述項目を持っている場合、データ名-*2* は、それらのレコード記述項目のうちの 1 つにだけ記述されている必要があります。 いずれかのレコード記述項目の中でデータ名-*2* によって参照される同じ文字位置は、そのファイルの他のすべてのレコード記述項目でも、キーとして暗黙のうちに参照されます。

EXTERNAL 節によって定義されたファイルの場合、そのファイルと関連付けられた実行単位内のすべてのファイル記述項目は、レコード内で同じ相対位置を同じ長さで指定するデータ名-*2* のデータ記述項目を持っている必要があります。

レコード・キー名-*1*

レコード・キー名-*1* には、データ名-*10* のクラスおよびカテゴリーがあります。

制約事項 : レコード・キー名 は、STL ファイル・システムでのみサポートされます。

データ名-*10*

データ名-*10* は、そのファイルに関連付けられているレコード記述項目の中で記述する必要があります。キーは、以下のデータ・カテゴリーのいずれかにすることができます。

- 英数字
- 数字
- 数字編集 (USAGE DISPLAY または NATIONAL)
- 英数字編集
- 英字
- 外部浮動小数点 (USAGE DISPLAY または NATIONAL)
- 内部浮動小数点
- DBCS
- 国別
- 国別編集

キー・データ項目のカテゴリーには関係なく、キーは英数字項目として扱われます。 キーの照合順序は、レコードの位置決め、または そのファイルに関連付けられているファイル位置標識の設定にキーが使用されたときの項目の 2 進値の順序によって決まります。

データ名-*10* のオカレンスはすべて、同じカテゴリーでなければなりません。

データ名-*10* をウィンドウ表示日付フィールドにすることはできません。

データ名-*10* は、可変長データ項目を指していてはなりません。データ名-*10* は修飾することができます。

索引付きファイルに可変長レコードが入っている場合、データ名-*10* は、そのレコードの最初の *n* バイト (*n* は、ファイルに指定されている最小レコード・サイズに等しい) に入っていなければなりません。詳しくは、152 ページの『RECORD 節』を参照してください。

SELECT 節内でファイル名-*1* によって参照されるファイル結合子が外部ファイル結合子である場合、このファイル結合子を参照する実行単位内のファイル制御項目はすべて、データ名-*2* および各データ名-*10* の同一データ記述項目と、関連レコード内のそれらの相対位置を持っていなければなりません。 外部ファイル結合子について詳しくは、166 ページの『EXTERNAL 節』を参照してください。

ALTERNATE RECORD KEY 節

ALTERNATE RECORD KEY 節 (フォーマット 2) は、索引付きファイル内のデータへの代替パスを提供するレコード内のデータ項目を指定します。

データ名-*3*
ALTERNATE RECORD KEY データ項目。

データ名-*3* は、そのファイルに関連付けられているレコード記述項目の中で記述する必要があります。キーは、以下のデータ・カテゴリーのいずれかにすることができます。

- 英数字
- 数字
- 数字編集 (USAGE DISPLAY または NATIONAL)
- 英数字編集
- 英字
- 外部浮動小数点 (USAGE DISPLAY または NATIONAL)
- 内部浮動小数点
- DBCS
- 国別
- 国別編集

キー・データ項目のカテゴリーには関係なく、キーは英数字項目として扱われます。 キーの照合順序は、レコードの位置決め、または そのファイルに関連付けられているファイル位置標識の設定にキーが使用されたときの項目の 2 進値の順序によって決まります。

データ名-*3* をウィンドウ表示日付フィールドにすることはできません。

データ名-*3* は、可変オカレンス・データ項目を含むグループ項目を参照することはできません。 データ名-*3* は修飾することができます。

データ名-*3* は、以下のデータ項目を参照してはいけません。

- 可変長データ項目。
- 左端のバイト位置が、基本レコード・キーまたは別の代替レコード・キーの左端のバイト位置に対応している項目。 いずれかのキーが SOURCE 句を使用して指定されている場合、この制限は適用されません。

索引ファイルに可変長レコードが含まれる場合、データ名-*3* は、そのファイルで指定されている最小レコード・サイズ内に含まれている必要はありません。 すなわち、データ名-*3* はレコードの最小レコード・サイズを 超えることも可能ですが、これはお勧めできません。

索引ファイルに可変長レコードが含まれる場合、データ名-*3* は、そのファイルで指定されている最小レコード・サイズ内に含まれている必要はありません。 すなわち、データ名-*3* はレコードの最小レコード・サイズを 超えることも可能ですが、これはお勧めできません。

索引付きファイルに可変長レコードが含まれている場合、データ名-*3* は、そのレコードの最初の *x* バイト (*x* はファイルに指定された最小レコード・サイズと等価) に入っていなければなりません。 データ名-*3* はレコードの最小レコード・サイズを超えることも可能ですが、これはお勧めできません。詳しくは、152 ページの『RECORD 節』を参照してください。

データ名-*3* のデータ記述とそのレコード内での相対位置は、 ファイルが定義されたときに使用されたものと同じでなければなりません。 ファイルの代替レコード・キーの個数も、 ファイルの作成時に使用された個数と同じでなければなりません。

データ名-*3* の左端の文字位置は、基本レコード・キーまたは別の代替レコード・キーの左端の文字位置と同一であってはなりません。

レコード・キー名-*2*

レコード・キー名-*2* には、データ名-*11* のクラスおよびカテゴリーがあります。

制約事項 : レコード・キー名 は、STL ファイル・システムでのみサポートされます。

データ名-*11*

データ名-*11* は、そのファイルに関連付けられているレコード記述項目の中で記述する必要があります。キーは、以下のデータ・カテゴリーのいずれかにすることができます。

- 英数字
- 数字
- 数字編集 (USAGE DISPLAY または NATIONAL)
- 英数字編集
- 英字
- 外部浮動小数点 (USAGE DISPLAY または NATIONAL)
- 内部浮動小数点
- DBCS
- 国別
- 国別編集

キー・データ項目のカテゴリーには関係なく、キーは英数字項目として扱われます。 キーの照合順序は、レコードの位置決め、または そのファイルに関連付けられているファイル位置標識の設定にキーが使用されたときの項目の 2 進値の順序によって決まります。

データ名-*11* のオカレンスはすべて、同じカテゴリーでなければなりません。

データ名-*11* をウィンドウ表示日付フィールドにすることはできません。

データ名-*11* は、可変長データ項目を指していてはなりません。データ名-*11* は修飾することができます。

索引付きファイルに可変長レコードが入っている場合、データ名-*11* は、そのレコードの最初の *x* バイト (*x* は、ファイルに指定されている最小レコード・サイズに等しい) に入っていなければなりません。詳しくは、152 ページの『RECORD 節』を参照してください。

DUPLICATES 句が指定されていない場合、ALTERNATE RECORD KEY データ項目の中に含まれる値は、ファイルの中のレコードの間で固有でなければなりません。

DUPLICATES 句が指定されている場合、 ALTERNATE RECORD KEY データ項目の中に含まれる値は、 ファイルの中のレコードの間で重複が可能です。 順次アクセスにおいて、キーの重複したレコードは、 ファイルの中にそれらのレコードが配置されている順に取り出されます。 ランダム・アクセスでキーの重複している一連のレコードの場合は、 そのうち最初に記述されたレコードしか取り出せません。

EXTERNAL 節によって定義されたファイルの場合、そのファイルと関連付けられた実行単位内のすべてのファイル記述項目は、レコード内で同じ相対位置を同じ長さで指定するデータ名-*3* のデータ記述項目を持っている必要があります。 ファイル記述項目は、同数の代替レコード・キーおよび同じ DUPLICATES 句を指定する必要があります。

SELECT 節内でファイル名-*1* によって参照されるファイル結合子が外部ファイル結合子である場合、このファイル結合子を参照する実行単位内のファイル制御項目はすべて、データ名-*3* および各データ名-*11* の同一データ記述項目と、関連レコード内のそれらの相対位置、同数の代替レコード・キー、および同一 DUPLICATES 句を持っていなければなりません。 外部ファイル結合子について詳しくは、166 ページの『EXTERNAL 節』を参照してください。

RELATIVE KEY 節

RELATIVE KEY 節 (フォーマット 3) 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。

データ名-*4*

これは、その記述に PICTURE 記号 P が含まれない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、この相対ファイルに関連したレコード記述項目で定義してはなりません。つまり、RELATIVE KEY はレコードの一部ではありません。 データ名-*4* は修飾することができます。

データ名-*4* をウィンドウ表示日付フィールドにすることはできません。

START ステートメントを使用する場合のみ、 ACCESS IS SEQUENTIAL についてデータ名-*4* は必須です。 ACCESS IS RANDOM と ACCESS IS DYNAMIC が使用されている場合、データ-4 は必ず指定しなければなりません。 START ステートメントが実行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開始すべきレコードを判別します。

データ名-*4* に値が指定されており、 START ステートメントが実行されていない場合は、その値は無視され、 処理はそのファイルの最初のレコードから開始されます。

相対ファイルを START ステートメントによって参照するときは、 そのファイルに対して RELATIVE KEY 節を指定する必要があります。

外部ファイルでは、データ名-*4* は外部データ項目を参照しなければならず、関連する各ファイル制御項目内の RELATIVE KEY 句は、それぞれその同じ外部データ項目を参照しなければなりません。

ACCESS MODE IS RANDOM 節は、SORT ステートメントや MERGE ステートメントの USING 句または GIVING 句の中で指定された ファイル名に対しては使用できません。

PASSWORD 節

PASSWORD 節は、構文チェックされますが、プログラムの実行には何も影響しません。

FILE STATUS 節

FILE STATUS 節は、ファイルに対するそれぞれの入出力操作の実行を監視します。

FILE STATUS 節を指定すると、このファイルを明示的または暗黙のうちに参照する入出力操作が実行されるたびに、システムはファイル状況キー・データ項目の中に値を入れます。 その値はそのステートメントの実行状況を示すものです。 (「状況キー」については、269 ページの『共通の処理機能』を参照。)

データ名-*1*

ファイル状況キー・データ項目は、WORKING-STORAGE SECTION、LOCAL-STORAGE SECTION、または LINKAGE SECTION で、以下の項目のいずれかとして定義することができます。

- 英数字カテゴリーの 2 文字のデータ項目
- 国別カテゴリーの 2 文字のデータ項目
- USAGE DISPLAY または NATIONAL で、数字カテゴリーの 2 桁のデータ項目 (外部 10 進データ項目)

データ名-*1* に、PICTURE 記号 'P' を含めることはできません。

データ名-*1* は修飾することができます。

ファイル状況キー・データ項目は、任意の位置に指定してはなりません。 すなわち、そのデータ項目の後に OCCURS DEPENDING ON 節を含むデータ項目があってはなりません。

データ名-*8*

ファイル・システムから戻される情報を表わします。 定義はファイル・システムおよびプラットフォームに固有であるため、データ名-*8* の特定の値に依存するアプリケーションは、プラットフォームをまたいで移植可能にならない可能性があります。

データ名-*8* の定義方法は、使用しているファイル・システムによって決まります。

ファイル・システム LSQ、QSAM、RSD、SFS、STL

データ名-*8* は、PICTURE 9(6) および USAGE DISPLAY の属性で定義する必要があります。 ただし、PICTURE X(*n*) で追加フィールドを定義できます。 ファイル・システムがフィードバック値を定義し、それらのフィードバック値は、100000 未満の場合には先行ゼロがある、6 桁の外部 10 進表記に変換されます。PICTURE X(*n*) を使用して追加フィールドを定義すると、ゼロ以外のフィードバック・コードを説明する追加情報が X(*n*) に含まれます。 (ほとんどのプログラムの場合、完全なメッセージ・テキストを表示するのに、 *n* の値は 100 で十分です。)

Db2 ファイル・システム

データ名 *8* を次のようなグループ項目として定義する必要があります。

```
01 FileStatus2.
  02 FS2-SQLCODE COMP PICTURE S9(9).
  02 FS2-SQLSTATE PICTURE X(5).
```

FS2-SQLCODE および FS2-SQLSTATE 内のランタイム値は、以前に完了した操作の SQL フィードバック情報を表します。

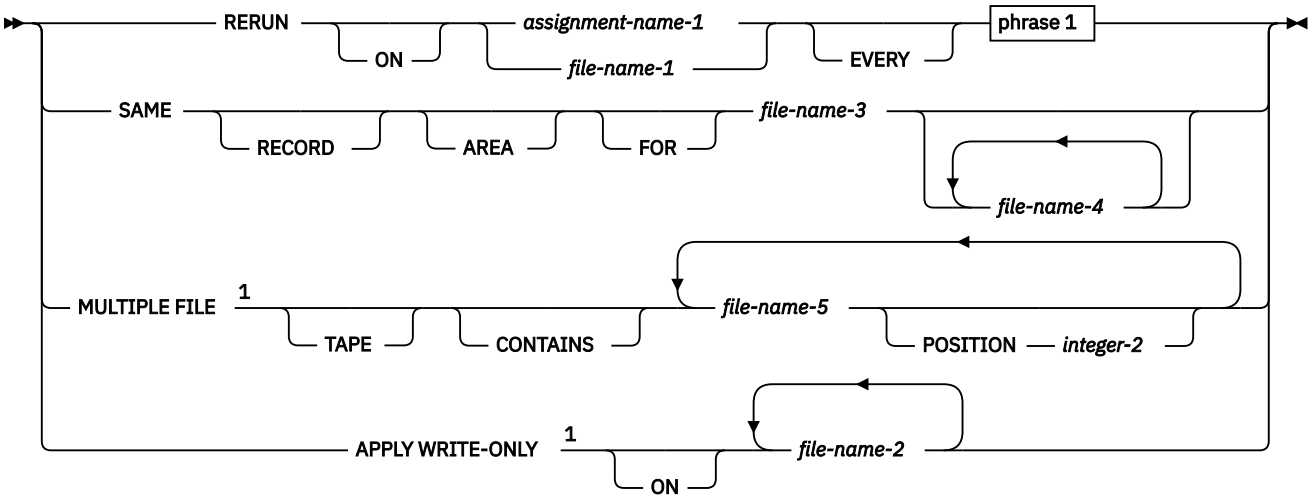
I-O-CONTROL 段落

入出力セクションの I-O-CONTROL 段落は、チェックポイントをいつ取るかを指定し、またさまざまなファイルが共用するストレージ域を指定します。 この段落は、COBOL プログラムではオプショナルです。

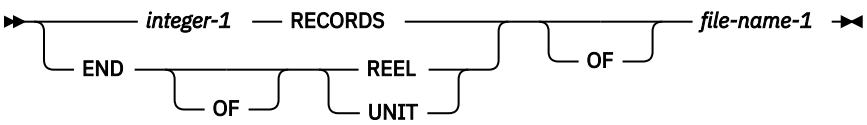
キーワード I-O-CONTROL は、この段落の冒頭に一度だけ使用することができます。 I-O-CONTROL というワードは、領域 A で開始し、分離文字ピリオドを後に付けなければなりません。

I-O-CONTROL 段落の中に節を記述する場合、その順序は任意です。 I-O-CONTROL 段落は、分離文字ピリオドによって終わります。

フォーマット: 順次 I-O-CONTROL 項目



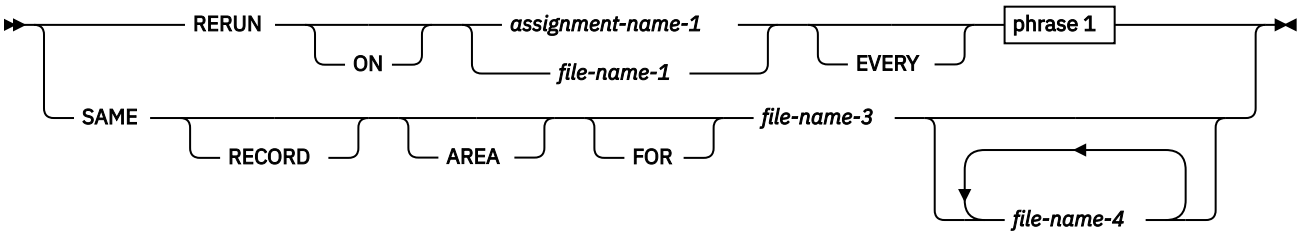
句 1



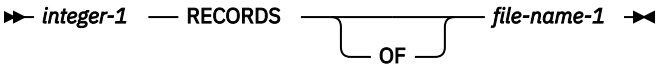
注:

[1] MULTIPLE FILE 節および APPLY WRITE-ONLY 節は、 構文チェックされますが、プログラムの実行には何も影響しません。

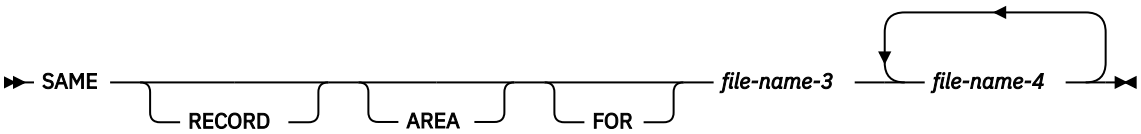
フォーマット: 相対および索引付き I-O-CONTROL 項目



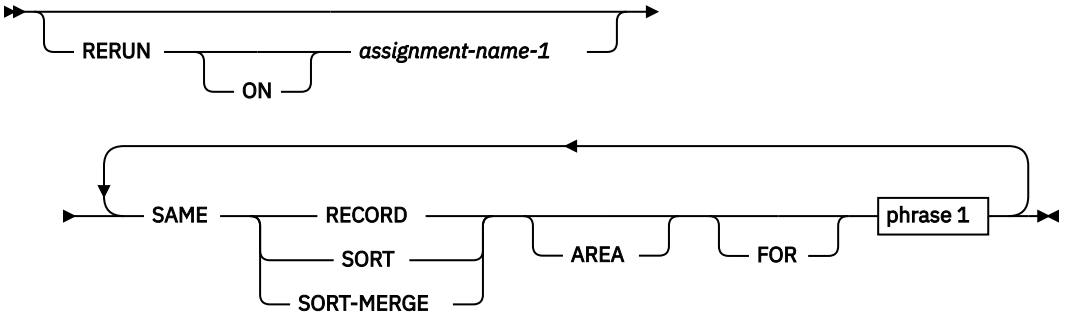
句 1



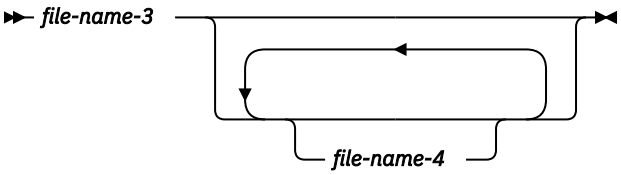
フォーマット: 行順次 I-O-CONTROL 項目



フォーマット: ソート/マージ I-O-CONTROL 項目



句 1



RERUN 節

RERUN 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。 それぞれの句の制約事項に従っていれば、 2 つ以上の RERUN 節を指定することができます。

RERUN 節は構文チェックされますが、プログラムの実行には何も影響しません。

次の場合、RERUN 節を使用しないでください。

- EXTERNAL 節を使用して記述されたファイルの場合
- RECURSIVE 節を指定したプログラム内

ファイル名-1
このファイルは、 順次編成のファイルでなければなりません。

割り当て名-*1*
チェックポイント・ファイル用の外部ファイル。 これは、含まれるプログラムと含んでいるプログラムのプログラム全体を通じて、 ASSIGN 節で指定された割り当て名と同じにすることはできません。

SORT/MERGE の考慮事項:

I-O-CONTROL 段落の中で RERUN 節を指定する場合、 チェックポイント・レコードは、プログラム中の各 SORT ステートメントや MERGE ステートメントの実行中に ソート/マージ・プログラムによって決定される論理的な間隔で書き込まれます。 RERUN 節を省略すると、チェックポイント・レコードは書き込まれません。

プログラム内で SORT/MERGE I-O-CONTROL 段落は 1 つだけ指定することができ、含まれるプログラム内では指定できません。 このフォーマットは、そのプログラム単位内にある SORT ステートメントおよび MERGE ステートメント全体に影響を及ぼします。

EVERY 整数-*1* RECORDS
チェックポイント・レコードは、処理されるファイル名-*1* の中の整数-*1* で指定したレコード数ごとに書き込まれます。

整数-*1* RECORDS 句を複数回指定する場合、 これらのうちの 2 つで同じファイル名-*1* を指定することはできません。

整数-*1* RECORDS 句を指定する場合、割り当て名-*1* を指定する必要があります。

EVERY END OF REEL/UNIT
チェックポイント・レコードは、ファイル名-*1* のボリュームの終わりが発生すると必ず書き込まれます。 用語 REEL と UNIT は、どちらを使用しても同じことです。

END OF REEL/UNIT 句を複数回指定する場合、 これらのうちの 2 つで同じファイル名-*1* を指定することはできません。

END OF REEL/UNIT 句は、ファイル名-*1* が順次編成ファイルである場合にのみ指定できます。

SAME AREA 節

SAME AREA 節は構文チェックされますが、プログラムの実行には何も影響しません。

SAME RECORD AREA 節

SAME RECORD AREA 節は、現在の論理レコードを処理するために 2 つ以上のファイルが同じ主記憶域を使用することを指定します。

SAME RECORD AREA 節で指定されたファイルは、同じ編成やアクセス方式である必要はありません。

ファイル名-3* 、 *ファイル名-4
これらは、同じプログラムのファイル制御段落中に指定しなければなりません。 ファイル名-*3* およびファイル名-*4* は、EXTERNAL 節で定義されたファイルを参照してはなりません。

それらのファイルはすべて同時にオープンすることができます。 共用ストレージ域にある論理レコードは、次のようにみなされます。

- SAME RECORD AREA 節内で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 節内で最後に読み取られた入力ファイルの論理レコード

1 つのプログラムの中に複数の SAME RECORD AREA 節を指定できます。 しかし、以下のことが言えます。

- 複数の SAME RECORD AREA 節の中で特定のファイル名を指定することはできません。
- SAME RECORD AREA 節の中に SAME AREA 節のファイル名が 1 つ以上ある場合は、 その SAME AREA 節のすべてのファイル名がその SAME RECORD AREA 節の中にもなければなりません。 ただし、その SAME AREA 節にないファイル名でも、 その SAME RECORD AREA 節に指定できます。
- SAME AREA 節のファイルは一度に 1 つしかオープンできないという規則は、すべてのファイルを同時にオープンしてもよいという SAME RECORD AREA の規則より優先します。

- SAME RECORD AREA 節が複数のファイルに対して指定される場合、これらのファイルのレコード記述項目またはファイル記述項目に GLOBAL 節を含めることはできません。
- RECORD CONTAINS 0 CHARACTERS 節を指定する場合、 SAME RECORD AREA 節を指定することはできません。

SAME RECORD AREA 節で指定されたファイルは、 同じ編成やアクセス方式である必要はありません。

SAME SORT AREA 節

SAME SORT AREA 節は構文チェックされますが、プログラムの実行には何も影響しません。

ファイル名-3 、 ファイル名-4
これらは、同じプログラムのファイル制御段落中に指定しなければなりません。 ファイル名-*3* および ファイル名-*4* は、EXTERNAL 節で定義されたファイルを参照してはなりません。

SAME SORT AREA 節を指定する場合、少なくとも指定した 1 つのファイル名はソート・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。

- 複数の SAME SORT AREA 節を指定できます。 しかし、1 つのソート・ファイルを 2 つ以上の節の中で指定することはできません。
- ソート・ファイルでないファイルが SAME AREA 節と 1 つ以上の SAME SORT AREA 節の両方で指定されている場合は、 SAME AREA 節のすべてのファイルが SAME SORT AREA 節の中になければなりません。
- SAME SORT AREA 節で指定されたファイルは、 同じ編成やアクセス方式である必要はありません。
- SAME SORT AREA 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 AREA 節または SAME RECORD AREA 節で指定するのでない限り、 相互にストレージを共有することはありません。
- この節の中で指定されたソート・ファイルまたはマージ・ファイルを参照する SORT ステートメントまたは MERGE ステートメントが実行される場合、この節の中で指定されたファイル名に関連付けられた非ソート・ファイルまたは非マージ・ファイルがオープン・モードであってはなりません。

SAME SORT-MERGE AREA 節

SAME SORT-MERGE AREA 節は、SAME SORT AREA 節と同じです 。

詳しくは、127 ページの『SAME SORT AREA 節』を参照してください。.

MULTIPLE FILE TAPE 節

MULTIPLE FILE TAPE 節 (フォーマット 1) は、複数のファイルが物理的に同一のテープ・リールを共用することを指定します。

この節は構文チェックされますが、プログラムの実行には何も影響しません。

APPLY WRITE-ONLY 節

APPLY WRITE-ONLY 節は構文チェックされますが、プログラムの実行には何も影響しません。

第 5 部 データ部

第 15 章 DATA DIVISION の概説

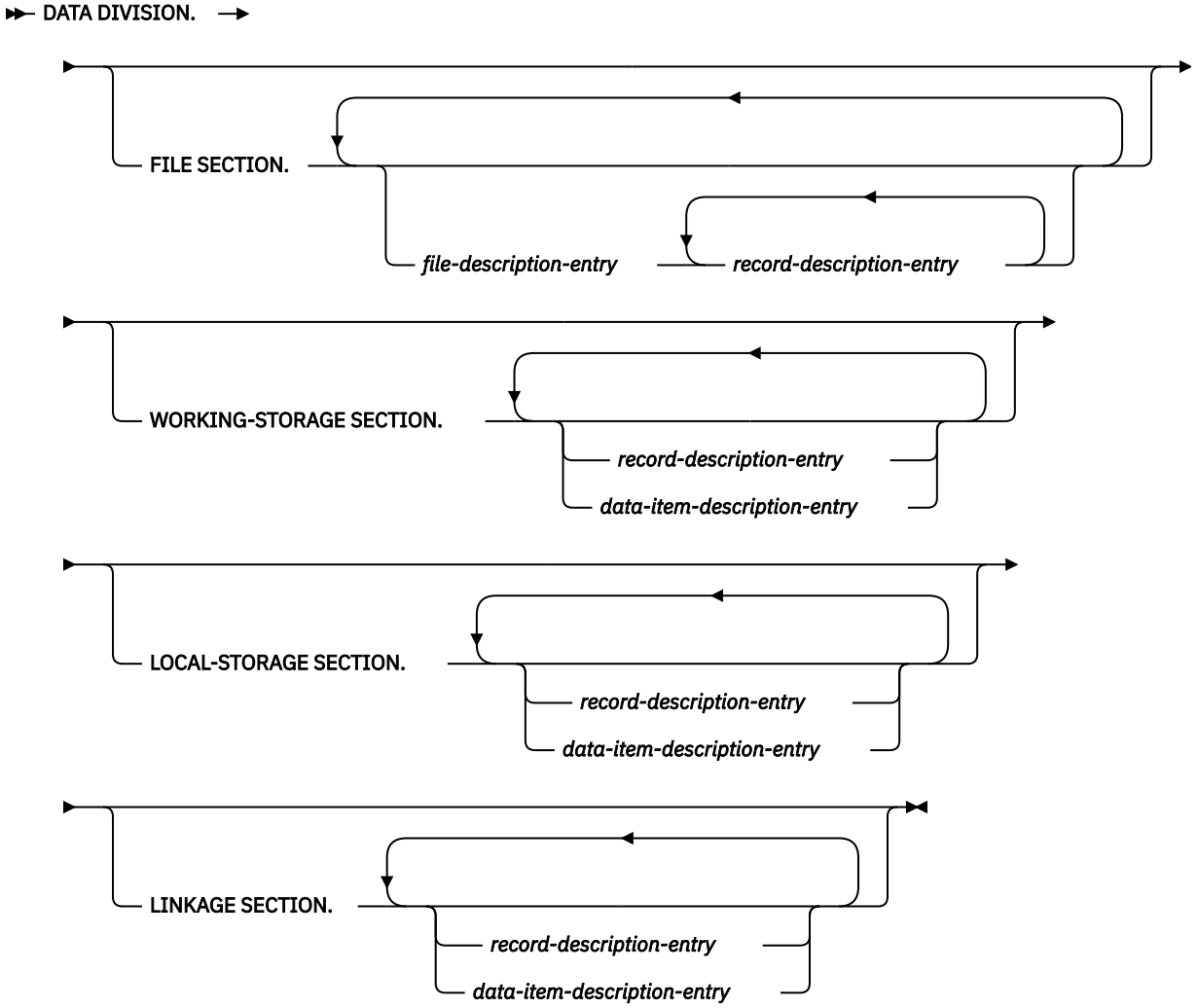
ここでは、プログラムの DATA DIVISION の構造を概説します。

DATA DIVISION のセクションには、COBOL プログラムの中にそれぞれ特定の論理機能があり、その論理機能が不要であればそのセクションは省略できます。セクションを含める場合には、 必ず次に示す順序で記述しなければなりません。 DATA DIVISION はオプショナルです。

プログラム・データ部

COBOL ソース・プログラムの DATA DIVISION は、 プログラムにより処理されるすべてのデータを、特定の構造により記述します。

フォーマット: プログラム・データ部



FILE SECTION

FILE SECTION は、データ・ファイルの構造を定義します。 FILE SECTION は FILE SECTION というヘッダーで開始し、その後に分離文字ピリオドを付けなければなりません。

ファイル記述項目
FILE SECTION の編成の最高レベルを表します。 これは、ファイルの物理構造と ID について情報を提供し、そのファイルに関連付けられるレコード名を示します。 ファイル記述項目の中で必要なフォーマットと節については、145 ページの『第 16 章 DATA DIVISION - ファイル記述項目』を参照してください。

レコード記述項目
ある特定のファイルに含まれている特定のレコードを記述する、または (TYPEDEF 節を使用して) タイプ名を記述する 1 組のデータ記述項目です (157 ページの『第 17 章 DATA DIVISION - データ記述項目』を参照)。

FILE SECTION のレコードは、英数字グループ項目、国別グループ項目、またはクラスが英字、英数字、DBCS、国別、数字の基本データ項目として記述しなければなりません。

レコード記述記入項目は複数指定できます。 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。

FILE SECTION 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ければ、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 SECTION 内で使用できます。

WORKING-STORAGE SECTION

WORKING-STORAGE SECTION は、データ・ファイルの一部ではないが、プログラムによって開発され処理されているデータ・レコードを記述します。また、WORKING-STORAGE SECTION は、ソース・プログラムの中で値が代入され、オブジェクト・プログラムの実行時には値が変わらないデータ項目も記述します。

WORKING-STORAGE SECTION は WORKING-STORAGE SECTION というセクション・ヘッダーで開始し、その後に分離文字ピリオドを付けなければなりません。タイプ名は WORKING-STORAGE SECTION 内で定義できます。

WORKING-STORAGE プログラム
プログラム の場合の WORKING-STORAGE SECTION には、実行単位全体を通して複数のプログラムによって共用される外部データ・レコードを記述することもできます。 FILE SECTION 内のレコード記述に使用されるすべての節は、 VALUE 節と EXTERNAL 節 (これらの節は FILE SECTION の中のレコード記述項目では指定できません) と同じく、WORKING-STORAGE SECTION 内のレコード記述に使用することができます。

WORKING-STORAGE SECTION には、レコード記述項目と、独立データ項目のデータ記述項目 (データ項目記述項目) が含まれています。

レコード記述項目
WORKING-STORAGE SECTION 内にあって、相互に一定の階層関係にあるデータ項目は、レベル番号によって構造化が指定されるレコード群にまとめる必要があります。 詳細については、157 ページの『第 17 章 DATA DIVISION - データ記述項目』を参照してください。

データ項目記述項目
WORKING-STORAGE SECTION にあって相互に階層関係を持たない独立した項目は、 それ以上細分する必要がない限りレコード群にまとめる必要はありません。 その代わりに、それらの項目は独立基本項目として分類および定義されます。 それぞれの項目は、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項目の中で定義されます。 詳細については、157 ページの『第 17 章 DATA DIVISION - データ記述項目』を参照してください。

LOCAL-STORAGE SECTION

LOCAL-STORAGE SECTION は、呼び出しのたびに割り振られ解放されるストレージを定義します。 LOCAL-STORAGE SECTION は、スタック・メモリーを使用します。

呼び出しのたびに、LOCAL-STORAGE SECTION で定義されたデータ項目が再度割り振られます。 VALUE 節を持つ各データ項目は、節で指定された値に初期設定されます。

ネストされたプログラムの場合、LOCAL-STORAGE SECTION で定義されたデータ項目が、最外部プログラムの呼び出しごとに割り振られます。 しかし、データ項目は、ネストされたプログラムが呼び出されるたびに VALUE 節で指定された値に初期設定されます。

LOCAL-STORAGE SECTION で定義されたデータ項目は、EXTERNAL 節を指定することはできません。

LOCAL-STORAGE SECTION は、LOCAL-STORAGE SECTION というヘッダーで開始し、その後に分離文字ピリオドを付ける必要があります。

LOCAL-STORAGE SECTION は、 再帰的プログラムおよび非再帰的プログラムで指定することができます。

LINKAGE SECTION

LINKAGE SECTION は、別のプログラムから利用できるデータを記述します。

レコード記述項目
詳細は、132 ページの『WORKING-STORAGE SECTION』を参照してください。

データ項目記述項目
詳細は、132 ページの『WORKING-STORAGE SECTION』を参照してください。

LINKAGE SECTION のレコード記述項目とデータ項目記述項目は、名前と記述を指定しますが、 データ域は別のところに存在しているため、プログラムの中にストレージは確保されません。タイプ名 は LINKAGE SECTION で定義できます。

LINKAGE SECTION では任意のデータ記述節を利用して項目を記述することができますが、次のような例外があります。

- レベル 88 項目以外の項目に VALUE 節を指定することはできません。
- EXTERNAL 節は指定できません。

LINKAGE SECTION で GLOBAL 節を指定することができます。

データ単位

データは、トピックに示すように、概念的な単位にグループ化されます。

- ファイル・データ
- プログラム・データ

ファイル・データ

ファイル・データは、ファイル内に含まれています。 ファイル とは、いずれかの入出力装置上に存在するデータ・レコードの集まりです。 ファイルは物理レコードのグループとみなすことができます。 また、論理レコードのグループともみなすことができます。 物理レコードと論理レコードの間の関係は、DATA DIVISION で記述します。

詳しくは、150 ページの『FILE SECTION』を参照してください。.

物理レコード は、ストレージへ (またはストレージから) 移動される際に 1 つのエンティティーとして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 は、そのサブディビジョンに論理関係があるデータの単位です。 論理レコードそれ自体が物理レコードとなる場合があります (すなわちデータの 1 物理単位に完全に含まれる)。 複数の論理レコード

が 1 つの物理レコード内に含まれる場合があり、 また 1 つの論理レコードがいくつかの物理レコードにまたがる場合もあります。

ファイル記述項目 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、ラベル付け情報など) を指定します。

レコード記述項目 は、ファイル内の論理レコード (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フォーマット)、データに代入さ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は、 論理レコードだけを使用できるようになります。 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・データは、ファイルから読み取られるのではなく、プログラムによって作られます。

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て、 一連のレコード記述項目によって定義することができます。 そのようにグループ化する必要のない項目は、独立データ記述項目 (データ項目記述項目) の中で 定義することができます。

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、DATA DIVISION の中で、レベル標識とレベル番号を使用する方式で定義します。

レベル標識 は、その記述項目を使用して、プログラム内の各ファイルを識別します。 レベル標識は、それに関連したデータ階層の最上位レベルを表します。 FD はファイル記述レベル標識で、SD はソート/マージ・ファイル記述レベル標識です。

レベル番号 は、その記述項目を使用して、特定のデータの性質を示します。 レベル番号は、データ階層を記述するために使用することが できます。この番号によって、このデータが特殊な目的を持っていることを示すことができます。 また、それらはレベル標識に関連 (またそれに従属) させたり、 独立して使用して内部データや 2 つ以上のプログラムに共通のデータを記述したりできます (レベル番号の規則については、160 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で、それをさらに分割し、より詳しいデータ参照ができます。

例えば、百貨店のカスタマー・ファイルの場合に、1 人の顧客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。 そのレコードはさらに、顧客名、顧客の住所、顧客番号、売上の部門番号、売上の単位数、売上高、前回の収支、およびその他の付属情報、という部分に分けることができるでしょう。

レコードの基本的なサブディビジョン (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から構成される場合と、レコードそれ自体が 1 つの基本項目である場合があります。

基本項目のセットを参照することが必要な場合があります。したがって、基本項目はひとまとめにしてグループ項目 にすることができます。 グループを組み合わせて、 1 つまたは 2 つ以上の小グループを含む、より大きいグループにすることもできます。 したがって、データ項目の 1 つの階層の中で、 1 つの基本項目が 2 つ以上のグループ項目に属することができます。

レベル番号のシステムは、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。 特別な目的に使用されるデータ項目を識別するために、特殊なレベル番号も使用されます。

レコード記述項目の中のデータのレベル

レコード内のグループ項目や基本項目にはそれぞれ別々の項目が必要で、その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 1 桁または 2 桁の整数であり、その値は 01 から 49 の整数か、3 つの特別なレベル番号 (66、77、または 88) のうちの 1 つです。 次に示すレベル番号は、レコードの構造化に使用します。

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 項目は、英数字グループ項目、国別グループ項目、または基本項目のいずれかにすることができます。 レベル番号は、領域 A から開始し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 から 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。 それらは領域 A または領域 B で開始することができます。この系列の中で包括度の低いデータ項目には、高い (必ずしも連続してはいない) レベル番号が割り当てられます。

グループ項目内のレベル番号間の関係は、そのグループ内のデータ階層を定義します。

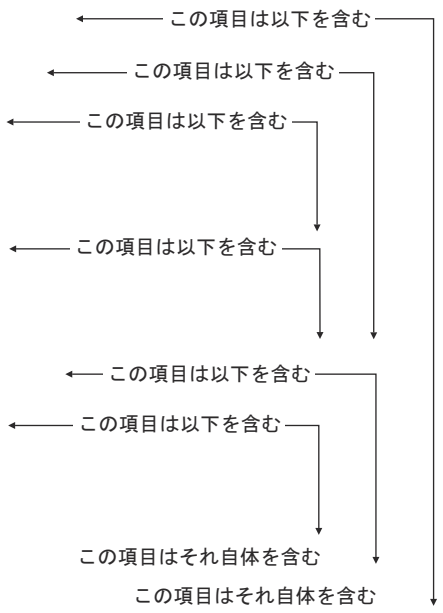
1 つのグループ項目には、 そのグループ項目の後にあるすべてのグループ項目と基本項目のうち、 そのグループのレベル番号以下のレベル番号が現れるまでのものが含まれます。

次の図は、レベル 01 の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であるグループを示しています。

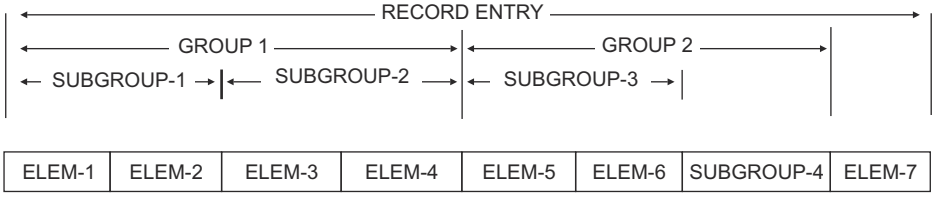
COBOL レコード記述項目は以下のように作成する。

```
01   RECORD-ENTRY.

  05   GROUP-1.

    10   SUBGROUP-1.

      15   ELEM-1 PIC... .
      15   ELEM-2 PIC... .

    10   SUBGROUP-2.

      15   ELEM-3 PIC... .
      15   ELEM-4 PIC... .

  05   GROUP-2.

    15   SUBGROUP-3.

      25 ELEM-5 PIC... .
      25 ELEM-6 PIC... .

    15   SUBGROUP-4 PIC... .

  05   ELEM-7 PIC... .
```

以下に示すように分割される。



レコード記述項目のストレージの配置を以下の図に示す。



また、階層内の同じレベルに対して、レベル番号が異なる従属項目を持つグループを定義することもできます。 例えば、以下の EMPLOYEE-RECORD の 05 EMPLOYEE-NAME と 04 EMPLOYEE-ADDRESS は、階層

内の同じレベルを定義します。 コンパイラーは、MAP 出力に示すように相対的な関係でレベルの再番号付けを行います。

```
01   EMPLOYEE-RECORD.
     05  EMPLOYEE-NAME.
         10  FIRST-NAME PICTURE  X(10).
         10  LAST-NAME  PICTURE  X(10).
     04  EMPLOYEE-ADDRESS.
         08  STREET     PICTURE  X(10).
         08  CITY       PICTURE  X(10).
```

以下のレコード記述項目は、上記のレコード記述項目と同じデータ階層を定義します。

```
01   EMPLOYEE-RECORD.
     02  EMPLOYEE-NAME.
         03  FIRST-NAME PICTURE  X(10).
         03  LAST-NAME  PICTURE  X(10).
     02  EMPLOYEE-ADDRESS.
         03  STREET     PICTURE  X(10).
         03  CITY       PICTURE  X(10).
```

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ます。

特殊なレベル番号

特別なレベル番号は、レコードを構築していない項目を識別します。

特別なレベル番号には次のものがあります。

66
RENAMES 節を含む必要のある項目を識別します。 そのような項目は、それ以前に定義されているデータ項目をグループ化し直します。 (詳細については、203 ページの『RENAMES 節』を参照。)

77
他の項目のサブディビジョンではなく、それ以上分割されないデータ項目記述項目 (WORKING-STORAGE SECTION、LOCAL-STORAGE SECTION、または LINKAGE SECTION の独立した項目) を識別します。 レベル 77 の項目は、領域 A から開始しなければなりません。

88
条件変数の特定の値と結び付けられている条件名項目を識別します。 (詳細については、220 ページの『VALUE 節』を参照。)

プログラムの中で参照される WORKING-STORAGE SECTION、LOCAL-STORAGE SECTION、および LINKAGE SECTION 内にあるレベル 77 とレベル 01 の項目には、固有のデータ名を付けなければなりません。これらはどちらも修飾することができないからです。プログラムの中で参照される従属データ名は、固有なものとして定義するか、あるいは修飾することによって 固有なものにし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。

字下げ

連続するデータ記述項目は、先行する項目と同じ桁から始めたり、レベル番号に応じて字下げしたりできます。

字下げは情報をわかりやすくする点で役立ちますが、字下げをしてもコンパイラーの処置は変わりません。

グループ項目のクラスとカテゴリー

COBOL for Linux には、2 種類のグループ、英数字グループと国別グループがあります。

GROUP-USAGE 節を指定していないグループは、英数字グループです。 英数字グループは、グループ内に含まれている基本データ項目の表現とは無関係に、英数字のクラスおよびカテゴリーを持ち、USAGE DISPLAY であるかのように扱われます。 多くの操作 (移動や比較など) では、データ表現の編集や変換が行われないことを除いては、英数字グループは英数字カテゴリーの基本項目のように扱われます。 その他の操作 (MOVE CORRESPONDING や ADD CORRESPONDING など) では、従属データ項目は別個の基本項目として処理されます。

英数字グループの内容は、CHAR(NATIVE) コンパイラー・オプションを使用した場合、固有 1 バイト文字で表現されるかのように扱われ、CHAR(EBCDIC) コンパイラー・オプションを使用した場合は 1 バイトの EBCDIC 文字として扱われます。

国別グループは、NATIONAL 句を指定した GROUP-USAGE 節によって、グループ・レベルで定義されます。すべての従属データ項目は、明示的または暗黙的に USAGE NATIONAL で記述する必要があり、従属グループは明示的または暗黙的に GROUP-USAGE NATIONAL を指定して定義する必要があります。

別の記述が行われていない限り、国別グループ項目は、USAGE NATIONAL で、クラスおよびカテゴリーが国別の、 PICTURE N(m) で記述されている基本データ項目として処理されます。ここで、m は国別文字位置にあるグループの長さです。 国別グループには国別文字のみが含まれるため、移動および比較では必要に応じてデータが変換されます。 コンパイラーは、適切な切り捨ておよび埋め込みを確実に行います。 その他の操作 (MOVE CORRESPONDING や ADD CORRESPONDING など) では、従属データ項目は別個の基本項目として処理されます。 詳しくは、169 ページの『GROUP-USAGE 節』を参照してください。

下の表は、グループ項目のクラスとカテゴリーを要約したものです。

表 *11.* **グループ項目のクラスとカテゴリー**

グループ記述	**グループのクラス**	**グループのカテゴリー**	**グループ内の基本項目の USAGE**	**グループの USAGE**
GROUP-USAGE 節の指定なし	英数字	英数字 (ただし、グループ内の基本項目はどのカテゴリーでも持つことができる)	任意	USAGE に関係する場合は、DISPLAY として扱われます。
明示的または暗黙的に GROUP-USAGE 節を指定	国別	国別	NATIONAL	NATIONAL

データのクラスとカテゴリー

COBOL プログラムで使用されるほとんどのデータとすべてのリテラルは、クラスとカテゴリーに分けられます。 データ・クラスは、データ・カテゴリーをグループ化したものです。 データ・カテゴリーは、データ記述項目または関数定義の属性によって決定されます。

データ・カテゴリーについて詳しくは、139 ページの『カテゴリーの記述』を参照してください。

以下の基本データ項目には、クラスとカテゴリーがありません。

- 指標データ項目
- USAGE POINTER、USAGE FUNCTION-POINTER、または USAGE PROCEDURE-POINTER

これ以外の基本データ項目のすべてのタイプには、138 ページの表 12 に示すようなクラスとカテゴリーがあります。

関数は、基本データ項目を参照し、(138 ページの表 13 に示すように) その関数のタイプに関連付けられたデータ・クラスとカテゴリーに属します。

リテラルには、139 ページの表 14 に示すようなクラスとカテゴリーがあります。 形象定数 (NULL を除く) には、その使用されている文脈で形象定数によって示されるリテラルまたは値によって決まる、クラスとカテゴリーがあります。 詳しくは、13 ページの『形象定数』を参照してください。

すべてのグループ項目には、それぞれ従属する基本項目が別のクラスおよびカテゴリーに属している場合であっても、クラスとカテゴリーがあります。 グループ項目の種別については、136 ページの『グループ項目のクラスとカテゴリー』を参照してください。

表 *12.* **基本データ項目のクラス、カテゴリー、および *USAGE***		
基本データ項目のクラス [2]	**カテゴリー**	**USAGE**
英字	英字	DISPLAY
英数字	英数字	DISPLAY
	英数字編集	DISPLAY
	数字編集	DISPLAY
ブール [1]	ブール [1]	DISPLAY
日時 [1]	日付 [1] 時刻 [1]	DISPLAY PACKED-DECIMAL
	タイム・スタンプ [1]	DISPLAY
DBCS[1]	DBCS[1]	DISPLAY-1
国別 [1]	国別 [1]	NATIONAL
	国別編集 [1]	NATIONAL
	数字編集 [1]	NATIONAL
数字	数字	DISPLAY (ゾーン 10 進数タイプ)
		NATIONAL (国別 10 進数タイプ)
		PACKED-DECIMAL (内部パック 10 進数タイプ)
		COMP-3 (内部 10 進数タイプ)
		BINARY
		COMP
		COMP-4
		COMP-5
	内部浮動小数点 [1]	COMP-1
		COMP-2
	外部浮動小数点 [1]	DISPLAY
		NATIONAL

1. IBM 拡張
2. すべてのカテゴリーに関して、グループ項目のクラスは英数字です。

表 *13.* **関数のクラスとカテゴリー**	
関数のタイプ	**クラスとカテゴリー**
英数字	英数字
国別	国別

表 *13.* **関数のクラスとカテゴリー** *(続き)*	
関数のタイプ	**クラスとカテゴリー**
整数	数字
数字	数字

表 *14.* **リテラルのクラスとカテゴリー**	
リテラル	**クラスとカテゴリー**
英数字 (16 進形式を含む)	英数字
DBCS	DBCS
国別 (16 進形式を含む)	国別
数字 (固定小数点と浮動小数点)	数字

カテゴリーの記述

データ項目のカテゴリーは、そのデータ記述項目の属性 (その PICTURE 文字ストリングや USAGE 節など) またはその関数定義によって設定されます。

各カテゴリーの意味を以下に示します。

英字

データ項目は、その PICTURE 文字ストリングによって英字カテゴリーとして記述されます。 PICTURE 文字ストリングの詳細については、187 ページの『英字項目』を参照してください。

英字カテゴリーのデータ項目は、英字データ項目として参照されます。

英数字

以下はそれぞれ、英数字カテゴリーのデータ項目です。

- その PICTURE 文字ストリングによって英数字として記述される基本データ項目。 PICTURE 文字ストリングの詳細については、188 ページの『英数字項目』を参照してください。
- 英数字グループ項目
- 英数字関数
- 以下の特殊レジスター
  - DEBUG-ITEM
  - SHIFT-OUT
  - SHIFT-IN
  - SORT-CONTROL
  - SORT-MESSAGE
  - WHEN-COMPILED
  - XML-EVENT

– XML-TEXT

英数字編集

データ項目は、その PICTURE 文字ストリングによって英数字編集カテゴリーとして記述されます。
PICTURE 文字ストリングの詳細については、188 ページの『英数字編集項目』を参照してください。

英数字編集カテゴリーのデータ項目は、英数字編集データ項目として参照されます。

ブール

ブール・データは、表示画面形式および外部記述プリンター・ファイルに関連付けられた標識の値の変更方法および引き渡し方法を提供する IBM 拡張です。 ブール値 0 は標識のオフ 状況であり、ブール値 1 は標識のオン 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、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" と定義されます。

ブール文字は 1 バイトを占めます。

形象定数 ZERO がブール・データ項目またはブール・リテラルに関連付けられている場合、これはブール・リテラル B"0" を表します。

予約語 ALL がブール・リテラルに有効です。

PICTURE 文字ストリングの詳細については、187 ページの『ブール項目』を参照してください。

ブール・カテゴリーのデータ項目は、ブール・データ項目と呼ばれます。

日付、時刻、タイム・スタンプ

データ項目は、FORMAT 節によってカテゴリー日付、時刻、またはタイム・スタンプとして記述されます。
FORMAT 節について詳しくは、166 ページの『FORMAT 節』を参照してください。

DBCS

データ項目は、その PICTURE 文字ストリングおよび NSYMBOL(DBCS) コンパイラー・オプションによって、または明示的な USAGE DISPLAY-1 節によって、DBCS カテゴリーとして記述されます。 PICTURE 文字ストリングの詳細については、188 ページの『DBCS 項目』を参照してください。

DBCS カテゴリーのデータ項目は、DBCS データ項目として参照されます。

外部浮動小数点

データ項目は、その PICTURE 文字ストリングによって外部浮動小数点カテゴリーとして記述されます。
PICTURE 文字ストリングの詳細については、189 ページの『外部浮動小数点項目』を参照してください。
外部浮動小数点データ項目は、USAGE DISPLAY または USAGE NATIONAL で記述できます。

USAGE DISPLAY のときは、項目は display 浮動小数点データ項目として参照されます。

USAGE NATIONAL のときは、項目は国別浮動小数点データ項目として参照されます。

数字クラスの外部浮動小数点データ項目は、特に除外されていない限り、数字データ項目への参照に含まれます。

内部浮動小数点

データ項目は、USAGE 節と COMP-1 または COMP-2 句によって、内部浮動小数点カテゴリーとして記述されます。

内部浮動小数点カテゴリーのデータ項目は、内部浮動小数点データ項目として参照されます。 数字クラスの内部浮動小数点データ項目は、特に除外されていない限り、数字データ項目への参照に含まれます。

国別

以下はそれぞれ、国別カテゴリーのデータ項目です。

- その PICTURE 文字ストリングおよび NSYMBOL(NATIONAL) コンパイラー・オプションによって、または明示的な USAGE NATIONAL 節によって国別カテゴリーとして記述されたデータ項目。 PICTURE 文字ストリングの詳細については、190 ページの『国別項目』を参照してください。
- GROUP-USAGE NATIONAL 節で明示的または暗黙的に記述されたグループ項目
- 国別関数
- 特殊レジスター XML-NTEXT

国別編集

データ項目は、その PICTURE 文字ストリングによって国別編集カテゴリーとして記述されます。
PICTURE 文字ストリングの詳細については、191 ページの『国別編集項目』を参照してください。

国別編集カテゴリーのデータ項目は、国別編集データ項目として参照されます。

数字

以下はそれぞれ、数字カテゴリーのデータ項目です。

- その PICTURE 文字ストリングによって数字として記述されているが、BLANK WHEN ZERO 節を使用して記述されていない基本データ項目。 PICTURE 文字ストリングの詳細については、192 ページの『数字項目』を参照してください。
- 以下のいずれかの USAGE で記述された基本データ項目
  - BINARY、 COMPUTATIONAL、 COMPUTATIONAL-4、 COMPUTATIONAL-5、 COMP、 COMP-4、または COMP-5
  - PACKED-DECIMAL、 COMPUTATIONAL-3、 または COMP-3
- 数字タイプの特殊レジスター
  - LENGTH OF
  - LINAGE-COUNTER
  - RETURN-CODE
  - SORT-CORE-SIZE
  - SORT-FILE-SIZE
  - SORT-MODE-SIZE
  - SORT-RETURN
  - TALLY
  - XML-CODE
- 数字関数
- 整数関数

数字カテゴリーのデータ項目は、数字データ項目として参照されます。

数字編集

以下はそれぞれ、数字編集カテゴリーのデータ項目です。

- その PICTURE 文字ストリングによって数字編集として記述されているデータ項目。 PICTURE 文字ストリングの詳細については、193 ページの『数字編集項目』を参照してください。
- その PICTURE 文字ストリングによって数字として記述され、BLANK WHEN ZERO 節を使用して記述されているデータ項目。

位置合わせの規則

データを基本項目に位置決めするときの標準位置合わせの規則は、受け取り項目のカテゴリーによって異なります。

受け取り項目とはデータの移動先項目のことです。 受け取り項目について詳しくは、328 ページの『基本移動』を参照してください。

数字
数字受け取り項目の場合には、次の規則が適用されます。

1. データは想定小数点位置に合わせられ、 必要なら切り捨てられるか 0 が埋め込まれます。 (想定小数点 とは、 論理的な意味はあるが、データの中に実際の小数点の文字としては存在しない小数点のことです。)
2. 想定小数点が明示的に指定されていない場合、受け取り項目は、 フィールドのすぐ右側に想定小数点が指定されているものとして扱われます。 その上で、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、必要なら右端または左端のいずれかが切り捨てられるか、または 0 が埋め込まれます。ただし、先行するゼロに置き換えられる編集処理の場合は別です。

内部浮動小数点
小数点は、フィールドのすぐ左側にあるものとみなされます。 データは、小数点の次の左端文字位置に合わせられ、 それに応じて指数もそろえられます。

外部浮動小数点
データは、左端文字位置に合わせられ、 それに応じて指数もそろえられます。

英数字、英数字編集、英字、DBCS
これらの受け取り項目の場合には、次の規則が適用されます。

1. データは左端文字位置に合わせられ、必要なら右端が切り捨てられるか、 または右端にスペースが埋め込まれます。
2. この受け取り項目に JUSTIFIED 節が 指定されている場合、上記の規則は、170 ページの『JUSTIFIED 節』に説明されているように修正されます。

国別、国別編集
これらの受け取り項目の場合には、次の規則が適用されます。

1. データは左端文字位置に合わせられ、必要なら右端が切り捨てられるか、 または右端にデフォルトの Unicode スペース (NX'2000') が埋め込まれます。 切り捨ては、 国別文字の境界で行われます。
2. この受け取り項目に JUSTIFIED 節が 指定されている場合、上記の規則は、170 ページの『JUSTIFIED 節』に説明されているように修正されます。

日付、時刻、タイム・スタンプ
これらの受け取り項目の場合には、次の規則が適用されます。

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 が指定されているクラス日時項目に関しては、データは右端の数字位置に合わせられ、(必要であれば) その左側にゼロが埋め込まれます。

文字ストリングと項目のサイズ

PICTURE 節で記述される項目の場合、基本項目のサイズは、PICTURE 文字ストリングと SIGN 節 (該当する場合) に記述された文字位置の数によってソース・コードで記述されます。 ただし、ストレージ・サイズは、その項目が実際に占有するバイト数 (PICTURE 文字ストリング、SIGN IS SEPARATE 節 (該当する場合)、および USAGE 節の組み合わせによって決定) によって決められます。

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 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。

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、国別編集、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 節 (指定されている場合) 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。

内部浮動小数点項目の場合、その USAGE 節によってストレージ内の項目のサイズが決められます。 USAGE COMPUTATIONAL-1 はその項目のために 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。

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へ移動する場合、コンパイラーは先行桁の切り捨てによって、短いほうの項目の PICTURE 文字ストリングで表されている桁数に合わせてデータを切り捨てます。 例えば、送り出しフィールドに PICTURE S99999 と指定されていて、その値が +12345 である場合に、 そのデータが PICTURE S99 と指定された BINARY の受け取りフィールドに移動されるとすると、 そのデータは切り捨てられて +45 になります。 追加情報については、213 ページの『USAGE 節』を参照してください。

TRUNC コンパイラー・オプションは、2 進数字項目に影響を及ぼすことがあります。 TRUNC については、「*COBOL for Linux on x86* プログラミング・ガイド」内の『*TRUNC*』 を 参照してください。

符号付きデータ

COBOL 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

演算符号は、符号付き数字項目に関連したものであり、その代数的な性質を示します。

代数符号の内部表現は、その項目の USAGE 節、SIGN 節 (存在する場合)、および操作環境によって決まります (内部表現について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『例*:* 数値データおよび内部表現』を参照してください。)

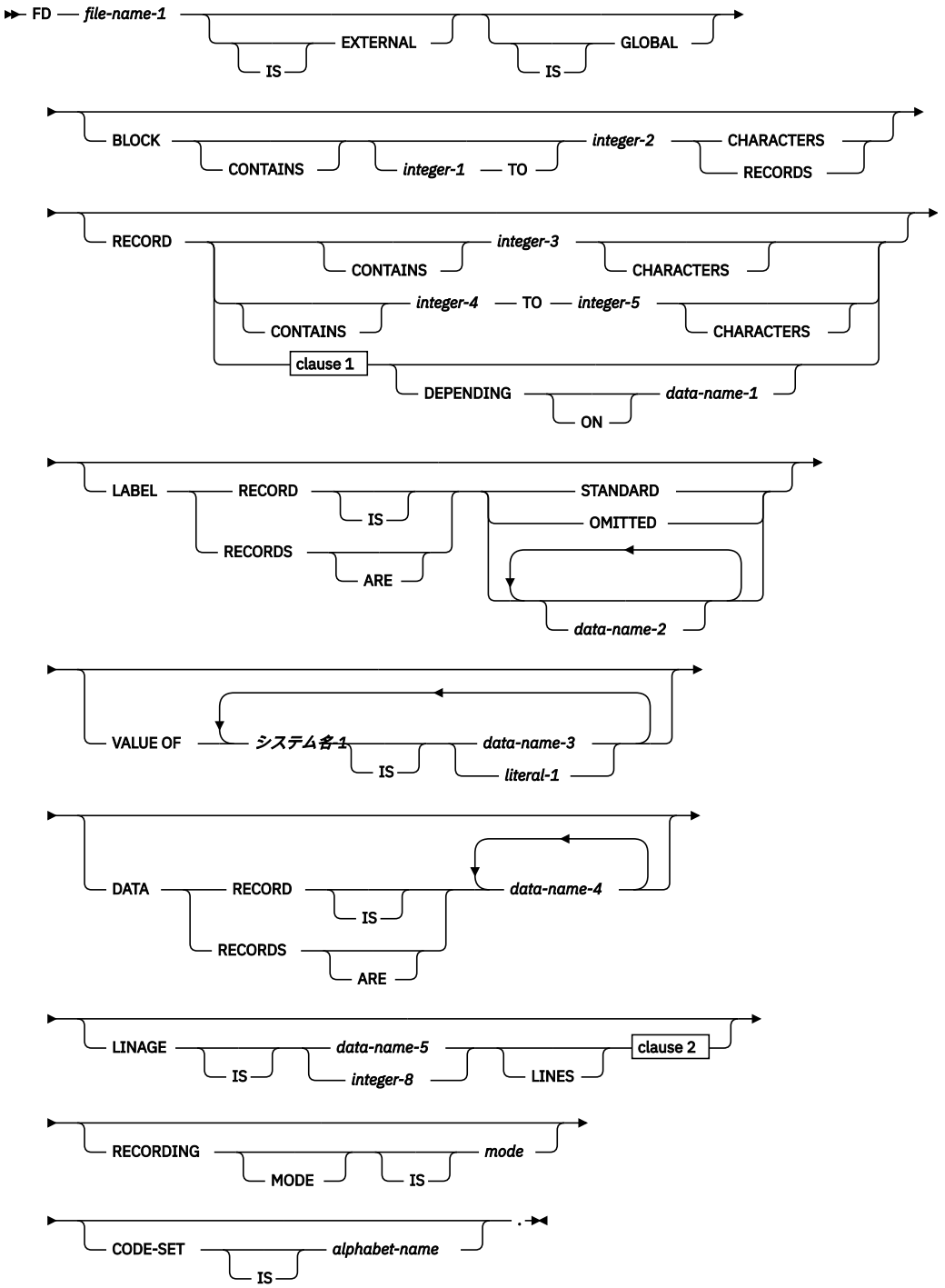
編集符号

編集符号は数字編集項目に関連したものです。 編集符号は、編集済み出力の項目の符号を識別する PICTURE 記号です。

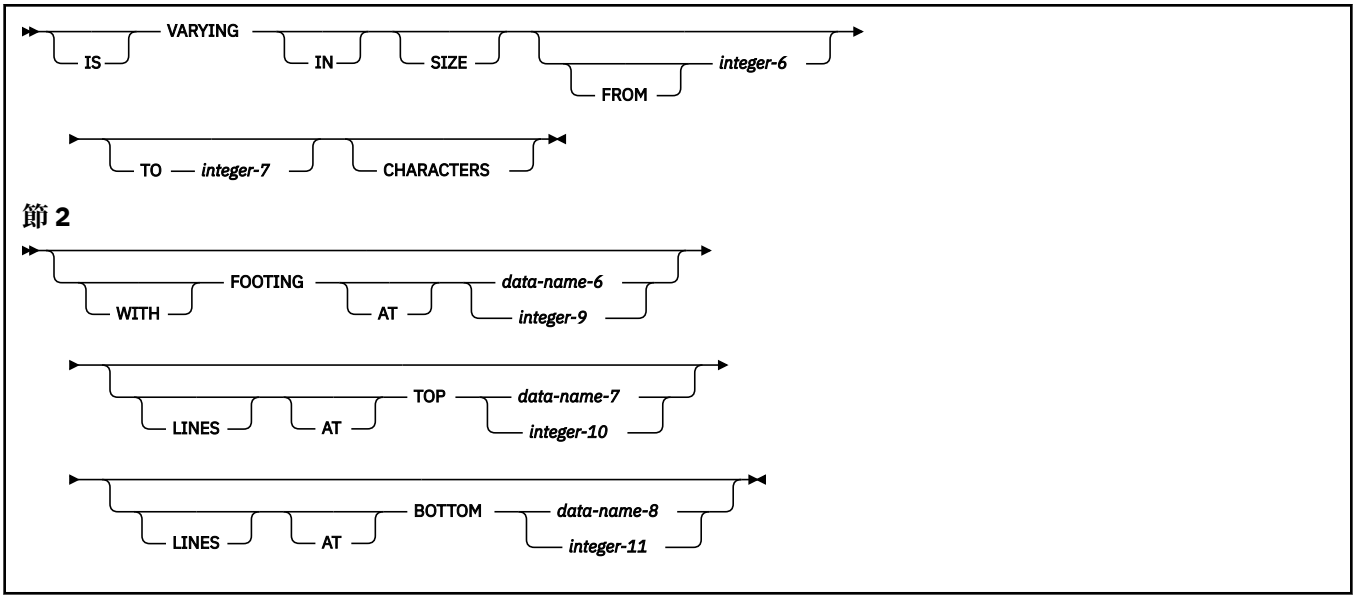
第 16 章 DATA DIVISION - ファイル記述項目

COBOL プログラムで、ファイル記述 *(FD)* 項目 (またはソート/マージ・ファイルの場合 はソート・ファイル記述 *(SD)* 項目) は、FILE SECTION の中の最高レベルの編成を表します。FD 項目や SD 項目の後にオプションの節をどのような順序で指定するかは、 重要なことではありません。

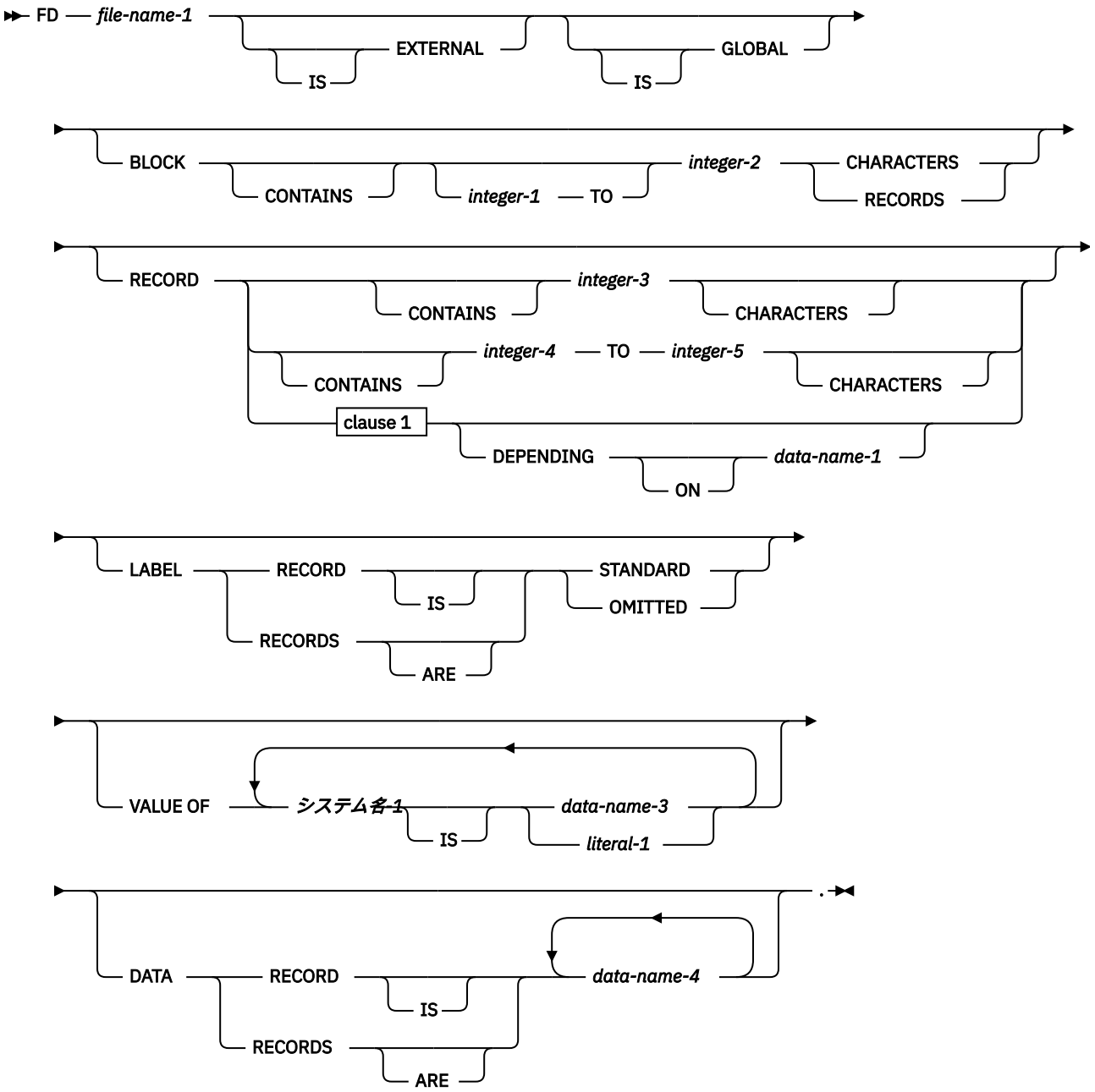
フォーマット 1: 順次ファイル記述項目



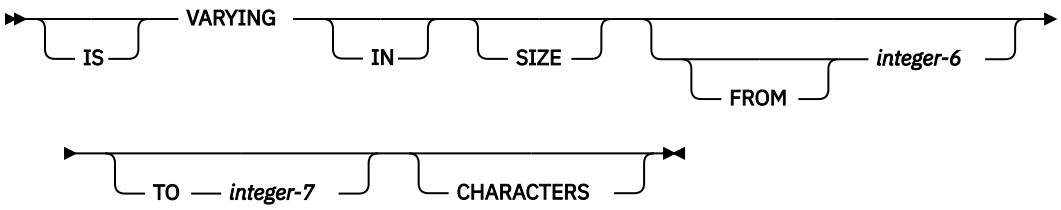
節 1


IS
VARYING
IN
SIZE
FROM
integer-6
TO
integer-7
CHARACTERS
節 2
WITH
FOOTING
AT
data-name-6
integer-9
LINES
AT
TOP
data-name-7
integer-10
LINES
AT
BOTTOM
data-name-8
integer-11

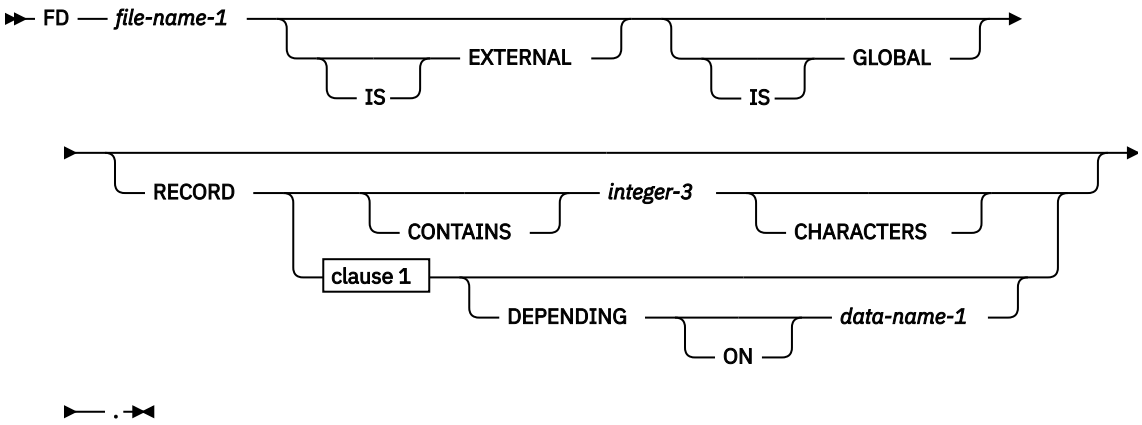
フォーマット 2: 相対および索引付きファイル記述項目



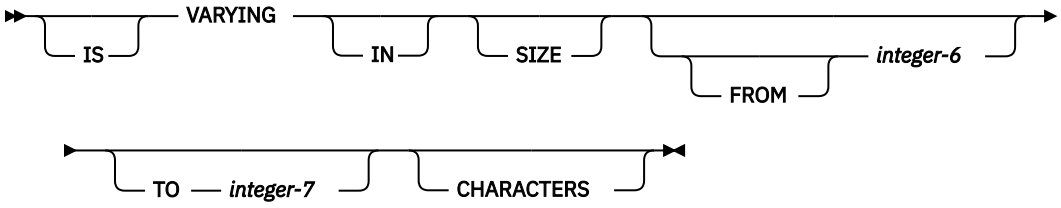
節 1



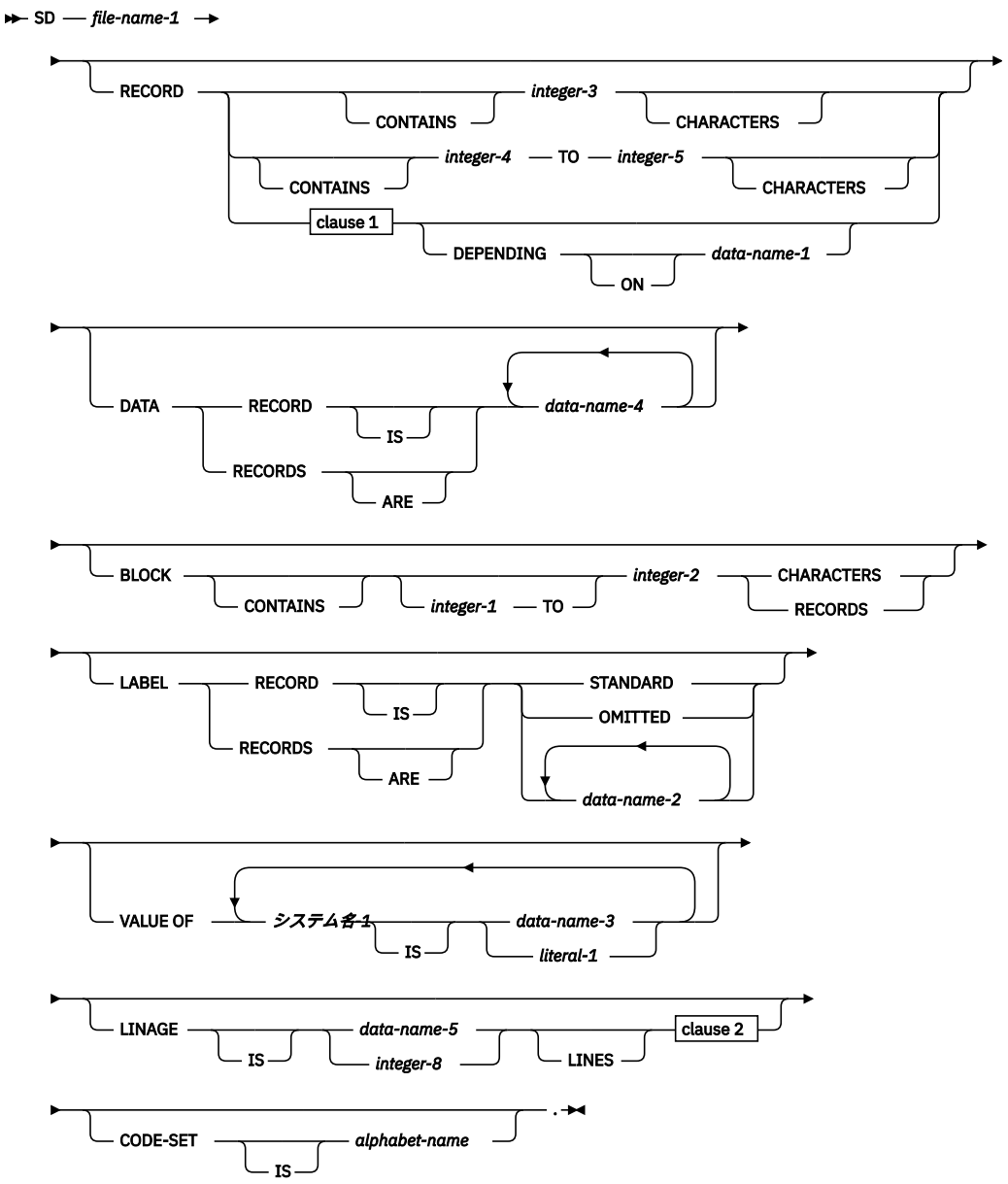
フォーマット 3: 行順次ファイル記述項目



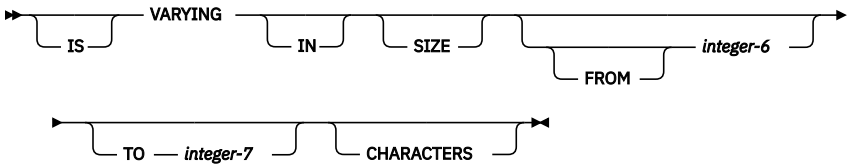
節 1



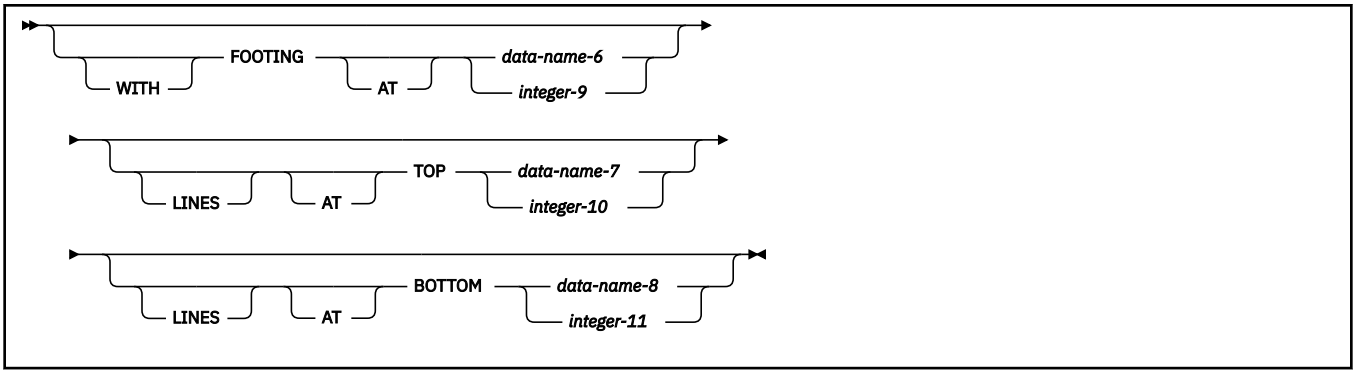
フォーマット 4: ソート/マージ・ファイル記述項目



節 1



節 2



FILE SECTION

FILE SECTION には、各入出力ファイルごとにレベル標識が必要です。 ソート・ファイルまたはマージ・ファイルを除くすべてのファイルで、FILE SECTION には FD 項目が含まれていなければなりません。 ソート・ファイルまたはマージ・ファイルについてはそれぞれ、FILE SECTION には必ず SD 項目が必要です。

ファイル名
これは、レベル標識 (FD または SD) の後に記入しなければなりません。 また関連する SELECT 節に指定された名前と同じでなければなりません。 ファイル名 は、ユーザー定義語の形成規則に従う必要があります。 したがって、少なくとも 1 文字は英字でなければなりません。 このプログラムの中でファイル名 は固有でなければなりません。

ファイル名 の後には、1 つ以上のレコード記述項目が付きます。 レコード記述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 の後の節はオプションであり、どのような順序でも指定できます。

FD (フォーマット 1、2、および 3)
FD 項目の最後の節の直後には、 分離文字ピリオドを付けなければなりません。

SD (フォーマット 4)
SD 項目は、プログラム中のソート・ファイルまたはマージ・ファイルのそれぞれに対して記述する必要があります。 SD 項目の最後の節の直後には、分離文字ピリオドを付けなければなりません。

次の例は、ソート・ファイルまたはマージ・ファイルで必要な FILE SECTION の項目を示しています。

```
SD  SORT-FILE.
01  SORT-RECORD  PICTURE X(80).
```

FILE SECTION のレコードは、英数字グループ項目、国別グループ項目、またはクラスが英字、英数字、DBCS、国別、または数字の基本項目として記述しなければなりません。

EXTERNAL 節

EXTERNAL 節は、ファイル結合子が外部にあることを指定し、2 つのプログラムがファイルを共有することによって、相互に連絡できるようにします。

あるファイルに関連付けられたストレージが、 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られている場合、そのファイルのファイル結合子は外部にあるといいます。 外部ファイルは、そのファイルを記述している実行単位の中のどのプログラムからでも参照できます。 ファイルの別々の記述を使用することによって、異なるプログラムからある外部ファイルを参照すると、 それは常に同じファイルを参照することになります。 1 つの実行単位の中で、 外部ファイルを代表するものはただ 1 つしかありません。

FILE SECTION の中で EXTERNAL 節は、ファイル記述項目の中でのみ指定できます。

ファイル記述項目に現れるレコードは、対応する外部ファイル記述項目内のものと名前が同じであ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項目の中で同じである必要はありません。

EXTERNAL 節の使用は、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。 EXTERNAL 節の使用に関する具体的な説明については、「*COBOL for Linux on x86* プログラミング・ガイド」の『*EXTERNAL* 節によるデータの共用』を参照してください。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。

GLOBAL 節

GLOBAL 節は、ファイル名によって指定されたファイル結合子がグローバル名であることを指定します。 グローバル・ファイル名は、それが宣言されているプログラムと、 そのプログラムの中に直接的または間接的に含まれるすべてのプログラムから使用できます。

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。

ファイル名は、そのファイル名に対するファイル記述項目の中で GLOBAL 節が指定されている場合、グローバル名になります。 レコード名は、そのレコード名が宣言されているレコード記述項目の中で GLOBAL 節を指定している場合、あるいは FILE SECTION 内の レコード記述項目の場合には、レコード記述項目と関連付けられているファイル名のファイル記述項目の中で GLOBAL 節を指定している場合、 グローバル名になります。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 節の使用について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド 」の 『入出力操作でのデータの使用』 および『名前のスコープ』 を参照してください。

次のような状況では、実行単位内の 2 つのプログラムがグローバル・ファイル結合子を参照できます。

- 外部ファイル結合子は、 そのファイル結合子を記述しているどのプログラムからでも参照することができます。
- あるプログラムとそれを含むプログラムは、 含むプログラムの中で、 あるいは含むプログラムを直接的または間接的に含むプログラムの中で、 関連付けられたグローバル・ファイル名を参照することによって、 グローバル・ファイル結合子を参照できます。

BLOCK CONTAINS 節

BLOCK CONTAINS 節は構文チェックされますが、プログラムの実行には何も影響しません。

整数-*1* 、 整数-*2*
ゼロ以外の符号なし整数でなければなりません。 これらは、以下のものを指定します。

CHARACTERS
データ・レコード内のデータ項目の USAGE には関係なく、 物理レコードを保管するために必要とされるバイト数を指定します。

整数-*2* だけを指定すると、それは物理レコードの正確なバイト数を指定します。 整数-*1* および整数-*2* を両方とも指定すると、それぞれ物理レコードの最小バイト数と最大バイト数を指定したことになります。

整数-*1* および整数-*2* には、 物理レコードに含まれるべき制御バイトと埋め込みバイトも含めてください。 (論理レコードには埋め込みバイトはありません。)

CHARACTERS 句がデフォルト値です。 CHARACTERS 句は、次の場合には必ず指定しなければなりません。

- 物理レコードに埋め込みバイトが含まれている場合。
- 物理レコードのサイズが間違って解釈されるような方法で、 論理レコードがグループ化されている場合。 例えば、100 バイトの可変長レコードを記述していて、4 レコードのブロックを書き込むたびに、50 バイト・レコードを 1 つと、 それに続けて 100 バイト・レコードを 3 つ書き込むとします。 この場合、RECORDS 句を指定していると、 コンパイラーはブロック・サイズを、実際のサイズである 370 バイトではなく 420 バイトと計算します。 (この計算には、ブロック記述子とレコード記述子が含まれています。)

RECORDS
これは、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。

コンパイラーは、ブロック・サイズとして最大サイズが 整数-*2* レコードのものを用意しなければならないものとみなし、さらに制御バイトに必要な余分なスペースを準備します。

RECORD 節

RECORD 節を使用する場合、レコード・サイズは、レコード内に含まれているデータ項目の USAGE には関係なく、レコードを内部的に保管するために必要なバイト数として指定しなければなりません。

例えば、DBCS 文字が 10 文字のレコードがある場合、RECORD 節は `RECORD CONTAINS 20 CHARACTERS` とする必要があります。 国別文字が 10 文字のレコードの場合、RECORD 節は `RECORD CONTAINS 20 CHARACTERS` とする必要があります。

レコード・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定されます。 (213 ページの『USAGE 節』と 206 ページの『SYNCHRONIZED 節』を参照してください。)

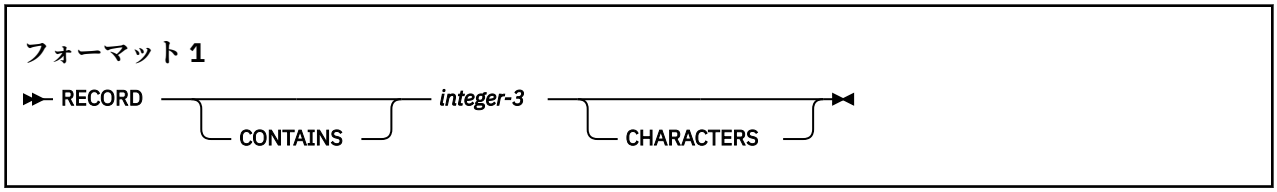
RECORD 節を省略した場合、コンパイラーはレコード記述からレコード長を決定します。 レコード記述内の項目の 1 つに OCCURS DEPENDING ON 節が含まれている場合、 コンパイラーは可変長項目の最大値を使用して、 レコードを内部的に保管するために必要なバイト数を計算します。

関連付けられているファイル結合子が外部ファイル結合子である場合、 そのファイル結合子に関連付けられている実行単位内のすべてのファイル記述項目には、 バイト数に関して同一の最大数が指定されていなければなりません。

以下のセクションでは、RECORD 節のフォーマット設定について説明します。

フォーマット 1

フォーマット 1 は、固定長レコードのバイトの数を指定します。



integer-3
ファイル内の各レコードに含まれるバイトの数を指定する符号なしの整数とする必要があります。

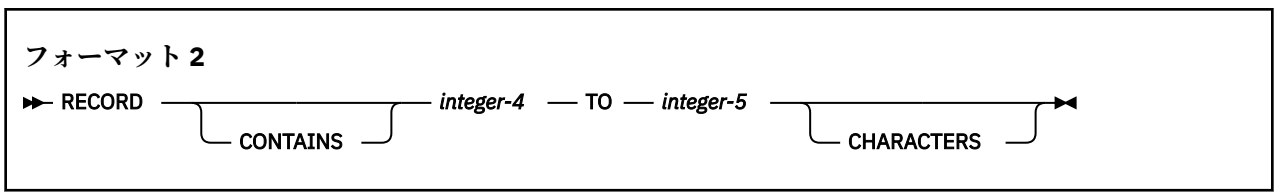
RECORD CONTAINS 0 CHARACTERS 節は構文チェックされますが、プログラムの実行には何も影響しません。

RECORD CONTAINS 0 節は SD 項目については指定しないでください。

フォーマット 2

フォーマット 2 は、固定長レコードまたは可変長レコードのバイトの数を指定します。

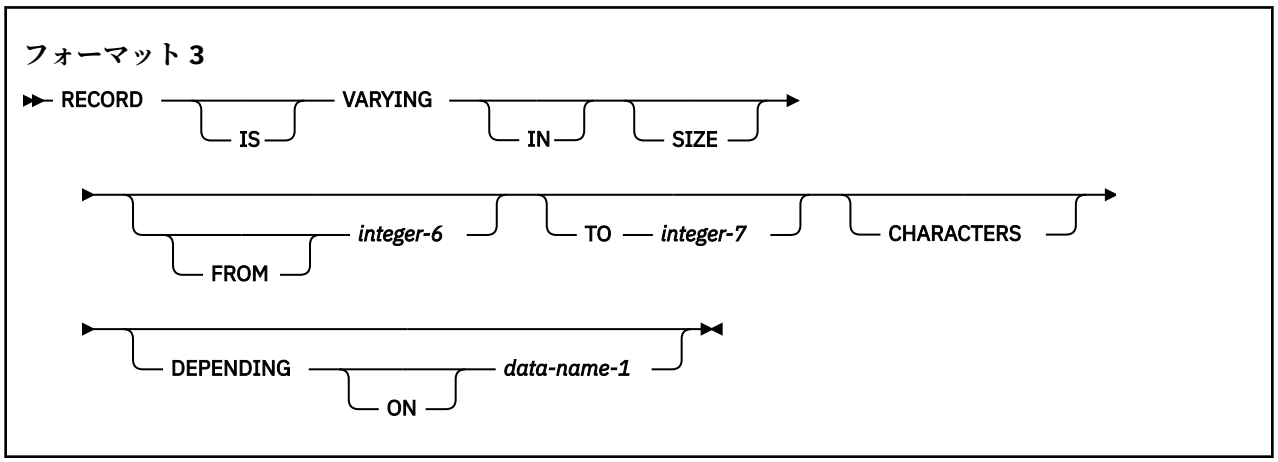
すべての 01 レコード記述項目に示されたレコード長が同一である場合には、 固定長レコードになります。 フォーマット 2 の RECORD CONTAINS 節が必要になることはありません。 最小と最大のレコード長はレコード記述項目によって決定されるからです。



整数-*4* 、 整数-*5*
符号なし整数でなければなりません。 整数-*4* には最小データ・レコード・サイズを指定し、整数-*5* には最大データ・レコード・サイズを指定します。

フォーマット 3

フォーマット 3 は、可変長レコードを指定するために使います。



整数-*6*
ファイル中のいずれかのレコードに含まれるバイトの最小数を指定します。 整数-*6* を指定しない場合、ファイル中のレコードに含まれるバイトの最小数は、 そのファイルの中のレコードに書き込まれたバイト数の最小のものに等しくなります。

整数-*7*
ファイル中のいずれかのレコードに含まれるバイトの最大数を指定します。 整数-*7* を指定しない場合、ファイル中のレコードに含まれるバイトの最大数は、 そのファイルの中のレコードに書き込まれたバイト数の最大のものに等しくなります。

レコード記述に関連したバイト数は、すべての基本データ項目 (再定義したものと名前を変更したものを除く) のバイトの個数と、 同期のために必要な暗黙の FILLER を加算して得られた合計によって決まります。テーブルを指定した場合は、次のようになります。

- レコードに記述されたテーブル・エレメントの最小数を上記の加算で使用することにより、 レコード記述に関連付けられるバイトの最小数が決定されます。
- レコードに記述されたテーブル・エレメントの最大数を上記の加算で使用することにより、 レコード記述に関連付けられるバイトの最大数が決定されます。

データ名-*1* を指定した場合は、次のようになります。

- データ名-*1* は、基本符号なし整数でなければなりません。
- データ名-*1* をウィンドウ表示日付フィールドにすることはできません。
- レコード内のバイトの数は、そのファイルに対して RELEASE、REWRITE、または WRITE のうちのいずれかのステートメントが実行される前に、データ名-*1* によって参照されるデータ項目の中に入れておかなければなりません。
- DELETE、RELEASE、REWRITE、START、または WRITE が実行されても、 また READ または RETURN ステートメントが正しく実行されなかった場合も、データ名-*1* によって参照されるデータ項目の内容は変わりません。
- ファイルに対して READ ステートメントまたは RETURN ステートメントが正しく実行された後、データ名-*1* によって参照されるデータ項目の内容は、今読み取られたレコード内のバイトの数を示しています。

RELEASE、REWRITE、または WRITE の各ステートメントの実行中に、 レコードの中のバイトの数は次の条件により決定されます。

- データ名-*1* を指定した場合、データ名-*1* によって参照されるデータ項目の内容によって。

- データ名-*1* が指定されず、レコードに可変オカレンス・データ項目が含まれない場合は、 レコード内のバイト位置の数によって。
- データ名-*1* が指定されず、 レコードに可変オカレンス・データ項目が含まれる場合は、固定位置と、 出力ステートメント実行時の出現数によって示されるテーブルの該当位置との合計によって。

READ ... INTO または RETURN ... INTO ステートメントの実行中に、暗黙の MOVE ステートメントの送り出しデータ項目として加わるカレント・レコード内のバイトの数は、 次の条件によって決定されます。

- データ名-*1* を指定した場合、データ名-*1* によって参照されるデータ項目の内容によって。
- データ名-*1* が指定されていない場合は、データ名-*1* が指定されていたとしたらそのデータ名-*1* によって参照されるデータ項目に移動されることになる値によって。

LABEL RECORDS 節

LABEL RECORDS 節は構文チェックされますが、プログラムの実行には何も影響しません。

以下の言語エレメントのいずれかを使用すると、警告メッセージが出されます。

- LABEL RECORD IS データ名
- USE ... AFTER ... LABEL PROCEDURE

LABEL RECORDS 節は、ラベルの有無を示します。 これがファイルに対して指定されていない場合、 そのファイルのラベル・レコードはシステムのラベル指定と一致している必要があります。

STANDARD
このファイルに対して、 システムの指定と一致したラベルが付いています。

STANDARD は、大容量記憶装置やテープ装置で指定できます。

OMITTED
このファイルに対してラベルは付いていません。

OMITTED は、テープ装置で使用することができます。

データ名-2
標準ラベルに加えて、ユーザー・ラベルが付いています。 データ名-*2* は、ユーザー・ラベル・レコードの名前を指定します。 データ名-*2* は、ファイルに関連したレコード記述項目の対象として指定する必要があります。

VALUE OF 節

VALUE OF 節は、ファイルと関連付けられているラベル・レコードの中の項目を記述します。

データ名-3
必要な場合には修飾しなければなりませんが、添え字付けはできません。 これは、WORKING-STORAGE SECTION に記述しなければなりません。 USAGE IS INDEX 節と共に記述することはできません。

リテラル-1
数字または英数字のリテラル、あるいは数字か英数字のカテゴリーに属する表意定数を指定できます。浮動小数点リテラルを指定することはできません。

VALUE OF 節は構文チェックされますが、プログラムの実行には何も影響しません。

DATA RECORDS 節

DATA RECORDS 節は構文チェックされますが、この節は、ファイルに関連付けられたデータ・レコードの名前に関する説明として使用されているにすぎません。

データ名-4
ファイルに関連付けられているレコード記述項目の名前。 データ名-*4* はタイプ名であってはなりません。

データ名に、同じ名前が関連付けられているレベル番号 01 のレコード記述は必須ではありません。

LINAGE 節

LINAGE 節は、論理ページの上下幅を行数で指定します。 オプションとして、さらにフッター域の開始行番号や、論理ページの上部マージンおよび下部マージンも指定できます (論理ページと物理ページは同じサイズであるとは限りません)。

LINAGE 節は、OUTPUT または EXTEND としてオープンされている順次ファイルに対して有効です。

整数はすべて符号なしでなければなりません。 データ名はすべて、符号なしの整数データ項目として記述する必要があります。

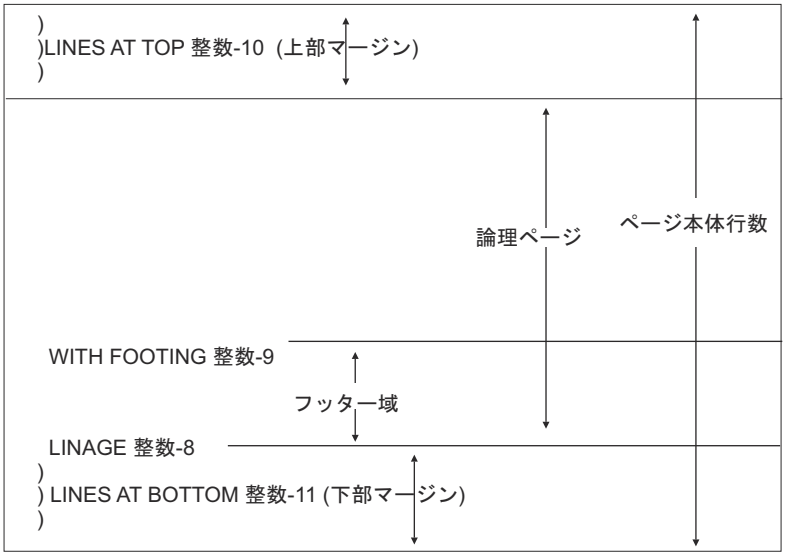
データ名-5 、 整数-8
ここには、この論理ページで書き込みまたは行送りができる行数を指定します。 これらの行によって表されるページ域を、ページ本体 といいます。 その値は 0 より大きくなければなりません。

WITH FOOTING AT
整数-*9* またはデータ名-*6* のデータ項目の値は、 ページ本体の中のフッター域の開始行番号を指定します。 フッター域の行番号は、 0 より大きくかつページ本体の最終行番号以下の値でなければなりません。 フッター域はそれら 2 つの行の間に置かれます。

LINES AT TOP
整数-*10* またはデータ名-*7* のデータ項目の値は、 論理ページの上部マージンを行数で指定します。 値として 0 も可能です。

LINES AT BOTTOM
整数-*11* またはデータ名-*8* のデータ項目の値は、 論理ページの下部マージンを行数で指定します。 値として 0 も可能です。

以下の図に、LINAGE 節の各句の使用法を示します。



LINAGE 節で指定された論理ページ・サイズは、 FOOTING 句を除く各句で指定された値の合計です。 LINES AT TOP 句を省略した場合、上部マージンとしての前提値はゼロになります。 同様に、 LINES AT BOTTOM 句を省略した場合、下部マージンとして前提値はゼロになります。 各論理ページは、先行する論理ページの直後に置かれ、 両者の間には余分なスペースはありません。

FOOTING 句を省略した場合、 前提値はページ本体の値に等しくなります (つまり整数-*8* またはデータ名-*5*)。

OPEN OUTPUT ステートメントが実行される時点で、整数-*8*、整数-*9*、整数-*10*、および整数-*11* の値が指定されていれば、これらの値を使用して、 このファイルの論理ページのページ本体、フッター域開始行、上部マージン、 下部マージンが決定されます。 (上の図を参照。) 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、 これらの値が使用されます。

このファイルに対して OUTPUT 句を伴う OPEN ステートメントが実行される時点で、最初の論理ページに関してのみ、データ名-*5*、データ名-*6*、データ名-*7*、データ名-*8* によって、ページ本体、フッター域の開始行、 上部マージン、下部マージンが決定されます。

ADVANCING PAGE 句を伴う WRITE ステートメントが実行される時点で、 またはページ・オーバーフロー条件が発生した時点で、データ名-*5*、データ-*6*、データ名-*7*、およびデータ名-*8* の値が指定されていれば、これらの値を指定して、次の論理ページのページ本体、フッター域の開始行、上部マージン、下部マージンが決定されます。

外部ファイル結合子がファイル記述項目に関連付けられている場合、 そのファイル結合子に関連付けられた実行単位内のすべてのファイル記述項目は、 次のようになっていなければなりません。

- いずれかのファイル記述項目に LINAGE 節が含まれている場合には、LINAGE 節があること。
- 整数-*8*、整数-*9*、整数-*10*、および 整数-*11* が指定されている場合には、それらの値が対応しており同一であること。
- データ名-*5*、データ名-*6*、データ名-*7*、データ名-*8* によって参照される外部データ項目が、 それぞれ対応している同一の外部データ項目であること。

外部ファイルの紙送り制御文字の動作については、400 ページの『ADVANCING 句』を参照してください。

SD の下の LINAGE 節は構文チェックされますが、プログラムの実行には何も影響しません。

LINAGE-COUNTER 特殊レジスター

LINAGE-COUNTER 特殊レジスターについては、19 ページの『LINAGE-COUNTER』を参照してください。

RECORDING MODE 節

順次ファイルのレコードの RECORDING MODE 節は、次のように扱われます。

F
レコード記述は固定長として妥当性検査されます。 レコード記述が可変の場合は、RECORDING MODE F を指定しないでください。

V
可変長レコード・フォーマットとみなされます (レコード記述が固定長であっても)。

U
構文チェックされますが、プログラムの実行には何も影響しません。.

S

- 非 QSAM ファイルでは、V と同様に扱われます。
- QSAM ファイルでは、レコードは可変長または固定長のいずれかであり、ブロックより大きくすることができます。 1 つのレコードが 1 ブロック内の残りのスペースよりも大きい場合、 そのブロックの残りのスペースを埋める分だけ、 そのレコードの一部が書き込まれます。 レコードの残りの部分は次のブロックに保管されます (必要なら次の複数のブロックにわたって保管されます)。 利用できるのは完全なレコードだけです。 あるブロックの中のレコードの各セグメントには、 それがそのレコード全体であるとしても、セグメント記述子フィールドが含まれ、各ブロックにはブロック記述子フィールドが含まれます。 これらのフィールドは DATA DIVISION には記述されていません。それは自動的に用意されます。 これらのフィールドを利用することはできません。

CODE-SET 節

CODE-SET 節は、構文チェックされますが、プログラムの実行には何も影響しません。

第 17 章 DATA DIVISION - データ記述項目

データ記述項目 は、データ項目の特性を指定します。 以降のセクションでは、データ記述項目のセットをレコード記述項目 と呼びます。 データ記述項目 という用語は、データ記述項目およびレコード記述項目を指します。

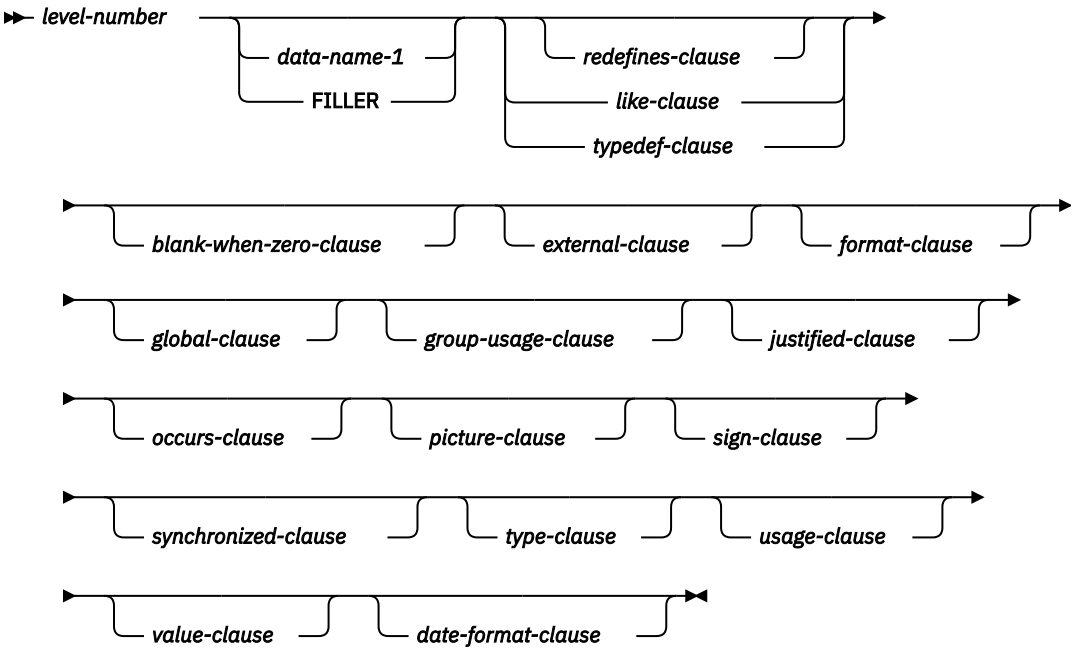
独立データ項目を定義するデータ記述項目は、レコードを構成しません。 これらの項目をデータ項目記述項目 といいます。

データ記述項目には 3 種類の一般形式があります。 また、データ記述項目にはすべて、末尾に分離文字ピリオドを付けなければなりません。

フォーマット 1

フォーマット 1 が、データ記述項目について DATA DIVISION の全セクションで使用されます。

フォーマット 1: データ記述項目



節はどのような順序でも記述できますが、以下の例外があります。

- データ名-*1* または FILLER を指定する場合、それはレベル番号の直後に置かなければなり ません。
- REDEFINES 節を指定する場合、データ名-*1* または FILLER のいずれかを指定するときには、 その直後に置か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、REDEFINES 節はレベル番号の直後に置かなければなりません。
- TYPEDEF 節は、指定するのであれば *data-name-1* の後に続く最初の項目でなければなりません。 TYPEDEF 節を FILLER とともに指定することはできません。 TYPEDEF 節と REDEFINES 節を両方とも *data-name-1* に対して指定することはできません。

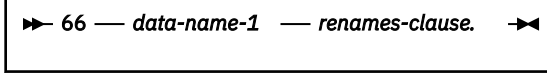
フォーマット 1 のレベル番号としては、01 から 49、または 77 のいずれかの番号が使用できます。

節を区切るためには、スペース、コンマ、またはセミコロンが必要です。

フォーマット 2

フォーマット 2 は、定義済み項目を再グループ化します。

フォーマット 2: RENAMES



レベル 66 の項目は、他のレベル 66 の項目の名前に変更することはできません。また、レベル 01、レベル 77、またはレベル 88 の項目についても、名前の変更はできません。

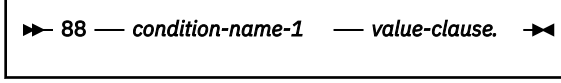
あるレコードに関連付けられているすべてのレベル 66 項目は、 そのレコードの中の最後のデータ記述項目の直後になければなりません。

詳しくは、203 ページの『RENAMES 節』を参照してください。

フォーマット 3

フォーマット 3 は、条件名を記述します。

フォーマット 3: 条件名



条件名-*1*
ある値、値の集合、または値の範囲を条件変数に関係付けるユーザー指定の名前。

レベル 88 の項目は、条件名に関連付けられている 条件変数用のデータ記述項目の 直後でなければなりません。

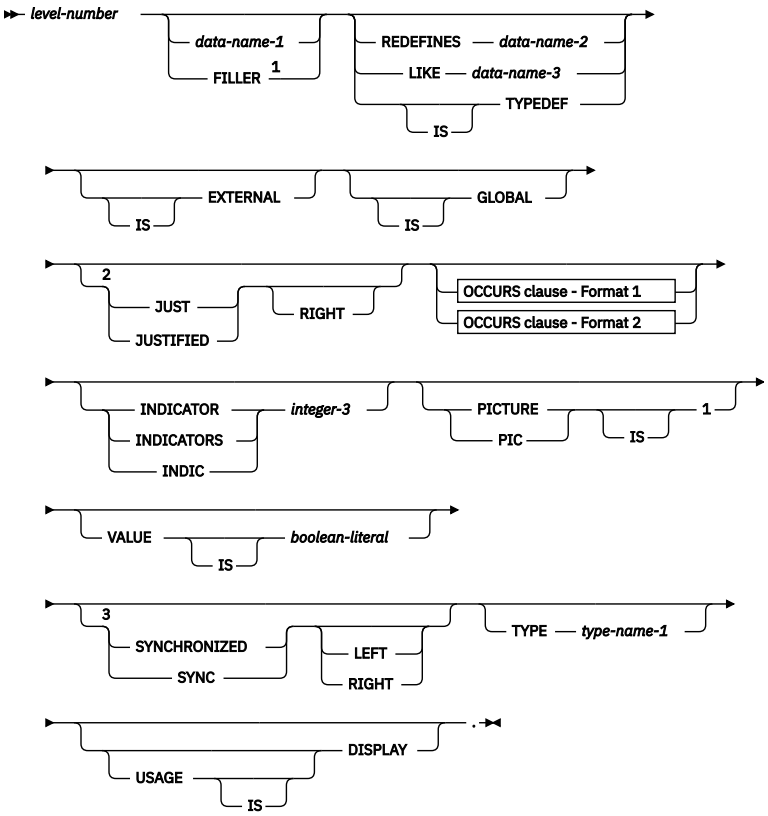
フォーマット 3 は、基本項目、国別グループ項目、または英数字グループ項目を記述するために使用することができます。 条件名項目の追加情報については、220 ページの『VALUE 節』および 241 ページの『条件名条件』を参照してください。

フォーマット 4

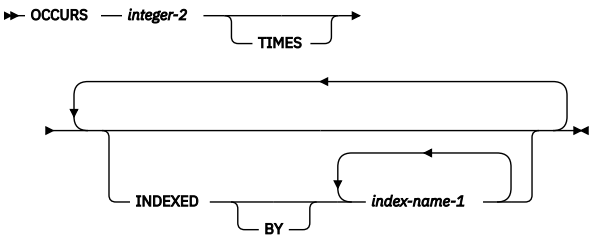
この形式は、ブール・データを記述します。ブール・データ項目は値を 1 または 0 に限定された項目です。

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。

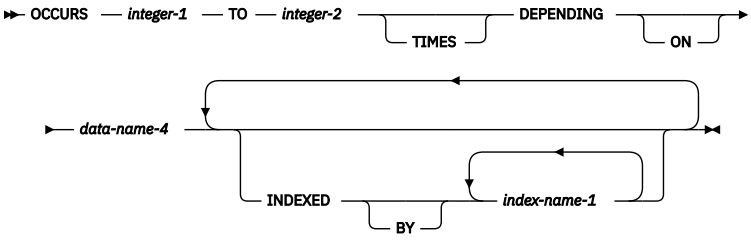
フォーマット 4: ブール・データ



OCCURS 節 - フォーマット 1



OCCURS 節 - フォーマット 2



注:

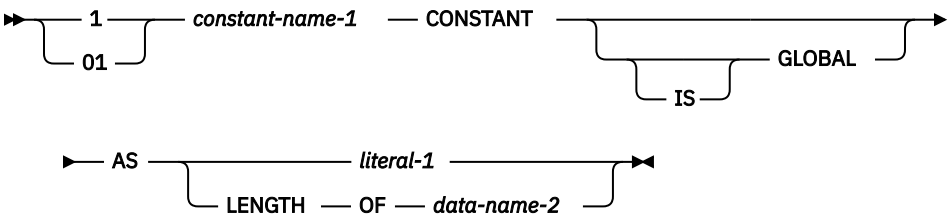
[1] TYPEDEF 節で使用できない
[2] 構文検査のみ
[3] 構文検査のみ

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。

形式 5

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。

フォーマット 5: 定数名



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、DATA DIVISION および PROCEDURE DIVISION 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。

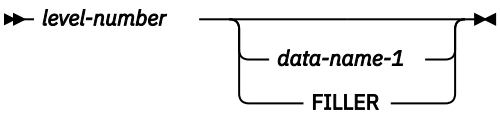
LENGTH OF 句が指定されている場合、定数名-*1* の値は、LENGTH 組み 込み関数で指定されているとおりに決定されます。ただし、例外として、 データ名-*2* が、OCCURS DEPENDING ON 文節で記述される可変長データ項目の 場合は、データ項目の最大サイズが使用されます。

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、データ記述項目、名前変更したり再定義した項目、 または条件名項目の冒頭に置かれます。

レベル番号は、1 から 49 までの整数値 (1 と 49 を含む) か、または特別なレベル番号値 66、77、 または 88 のいずれかになります。

フォーマット



レベル番号

01 および 77 は、領域 A で開始しなければならず、その後に 分離文字ピリオドを付けるか、またはスペースとその後に関連データ名、 FILLER、または該当するデータ記述節を付けなければなりません。

レベル番号 02 から 49 は、領域 A または領域 B で開始でき、 その後にスペースまたは分離文字ピリオドを付けなければなりません。

レベル番号 66 と 88 は、領域 A または領域 B から開始でき、 その後にスペースを付けなければなりません。

1 桁のレベル番号 1 から 9 は、レベル番号 01 から 09 で置き換えることができます。

連続するデータ記述項目は、最初の項目と同じ桁から始めることも、 またはレベル番号に合わせて字下げをすることもできます。 字下げしても、レベル番号の大きさに変わりはありません。

レベル番号を字下げする際には、新しいレベル番号が出てくるたびに領域 A の 右側にいくつでもスペースを付けて開始できます。右側に字下げする際の限度は領域 B の幅までで、他に制限はありません。

詳しくは、134 ページの『データのレベル』を参照してください。

データ名-*1*
記述されるデータを明示的に識別します。

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。 データ名-*1* は、レベル番号の後に付く最初のワードでなければなりません。

データ項目は、プログラムの実行中に変更することができます。

データ名-*1* は、レベル 66 とレベル 88 の項目に対して指定しなければなりません。 また、これは、GLOBAL 節や EXTERNAL 節を含む項目の場合、ファイル記述項目と関連付けられたレコード記述項目が GLOBAL 節および EXTERNAL 節を持っているときにも指定しなければなりません。

FILLER
プログラムの中で明示的に参照されないデータ項目です。 このキーワード FILLER は、オプショナルです。 指定する場合、FILLER はレベル番号の後に付く最初のワードでなければなりません。

FILLER というキーワードは、条件変数と共に使用できます。ただし、それが可能なのは、その条件変数に対して明示的な参照が行われるのではなく、その条件変数が想定している値に対してのみ明示的な参照が行われる場合です。 FILLER は条件名と一緒に使用することはできません。

MOVE CORRESPONDING ステートメント、ADD CORRESPONDING ステートメント、または SUBTRACT CORRESPONDING ステートメントの中では、FILLER 項目は無視されます。

INITIALIZE ステートメントでは、次のようになります。

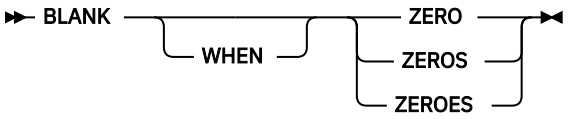
- FILLER 句が指定されていない場合、FILLER 基本項目は無視されます。
- FILLER 句が指定されている場合、明示 FILLER 節または暗黙 FILLER 節を持つ受信基本データ項目が初期化されます。

データ名-*1* または FILLER 節が省略された場合、記述されたデータ項目は、 FILLER が指定されたものとして扱われます。

BLANK WHEN ZERO 節

BLANK WHEN ZERO 節は、項目の値が 0 の時は、その項目にスペースだけが入ることを指定します。

フォーマット



BLANK WHEN ZERO 節は、その PICTURE 文字ストリングによって数字編集または数字カテゴリーとして記述されている (PICTURE 記号を S または * を指定しない) 基本項目に対してのみ指定できます。 これらの項目は、USAGE DISPLAY または USAGE NATIONAL として暗黙的もしくは明示的に記述されなければなりません。

その PICTURE 文字ストリングによって数字として定義されている項目に対して指定されている BLANK WHEN ZERO 節は、この項目を数字編集カテゴリーとして定義します。

BLANK WHEN ZERO 節は、日付フィールドに対しては指定してはなりません。

DATE FORMAT 節

DATE FORMAT 節は、データ項目がウィンドウ表示日付フィールドまたは拡張日付フィールドであることを指定します。

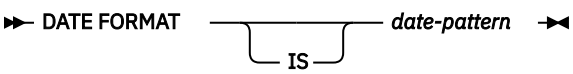
ウィンドウ表示日付フィールド
YY を含む DATE FORMAT 節によって指定されたウィンドウ表示 (2 桁) 年が入れられます。

拡張日付フィールド
YYYY を含む DATE FORMAT 節によって指定された拡張 (4 桁) 年が入れられます。

NODATEPROC コンパイラー・オプションが有効であると、DATE FORMAT 節は構文検査されますが、プログラムの実行には影響はありません。NODATEPROC は、日付処理を使用不可にします。 このセクションで、DATE FORMAT 節および日付フィールドについて述べている規則と制約事項は、 DATEPROC コンパイラー・オプションが有効な場合のみ適用されます。

DATE FORMAT 節は、USAGE NATIONAL で記述されたデータ項目に対しては指定してはなりません。

フォーマット



日付パターン は、ウィンドウ表示西暦年または拡張西暦年を表す YYXXXX などの文字ストリングであり、オプションとして月日など日付の他の部分を表す 1 から 4 文字が最初または最後に付く場合があります。

日付パターン・ストリング	データ項目に含まれるもの
YY	ウィンドウ表示 (2 桁) 年。
YYYY	拡張 (4 桁) 年。
X	1 文字。例えば、1 学期または四半期 (1 から 4) などを表す数字。
XX	2 文字。例えば、月を表す数字 (01 から 12)。
XXX	3 文字。例えば、1 年のうちの何日目かを表す数字 (001 から 366)。
XXXX	4 文字。例えば、月を表す 2 桁 (01 から 12) と月の何日かを表す 2 桁 (01 から 31)。

日付フィールドおよび関連名の概要については、69 ページの『第 10 章 2000 年言語拡張および日付フィールド』を参照してください。 アプリケーションでの日付フィールドの使用について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『*2000* 年言語拡張 *(MLE)*』を参照してください。

ウィンドウ表示日付フィールドのセマンティクス

ウィンドウ表示日付フィールドが算術式または算術ステートメントでオペランドとして使用されると、世紀ウィンドウに関しては自動拡張されます。 しかし、ウィンドウ化日付を増大または減少した結果は、 その後の計算、比較、および保管操作においては、 やはりウィンドウ化日付として扱われます。

ウィンドウ表示日付フィールドが以下の状況で使用された場合、それは拡張日付フォーマットに変換されたかのように扱われます。

- 他方のオペランドが拡張日付であるような減算のオペランド
- 比較条件のオペランド
- 算術または MOVE ステートメントの送り出しフィールド

拡張日付フォーマットへの変換の詳細は、ウィンドウ表示日付フィールドが数字であるか、 英数字であるかによって異なります。

世紀ウィンドウに 19*nn* の開始年を与えた場合、数字ウィンドウ表示日付フィールドの年部分 (*yy*) は、次のように拡張されたかのように扱われます。

- *yy* が *nn* より小さい場合は、*yy* に 2000 を加算します。
- *yy* が *nn* 以上の場合は、*yy* に 1900 を加算します。

符号付き数字ウィンドウ表示日付フィールドの場合、これは、年の表示が 2 種類あり得るということを意味します。 例えばウィンドウ表示西暦年の値 99 と -01 は両方とも 1999 として扱われます。 これは 1900 + 99 = 2000 + -01 だからです。

英数字のウィンドウ表示日付フィールドも同様に扱われますが、1900 または 2000 を加算する代わりに、19 または 20 の接頭部が使用されます。

例えば、比較条件のオペランドとして使用された場合、ウィンドウ表示日付フィールドは以下のようになります。

```
01  DATE-FIELD  DATE FORMAT YYXXXX PICTURE 9(6)
                VALUE IS 450101.
```

上記のように定義されたウィンドウ表示日付フィールドは、次のどちらかの値の拡張日付フィールドであるかのように扱われます。

- 19450101 (世紀ウィンドウ開始年が 1945 以前の場合)
- 20450101 (世紀ウィンドウ開始年が 1945 より後の場合)

日付フィールドの使用に関する制約事項

日付フィールドを異なるコンテキストで使用する場合には、制約事項がいくつかあります。

コンテキストは以下のとおりです。

- DATE FORMAT 節と他の節との結合
- 日付フィールドであるグループ項目
- 日付フィールドを非日付データとして扱う言語エレメント
- ウィンドウ表示日付フィールドを引数として受け入れない言語エレメント
- 日付フィールドを引数として受け入れない言語エレメント

日付フィールドを他の関連で使用する場合の制約事項については、以下を参照してください。

- 236 ページの『日付フィールドを使用する算術計算』
- 250 ページの『日付フィールドの比較』
- 280 ページの『ADD ステートメント』
- 389 ページの『SUBTRACT ステートメント』
- 327 ページの『MOVE ステートメント』

DATE FORMAT 節と他の節との結合

DATE FORMAT 節を他の節と結合する場合には、制約事項があります。

DATE FORMAT 節と結合可能な USAGE 節の句は以下の句のみです。

- BINARY
- COMPUTATIONAL[1]
- COMPUTATIONAL-3
- COMPUTATIONAL-4
- DISPLAY
- PACKED-DECIMAL

[1]TRUNC(BIN) コンパイラー・オプションが有効である場合、USAGE COMPUTATIONAL は DATE FORMAT 節と結合できません。

PICTURE 文字ストリングでは、DATE FORMAT 節と同じ文字数または桁数を指定しなければなりません。英数字の日付フィールドの場合、許可される PICTURE 文字ストリング記号は、A、9、および X だけです (少なくとも 1 つの X がなければなりません)。数字日付フィールドの場合、可能な PICTURE 文字ストリング記号は 9 と S だけです。

以下の節は、DATE FORMAT で定義されたデータ項目には使用できません。

- BLANK WHEN ZERO
- JUSTIFIED
- SIGN 節の SEPARATE CHARACTER 句

EXTERNAL 節は、ウィンドウ表示日付フィールドまたはウィンドウ表示日付フィールド従属項目を含むグループ項目には使用できません。

次の節を DATE FORMAT と結合するときには、いくつかの制約事項が適用されます。

- 199 ページの『REDEFINES 節』
- 220 ページの『VALUE 節』

日付フィールドであるグループ項目

グループ項目が DATE FORMAT 節を使用して定義されている場合は、ある制約事項が適用されます。

制約事項は以下のとおりです。

- グループ内の基本項目は、すべて USAGE DISPLAY でなければなりません。
- グループ項目の長さは、DATE FORMAT 節の日付パターン と同じ文字数でなければなりません。
- グループが USAGE DISPLAY を使用する日付フィールドだけから構成され、グループ項目と単一従属項目に DATE FORMAT 節が指定されている場合、両方の DATE FORMAT 節は同一でなければなりません。
- グループを細分化する従属項目がグループ項目に含まれている場合には、 以下の制約事項が適用されます。
  - 指定された (FILLER ではない) 従属項目が、グループ項目日付フィールドの年部分から構成され、かつ DATE FORMAT 節がある場合、DATE FORMAT 節は、YY または YYYY (グループ項目と同じ文字数の年) でなければなりません。
  - グループ項目が YYXXXX、YYYYXXXX、XXXXYY、または XXXXYYYY の DATE FORMAT 節の指定されているグレゴリオ暦で、 名前付き従属日付データ項目がグレゴリオ暦の年および月の部分を構成する場合、その DATE FORMAT 節はそれぞれ YYXX、YYYYXX、XXYY、 または XXYYYY でなければなりません (または、YYYYXXXX のグループ日付フォーマットの場合、YYXX の従属日付データは以下の説明のようになります)。
  - ウィンドウ表示日付フィールドが従属項目がグループ項目の後 2 文字で開始する場合、 日付が年が最後にくるフォーマットでなく従属項目の日付フォーマットに X がないか、グループ項目と同じ数の X が Y の後に続くか、 またはグループ日付フォーマット YYYYXXXX の下の YYXX である場合に、それは拡張日付フィールド・グループに従属します。
  - 上記の制限事項で説明したように、DATE FORMAT 節を指定できる従属項目は、 グループ項目の年の部分を定義する項目、拡張日付フィールド・グループ項目のウィンドウ表示部分、 またはグレゴリオ日付グループ項目の年と月の部分だけです。

  以下の例では、有効なグループ項目を定義します。

```
01  YYMMDD     DATE FORMAT YYXXXX.
    02  YYMM   DATE FORMAT YYXX.
        03  YY DATE FORMAT YY PICTURE 99.
        03                    PICTURE 99.
    02  DD                    PICTURE 99.
```

日付フィールドを非日付データとして扱う言語エレメント

日付フィールドが以下の言語エレメントで使用されると、それらは非日付データとして扱われます。 すなわち、DATE FORMAT は無視され、日付データ項目の内容は自動拡張を受けずに使用されます。

- 環境部 FILE-CONTROL 段落:
  - SELECT ... ASSIGN USING データ名
  - SELECT ... PASSWORD IS データ名
  - SELECT ... FILE STATUS IS データ名
- データ部項目:
  - LABEL RECORD IS データ名
  - LABEL RECORDS ARE データ名
  - LINAGE IS データ名 FOOTING データ名 TOP データ名 BOTTOM データ名
- クラス条件
- 符号条件
- DISPLAY ステートメント

ウィンドウ表示日付フィールドを引数として受け入れない言語エレメント

ウィンドウ表示日付フィールドを引数として使用する場合には、制約事項があります。

ウィンドウ表示日付フィールドは、以下のものとして使用することはできません。

- 環境部 FILE-CONTROL 段落の以下のフォーマットにおけるデータ名:
  - SELECT ... RECORD KEY IS
  - SELECT ... ALTERNATE RECORD KEY IS
  - SELECT ... RELATIVE KEY IS
- データ部のファイル記述 (FD) またはソート記述 (SD) 項目の RECORD IS VARYING DEPENDING ON 節におけるデータ名
- データ部データ定義項目の OCCURS DEPENDING ON 節のオブジェクト
- データ部データ定義項目の OCCURS 節の ASCENDING KEY または DESCENDING KEY 句のキー
- 以下のステートメントにおけるデータ名または ID:
  - CANCEL
  - GO TO ... DEPENDING ON
  - INSPECT
  - SET
  - SORT
  - STRING
  - UNSTRING
- CALL ステートメントで、プログラム名を含む ID として
- PERFORM ステートメントの TIMES および VARYING 句の ID として (ウィンドウ表示日付フィールドは PERFORM 条件の中では指定可能)
- 逐次 (フォーマット 1) SEARCH ステートメントの VARYING 句の ID、 または 2 進 (フォーマット 2) SEARCH ステートメントの ID として (ウィンドウ表示日付フィールドは SEARCH 条件の中では指定可能)
- WRITE ステートメントの ADVANCING 句における ID
- 組み込み関数 (UNDATE 組み込み関数を除く) への引数

ウィンドウ表示日付フィールドは、MERGE または SORT ステートメントで昇順キーまたは降順キーとして使用することはできません。

日付フィールドを引数として受け入れない言語エレメント

ウィンドウ表示日付フィールドおよび拡張日付フィールドを使用する場合には、制約事項があります。

ウィンドウ表示日付フィールドまたは拡張日付フィールドのいずれも使用することができません。

- DIVIDE ステートメント (GIVING または REMAINDER 節の ID として使用する場合を除く)
- MULTIPLY ステートメント (GIVING 節の ID として使用する場合を除く)

(日付フィールドは、除算または乗算のオペランドとして使用することはできません。)

EXTERNAL 節

EXTERNAL 節は、データ項目と関連付けられたストレージが、実行単位内の特定のプログラムではなく、その実行単位に関連付けられるということを指定します。

外部データ項目は、そのデータ項目について記述する実行単位の中のどのプログラムからでも参照できます。 データ項目の別個の記述を使用して異なるプログラムから外部データ項目を参照することは、いつでも同じデータ項目を参照することです。 1 つの実行単位内では、外部データ項目を代表するものは 1 つしかありません。

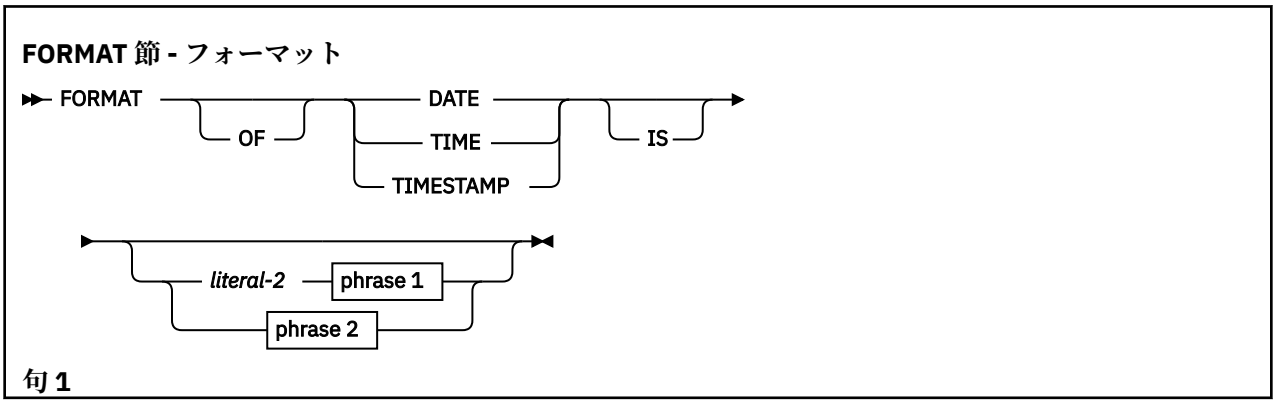
EXTERNAL 節は、レベル番号が 01 のデータ記述項目でのみ指定できます。 それは、プログラムの WORKING-STORAGE SECTION にあるデータ記述項目でのみ指定できます。 これは LINKAGE SECTION データ記述項目、LOCAL-STORAGE SECTION データ記述項目、FILE SECTION データ記述項目のいずれにおいても指定できません。あるデータ記述項目によって記述されているデータ項目があり、そのデータ記述項目が、外部レコードを記述している データ記述項目に従属している場合は、そのようなデータ項目にも外部属性が与えられます。 外部データ・レコードにある索引は、外部属性を処理しません。

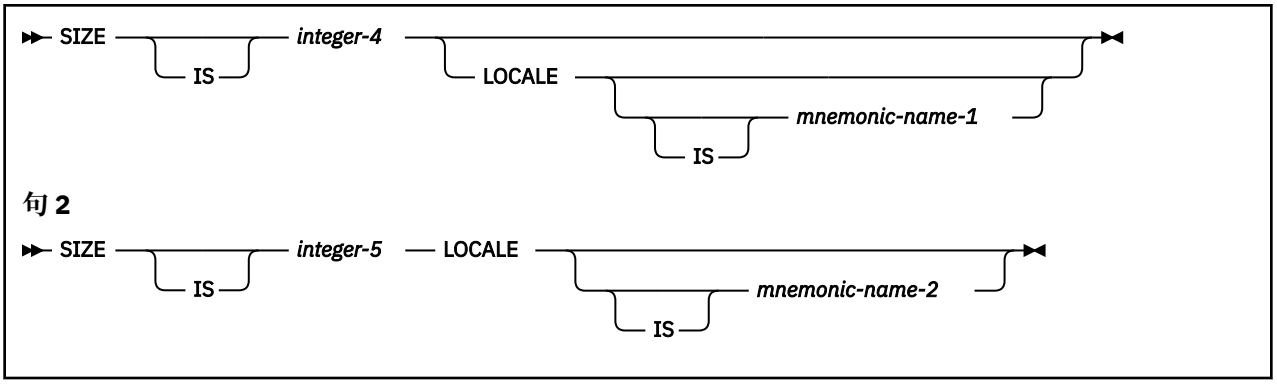
データ名節によって指定されたレコードに含まれるデータは外部データであり、それを記述する実行単位、または場合によっては それを再定義している実行単位の中のどのプログラムからでもアクセスして処理することができます。 このデータは、次のような規則に従います。

- 実行単位内の 2 つ以上のプログラムが同じ外部データ・レコードを記述している場合は、それに関連した レコード記述項目の各レコード名は同じでなければならず、これらのレコードは同数のバイトを定義していなければなりません。 しかし、外部レコードを記述している 1 つのプログラムに、外部レコード全体を再定義する REDEFINES 節を含むデータ記述項目が含ま れていることがあり、その場合には実行単位内の他のプログラムで同じ ように全体を再定義する必要はありません。
- EXTERNAL 節を使用しても、 関連するデータ名がグローバル名であることを意味するわけではありません。

FORMAT 節

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『リテラル-*8* で使用することができる変換指定子』の表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 98 ページの『FORMAT 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

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。

SIZE 句について、より詳細な説明が必要な場 合は、100 ページの『SIZE 句』を参照してください。

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* また簡略名-*2* の詳細は 168 ページの『LOCALE 句』および 101 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。

日時クラス項目の USAGE

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。 現行ロケールを判別する方法について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」 の『*CEELOCT:* 現在の現地日時の取得』 を参照してください。

GLOBAL 節

GLOBAL 節は、データ名または定数名が、そのデータ名を定義するプログラム内に含まれているすべてのプログラムに使用可能であることを指定します (ただし、その中に含まれているプログラム自体がその名前を定義している場合を除きます)。グローバル名に従属するデータ名または定数名またはグローバル名と関連付けられた条件名や指標は、すべてグローバル名です。

あるデータ名または定数名がそれによって定義されているデータ記述項目か、そのデータ記述項目が従属している別の項目のどちらかに GLOBAL 節が指定されている場合、そのデータ名または定数名はグローバル名です。GLOBAL 節は WORKING-STORAGE SECTION、FILE SECTION、LINKAGE SECTION、および LOCAL-STORAGE SECTION の中で指定することができます。ただし、レベル番号が 01 のデータ記述項目の中でなければなりません。

同じ DATA DIVISION の中では、同じデータ名または定数名が指定されている任意の 2 つのデータ項目に関するデータ記述項目に GLOBAL 節を含めてはなりません。

グローバル名を記述してあるプログラム内に直接的または間接的に含まれるプログラムの中のステートメントは、そのグローバル名を再度 記述しなくても参照することができます。

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 節内のグローバル・タイプ名を使用して定義されたデータ項目がグローバル属性を獲得することはありません。

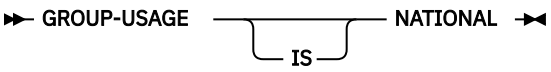
実行単位内の 2 つのプログラムは、次のような場合には共通データを参照することができます。

- 外部データ・レコードのデータ内容が、そのデータ・レコードを外部として記述しているどのようなプログラムからでも参照できるとき。
- あるプログラムが別のプログラム内に含まれている場合、両方のプログラムは、次のどちらのデータも参照できる。すなわち、 含んでいる方のプログラムの中にあるグローバル属性データ、またはその含んでいるプログラムを直接的または間接的に 含んでいるいずれかのプログラムの中にあるグローバル属性データ。

GROUP-USAGE 節

NATIONAL 句のある GROUP-USAGE 節は、その項目によって定義されるグループ項目が国別グループ項目であることを指定します。 国別グループ項目は、すべての従属データ項目および従属グループ項目内に国別文字を含んでいます。

フォーマット



GROUP-USAGE NATIONAL が指定されている場合

- 項目のサブジェクトは国別グループ項目です。 国別グループのクラスおよびカテゴリーは国別です。
- USAGE 節は、項目のサブジェクトに対しては指定してはなりません。 USAGE NATIONAL 節は暗黙指定されます。
- USAGE NATIONAL 節は、USAGE NATIONAL 節で記述されていない従属基本データ項目に対して暗黙指定されます。
- すべての従属基本データ項目は、明示的または暗黙的に USAGE NATIONAL で記述される必要があります。
- 符号付き数値データはすべて、SIGN IS SEPARATE 節で記述される必要があります。
- GROUP-USAGE NATIONAL 節は、GROUP-USAGE NATIONAL 節で記述されていないすべての従属グループ項目に対して暗黙指定されます。
- すべての従属グループ項目は、明示的または暗黙的に GROUP-USAGE NATIONAL 節で記述される必要があります。
- JUSTIFIED 節を指定することはできません

別の記述が行われていない限り、国別グループ項目は USAGE が国別でクラスおよびカテゴリーが国別の、PICTURE N(*m*) で記述されている基本データ項目として処理されます。ここで、*m* は国別文字位置にあるグループの長さです。

使用上の注意: 国別グループを使用する場合、コンパイラーは、MOVE や INSPECT などのステートメントについて、グループ項目の適切な切り捨ておよび埋め込みを確実に行うことができます。 GROUP-USAGE

NATIONAL 節を指定しないで定義されるグループは、英数字グループです。 英数字グループの内容は、すべての国別文字を含め、英数字データとして取り扱われ、国別文字データの無効な切り捨てや誤った処理につながる可能性があります。

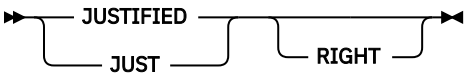
下記の表では、国別グループ項目がグループ項目として処理される場合を要約しています。

表 *15.* **国別グループ項目がグループとして処理される場合**	
言語機能	**国別グループ項目の処理**
名前の修飾	国別グループ項目の名前を使用して、国別グループ内の基本データ項目および従属グループ項目の名前を修飾することができます。 国別グループの修飾の規則は、英数字グループの修飾の規則と同じです。
RENAMES 節	THROUGH 句で指定された国別グループ項目の場合の規則は、THROUGH 句で指定された英数字グループの規則と同じです。 結果は英数字グループ項目になります。
CORRESPONDING 句	国別グループ項目は、CORRESPONDING 句の規則に従って、グループとして処理されます。 国別グループ内の基本データ項目は、英数字グループ内で定義されている場合と同様に処理されます。
INITIALIZE ステートメント	国別グループ項目は、INITIALIZE ステートメントの規則に従って、グループとして処理されます。 国別グループ内の基本項目は、英数字グループ内で定義されている場合と同様に初期化されます。
XML GENERATE ステートメント	FROM 句に指定された国別グループ項目は、XML GENERATE ステートメントの規則に従って、グループとして処理されます。 国別グループ内の基本項目は、英数字グループ内で定義されている場合と同様に処理されます。

JUSTIFIED 節

JUSTIFIED 節は、英字、英数字、DBCS または国別カテゴリーの受け取り項目における 標準の位置合わせ規則を変更します。

フォーマット



JUSTIFIED 節を指定する際に使用できるレベルは、基本レベルだけです。 JUST は JUSTIFIED の省略形で、両者の意味は同じです。

次の場合、JUSTIFIED 節は指定できません。

- 数字、数字編集、英数字編集、または国別編集カテゴリーのデータ項目の場合
- 編集済み DBCS 項目の場合
- 指標データ項目の場合
- USAGE FUNCTION-POINTER、USAGE POINTER、または USAGE PROCEDURE-POINTER として記述されている項目の場合
- 外部浮動小数点項目および内部浮動小数点項目の場合
- 日付フィールドの場合
- レベル 66 (RENAMES) およびレベル 88 (条件名) の項目を指定する場合
- TYPE 節が指定されている項目の場合

受け取り項目で JUSTIFIED 節を指定すると、データは受け取り項目の中で右端の文字位置に位置合わせされます。 また、次のようになります。

- 送り出し項目が受け取り項目よりも大きい場合、左端の文字が切り捨てられます。
- 送り出し項目が受け取り項目よりも小さい場合、 左側の使用されていない文字位置は、スペースが埋め込まれます。 DBCS 項目の場合、それぞれの未使用位置には DBCS スペースが埋め込まれます。 USAGE NATIONAL で記述された項目の場合は、それぞれの未使用位置にはデフォルトの Unicode スペース (NX'2000') が埋め込まれます。 それ以外の場合は、それぞれの未使用位置には英数字スペースが埋め込まれます。

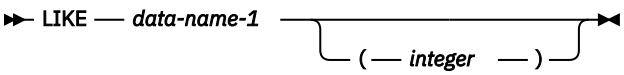
JUSTIFIED 節を省略した場合、標準位置合わせの規則に従います (142 ページの『位置合わせの規則』を参照)。

JUSTIFIED 節は、VALUE 節によって決められている初期設定値には影響を及ぼしません。

LIKE 文節

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。

フォーマット



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。

注 : 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ます。

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 166 ページの『FORMAT 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。

表 *16.* 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-4 COMPUTATIONAL-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)
```

OCCURS 節

テーブル操作に使用される DATA DIVISION 言語エレメントは、OCCURS 節と INDEXED BY 句です。

INDEXED BY 句の説明については、176 ページの『INDEXED BY 句』を参照してください。 .

OCCURS 節は、 指標付けまたは指標付けによって各エレメントを参照することのできるテーブルを指定します。 また、この節を使用すると、 繰り返されるデータ項目に対して別々の項目を指定する必要はなくなります。

OCCURS 節のフォーマットには、 固定長テーブルと可変長テーブルがあります。

OCCURS 節のサブジェクト は、その OCCURS 節を含んでいるデータ項目のデータ名です。 OCCURS 節それ自体を除き、そのサブジェクトと共に書かれたデータ記述節は、 記述された項目のそれぞれのオカレンスごとに適用されます。

OCCURS 節のサブジェクト、またはこのサブジェクトに従属しているいずれかのデータ項目が参照される場合は、添え字付けされるか指標付けされる必要があります。ただし、次の例外があります。

- 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用されている場合。
- OCCURS 節のサブジェクトが、形式 2 SORT ステートメントのサブジェクトとして使用されている場合。
- そのサブジェクトまたはその従属データ項目が ASCENDING/DESCENDING KEY 句のオブジェクトである場合。

- その従属データ項目が REDEFINES 文節のオブジェクトである場合。
- 従属データ項目が、LENGTH OF 特殊レジスターのサブジェクトとして使用されている場合。 詳しくは、18 ページの『LENGTH OF』を参照してください。

添え字付けまたは指標付けがなされている場合、ALL 添え字が組み込み関数で使用されていない限り、サブジェクトはテーブル内の 1 つのオカレンス項目を参照します。

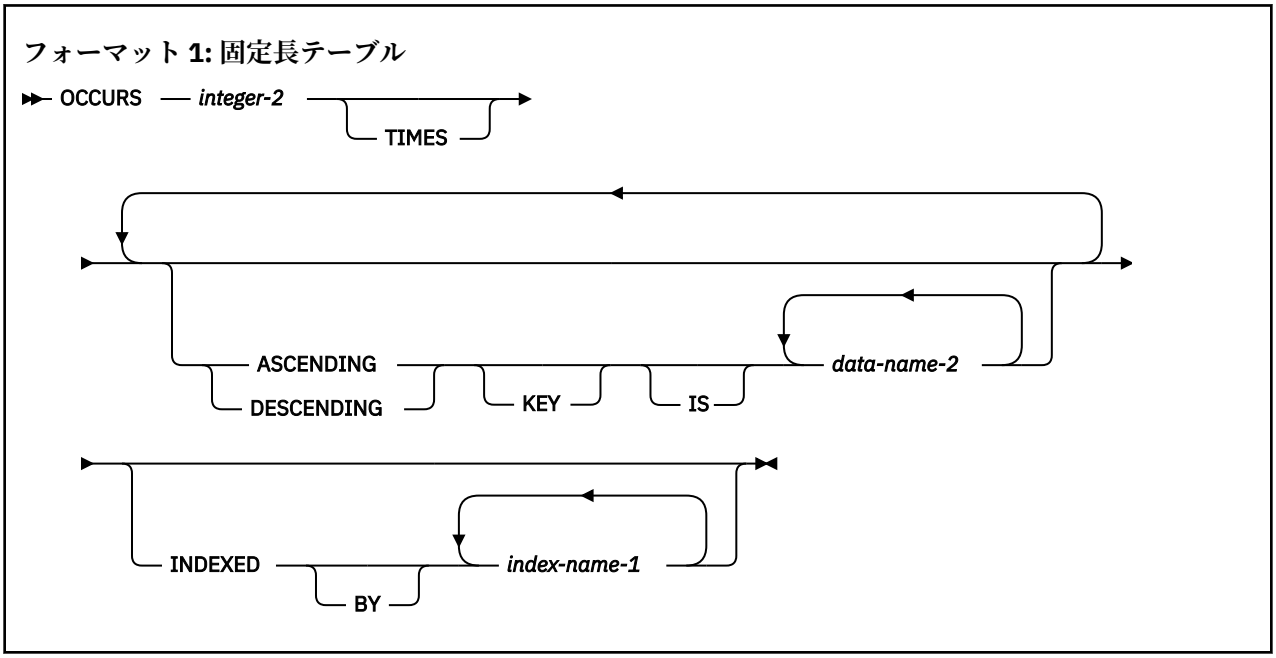
OCCURS 節は、次のようなデータ記述項目では指定できません。

- レベル番号が 01、66、77、または 88 の場合
- 再定義データ項目を記述している場合 (ただし、再定義された項目が OCCURS 節を含む項目に従属することは可能)。 199 ページの『REDEFINES 節』を参照してください。 .

固定長テーブル

固定長テーブルを指定するには、OCCURS 節を使用します。

7 つの添え字または指標が使用できるため、フォーマット 1 の OCCURS 節は、 最外部のレベルと 6 つのネストしたレベルが可能です。 フォーマット 1 の OCCURS 節は、OCCURS DEPENDING ON 節への従属節として指定できます。 この方法を使用すると、テーブルは 7 次元まで指定できます。



整数-2
正確なオカレンス項目数。 整数-2 は 0 より大きくなければなりません。

ASCENDING KEY 句と DESCENDING KEY 句

データは、データ名-2 に含まれる値に従い、指定されたキーワードに応じて昇順または降順に並べられます。 データ名は重要度の降順にリストされます。

順序は、オペランドの比較の規則に従って決められます (242 ページの『比較条件』を参照)。 ASCENDING KEY データ項目および DESCENDING KEY データ項目は、OCCURS 節、テーブル・エレメントの二分探索のための SEARCH ALL ステートメント、および形式 2 SORT ステートメントで使用されます。 別の方法として、形式 2 SORT ステートメントでキーを指定できます。

データ名-2
これは、サブジェクト項目の名前、またはサブジェクト項目に従属する項目の名前でなければなりません。 データ名-2 をウィンドウ表示日付フィールドにすることはできません。 データ名-2 は修飾することができます。

データ名-*2* がサブジェクト項目を指定している場合、 その項目の全体が ASCENDING KEY または DESCENDING KEY となり、 そのテーブル・エレメントに対して指定できる唯一のキーとなります。

データ名-*2* がサブジェクト項目の名前を指定しない場合、データ名-*2* は次のようになります。

- テーブル項目自体のサブジェクトに従属しなければなりません。
- OCCURS 節を含む他の項目に従属したりその後に指定することはできません。
- それら自体が、OCCURS 節を含むことはできません。

データ名-*2* では、OCCURS DEPENDING ON 節を含む従属項目は使用できません。

ASCENDING KEY 句または DESCENDING KEY 句を指定する場合、次の規則が適用されます。

- キーは、レベルの高いものから順に記入する必要があります。
- 1 つのテーブル・エレメントに対するキーの総数は 12 以下でなければなりません。
- テーブルの中のデータを、使用中の照合シーケンスに従って昇順または降順に並べておく必要があります。
- キーは、以下のいずれかの USAGE で記述される必要があります。
  - BINARY
  - DISPLAY
  - DISPLAY-1
  - NATIONAL
  - PACKED-DECIMAL
  - COMPUTATIONAL
  - COMPUTATIONAL-1
  - COMPUTATIONAL-2
  - COMPUTATIONAL-3
  - COMPUTATIONAL-4
  - COMPUTATIONAL-5
- USAGE NATIONAL で記述されたキーは、国別、国別編集、数字編集、数字、または外部浮動小数点のいずれかのカテゴリーになります。
- 1 つのテーブル・エレメントに関連付けられたすべてのキーの長さの合計は、256 以下でなければなりません。
- キーが修飾子なしで指定され、しかもそれが固有名でない場合、そのキーは、 暗黙のうちに OCCURS 節のサブジェクトおよび OCCURS 節のサブジェクトにかかわるすべての修飾子で修飾されます。

次の例は、ASCENDING KEY データ項目の指定方法を示したものです。

```
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 のキーは、その項目に従属し、WEEK-RECORD のキーは、その従属項目へ従属しています。

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は、SEARCH ALL ステートメント実行の結果は予測できません。

INDEXED BY 句

INDEXED BY 句は、テーブルで使用することができる指標を指定します。 INDEXED BY 句の指定がないテーブルは、別のテーブルに関連付けた索引付け名を使用して、指標付けをして 参照できます。

指標付けの使用について詳しくは、61 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。

通常は、指標はテーブルを含むプログラムに関連した静的メモリーに割り振られます。 したがって、プログラムに再入すると、指標は最後に使用された状態になります。 しかし、次の場合に指標は呼び出しごとに割り振られます。 したがって、次のセクション中のテーブルに関する指標の項目ごとに指標の値を設定しなければなりません。

- LOCAL-STORAGE SECTION
- LINKAGE SECTION は、以下のとおりです。
  - RECURSIVE 節を指定してコンパイルしたプログラム

外部データ・レコードで指定されている指標には、外部属性はありません。

指標名-*1*

各指標名は、プログラムの使用に備えてコンパイラーが作成する指標を指定します。 これらの指標名はデータ名ではない ので、 COBOL プログラム中の別の場所では識別されません。その代わりに、オブジェクト・プログラムの専用特殊レジスターとみなされます。 それらはデータではなく、データ階層の一部でもありません。

未参照の指標名を固有に定義する必要はありません。

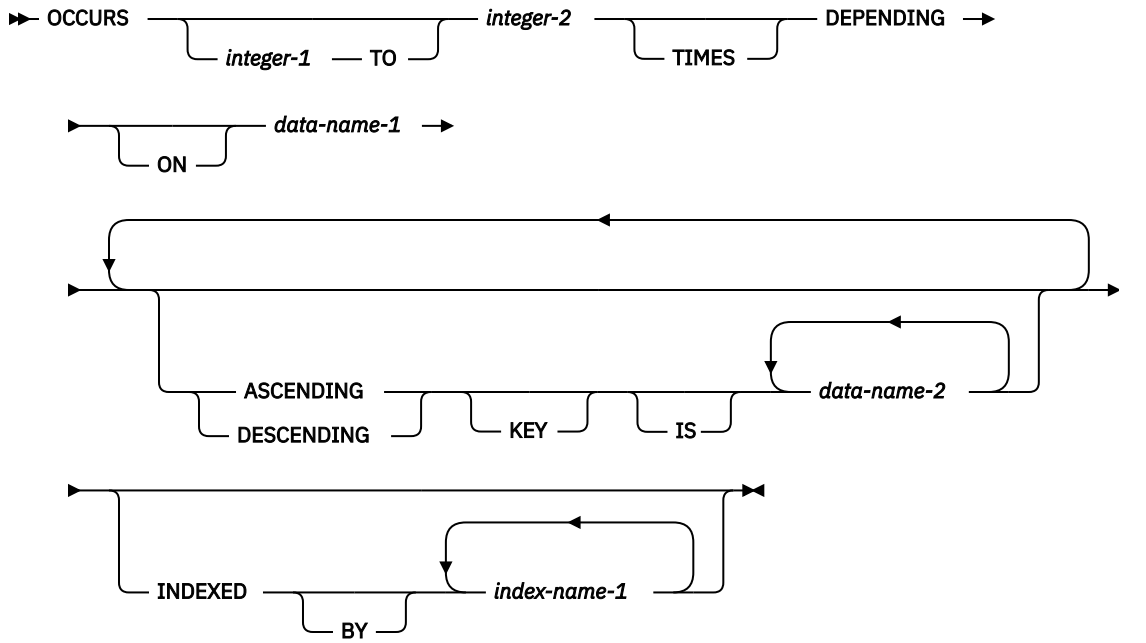
1 つのテーブル項目の中では、12 個までの指標名を指定できます。

グローバル属性のデータ項目が指標によってアクセスされるテーブルを含んでいる場合は、その指標もグローバル属性を持ちます。 したがって、指標名の有効範囲は、その索引が定義されているテーブルに名前を付けるデータ名の有効範囲と同じです。

可変長テーブル

可変長テーブルは、OCCURS DEPENDING ON 節を使用して指定できます。

フォーマット 2: 可変長テーブル



整数-*1*

最小の出現数。

整数-*1* の値は 0 以上でなければならず、整数-*2* の値未満でなければなりません。

整数-*1* を省略すると、その値は 1 と見なされ、そのキーワード TO も 省略しなければなりません。

整数-*2*

最大の出現数。

整数-*2* は 整数-*1* より大きくなければなりません。

サブジェクト項目の長さ は固定長です。 サブジェクト項目の繰り返される回数 のみが変数です。

OCCURS DEPENDING ON 節

OCCURS DEPENDING ON 節は、可変長テーブルを指定します。

データ名-*1*

OCCURS DEPENDING ON 節のオブジェクト、つまり、現行のサブジェクト 項目のオカレンス項目数を表す現行値を持つデータ項目を指定してください。 そのオカレンス項目数がオブジェクトの値を超えるような項目の内容は未定義です。

OCCURS DEPENDING ON 節 (データ名-*1*) のオブジェクトでは、整数データ項目を記述しなければなりません。 オブジェクトをウィンドウ表示日付フィールドにすることはできません。

OCCURS DEPENDING ON 節のオブジェクトは、 テーブルの範囲内の、どのストレージ位置 (つまりテーブル内の最初の文字位置から、テーブル内の最後の文字位置までのどのストレージ位置) にも置くことはできません。

OCCURS DEPENDING ON 節のオブジェクトの位置は自由にはなりません。 OCCURS DEPENDING ON 節を含む項目の後にこのオブジェクトを置くことはできません。

EXTERNAL 節を含んでいるレコード記述項目に含まれるデータ記述項目で OCCURS 節を指定する場合、データ名-*1* は、それを指定する場合、外部属性を持つデータ項目を参照しなければなりません。 データ名-*1* は、項目のサブジェクトと同じ DATA DIVISION に記述しなければなりません。

GLOBAL 節を含むデータ記述項目に従属するデータ記述項目の中で OCCURS 節を指定する場合、データ名-*1* は、それを指定する場合はグローバル名でなければなりません。 データ名-*1* は、項目のサブジェクトと同じ DATA DIVISION に記述しなければなりません。

OCCURS 節で使用されるすべてのデータ名を修飾できます。 ただし、添え字または指標を付けることはできません。

グループ項目、または従属 OCCURS DEPENDING ON 項目を含むデータ項目、または OCCURS DEPENDING ON 項目の後にあるが従属していないデータ項目が参照されるとき、OCCURS DEPENDING ON 節のオブジェクトの値は、 整数-*1* から整数-*2* の範囲内になければなりません。

オブジェクトの値が整数-*1* から整数-*2* までの範囲になければ、動作は確定しません。

従属する OCCURS DEPENDING ON 項目を含んでいるグループ項目が参照される場合、 その演算で使用されるテーブル区域の部分は、次のようにして決定されます。

- オブジェクトがグループの外にあるときは、 演算の開始に当たってオブジェクトによって指定されたテーブル区域の部分だけが使用されます。
- オブジェクトが同じグループ内に含まれ、そのグループ・データ項目が送り出し項目として参照される場合、 演算の開始に当たってオブジェクトの値によって指定されたテーブル区域だけが、演算で使用されます。
- オブジェクトが同じグループに含まれ、かつグループ・データ項目が受け取り項目として参照される場合、 グループ項目の最大長が演算で使用されます。

最大長の規則の適用によって影響が出るステートメントは、以下のものです。

- ACCEPT *ID* (フォーマット 1 とフォーマット 2)
- CALL ... USING BY REFERENCE *ID*
- MOVE ... TO *identifier*
- READ ... INTO *ID*
- RELEASE *ID* FROM ...
- RETURN ... INTO *identifier*
- REWRITE *ID* FROM ...
- STRING ... INTO *identifier*
- UNSTRING ... INTO *identifier* DELIMITER IN *identifier*
- WRITE *ID* FROM ...

可変長グループ項目の後に非従属項目が続かない場合、CALL ... USING BY REFERENCE *ID* の ID としてグループの最大長が出現したときにその最大長が使用されます。 したがって、グループの位置が変化する場合以外は、 OCCURS DEPENDING ON 節のオブジェクトを設定する必要はありません。

グループ項目の後に非従属項目が続く場合、最大長ではなく実際の長さが使用されます。 項目のサブジェクトが参照されるとき、または項目のサブジェクトに従属するか、それを従属させているデータ項目が参照されるとき、OCCURS DEPENDING ON 節のオブジェクトは、整数-*1* から整数-*2* の範囲内になければなりません

OCCURS DEPENDING ON 節の使用法によっては、複合 *OCCURS DEPENDING ON* (ODO) 項目が発生します。 以下の項目が、複合 ODO 項目を構成します。

- OCCURS DEPENDING ON 節を使用して記述されたデータ項目。後ろに OCCURS 節を使用して (または使用せずに) 記述された非従属基本データ項目が付く。
- OCCURS DEPENDING ON 節を使用して記述されたデータ項目。後ろに非従属グループ項目が付く。
- OCCURS DEPENDING ON 節を使用して記述された、1 つ以上の従属項目を含むグループ項目。
- OCCURS 節または OCCURS DEPENDING ON 節を使用して記述されたデータ項目で、OCCURS DEPENDING ON 節を使用して記述された従属データ項目を含む (可変長エレメントを含むテーブル)

- 可変長エレメントを含むテーブルに関連した指標名

OCCURS DEPENDING ON 節のオブジェクトは、複合 ODO 項目の後に続く非従属項目にすることはできません。

OCCURS DEPENDING ON 節を使用して記述された項目に続く非従属項目は、すべて可変位置項目 です。すなわち、その位置は、 OCCURS DEPENDING ON オブジェクトの値によります。

ファイル記述 (FD) 項目の中で暗黙の再定義が使用される場合、従属レベルの項目に OCCURS DEPENDING ON 節を含めることができます。

INDEXED BY 句を、OCCURS DEPENDING ON 節を含む従属項目を持つテーブルに対して指定することができます。

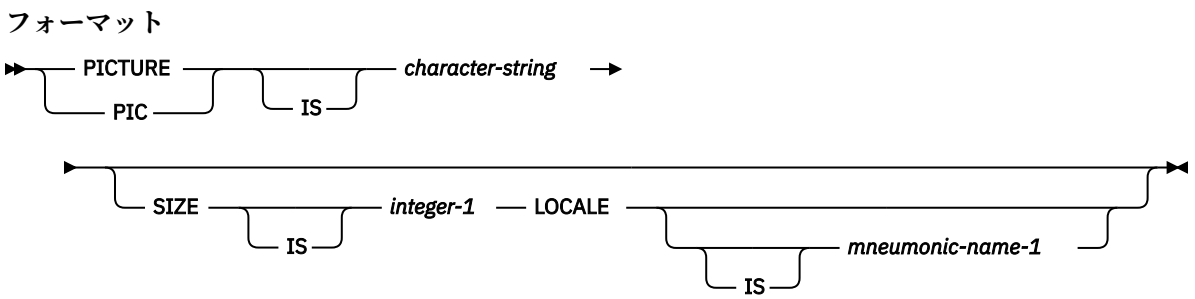
複合 OCCURS DEPENDING ON について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『複合 *OCCURS DEPENDING ON*』を参照してください。

ASCENDING KEY 句、DESCENDING KEY 句、および INDEXED BY 節については、174 ページの『固定長テーブル』で解説しています。

PICTURE 節

PICTURE 節は、基本項目の一般的な特性および編集上の要件を指定します。

フォーマット



PICTURE または PIC

PICTURE 節は、以下の場合を除き、 すべての基本項目ごとに指定しなければなりません。

- 指標データ項目
- LIKE、RENAMES、または TYPE 節のサブジェクト
- USAGE POINTER、USAGE FUNCTION-POINTER、または USAGE PROCEDURE-POINTER
- 内部浮動小数点データ項目
- 日付、時刻、またはタイム・スタンプのデータ項目

これらの除外例の場合には、ピクチャー節は使用できません。

PICTURE 節は、基本レベルにおいてのみ指定できます。

PIC は PICTURE の省略形で、同じ意味を持っています。

文字ストリング

文字ストリング は、ピクチャー記号として使用される特定の COBOL 文字から構成されます。 これらの文字の許容される組み合わせによって、基本データ項目のカテゴリーが決定されることになります。ロケール句が指定されている場合を除き、これらの文字の許容される組み合わせによって、基本データ項目のカテゴリーが決定されることになります。 PICTURE 節の LOCALE 句はカテゴリー数字編集項目を定義するものです。

文字ストリング の長さは、50 文字までです。

LOCALE
180 ページの『LOCALE 句』を参照してください。

LOCALE 句

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 142 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。

PICTURE 節で使用される記号

PICTURE 文字ストリング内で使用される句読記号は、句読記号とはみなされず、 PICTURE 文字ストリングの記号とみなされます。

DECIMAL-POINT IS COMMA を SPECIAL-NAMES 段落に指定すると、 PICTURE 文字ストリングおよび数字リテラルにおいて、ピリオドとコンマの機能を交換することができます。

PICTURE 記号を表す次の大文字に対応する小文字の英字は、 PICTURE 文字ストリングの中で、大文字で表されたものと同じ働きをします。

```
A, B, E, G, N, P, S,  V, X, Z, CR, DB
```

他のすべての小文字は、対応する大文字の表示と同じではありません。

各 PICTURE 節の記号の意味が下表に定義されています。

- LOCALE 句が指定されていない場合は、181 ページの表 17 を参照してください。
- LOCALE 句が指定されている場合は、183 ページの表 18 を参照してください。

サイズ という見出しは、項目内の文字位置の数を判別する際の項目のカウント方法を示します。文字位置のタイプは、項目に指定された USAGE 節によって決まります。

USAGE	文字位置のタイプ	文字当たりのバイト数
DISPLAY	英数字	1
DISPLAY-1	DBCS	2

USAGE	文字位置のタイプ	文字当たりのバイト数
NATIONAL	国別	2
その他すべて	概念上	該当なし

表 *17.* ***LOCALE*** **句が指定されていない場合の** ***PICTURE*** **節の記号の意味**

記号	意味	サイズ
A	ラテン・アルファベットまたはスペースのみを入れることのできる文字位置。	「A」はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
B	Usage DISPLAY の場合は、英数字スペースが挿入される文字位置。 Usage DISPLAY-1 の場合は、DBCS スペースが挿入される文字位置。 Usage NATIONAL の場合は、国別スペースが挿入される文字位置。	「B」はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
E	外部浮動小数点項目の指数の開始を示します。外部浮動小数点項目の詳細については、186 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。	「E」はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
G	DBCS 文字位置。	「G」はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
N	USAGE DISPLAY-1 を使用して指定した場合、または USAGE が指定されていないときに NSYMBOL(DBCS) コンパイラー・オプションが有効な場合の DBCS 文字位置。 国別カテゴリーの場合は、USAGE NATIONAL を使用して指定したとき、または USAGE が指定されていないときに NSYMBOL(NATIONAL) コンパイラー・オプションが有効なときの国別文字位置。 国別編集カテゴリーの場合は、国別文字位置。	「N」はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
P	想定小数部の位取り位置。 データ項目内の数字に小数点がない場合に、想定小数点の位置を指定するために使用します。 詳しくは、185 ページの『P 記号』を参照してください。	データ項目サイズの計算に入れられません。 位取り位置の文字は、数字編集項目、または算術オペランドとして使用される項目の最大桁数を判別する際には、桁数に含められます。 値のサイズは、 PICTURE 文字ストリングによって表される桁位置の数になります。
S	演算符号が存在することを示す標識 (符号を表すものではなく、したがって当然位置を示すものでもありません)。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 節が SEPARATE CHARACTER 句を指定する場合を除き (1 文字位置としてカウントされる)、基本項目の計算に入れられません。
V	想定小数点の位置を表す標識。 文字位置を表しません。 想定小数点が、ストリングの中の右端の記号の右側にあるとき、その V は冗長です。	基本項目サイズの計算には入れられません。
X	コンピューターの英数字文字セットの任意の使用可能文字を入れることのできる文字位置。	「X」はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。

表 *17.* ***LOCALE*** **句が指定されていない場合の** ***PICTURE*** **節の記号の意味** *(続き)*

記号	意味	サイズ
Z	先頭の数字位置。 その位置に 0 が入っていると、その 0 はスペース文字で置き換えられます。	「Z」はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
9	数表示を含む文字位置。	9 は、それぞれ項目の値の 1 つの 10 進数です。 USAGE が DISPLAY および NATIONAL の場合、「9」はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
0	数字の 0 が挿入される文字位置。	「0」はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
1	ブール値 B"1" または B"0" を含む文字位置。 USAGE は明示的または暗黙的に DISPLAY として定義されていなければなりません。	ブール値はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
/	スラッシュ文字が挿入される文字位置。	スラッシュ文字 (/) はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
,	コンマが挿入される文字位置。	コンマ (,) はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
.	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、PICTURE 節はそのデータ記述項目の最後の節でなければならず、直後に分離文字ピリオドが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のものが使用されます。 **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。 この交換では、ピリオドやコンマが PICTURE 節内のどこにあっても、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。	ピリオド (.) はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
+ - CR DB	編集符号制御記号。 それぞれの記号は、編集符号制御記号が入れられる文字位置を表します。	編集符号記号で使用される文字はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
*	金額変造防止記号。先頭部分の数字位置で、その部分に 0 が入っているときにアスタリスクが入れられます。	アスタリスク (*) はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
cs	*cs* は、任意の有効な通貨記号にできます。 通貨記号は通貨符号値が入れられる文字位置を表します。 デフォルトの通貨記号は、コンパイル時に有効であるコード・ページの 値 X'24' を割り当てられた文字です。 本書において、デフォルト通貨記号はドル記号 ($) で表され、*cs* は任意の有効な通貨記号を表します。 詳しくは、186 ページの『通貨記号』を参照してください。	通貨記号が最初に出現した時点で、通貨符号値の文字数がデータ項目のサイズに加算されます。 それ以降の出現のたびに、1 文字位置がデータ項目のサイズに加算されます。

表 *18.* ***LOCALE*** **句が指定されている場合の** ***PICTURE*** **節の記号の意味**		
記号	**意味**	**サイズ**
9	数字を含み、編集項目に入れることができる数字の数にカウントされる文字位置。	9 は、それぞれ項目の値の 1 つの 10 進数です。USAGE が DISPLAY および NATIONAL の場合、「9」はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
.	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、PICTURE 節はそのデータ記述項目の最後の節でなければならず、直後に分離文字ピリオドが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のものが使用されます。 **注 :** ある種のプログラムでは、SPECIAL-NAMES 段落で DECIMAL-POINT IS COMMA 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。 この交換では、ピリオドやコンマが PICTURE 節内のどこにあっても、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。	ピリオド (.) はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
+	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、編集項目は符号なしになります。	+ はそれぞれ、1 つの文字位置としてデータ項目のサイズにカウントされます。
cs	文字ストリング内の通貨記号は、指定されたロケールに関連付けられた通貨ストリングを編集項目に含めることを指示します。	通貨記号が最初に出現した時点で、通貨符号値の文字数がデータ項目のサイズに加算されます。それ以降の出現のたびに、1 文字位置がデータ項目のサイズに加算されます。

184 ページの図 1 は、LOCALE 句が指定されていない場合に、PICTURE 節の記号を指定する際に従わなければならない順序を示しています。 図の終わりにある注を参照してください。 185 ページの図 2 は、LOCALE 句が指定されている場合に、PICTURE 節の記号を指定する際に従わなければならない順序を示しています。

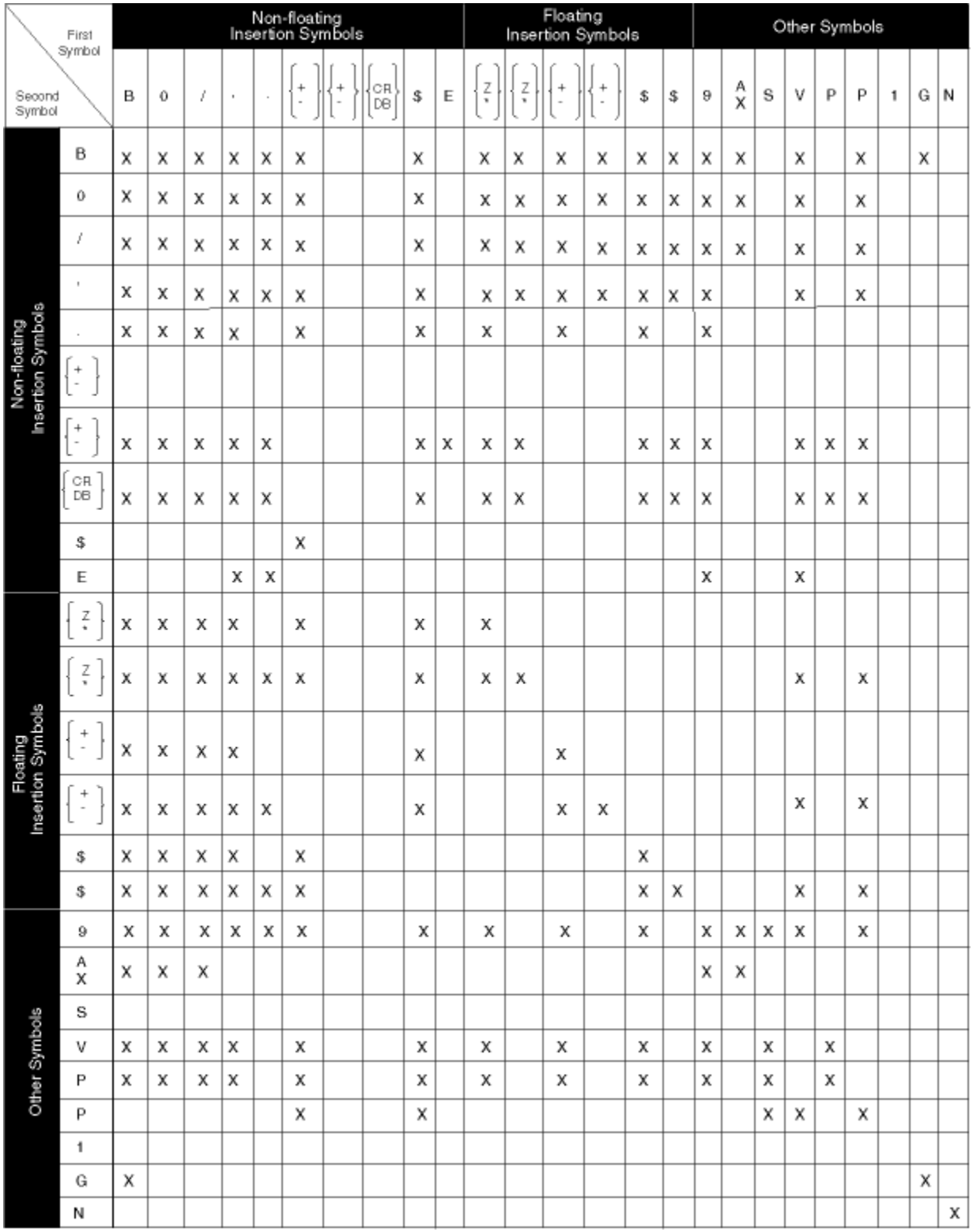


図 *1. LOCALE* 句が指定されていない場合の *PICTURE* 節の記号の順序

184 ページの図 1 に対する注:

1. 交差部分の X は、最上列の記号が、当該文字ストリング内で、行の左にある記号の左側のどこにあっても良いことを示しています。
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. 記号 A、X、Z、9、または * の少なくとも 1 つ、あるいは記号 +、-、または $ 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。
4. 記号 G または N は、PICTURE 文字ストリングに単独で入れることができます。
5. 非浮動挿入記号 +、-、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。 左端の列と最上段の行に示された各記号は、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 回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。

First Symbol / Second Symbol		Symbols			
		9	CS	.	+
Symbols	9	X	X	X	X
	CS				X
	.	X	X		X
	+				

図 *2. LOCALE* 句が指定されている場合の *PICTURE* 節の記号の順序

P 記号

記号 P は位取り位置を指定し、想定小数点を暗黙指定します (一連の P が左端の PICTURE 文字であればそれらの P の左側、右端の PICTURE 文字であればそれらの P の右側)。

想定小数点の記号 V は、このような PICTURE 記述内の左端または右端の文字としては冗長です。

記号 P は、PICTURE 文字ストリング内の左端または右端の桁位置に、連続した P のストリングとしてのみ指定できます。

PICTURE 文字ストリングに記号 P を含んでいるデータ項目を参照するある種の演算では、そのデータ項目の実際の文字表現ではなく、データ項目の代数値が使用されます。 代数値は所定の位置に小数点を、記号 P で指定された桁位置に 0 を想定したものです。値のサイズは、 PICTURE 文字ストリングによって表される桁位置の数になります。 このような演算には以下があります。

- 数字の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あり、その PICTURE 文字ストリングが記号 P を含んでいる MOVE ステートメント
- 送り出しオペランドが数字編集であり、PICTURE 文字ストリングが記号 P を含み、受け取りオペランドが数字または数字編集である MOVE ステートメント
- 送り出しと受け取りの両方のオペランドが数字である比較演算

上記以外のすべての演算では、記号 P で指定された桁位置は無視され、オペランドのサイズのカウントには入れられません。

通貨記号

PICTURE 文字ストリング内の通貨記号は、デフォルトの通貨記号 $ で表されるか、 CURRENCY コンパイラー・オプションで、または ENVIRONMENT DIVISION の SPECIAL-NAMES 段落の CURRENCY SIGN 節のいずれかで 指定する単一文字によって表されます。

CURRENCY SIGN 節が指定されている場合、CURRENCY および NOCURRENCY コンパイラー・オプションは無視されます。 CURRENCY SIGN 節が指定されない場合に NOCURRENCY コンパイラー・オプションが有効であれば、デフォルトの通貨符号値および 通貨記号としてドル記号 ($) が使用されます。 CURRENCY SIGN 節については、97 ページの『CURRENCY SIGN 節』を参照してください。 CURRENCY および NOCURRENCY コンパイラー・オプションについて詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」内の『*CURRENCY*』 を 参照してください。

通貨記号を、PICTURE 文字ストリング内で繰り返し、 挿入が一定しないケースを指定することができます。 同じ PICTURE 文字ストリングで異なる通貨記号を使用することはできません。

他のすべての PICTURE 記号とは異なり、通貨記号は大/小文字が区別されます。例えば、'D' と 'd' は異なる通貨記号を指定します。

通貨記号は、USAGE DISPLAY で数字編集項目を定義する目的でのみ使用できます。

文字ストリングの表現

このトピックでは、PICTURE 文字ストリング内に 1 回以上現れる記号について説明します。

2 回以上指定できる記号
次の記号は、1 つの PICTURE 文字ストリングの中に 2 回以上指定することができます。

```
A  B  G  N  P  X   Z  9  0  /  ,  +  -  *  cs
```

記号 A、G、N、X、Z、9、または * の少なくとも 1 つ、または記号 +、–、または *cs* の少なくとも 2 つは、PICTURE ストリングの中になければなりません。

これらの記号のすぐ後にある小括弧で囲まれた、符号なしのゼロ以外の整数は、 その記号が連続する回数を指定します。

例: 次の 2 つの PICTURE 節は、同じことを指定しています。

```
PICTURE IS $99999.99CR
PICTURE IS $9(5).9(2)CR
```

1 回だけ指定できる記号
次の記号は、1 つの PICTURE 文字ストリングの中に 1 回だけしか指定できません。

```
E  S  V  .  CR  DB  1
```

PICTURE 記号の V を除き、PICTURE 文字ストリングに上記のいずれかの記号が出現した場合、それぞれは、データ項目内にその文字または一連の有効な文字があることを表します。

LOCALE 句が指定されている場合、複数回指定できる記号は 9 のみです。 LOCALE 句が指定されている場合、以下の記号は、1 つの PICTURE 文字ストリング内で 1 回のみ指定できます。

```
.  +  cs
```

データ・カテゴリーと PICTURE の規則

PICTURE 記号を許容された範囲で組み合わせることによって、その項目のデータ・カテゴリーが決まります。

データ・カテゴリーは以下のとおりです。

- 英字
- 英数字
- 英数字編集
- ブール
- DBCS
- 外部浮動小数点
- 国別
- 国別編集
- 数字
- 数字編集

注 :

- 内部浮動小数点カテゴリーは、COMP-1 または COMP-2 句を指定する USAGE 節によって定義されます。
- 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 だけを含めることができます。

項目の内容は、ラテン・アルファベットとスペース文字のみで構成されていなければなりません。

その他の節

USAGE DISPLAY を指定するか、または暗黙に指定されている必要があります。

関連した VALUE 節では、英字のみ、SPACE、 または形象定数の値を持つシンボリック文字を含む英数字リテラルを指定しなければなりません。

有効なランタイム・ロケールは、DBCS 文字を含むコード・ページを示している必要があります。 ロケールについては、563 ページの『付録 H ロケールの考慮事項』を参照してください。

DBCS データ項目には 1 バイト文字を含めないでください。

DBCS データ項目に埋め込みが必要な場合、次の規則が適用されます。

- データ域が満たされるまで (データ項目に割り振られた 2 バイト文字数を基準)、埋め込みは 2 バイトのスペース文字を使用して行われます。
- 埋め込みに奇数バイトが必要な場合 (例えば、英数字グループ項目を DBCS データ項目に移動する場合)、埋め込みは 1 バイトのスペース文字を使用して行われます。

ブール項目

次の規則が適用されます。

- PICTURE 文字ストリングには、記号 1 だけを含めることができます。
- 「1」の 1 文字のみを指定できます。
- 項目の USAGE には DISPLAY のみを使用できます。
- 関連する VALUE 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する必要があります。
- 以下の節はブール項目には指定できません。
  - SIGN 節
  - BLANK WHEN ZERO 節
  - ASCENDING/DESCENDING KEY 節

英数字項目

PICTURE 文字ストリングは、特定の記号で構成されていなければなりません。

記号は以下のとおりです。

- 1 つ以上の、記号 X のオカレンス
- 記号 A、X、および 9 の組み合わせ (A だけまたは 9 だけで構成される文字ストリングは、 英数字項目を定義するものではありません。)

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、 コンピューターの文字セットで許容された任意の文字にすることができます。

その他の節

USAGE DISPLAY を指定するか、または暗黙に指定されている必要があります。

関連する VALUE 節では、英数字リテラル、または以下の形象定数のいずれか 1 つを指定しなければなりません。

- ZERO
- SPACE
- QUOTE
- HIGH-VALUE
- LOW-VALUE
- シンボリック文字
- ALL 英数字リテラル

英数字編集項目

PICTURE 文字ストリングには、記号 `A X 9 B 0 /` を含めることができます。

ストリングには、少なくとも 1 つの A または X、および少なくとも 1 つの B または 0 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、コンピューターの文字セットで許容される 2 つ以上の任意の文字を必要とします。

その他の節

USAGE DISPLAY を指定するか、または暗黙に指定されている必要があります。

関連する VALUE 節では、英数字リテラル、または以下の形象定数のいずれか 1 つを指定しなければなりません。

- ZERO
- SPACE
- QUOTE
- HIGH-VALUE
- LOW-VALUE
- シンボリック文字
- ALL 英数字リテラル

リテラルは指定されたとおりに扱われ、編集はされません。

DBCS 項目

PICTURE 文字ストリングには、記号 G、G と B、または N を含めることができます。それぞれの G、B または N は、1 文字の DBCS 文字位置を表すことができます。

関連する VALUE 節は、DBCS リテラル、形象定数 SPACE、または形象定数 ALL *DBCS* リテラル を含まなければなりません。

その他の節

有効なランタイム・ロケールは、DBCS 文字を含むコード・ページを示している必要があります。 ロケールについては、563 ページの『付録 H ロケールの考慮事項』を参照してください。

DBCS データ項目には 1 バイト文字を含めないでください。

DBCS データ項目に埋め込みが必要な場合、次の規則が適用されます。

- データ域が満たされるまで (データ項目に割り振られた 2 バイト文字数を基準)、埋め込みは 2 バイトのスペース文字を使用して行われます。
- 埋め込みに奇数バイトが必要な場合 (例えば、英数字グループ項目を DBCS データ項目に移動する場合)、埋め込みは 1 バイトのスペース文字を使用して行われます。

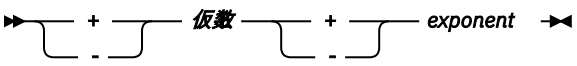
PICTURE 記号 G を使用する場合には、USAGE DISPLAY-1 を指定しなければなりません。 PICTURE 記号 N が使用され、NSYMBOL(DBCS) コンパイラー・オプションが有効であるときに、USAGE 節を省略した場合には、USAGE DISPLAY-1 が暗黙に指定されます。

外部浮動小数点項目

データ項目は、その PICTURE 文字ストリングによって外部浮動小数点カテゴリーとして記述されます。

PICTURE 文字ストリングについては以下に詳しい説明があります。

フォーマット



+ または -
符号文字は、仮数のすぐ前と指数のすぐ前に付けなければなりません。

+ 符号は、出力において正の値を正符号で、 負の値を負符号で表すことを指示します。

- 符号は、出力において正の値の符号部分をブランクで、 負の値を負符号で表すことを指示します。

それぞれの符号位置は、ストレージにおいて 1 バイトを占有します。

仮数
仮数は記号として次のものを含むことができます。

```
9 . V
```

実際の小数点はピリオド (.) で表せますが、 想定小数点は V で表されます。

仮数には実際の小数点か想定小数点のどちらかがなければなりません。 小数点は先頭、中間、末尾のどれでも可能です。

仮数には、1 から 16 桁の数字を含めることができます。

E
指数を示します。

指数 *(exponent)*
指数は、記号 99 で構成されなければなりません。

例: Pic -9v9(9)E-99

USAGE 節の DISPLAY 句と浮動小数点 PICTURE 文字ストリングは、項目を *DISPLAY* 浮動小数点データ項目として定義します。

USAGE 節の NATIONAL 句と浮動小数点 PICTURE 文字ストリングは、項目を国別浮動小数点データ項目 として定義します。

USAGE DISPLAY を指定して定義される項目では、V 以外のピクチャー記号はそれぞれ、項目内の 1 つの英数字文字位置を定義します。

USAGE NATIONAL を指定して定義される項目では、V 以外のピクチャー記号はそれぞれ、項目内の 1 つの国別文字位置を定義します。

その他の節

USAGE 節の DISPLAY 句または NATIONAL 句は、指定するかまたは暗黙に指定されている必要があります。

LIKE 節、OCCURS 節、REDEFINES 節、RENAMES 節、および TYPEDEF 節は、外部浮動小数点項目に関連付けることができます。

SIGN 節は説明文として受け入れられ、符号の表現には作用しません。

SYNCHRONIZED 節は説明文として扱われます。

以下の節は、外部浮動小数点項目では無効です。

- BLANK WHEN ZERO
- JUSTIFIED
- VALUE

国別項目

PICTURE 文字ストリングには、ピクチャー記号 N の、1 つ以上のオカレンスを含めることができます。

上記の規則は、NSYMBOL(NATIONAL) コンパイラー・オプションが有効であるとき、 および USAGE NATIONAL 節が指定されているときに適用されます。 USAGE NATIONAL 節が存在しないときに、NSYMBOL(DBCS) コンパイラー・オプションが有効である場合には、 ピクチャー記号 N が DBCS 文字を表し、DBCS 項目の PICTURE 節の規則が適用されます。

それぞれの N は、単一の国別文字位置を表します。

関連する VALUE 節では、英数字リテラル、国別リテラル、または以下の形象定数のいずれか 1 つを指定しなければなりません。

- ZERO
- SPACE
- QUOTE
- HIGH-VALUE
- LOW-VALUE
- シンボリック文字
- ALL 英数字リテラル
- ALL 国別リテラル

その他の節

USAGE 節では、NATIONAL 句のみを指定できます。 PICTURE 記号 N が使用され、NSYMBOL(NATIONAL) コンパイラー・オプションが 有効であるときに、Usage 節を省略した場合には、USAGE NATIONAL が暗黙に指定されます。

以下の節は、使用できます。

- JUSTIFIED
- EXTERNAL
- GLOBAL
- OCCURS
- REDEFINES
- RENAMES

- SYNCHRONIZED
- TYPEDEF

以下の節は、使用できません。

- BLANK WHEN ZERO
- DATE FORMAT
- FORMAT
- SIGN
- TYPE

国別編集項目

PICTURE 文字ストリングには、少なくとも 1 つの記号 N、および記号 B 0 (ゼロ) または / (スラッシュ) の少なくとも 1 つのインスタンスが含まれていなければなりません。

それぞれの記号は、単一の国別文字位置を表します。

関連する VALUE 節では、英数字リテラル、国別リテラル、または以下の形象定数のいずれか 1 つを指定しなければなりません。

- ZERO
- SPACE
- QUOTE
- HIGH-VALUE
- LOW-VALUE
- シンボリック文字
- ALL 英数字リテラル
- ALL 国別リテラル

リテラルは指定されたとおりに扱われ、編集はされません。

NSYMBOL(NATIONAL) コンパイラー・オプションは、国別編集カテゴリーのデータ項目の定義には影響しません。

その他の節

USAGE NATIONAL を指定するか、または暗黙に指定されている必要があります。

以下の節は、使用できます。

- JUSTIFIED
- EXTERNAL
- GLOBAL
- OCCURS
- REDEFINES
- RENAMES
- SYNCHRONIZED
- TYPEDEF

以下の節は、使用できません。

- BLANK WHEN ZERO
- DATE FORMAT
- FORMAT
- SIGN

- TYPE

数字項目

数字項目にはいくつかのタイプがあります。

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)

下記の表に示すように、数字項目のタイプは USAGE 節によって定義されます。

表 *19.* **数値のタイプ**	
タイプ	**USAGE 節**
2 進数	BINARY、COMP、COMP-4、または COMP-5
内部 10 進数	PACKED-DECIMAL、COMP-3
ゾーン 10 進数 (外部 10 進数)	DISPLAY
国別 10 進数 (外部 10 進数)	NATIONAL

数値日付フィールドの場合、PICTURE 文字ストリングには記号 9 および S のみを使用できます。 その他すべての数値フィールドの場合、PICTURE 文字ストリングには記号 9、P、S、および V のみを使用できます。

記号 S は、PICTURE 文字ストリングの左端の文字としてのみ書くことができます。

記号 V は、PICTURE 文字ストリング内で一度だけ書くことができます。

2 進数項目の場合は、数字の桁数は 1 から 18 桁でなければなりません。 パック 10 進数項目およびゾーン 10 進数項目の場合、数字の桁数は、ARITH(COMPAT) コンパイラー・オプションが有効なときは 1 から 18 桁で、ARITH(EXTEND) コンパイラー・オプションが有効なときは 1 から 31 桁でなければなりません。

数字日付フィールドの場合、桁数は DATE FORMAT 節で指定された文字数と一致していなければなりません。

符号なしの場合、 標準データ・フォーマットの項目の内容は、0 から 9 のアラビア数字の組み合わせを入れなければなりません。 符号付きの場合、+、-、またはその他の表現の演算符号を含めることができます。

有効範囲の例

```
 PICTURE   値の有効範囲
   9999     0 から 9999
    S99    -99 から +99
 S999V9    -999.9 から +999.9
 PPP999     0 から .000999
S999PPP    -1000 から -999000 および
           +1000 から +999000 または 0
```

その他の節

項目の USAGE は、DISPLAY、 NATIONAL、 BINARY、 COMPUTATIONAL、 PACKED-DECIMAL、 COMPUTATIONAL-3、 COMPUTATIONAL-4、 または COMPUTATIONAL-5 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。

TRUNC コンパイラー・オプションは、数値データ項目の使用に影響を与えます。詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『*TRUNC*』を参照してください。

数字編集項目

PICTURE 文字ストリングには、特定の記号を含めることができます。

記号は以下のとおりです。

```
B  P  V  Z  9  0  /  ,  .  +  -  CR  DB  *  cs
```

許容される記号の組み合わせは、PICTURE 節の許容された記号順序 (180 ページの『PICTURE 節で使用される記号』の表を参照) および編集規則 (193 ページの『PICTURE 節の編集』を参照) によって決められます。

次の規則が適用されます。

- 項目には BLANK WHEN ZERO 節を指定するか、または次の記号のうち少なくとも 1 つをストリングに含める必要があります。

  ```
  B  /  Z  0  ,  .  *  +  -  CR  DB  cs
  ```

- 以下の記号のうちの 1 つのみを、PICTURE 文字ストリングに書くことができます。

  ```
  +  -  CR  DB
  ```

- ARITH(COMPAT) コンパイラー・オプションが有効な場合は、 文字ストリング内で表される数字の桁数は、1 から 18 桁でなければなりません。 ARITH(EXTEND) コンパイラー・オプションが有効な場合は、文字ストリング内で表される数字の桁数は、1 から 31 桁でなければなりません。
- ストリング内の文字位置の総数 (編集文字の桁数を含める) 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。

その他の節

USAGE DISPLAY または NATIONAL を指定するか、または暗黙に指定されている必要があります。

項目の USAGE が DISPLAY の場合、関連付けられた VALUE 節には、英数字リテラルまたは形象定数を指定する必要があります。値は編集せずに割り当てられます。

項目の USAGE が NATIONAL の場合、関連付けられた VALUE 節には、英数字リテラル、国別リテラル、または形象定数を指定する必要があります。 値は編集せずに割り当てられます。

PICTURE 節の編集

PICTURE 節で編集を行う一般的な方法には、挿入による編集と、抑止および置換による編集の 2 種類があります。

挿入による編集には、以下のタイプの編集があります。

- 単純挿入
- 特別挿入
- 固定挿入
- 浮動挿入

抑止および置換による編集には、以下のタイプの編集があります。

- ゼロ抑制とアスタリスクによる置換
- ゼロ抑制とスペースによる置換

個々の項目に関して許容される編集のタイプは、その項目のデータ・カテゴリー によって異なります。 各カテゴリーに対してどのような編集のタイプが有効であるかを、以下の表に示します。 *cs* は、任意の有効な通貨記号を表します。

表 *20.* **データ・カテゴリー**		
データ・カテゴリー	**編集のタイプ**	**挿入記号**
英字	なし	なし
英数字	なし	なし
英数字編集	単純挿入	B 0 /
ブール	なし	なし
DBCS	単純挿入	B
外部浮動小数点	特別挿入	.
国別	なし	なし
国別編集	単純挿入	B 0 /
数字	なし	なし
数字編集	単純挿入 特別挿入 固定挿入 浮動挿入 ゼロ抑制 置換	B 0 / , . *cs* + - CR DB *cs* + - Z * Z * + - *cs*

編集のタイプについて、次のセクションで説明します。

- 194 ページの『単純挿入による編集』
- 195 ページの『特別挿入による編集』
- 195 ページの『固定挿入による編集』
- 196 ページの『浮動挿入による編集』
- 198 ページの『ゼロ抑制と置換による編集』

単純挿入による編集

このタイプの編集は、英数字編集、数字編集、および DBCS 項目に対して有効です。

各挿入記号は、項目のサイズに数えられ、それに等価の文字が挿入される項目内の位置を表します。 編集済み DBCS 項目では、挿入記号 B は、それぞれ項目のサイズに数えられ、 DBCS スペースが挿入される項目内の位置を表します。

次に例を示します。

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,*b*234,*b*000[1]
99,999	12345	12,345
GGBBGG	D1D2D3D4	D1D2*bbbb*D3D4[1]

PICTURE	データの値	編集結果
注: 1. 記号 *b* はブランク・スペースを表します。		

特別挿入による編集

このタイプの編集は、数字編集項目、または外部浮動小数点項目に対して有効です。

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。

注 : DECIMAL-POINT IS COMMA 節が指定されている場合は、ピリオドの代わりにコンマが使用されます。

ピリオド挿入記号は、項目のサイズに数えられ、その項目内で実際の小数点が挿入されるはずの位置を表します。

1 つの PICTURE 文字ストリングの中に、実際の小数点か、 または想定小数点として記号 V か、そのどちらか (両方ではなく) を指定しなければなりません。

次に例を示します。

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

固定挿入による編集は、数字編集項目にのみ有効です。

次のような挿入記号が使用されます。

- *cs*
- + - CR DB (編集用符号制御記号)

固定挿入による編集では、PICTURE 文字ストリングの中に 1 つの通貨記号と 1 つの編集用符号制御記号だけを指定することができます。

前に + または - の記号が付く場合を除き、 この文字ストリングの最初の文字は、通貨記号でなければなりません。

+ または - のどちらかが記号として使用されている場合、それは、 文字ストリングの中で最初か最後の文字でなければなりません。

CR または DB が記号として使用されている場合、それは、 文字ストリングの中で右端の 2 つの文字位置を占有します。 これら 2 つの文字位置が記号 CR または DB を含む場合、 大文字の英字が挿入文字です。

編集用符号制御記号によって作られる結果は、次に示すように、 データ項目の値に応じて異なります。

PICTURE 文字ストリング内の編集記号	結果: データ項目正の値またはゼロ	結果: データ項目負の値
+	+	-
-	スペース	-
CR	2 つのスペース	CR
DB	2 つのスペース	DB

次に例を示します。

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
-$999.99	-123.456	-$123.45
-$999.99	+123.456	$123.45
$9999.99CR	+123.45	$0123.45
$9999.99CR	-123.45	$0123.45CR

浮動挿入による編集

浮動挿入による編集は、数字編集項目にのみ有効です。

次に示す記号が使用されます。

```
cs + -
```

1 つの PICTURE 文字ストリング内では、これらの記号は、 浮動挿入文字として相互に排他的で、これらのうち 1 種類しか使用できません。

浮動挿入による編集を指定するには、許容される浮動挿入記号を少なくとも 2 つ含むストリングを使用することによって、 文字を実際に挿入することのできる左端の文字位置を表します。

文字ストリングの中の左端の浮動挿入記号は、実際に文字がデータ項目の中に現れる左端の限界を表します。 右端の浮動挿入記号は、実際に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。 ゼロ以外の数字データは、この限界とそれより右にあるすべての文字に置き変わることができます。

浮動挿入記号のストリングの中およびすぐ右側にある単純挿入記号 (B 0 / ,) は、 いずれも浮動文字ストリングの一部とみなされます。 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。

切り捨てのオカレンスを回避するために、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
- 受け取り項目の非浮動挿入記号の数
- 浮動挿入記号用の 1 文字位置

浮動挿入による編集の表現

PICTURE 文字ストリングで浮動挿入による編集を表す方法は、2 とおりあります。 したがって、次のように 2 とおりの編集が行われます。

1. 小数点の左側にある任意の先行数字文字位置またはそのすべてが、浮動挿入記号により表されます。 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、 浮動挿入文字が 1 つ置かれます。 挿入された文字の左側の文字位置は、スペースで埋められます。

   PICTURE 文字ストリングの中のすべての数字文字位置が挿入文字によって表されている場合、 少なくともそれら挿入文字のうちの 1 つは、小数点の左側になければなりません。
2. すべての数字文字位置を、浮動挿入記号によって表します。 編集が行われると、次のようになります。
   - データの値が 0 であれば、その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、その結果は規則 1 の場合と同様になります。

次に例を示します。

PICTURE	データの値	編集結果
$$$$.99	.123	$.12
$$$9.99	.12	$0.12
$,$$$,999.99	-1234.56	$1,234.56
+,+++,999.99	-123456.789	-123,456.78
$$,$$$,$$$.99CR	-1234567	$1,234,567.00CR
++,+++,+++.+++	0000.00	

ゼロ抑制と置換による編集

ゼロ抑制と置換による編集は、数字編集項目にのみ有効です。

ゼロ抑制による編集では、記号 Z および * が使用されます。 これらの記号は、PICTURE 文字ストリングの中で相互に排他的でどちらか一方しか使用できません。

以下に示す記号は、PICTURE 文字ストリングの中で浮動置換記号として相互に排他的でいずれか 1 つしか使用できません。

Z * + - *cs*

ゼロ抑制と置換による編集を行うには、許容されている記号を 1 つ以上含むストリングを使用して、 ゼロ抑制と置換による編集が可能な左端の文字位置を現します。

浮動編集記号のストリング内か、またはそのすぐ右側の単純挿入記号 (B 0 / ,) は、 そのストリングの一部とみなされます。 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。

ゼロ抑制の表現

PICTURE 文字ストリングには、ゼロ抑制を表現する方法が 2 とおりあり、 それに応じて 2 とおりの編集を行うことができます。

1. 小数点の左側にある任意の先行数字文字位置、またはそれらの位置のすべてを抑止記号によって表します。 編集が行われると、データ内で抑止記号と同じ文字位置に現れる先行ゼロは、置換用文字で置換されます。 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。
   - 抑止記号に対応していない文字
   - ゼロ以外のデータを含む文字
   - 小数点
2. PICTURE 文字ストリング内のすべての数字文字位置を抑止記号で表します。 編集が行われて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 0 の場合は、次のようになります。
   - Z が指定されていた場合には、データ全体にスペースが入れられます。
   - * が指定されていた場合には、実際の小数点を除いて、データ項目全体にアスタリスクが入れられます。

次に例を示します。

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

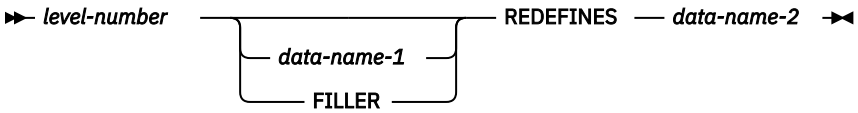
PICTURE	データの値	編集結果
Z,ZZZ.ZZ+	+123.456	123.45+
*,***.**+	-123.45	**123.45-
,*,***.**+	+12345678.9	12,345,678.90+
$Z,ZZZ,ZZZ.ZZCR	+12345.67	$ 12,345.67
$B*,***,***.**BBDB	-12345.67	$ ***12,345.67 DB

同じ項目に対して、抑止記号としてのアスタリスク (*) と BLANK WHEN ZERO 節の両方を指定することはできません。

REDEFINES 節

REDEFINES 節によって、異なるデータ記述項目を使用してコンピューターの同じストレージ域を記述することができます。

フォーマット



(レベル番号、データ名-*1*、および FILLER は、REDEFINES 節の一部ではありませんが、表現を明確にするためにのみこのフォーマットの中に加えたものです。)

指定する場合には、REDEFINES 節は、データ名-*1* または FILLER に続く最初の項目で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 節は、レベル番号に続く最初の項目である必要があります。

データ名-*1*、 FILLER

データ名-*2* で識別されるデータ域に関する記述を示し、どちらか一方を選択します。 データ名-*1* は再定義する 項目、すなわち REDEFINES のサブジェクト です。

データ名-*1* およびその従属項目のいずれにも、VALUE 節または TYPE 節を含めることはできません。 データ名-*1* に TYPEDEF 節が含まれていてはなりません。

データ名-*2*

再定義される 項目、すなわち REDEFINES のオブジェクト を示します。

データ名-*2* のデータ記述項目には、 REDEFINES 節を含めることができます。

データ名-*2* のデータ記述項目には、 OCCURS 節を含めることはできません。 ただし、データ名-*2* は、データ記述項目に OCCURS 節が指定されている項目に従属しているということは可能です。 このような場合には、REDEFINES 節の中でデータ名-*2* を参照するとき、この参照には添え字付けをすることはできません。

再定義された項目および各種の従属項目に TYPE 節が含まれていてはなりません。

データ名-*1* およびデータ名-*2* のいずれにも、OCCURS DEPENDING ON 節を含めることはできません。

データ名-*1* およびデータ名-*2* は、階層内でレベルが同一でなければなりませんが、レベル番号は同じでなくてもかまいません。 データ名-*1* およびデータ名-*2* はいずれも、レベル番号 66 または 88 で定義することはできません。

データ名-*1* およびデータ名-*2* は、それぞれ任意の USAGE を指定して記述することができます。

再定義は、データ名-*1* から開始し、レベル番号がデータ名-*1* のレベル番号以下になったところで終了します。 データ名-*1* とデータ名-*2* のレベル番号より低いレベル番号の項目が、これら項目の間にあってはなりません。 次に例を示します。

```
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.
```

A が再定義されている項目で、B は再定義している項目です。 再定義は B で開始し、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 が検出されると、再定義は終了します。

REDEFINES 節を含むデータ記述項目で GLOBAL 節が使用される場合、データ名-*1* (再定義する項目) のみが GLOBAL 属性を処理します。 例えば、以下の記述では、項目 B のみが GLOBAL 属性を処理します。

```
05  A PICTURE X(6).
05  B REDEFINES A GLOBAL PICTURE X(4).
```

EXTERNAL 節は、REDEFINES 節と同じデータ記述項目上には指定することができません。

再定義されるデータ項目 (データ名-*2*) は、外部データ・レコードとして宣言された場合、再定義するデータ項目 (データ名-*1*) のサイズは再定義されるデータ項目のサイズより大きくすることはできません。 再定義されるデータ項目が外部データ・レコードとして宣言されない場合は、そのような制約はありません。

以下の例は、再定義する項目 B は、再定義される項目 A より多くのストレージを占有できることを示しています。REDEFINED 節のストレージのサイズはバイト数で決まります。 項目 A は 6 バイトのストレージを占有し、項目 B はカテゴリーが国別のデータ項目であり、8 バイトのストレージを占有します。

```
05  A PICTURE X(6).
05  B REDEFINES A GLOBAL PICTURE N(4).
```

同一のストレージ域に対して何回でも再定義することが可能です。 ストレージ域の新しい記述を行う項目は、再定義されるストレージ域の記述のすぐ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介在させることはできません。 必要ではありませんが、複数の再定義すべてで、このストレージ域を定義した元の項目のデータ名を使用することができます。 例えば、次のように指定します。

```
 05  A              PICTURE 9999.
 05  B REDEFINES A  PICTURE 9V999.
 05  C REDEFINES A  PICTURE 99V99.
```

また、以下の例に示すように、複数の再定義で、直前の定義の名前を使用することもできます。

```
 05  A              PICTURE 9999.
 05  B REDEFINES A  PICTURE 9V999.
 05  C REDEFINES B  PICTURE 99V99.
```

FD 項目に従属するレベル 01 項目が複数書き込まれているときは、暗黙の再定義 とも言われる状態が発生します。 つまり、2 番目のレベル 01 の項目が、1 番目の項目に対して割り振られていたストレージを暗黙のうちに再定義します。 このようなレベル 01 の項目には、REDEFINES 節を指定することはできません。

FD 項目に従属するレベル 01 項目が複数書き込まれているとき (およびレベル 01 項目がタイプ名ではないとき) は、暗黙の再定義 と呼ばれる状態が発生します。 つまり、2 番目のレベル 01 の項目が、 1 番目の項

目に対して割り振られていたストレージを暗黙のうちに再定義します。 このようなレベル 01 の項目には、REDEFINES 節も TYPE 節も指定できません。 さらに、TYPE 節は、いかなるレベル 01 項目に従属するいかなる項目においても指定してはなりません。

データ項目が、 ファイル記述 (FD) 項目の複数の 01 レベル・レコードを暗黙に再定義する場合、 再定義している項目または再定義されている項目に従属している項目には、OCCURS DEPENDING ON 節を含めることができます。

REDEFINES 節の考慮事項

このトピックでは、REDEFINES 節の使用に関する考慮事項について説明します。

領域を再定義する場合、常にその領域の記述はすべて有効です。つまり、再定義で前の記述が取り替えられることはありません。 したがって、B REDEFINES C が指定されていた場合、 2 つのプロシージャー・ステートメント MOVE X TO B または MOVE Y TO C は、どちらもプログラム内の任意の地点で実行可能です。 最初の例では、B として記述されている領域は X の値とフォーマットを受け入れます。2 番目の例では、同じ物理領域 (ここでは C として記述されている) は Y の値とフォーマットを受け入れます。最初のステートメントの直後に 2 番目のステートメントが実行されると、当該ストレージ域で X の値が Y の値に置き換えられることに注意してください。

再定義するデータ項目の USAGE は、 再定義されるデータ項目の USAGE と同じである必要はありません。ただし、同じでなくても、既存のデータのフォーマットや内容が変更されることはありません。 次に例を示します。

```
05  B                 PICTURE  99 USAGE DISPLAY VALUE 8.
05  C REDEFINES B     PICTURE S99 USAGE COMPUTATIONAL-4.
05  A                 PICTURE S99 USAGE COMPUTATIONAL-4.
```

B を再定義しても、ストレージ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
```

最初のステートメントを実行すると、値 8 が A に加えられます (なぜなら、B は USAGE DISPLAY を持っているからです)。 次のステートメントを実行すると、-3848 の値が A に加えられ (なぜなら、C は USAGE COMPUTATIONAL-4 を持っているからです)、このストレージ域のビット構成は、2 進値で -3848 となります。 この例は、再定義の使い方を誤ると、 予想外の結果または正しくない結果が生じることを示したものです。

REDEFINES 節の使用例

REDEFINES 節は、再定義される領域の範囲内にある (従属する) 項目に対して指定することができます。

以下の例では、WEEKLY-PAY は SEMI-MONTHLY-PAY を再定義します (これは REGULAR-EMPLOYEE の範囲内にありますが、REGULAR-EMPLOYEE は TEMPORARY-EMPLOYEE によって再定義されます)。

```
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.
```

以下の例の CODE-H REDEFINES HOURLY-PAY のように、REDEFINES 節は、再定義する項目に従属している項目についても指定することができます。

```
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.
```

1 つのストレージ域内のデータ項目を、その長さを変えずに再定義することができます。 次に例を示します。

```
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.
```

1 つのストレージ域内のデータ項目の長さや型を再指定することもできます。 次に例を示します。

```
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.
```

データ項目には、再定義される項目より大きい長さを指定することもできます。 次に例を示します。

```
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 X(4).
```

このことが、再定義される項目 NAME-2 の長さを変更することはありません。

予想外の結果

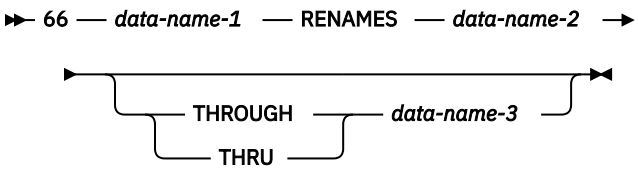
このトピックに示された条件で、予想外の結果が生じる場合があります。

- 再定義する項目が、再定義される項目に移動されたとき (すなわち、B REDEFINES C およびステートメント MOVE B TO C が実行された場合)。
- 再定義される項目が、再定義する項目に移動されたとき (すなわち、B REDEFINES C およびステートメント MOVE C TO B が実行された場合)。

RENAMES 節

RENAMES 節は、基本データ項目の代替グループ (重複することもある) を指定します。

フォーマット



RENAMES 節を含むデータ記述項目では、特別なレベル番号 66 を指定する必要があります。 (レベル番号 66 とデータ名-*1* は、RENAMES 節自体の一部ではありませんが、表現を明確にするためにのみこのフォーマットの中に加えたものです。)

1 つの論理レコードに対して、1 つ以上の RENAMES 節を記述できます。 ある論理レコードに関連付けられているすべての RENAMES 項目は、そのレコードの中の最後のデータ記述項目の直後に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を識別します。

レベル 66 の項目で、レベル 01、レベル 77、レベル 88、 または別のレベル 66 の項目の名前を変更することはできません。

データ名-*1* を修飾子として使用することはできません。 レベル標識項目またはレベル 01 項目の名前で修飾することだけができます。

データ名-2、データ名-*3*
基本データ項目の元のグループを識別します。したがって、これらの両方には関連するレベル 01 項目中の基本項目またはグループ項目の名前を指定しなければならず、 同じデータ名を指定できません。 これらのデータ名は両方とも修飾できます。

データ名-*2* とデータ名-*3* は、それぞれ以下の項目のいずれかを参照できます。

- 基本データ項目
- 英数字グループ項目
- 国別グループ項目

データ名-*2* またはデータ名-*3* が国別グループ項目を参照する場合、参照される項目はグループとして (国別カテゴリーの基本データ項目としてではなく) 処理されます。

データ名-*2* とデータ名-*3*、またはこれらが従属しているグループ項目についてのデータ項目で、OCCURS 節を指定することはできません。 また、データ名-*2* とデータ名-*3* の間に定義されているどの項目に対しても、OCCURS DEPENDING 節を指定することはできません。

TYPE 節は、データ名-*2* やデータ名-*3* データ記述内で指定してはなりません。 同様に、TYPE 節はデータ名-*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いずれ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 節を使用して定義されているグループ項目に従属している場合、データ名-*1* は同じグループ項目に従属していなければなりません。

キーワードの THROUGH と THRU は同じ意味です。

THROUGH 句が指定される場合

- データ名-*1* は、以下のようなすべての基本項目を含む英数字グループ項目を定義します。
  - データ名-*2* (それが基本項目である場合)、またはデータ名-*2* (それがグループ項目である場合) の中の最初の基本項目で開始します。
  - データ名-*3* (それが基本項目である場合)、またはデータ名-*3* (それが英数字グループ項目または国別グループ項目である場合) の中の最後の基本項目で終了します。

- 開始項目から終了項目によって占有されるストレージ域は、データ名-*1* によって占有されるストレージ域になります。

使用上の注意: THROUGH 句を指定して定義されたグループには、USAGE NATIONAL のデータ項目を含めることができます。

データ名-*3* の左端の文字をデータ名-*2* の左端の文字より前に置くことはできません。 また、データ名-*3* の右端の文字をデータ名-*2* の右端の文字より優先させることはできません。 このことは、データ名-*3* がデータ名-*2* に完全に従属することはできないことを意味します。

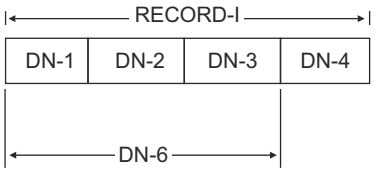
THROUGH 句を指定しないときは、

- データ名-*2* によって占有されるストレージ域は、データ名-*1* によって占有されるストレージ域になります。
- データ名-*2* のデータ属性はすべて、データ名-*1* のデータ属性になります。 つまり、次のようになります。
  - データ名-*2* が英数字グループ項目である場合は、データ名-*1* は英数字グループ項目です。
  - データ名-*2* が国別グループ項目である場合は、データ名-*1* は国別グループ項目です。
  - データ名-*2* が基本項目である場合は、データ名-*1* は基本項目として扱われます。

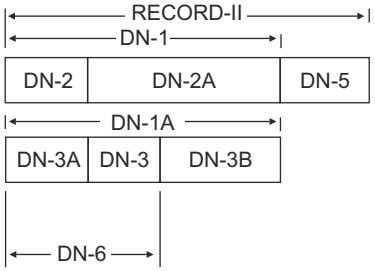
以下の図に、有効な RENAMES 節の指定と無効な指定を示します。

COBOL 仕様 **ストレージ・レイアウト**

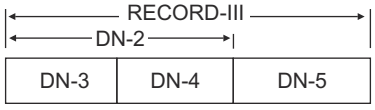
例 1 (有効)
```
01   RECORD-I.
    05DN-1... .
    05 DN-2...  .
    05 DN-3...  .
    05 DN-4...  .
66   DN-6 RENAMES DN-1 THROUGH DN-3.
```



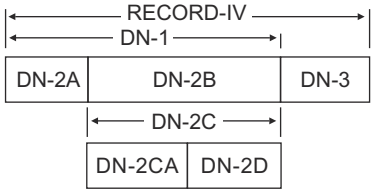
例 2 (有効)
```
01   RECORD-II.
    05 DN-1.
      10 DN-2...  .
      10 DN-2A...  .
    05 DN-1A REDEFINES DN-1.
      10 DN-3A...  .
      10 DN-3...  .
      10 DN-3B...  .
    05 DN-5...  .
66   DN-6 RENAMES DN-2 THROUGH DN-3.
```



例 3 (無効)
```
01   RECORD-III.
    05 DN-2.
      10 DN-3...  .
      10 DN-4...  .
    05 DN-5...  .
66   DN-6 RENAMES DN-2 THROUGH DN-3.  DN-6 は不確定
```



例 4 (無効)
```
01   RECORD-IV.
    05 DN-1.
      10 DN-2A...  .
      10 DN-2B...  .
      10 DN-2C REDEFINES DN-2B.
        15 DN-2CA...  .
  15 DN-2D...  .
    05 DN-3...  .
66   DN-4 RENAMES DN-1 THROUGH DN-2CA.  DN-4 は不確定
```

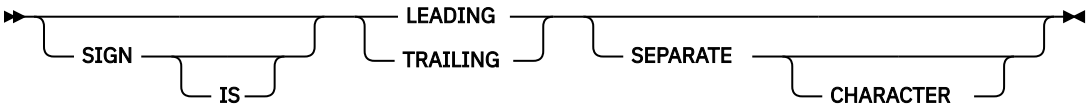


SIGN 節

SIGN 節は、符号付き数字項目に適用される演算符号の表示位置とモードを指定します。

SIGN 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。

フォーマット



SIGN 節は、以下の項目についてのみ指定できます。

- USAGE が DISPLAY または NATIONAL で、PICTURE 文字ストリングで S を使用して記述されている基本数字データ項目、または
- 従属項目としてそのような基本項目を少なくとも 1 つ含むグループ項目

SIGN 節がグループ・レベルで指定されている場合、その SIGN 節は、USAGE が DISPLAY または NATIONAL の従属符号付き数字基本データ項目のみに適用されます。 そのようなグループには、SIGN 節の影響を受けない項目を含めることもできます。 SIGN 節が、SIGN 節を持つグループ項目に従属するグループ項目または基本項目に対して指定されている場合、従属項目に対する SIGN 節はその従属項目に優先します。

外部浮動小数点項目に対する SIGN 節は、説明文として扱われます。

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。

CODE-SET 節を FD 項目の中で指定する場合、そのファイル記述項目と関連付けられた符号付き数字データ記述項目は、 SIGN IS SEPARATE 節と共に記述する必要があります。

SEPARATE CHARACTER 句を指定しない場合、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 桁位置 (いずれか、指定されている方) に関連付けられているとみなされます。 (この場合、SIGN IS TRAILING を指定すると、 コンパイラーによる標準の処置と等しくなります。)
- PICTURE 文字ストリングの中の文字 S は、 その項目のサイズ (標準データ・フォーマットの文字に関して) を判別する際にカウントには入れられません。

SEPARATE CHARACTER 句を指定する場合、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) とみなされます。 この文字位置は、数字文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に関して) を判別する際にカウントに入れられます。
- + は、正の演算符号として使用される文字です。
- - は、負の演算符号として使用される文字です。

SEPARATE CHARACTER 句は、日付フィールドには指定できません。

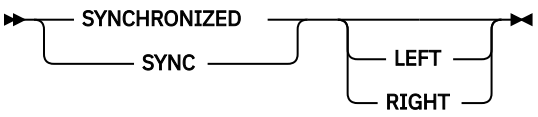
SIGN 節は、FORMAT 節が指定されている場合には指定できません。

TYPE 節は SIGN 節と同じデータ記述項目に指定できません。

SYNCHRONIZED 節

SYNCHRONIZED 節は、ストレージ内の自然境界に基本項目の位置合わせを行うように指定します。

フォーマット



SYNC は、SYNCHRONIZED の省略形で意味は同じです。

SYNCHRONIZED 節は必須ではありませんが、 算術演算に使用される 2 進数項目に対してシステムによってはパフォーマンスが向上します。

SYNCHRONIZED 節は、基本項目に対して、またはレベル 01 グループ項目に対して指定できます。その場合、そのグループ項目内のすべての基本項目が同期化されます。

LEFT
これは、基本項目が配置されるシステム設定の境界の左文字位置から開始するように、基本項目を位置付けることを指定します。

RIGHT
これは、基本項目がシステム設定境界に配置される際に、システム設定境界の右文字位置で終わるように、その基本項目を位置付けることを指定します。

LEFT 句と RIGHT 句が指定されたときは、構文チェックが行われますが、それはプログラムの実行に何も影響しません。

基本項目の長さは、SYNCHRONIZED 節を指定しても変化することはありません。

以下の図に、SYNCHRONIZE 節が他の言語エレメントに与える影響を示します。

表 *21.* ***SYNCHRONIZE*** **節が他の言語エレメントに与える影響**

言語エレメント	コメント
OCCURS 節	OCCURS 節のスコープ内の項目について指定すると、その項目の各オカレンスが同期化されます。
USAGE DISPLAY または PACKED-DECIMAL	各項目が構文チェックされますが、SYNCHRONIZED 節の実行には何も影響しません。
USAGE NATIONAL	各項目が構文チェックされますが、SYNCHRONIZED 節の実行には何も影響しません。
USAGE BINARY または COMPUTATIONAL	REDEFINES 節を含む項目の従属基本項目のうち最初のものについて指定した場合は、その項目に未使用の文字位置を追加する必要はまったくありません。 SYNCHRONIZED 節が、従属データ項目 (レベル番号が 02 から 49 のデータ項目) に対して指定されていないときは、 位置合わせに関して次の点を考慮する必要があります。 • その項目は、USAGE が BINARY であり、PICTURE が S9 から S9(4) の範囲にあれば、 レコード開始に関連して 2 の倍数の変位に位置合わせされます。 • USAGE が BINARY であり、PICTURE が S9(5) から S9(18) の範囲にある場合、または USAGE が INDEX であれば、その項目はレコードの先頭を基準にして 4 の倍数の変位に位置合わせされます。 SYNCHRONIZED 節が 2 進数項目に対して指定されない場合、遊びバイト用のスペースは確保されません。

表 *21.* ***SYNCHRONIZE* 節が他の言語エレメントに与える影響** *(続き)*	
言語エレメント	**コメント**
USAGE POINTER、PROCEDURE-POINTER、FUNCTION-POINTER	ADDR(32) コンパイラー・オプションが指定されている場合、データはフルワード境界に位置合わせされます。
USAGE COMPUTATIONAL-1	データはフルワード境界に位置合わせされます。
USAGE COMPUTATIONAL-2	データはダブルワード境界に位置合わせされます。
USAGE COMPUTATIONAL-3	データは、PACKED-DECIMAL 項目の SYNCHRONIZED 節と同様に扱われます。
USAGE COMPUTATIONAL-4	データは、COMPUTATIONAL 項目の SYNCHRONIZED 節と同様に扱われます。
USAGE COMPUTATIONAL-5	データは、COMPUTATIONAL 項目の SYNCHRONIZED 節と同様に扱われます。
DBCS と外部浮動小数点項目	各項目が構文チェックされますが、SYNCHRONIZED 節の実行には何も影響しません。

REDEFINES 節	REDEFINES 節を含む項目の場合は、再定義される側のデータ項目を、 再定義する側のデータ項目と適切に境界の位置合わせをしなければなりません。 例えば、次のように書いた場合、データ項目 A はフルワード境界から開始するようにする必要があります。
	`02 A PICTURE X(4).`
	`02 B REDEFINES A PICTURE S9(9) BINARY SYNC.`

FORMAT 節	各項目が構文チェックされますが、SYNCHRONIZED 節の実行には何も影響しません。

FILE SECTION では、コンパイラーは、SYNCHRONIZED 節を含むレベル 01 のレコードはすべてバッファー内でダブルワード境界に位置合わせされているものと想定します。 1 ブロックに複数のレコードがあるときは、正しい境界に位置合わせするために、 レコード間に必要なだけ遊びバイトを用意しなければなりません。

WORKING-STORAGE SECTION では、 コンパイラーはすべてのレベル 01 の項目をダブルワード境界に位置合わせします。

LINKAGE SECTION では、2 進数項目の位置合わせを行うために、 すべてのレベル 01 の項目はダブルワード境界から始まるものとみなされます。 したがって、CALL ステートメントを使用する場合は、 そのステートメントの USING 句の該当のオペランドが、 対応するように位置合わせされている必要があります。

SYNCHRONIZED 節は TYPE 節と同じデータ記述項目には指定できません。

遊びバイト

遊びバイトには、次の 2 種類があります。

- レコード内の 遊びバイト: レコード内のそれぞれの同期項目の前に置かれる未使用の文字位置。
- レコード間の 遊びバイト: ブロック化された論理レコードの間に置かれる未使用の文字位置。

レコード内の遊びバイト

データ記述中の 2 進数項目が本来の境界にない場合、コンパイラーはレコード内に遊びバイトを挿入して、すべての SYNCHRONIZED 項目が適切な境界にあるようにします。

ファイル内のレコードの長さを把握しているのは重要なことであり、 遊びバイトが必要であるかどうかを判別したり、 必要であればコンパイラーがバイトをいくつ追加すればよいか判別しなければなりません。コンパイラーの使用するアルゴリズムは、次のとおりです。

- 2 進数項目の前にあるすべての基本データ項目が占める総バイト数 (以前加えられた遊びバイトがあればそれを含む) を計算します。
- この合計を *m* で除算します。
  - *m* = 2 (4 桁以下の 2 進数項目の場合)。
  - *m* = 4 (5 桁以上の 2 進数項目、および COMPUTATIONAL-1 データ項目の場合)。
  - *m* = 4 または 8 (USAGE INDEX、 USAGE POINTER、 USAGE PROCEDURE-POINTER、または USAGE FUNCTION-POINTER を使用して記述されたデータ項目の場合。)
  - COMPUTATIONAL-1*m* = 8 (COMPUTATIONAL-2 データ項目の場合)。
- この除算の剰余 (*r*) が 0 の場合、遊びバイトは必要ありません。 この剰余が 0 でない場合、加えるべき遊びバイト数は、 *m* - *r* です。

これらの遊びバイトは、各レコードごとに、2 進数項目の前にある基本データ項目のすぐ後に追加されます。 これらは、SYNCHRONIZED 2 進数項目の直前にある基本項目と同じレベル番号を持つ項目を構成するかのように定義され、それらを含むグループのサイズに数えられます。

次に例を示します。

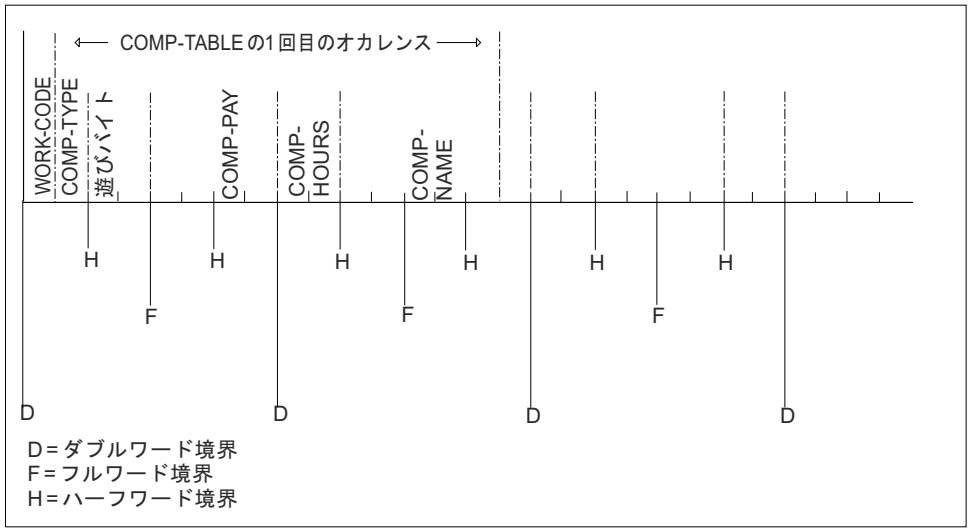
```
01  FIELD-A.
    05  FIELD-B                    PICTURE X(5).
    05  FIELD-C.
        10  FIELD-D                PICTURE XX.
       [10  SLACK-BYTES             PICTURE X.  INSERTED BY COMPILER]
        10  FIELD-E  COMPUTATIONAL  PICTURE S9(6) SYNC.
01  FIELD-L.
    05  FIELD-M                    PICTURE X(5).
    05  FIELD-N                    PICTURE XX.
   [05  SLACK-BYTES                 PICTURE X.  INSERTED BY COMPILER]
    05  FIELD-O.
        10  FIELD-P COMPUTATIONAL   PICTURE S9(6) SYNC.
```

OCCURS 節の指定のあるグループ項目が定義されており、 そのグループ項目の中に SYNCHRONIZED 2 進データ項目が含まれている場合も、コンパイラーは遊びバイトを追加することができます。 遊びバイトを追加するかどうかを決定するために、コンパイラーは次の処置を取ります。

- コンパイラーは、必要なレコード内の遊びバイトをすべて含めて、 グループのサイズを計算します。
- この合計を、グループ内の基本項目に必要な最大の *m* で除算します。
- *r* が 0 であれば、遊びバイトは必要ありません。 *r* が 0 でなければ、 遊びバイトとして *m* - *r* バイトを加える必要があります。

OCCURS 節を含むグループ項目のオカレンスのたびに、その終わりのところで遊びバイトが挿入されます。例えば、次のように定義されているレコードは、ストレージの中では、レコードの後の図に示すようになります。

```
01  WORK-RECORD.
    05  WORK-CODE                  PICTURE X.
    05  COMP-TABLE OCCURS 10 TIMES.
        10  COMP-TYPE              PICTURE X.
       [10  SLACK-BYTES             PIC XX.  INSERTED BY COMPILER]
        10  COMP-PAY               PICTURE S9(4)V99 COMP SYNC.
        10  COMP-HOURS             PICTURE S9(3) COMP SYNC.
        10  COMP-NAME              PICTURE X(5).
```

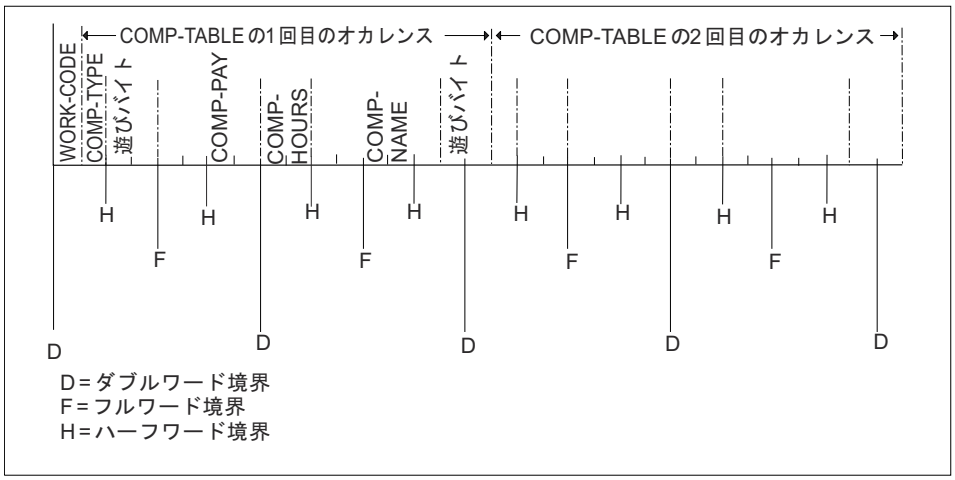


COMP-PAY と COMP-HOURS を適切な境界に位置合わせするために、コンパイラーはレコード内に 2 つの遊びバイトを追加しました。

上記の例の場合、さらに調整をしなければ、COMP-TABLE の 2 番目のオカレンスは、 ダブルワード境界の 1 バイト手前から始まることになってしまいます。 そして、2 番目以降のオカレンス項目では、COMP-PAY と COMP-HOURS が正しく位置合わせされません。 したがって、コンパイラーはグループの最後に遊びバイトを加える必要があります。 これによって、レコードは次のように記述されたかのようになります。

```
01  WORK-RECORD.
    05  WORK-CODE                  PICTURE X.
    05  COMP-TABLE OCCURS 10 TIMES.
        10  COMP-TYPE              PICTURE X.
       [10  SLACK-BYTES             PIC XX.  INSERTED BY COMPILER]
        10  COMP-PAY               PICTURE S9(4)V99 COMP SYNC.
        10  COMP-HOURS             PICTURE S9(3)  COMP SYNC.
        10  COMP-NAME              PICTURE X(5).
       [10  SLACK-BYTES             PIC XX.  INSERTED BY COMPILER]
```

この例では、2 番目の COMP-TABLE のオカレンス (およびそれ以降) は、ダブルワード境界を 1 バイト超えたところから開始されます。 COMP-TABLE の最初のオカレンスに関するストレージのレイアウトは、以下の図に示したように見えることになります。



このテーブルの後続の各オカレンスは、 これで最初のオカレンスと同じ相対位置で開始されることになります。

レコード間の遊びバイト

レコード内のすべての基本データ項目の長さ (すべての遊びバイトを含めて) を加算します。 次に、この合計をレコード内の基本項目のいずれかに対応する *m* の最高値で除算します。

r (剰余) が 0 であれば、遊びバイトは不要です。 *r* が 0 でなければ、遊びバイトとして *m* - *r* バイトが必要です。 これらの遊びバイトは、レコードの終わりにレベル 02 の FILLER を記述することによって指定することができます。

以下のようなレコード記述があるとします。

```
01  COMP-RECORD.
    05  A-1    PICTURE X(5).
    05  A-2    PICTURE X(3).
    05  A-3    PICTURE X(3).
    05  B-1    PICTURE S9999  USAGE COMP SYNCHRONIZED.
    05  B-2    PICTURE S99999 USAGE COMP SYNCHRONIZED.
    05  B-3    PICTURE S9999  USAGE COMP SYNCHRONIZED.
```

A-1、A-2、および A-3 のバイト数の合計は 11 です。 B-1 は 4 桁の COMPUTATIONAL 項目ですから、遊びバイトとして 1 バイトが B-1 の前に加えられる必要があります。 このバイトを加えると、B-2 の前にあるバイト数の合計は 14 になります。 B-2 は、長さが 5 桁の COMPUTATIONAL 項目ですから、その前に遊びバイトとして 2 バイトなければなりません。 B-3 の前には遊びバイトは不要です。

上記の考察によって書き換えたレコード記述項目は、今度は次のようになります。

```
01  COMP-RECORD.
    05  A-1            PICTURE X(5).
    05  A-2            PICTURE X(3).
    05  A-3            PICTURE X(3).
   [05  SLACK-BYTE-1    PICTURE X.   INSERTED BY COMPILER]
    05  B-1            PICTURE S9999 USAGE COMP SYNCHRONIZED.
   [05  SLACK-BYTE-2    PICTURE XX.  INSERTED BY COMPILER]
    05  B-2            PICTURE S99999 USAGE COMP SYNCHRONIZED.
    05  B-3            PICTURE S9999  USAGE COMP SYNCHRONIZED.
```

COMP-RECORD には合計 22 バイトありますが、上記の規則により、 *m* = 4 および *r* = 2 となります。したがって、ブロック化レコードの適切な位置合わせを得るには、レコードの終わりに遊びバイトを 2 バイト加えなければなりません。

したがって、最終的なレコード記述項目は、次のようになります。

```
01  COMP-RECORD.
    05  A-1             PICTURE X(5).
    05  A-2             PICTURE X(3).
    05  A-3             PICTURE X(3).
   [05  SLACK-BYTE-1    PICTURE X.   INSERTED BY COMPILER]
    05  B-1             PICTURE S9999 USAGE COMP SYNCHRONIZED.
   [05  SLACK-BYTE-2    PICTURE XX.  INSERTED BY COMPILER]
    05  B-2             PICTURE S99999 USAGE COMP SYNCHRONIZED.
    05  B-3             PICTURE S9999  USAGE COMP SYNCHRONIZED.
    05  FILLER          PICTURE XX.  [SLACK BYTES YOU ADD]
```

TYPE 節

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 211 ページの『TYPEDEF 文節』で説明しています。

フォーマット

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。

TYPEDEF 文節

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 210 ページの『TYPE 節』を参照してください。

フォーマット

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。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『複合 *OCCURS DEPENDING ON*』を参照してください。

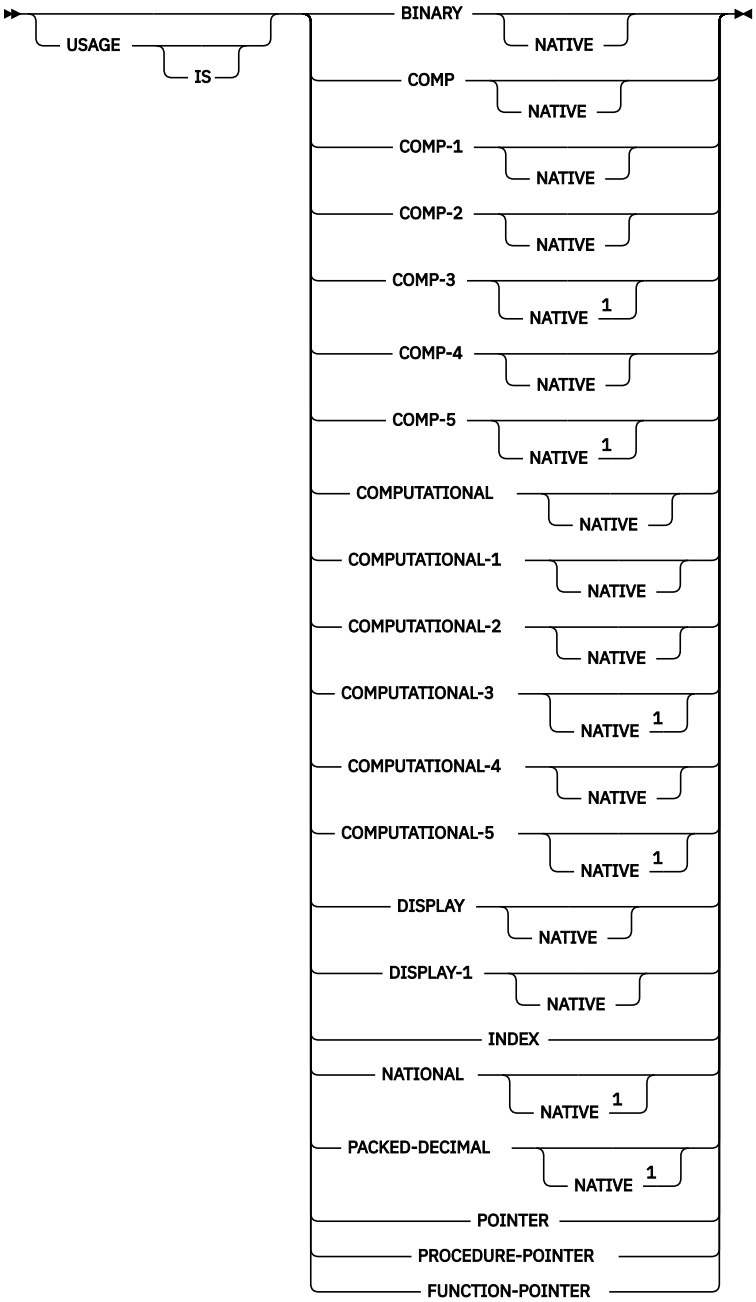
TYPEDEF 文節を指定できるのは、プログラムの作業用ストレージ・セクション、 ローカル・ストレージ・セクション、リンケージ・セクション、またはファイル・セクションの中だけです。

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。

USAGE 節

USAGE 節は、ストレージでデータが表されるフォーマットを指定します。

フォーマット 1



注:

[1] NATIVE 句は、 COMP-3、 COMPUTATIONAL-3、 COMP-5、 COMPUTATIONAL-5、 NATIONAL、および PACKED-DECIMAL データ項目の場合にはコメントとして扱われます。

USAGE 節は、66 および 88 以外の任意のレベル番号を持つデータ記述項目に対して指定することができます。

グループ・レベルで指定した場合、そのグループ内の各基本項目ごとに USAGE 節は適用されます。 基本項目の USAGE は、その基本項目が属するグループの USAGE と矛盾するものであってはなりません。

USAGE 節は、GROUP-USAGE NATIONAL 節が指定されているグループ・レベル項目の中に指定してはなりません。

グループ・レベル項目に対して GROUP-USAGE NATIONAL 節が指定または暗黙指定されている場合は、そのグループ内のすべての基本項目に対して USAGE NATIONAL を指定または暗黙指定する必要があります。詳しくは、169 ページの『GROUP-USAGE 節』を参照してください。

USAGE 節がグループまたは基本レベルのいずれかで指定されないと、USAGE 節は暗黙に以下のように指定されます。

- PICTURE 節が G および N 以外の記号のみを含むときは、Usage DISPLAY
- PICTURE 節に 1 つ以上の記号 N のみが含まれ、NSYMBOL(NATIONAL) コンパイラー・オプションが有効なときは、Usage NATIONAL
- PICTURE 節に 1 つまたは複数の記号 N が含まれ、 NSYMBOL(DBCS) コンパイラー・オプションが有効なときは、Usage DISPLAY-1

DATE FORMAT 節を使用して定義されたデータ項目の場合、USAGE として使用できるのは、 DISPLAY および COMP-3 (またはその等価の COMPUTATIONAL-3 および PACKED-DECIMAL) だけです。 詳しくは、163 ページの『DATE FORMAT 節と他の節との結合』を参照してください。

TYPE 節は USAGE 節と同じデータ記述項目に指定できません。

TYPE 節が指定されているデータ記述項目は、USAGE 節を含むデータ記述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.
```

計算用項目

計算用項目は、算術演算で使用される値です。 この項目は数字でなければなりません。 グループ項目が USAGE COMPUTATIONAL で記述されている場合、そのグループ内の基本項目はこの USAGE になります。

計算用項目の最大長は、10 進数で 18 桁です (PACKED-DECIMAL 項目を除く)。 ARITH(COMPAT) コンパイラー・オプションが有効な場合は、 PACKED-DECIMAL 項目の最大長は 10 進数の 18 桁です。
ARITH(EXTEND) コンパイラー・オプションが有効な場合は、 PACKED-DECIMAL 項目の最大長は 10 進数の 31 桁です。

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 つ以上の数字文字位置

S
1 つの演算符号

V
1 つの暗黙の小数点

P
1 つ以上の 10 進数位取り位置

COMPUTATIONAL-1 項目と COMPUTATIONAL-2 項目 (内部浮動小数点) は、PICTURE ストリングを持つことはできません。

BINARY
これは 2 進数データ項目を指定します。 これらの項目は、0 から 9 の 10 進数字と 1 つの符号から構成される 10 進数です。 負の数は、同じ絶対値を持つ正の数の 2 の補数として表されます。

2 進数項目によって占有されるストレージの大きさは、PICTURE 節で定義された 10 進数の桁数によって異なります。

PICTURE 節の示す桁	占有するストレージ
1 から 4	2 バイト (ハーフワード)
5 から 9	4 バイト (フルワード)
10 から 18	8 バイト (ダブルワード)

デフォルトでは、バイナリー・データはプラットフォームのネイティブの 2 進数表現形式を使用します。Linux システムの場合、ネイティブの 2 進数表現はリトル・エンディアン 形式 (最下位アドレスに最下位桁) になります。バイナリー・データ項目のエンディアンネス形式を変更する場合は、BINARY コンパイラー・オプションを参照してください。

BINARY, COMPUTATIONAL, および COMPUTATIONAL-4 のデータ項目は、 TRUNC コンパイラー・オプションによって影響を受けることがあります。 このコンパイラー・オプションの影響について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『*TRUNC*』を参照してください。 『*TRUNC*』および『*BINARY*』 を参照してください。

PACKED-DECIMAL
これは、内部 10 進項目を指定します。 これらの項目は、ストレージの中でパック 10 進数フォーマットで示されます。 最後の文字位置を除く各文字位置は 2 桁からなり、最後の文字位置は最下位桁と符号で占められます。 このような項目は、0 から 9 までの任意の数字に符号を付けて、18 桁までの 10 進数の値を表すことができます。

この符号表現は、ゾーン 10 進数フィールドで 4 ビットの符号表現を行うのと同じビット構成を使用します。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」内の 『ゾーン *10* 進およびパック *10* 進データの符号表現』 を参照してください。

PACKED-DECIMAL は、変換指定子のみが含まれる FORMAT リテラルを持つ日付項目および時刻項目にも指定できます。 この変換指定子に含めることができるのは数字のみでなければなりません。

COMPUTATIONAL または COMP (2 進数)
これは BINARY と等価です。 COMPUTATIONAL 句は BINARY と同期します。

COMPUTATIONAL-1 または COMP-1 (浮動小数点)
内部浮動小数点項目に対して指定します (単精度)。 COMP-1 項目は 4 バイト長です。

COMP-1 データ項目は、FLOAT(NATIVE|BE|LE) コンパイラー・オプションの影響を受けます。 詳しくは、*COBOL for Linux on x86* プログラミング・ガイド内の *FLOAT* を参照してください。

COMPUTATIONAL-2 または COMP-2 (長精度浮動小数点)
内部浮動小数点項目に対して指定します (倍精度)。 COMP-2 項目の長さは 8 バイトです。

COMP-2 データ項目は、FLOAT(NATIVE|BE|LE) コンパイラー・オプションの影響を受けます。 詳しくは、*COBOL for Linux on x86* プログラミング・ガイド内の *FLOAT* を参照してください。

COMPUTATIONAL-3 または COMP-3 (内部 10 進数)
これは PACKED-DECIMAL と等価です。

COMPUTATIONAL-4 または COMP-4 (2 進数)
これは BINARY と等価です。

COMPUTATIONAL-5 または COMP-5 (固有 2 進数)
これらのデータ項目は、ストレージ内ではバイナリー・データとして 表現されます。 このデータ項目には、(USAGE BINARY データの場合のように) 項目 に対する picture に入っている 9 の数で示される値に制限されず、ネイティブ・バイナリー 表記 (2、4 または 8 バイト) で本来表すことのできる値まで入れることができます。 数値データを COMP-5 項目に移動または保管すると、 COBOL の picture サイズによる制限ではなく、2 進数フィールド・サイズによっ て切り捨てが行われます。 COMP-5 項目が参照された場合は、 フル 2 進数フィールド・サイズがその操作で使用されます。

TRUNC(BIN) コンパイラー・オプションを指定すると、すべてのバイナリー・ データ項目 (USAGE BINARY、COMP、COMP-4) は、USAGE COMP-5 で宣言されたかのように処理されます。

以下の表には、PICTURE 文字ストリングのいくつか、結果のストレージ表記、および USAGE COMP-5 で記述されたデータ項目の値の範囲を示しています。

PICTURE	ストレージ表記	数値
S9(1) から S9(4)	2 進数ハーフワード (2 バイト)	-32768 から +32767
S9(5) から S9(9)	2 進数フルワード (4 バイト)	-2,147,483,648 から +2,147,483,647
S9(10) から S9(18)	2 進数ダブルワード (8 バイト)	-9,223,372,036,854,775,808 から +9,223,372,036,854,775,807
9(1) から 9(4)	2 進数ハーフワード (2 バイト)	0 から 65535
9(5) から 9(9)	2 進数フルワード (4 バイト)	0 から 4,294,967,295
9(10) から 9(18)	2 進数ダブルワード (8 バイト)	0 から 18,446,744,073,709,551,615

COMP-5 データ項目の picture では、位取り係数 (つまり小数点位、または 暗黙の整数の桁) を指定できます。 この場合、上の表にリストされている最大容量は、 それに応じて調節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、ストレージ内ではバイナリーのハーフワードとして表現され、-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: 算術ステートメントで ON SIZE ERROR 句を使用し、受け取り側が USAGE COMP-5 で定義されている場合、受け取り側に格納できる最大値は、項目の 10 進 PICTURE 文字ストリングで暗黙指定される値です。 この最大値を超える値を保管しようとすると、サイズ・エラー状態となります。

DISPLAY 句

データ項目は、文字形式で保管され、1 文字はそれぞれ 8 ビット・バイトです。 これは、印刷出力の際に使用されるフォーマットに対応します。 DISPLAY は、明示的にも暗黙的にも指定することができます。

USAGE IS DISPLAY は、次に示すような種類の項目で有効です。

- 英字
- 英数字
- 英数字編集
- ブール
- 日付、時刻、およびタイム・スタンプ
- 数字編集
- 外部浮動小数点
- 外部 10 進数

英字、英数字、英数字編集、ブール、および数字編集の各項目については、186 ページの『データ・カテゴリーと PICTURE の規則』に説明があります。

USAGE DISPLAY が指定された外部 10 進数項目は、ゾーン *10* 進数 項目と呼ばれることもあります。 数字の各桁は、1 バイトで表されます。 各バイトの上位 4 ビットはゾーン・ビットです。 最下位バイトの上位 4 ビットは項目の符号を表します。 各バイトの下位 4 ビットに数字の値が含まれます。

ARITH(COMPAT) コンパイラー・オプションが有効な場合は、 外部 10 進数項目の最大長は 18 桁です。
ARITH(EXTEND) コンパイラー・オプションが有効な場合は、 外部 10 進数項目の最大長は 31 桁です。

外部 10 進数項目の PICTURE 文字ストリングに含めることができるのは、次のものに限ります。

- 1 つ以上の記号 9
- 演算符号 S
- 想定小数点 V
- 1 つ以上の記号 P

CHAR(EBCDIC) コンパイラー・オプションの効果

CHAR(EBCDIC) コンパイラー・オプションを使用した場合、文字データが NATIVE 句で定義されない限り、DISPLAY または DISPLAY-1 句で定義されたデータ項目は、EBCDIC として扱われます。

DISPLAY-1 句

DISPLAY-1 句は、項目を DBCS として定義します。 データ項目は、文字形式で保管され、1 文字はそれぞれ 2 バイトのストレージを占有します。

FUNCTION-POINTER 句

FUNCTION-POINTER 句は、項目を関数ポインター・データ項目 として定義します。 関数ポインター・データ項目には、プロシージャー入り口点の記述子のアドレスを入れることができます。

関数ポインターは、4 バイトの基本項目か、 関数ポインターの機能は、プロシージャー・ポインターの機能と同じです。. したがって、関数ポインターは、C 関数ポインターとの相互運用が容易になっています。

関数ポインターは、次のいずれかの関数記述子を指すか、 または NULL を含むことができます。

- 最外部プログラムの PROGRAM-ID 段落によって定義される、COBOL プログラムの 1 次入り口点
- COBOL ENTRY ステートメントによって定義される、COBOL プログラムの代替入り口点
- 非 COBOL プログラムの入り口点

関数ポインター・データ項目の VALUE 節には、NULL または NULLS のみを含めることができます。

関数ポインターは、219 ページの『PROCEDURE-POINTER 句』で定義されているように、同じ内容においてプロシージャー・ポインターとして使用することができます。

INDEX 句

INDEX 句を使用して定義されたデータ項目を指標データ項目 といいます。

指標データ項目 は 4 バイトの基本項目か、 これを使用すると、指標名の値を保存して後で参照することができます。 指標データ項目 は、必ずしも特定のテーブルと結び付いている必要はありません。 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。 この値は、テーブル内のオカレンス番号に対応しています。

指標データ項目に対する直接参照は、SEARCH ステートメント、 SET ステートメント、比較条件、PROCEDURE DIVISION のヘッダーの USING 句、または CALL ステートメントまたは ENTRY ステートメントの USING 句を介してのみ行うことができます。

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の中で参照される英数字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・オカレンス項目を表す値を保存しますが、それ自体は、テーブルの一部として定義されているわけではありません。 次のようなケースで指標データ項目を参照した場合、値の変換は実行されません。

- SEARCH または SET ステートメントで直接参照
- MOVE ステートメントで間接参照
- 入出力ステートメントで間接参照

指標データ項目を条件変数にすることはできません。

DATE FORMAT、JUSTIFIED、PICTURE、BLANK WHEN ZERO、VALUE、TYPE、または FORMAT 節は、USAGE IS INDEX 節を使用して記述されたグループ項目または基本項目を記述するために使用することはできません。

SYNCHRONIZED は、USAGE IS INDEX と共に使用することによって、指標データ項目を効率的に使用することができます。

NATIONAL 句

NATIONAL 句は、ストレージ内で UTF-16 (CCSID 1200) で表される内容を持つ項目を定義します。データ項目のクラスとカテゴリーは、関連付けられている PICTURE 節で指定されているピクチャー記号によって決まります。

POINTER 句

USAGE IS POINTER を使用して定義されたデータ項目は、ポインター・データ項目 と呼ばれます。ポインター・データ項目 は 4 バイトの基本項目か、

ポインター・データ項目を使用すると、限定的な基底アドレッシングが可能になります。 ポインター・データ項目は、他のポインター項目と内容が等しいかどうかを比較でき、また他のポインター項目に移動できます。

ポインター・データ項目は、以下の場合でのみ使用できます。

- SET ステートメントの中 (フォーマット 5 およびフォーマット 8 の場合のみ)
- 比較条件の中
- CALL ステートメントの USING 句、ENTRY ステートメント、または PROCEDURE DIVISION のヘッダーの中

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の中で参照される英数字グループの一部にすることができます。 ただし、ポインター・データ項目があるグループの一部である場合でも、 上記ステートメントの実行時に値の変換は起こりません。

ポインター・データ項目は、 REDEFINES 節のサブジェクトまたはオブジェクトにすることができます。

SYNCHRONIZED は、USAGE IS POINTER と共に使用することによって、 ポインター・データ項目の効率的な使用を行うことができます。

ポインター・データ項目に対する VALUE 節には、 NULL または NULLS のみを入れることができます。

ポインター・データ項目を条件変数にすることはできません。

ポインター・データ項目は、どのクラスやカテゴリーにも属しません。

次の表は、USAGE IS POINTER で定義されたグループ項目または基本項目を記述するために使用できる/できない節をリストしたものです。

表 *22. USAGE IS POINTER* と一緒に使用できる節と使用できない節

USAGE IS POINTER で使用できる	**USAGE IS POINTER で使用できない**
GLOBAL 文節 EXTERNAL 節 OCCURS 文節 LIKE 節	DATE FORMAT 節 JUSTIFIED 節 PICTURE 文節 BLANK WHEN ZERO 節 TYPE 節 FORMAT 節

ポインター・データ項目は CORRESPONDING 句の処理では無視されます。

ポインター・データ項目を、ファイルに書き込むことは可能ですが、 以降そのポインターを含むレコードの読み取りにおいて、 含まれているアドレスはもはや正しいポインター位置を表すとは限りません。

USAGE IS POINTER は、ADDRESS OF 特殊レジスターに対して、暗黙のうちに指定されます。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『テーブル *(*配列*)* およびポインターの使用』を参照してください。

PROCEDURE-POINTER 句

PROCEDURE-POINTER 句は、項目をプロシージャー・ポインター・データ項目 として定義します。 プロシージャー・ポインター・データ項目には、プロシージャー入り口点の記述子のアドレスを入れることができます。

プロシージャー・ポインター・データ項目は 4 バイトの基本項目です。

プロシージャー・ポインターは、次のいずれかの関数記述子を指すか、または NULL を含むことができます。

- コンパイル単位のプログラムのうち、最外部のプログラムの PROGRAM-ID 段落によって定義された COBOL プログラムの 1 次入り口点
- COBOL ENTRY ステートメントによって COBOL プログラムのために定義されている代替となる入り口点
- 非 COBOL プログラムの入り口点

プロシージャー・ポインター・データ項目は、以下の中でのみ使用できます。

- SET ステートメントの中 (フォーマット 6 の場合のみ)
- CALL ステートメントの中
- 比較条件の中
- ENTRY ステートメントの USING 句、または PROCEDURE DIVISION のヘッダーの中

プロシージャー・ポインター・データ項目は、 他のプロシージャー・ポインター・データ項目と等しいかどうか比較したり、 他のプロシージャー・ポインター・データ項目に移すことができます。

プロシージャー・ポインター・データ項目は、 MOVE ステートメントや入出力ステートメントの中で参照されるグループの一部にすることができます。 ただし、それらのステートメントの実行時に値の変換は実行されません。 プロシージャー・ポインター・データ項目をファイルに書き込むことは可能ですが、 以降そのプロシージャー・ポインターを含むレコードを読むと、 プロシージャー・ポインターの値が正しくないという可能性があります。

プロシージャー・ポインター・データ項目は、 REDEFINES 節のサブジェクトにもオブジェクトにもすることができます。

SYNCHRONIZED は、USAGE IS PROCEDURE-POINTER と共に使用することによって、 プロシージャー・ポインター・データ項目を効率的に位置合わせすることができます。

節 GLOBAL、EXTERNAL、OCCURS、および LIKE は、USAGE IS PROCEDURE-POINTER とともに使用できます。

プロシージャー・ポインター・データ項目に対する VALUE 節は NULL または NULLS のみを含むことができます。

DATE FORMAT、JUSTIFIED、PICTURE、FORMAT 節、TYPE 節、および BLANK WHEN ZERO 節は、USAGE IS PROCEDURE-POINTER 節を使用して定義されたグループ項目または基本項目を記述するために使用することはできません。

プロシージャー・ポインター・データ項目を条件変数に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、どのクラスやカテゴリーにも属しません。

プロシージャー・ポインター・データ項目は、CORRESPONDING 演算では無視されます。

NATIVE 句

NATIVE 句を使用すると、文字データと浮動小数点データを z/OS® または OS/390® と Linux プラットフォームで表わされるように混在させることができます。NATIVE 句は CHAR(EBCDIC) および FLOAT(BE) コンパイラー・オプションをオーバーライドします。これらはホストのデータ・タイプの使用法を示します。

プログラム内でホストのデータ・タイプと固有データ・タイプ (ASCII と EBCDIC、および IEEE 浮動小数点) を使用できるようになるのは、こうしたデータ項目が NATIVE 句で特に定義されている場合のみです。

NATIVE を指定しても、データ項目のクラスまたはカテゴリーは変更されません。

数値データ項目は、内部表現に関係なく、その論理数値に基づいて算術演算 (数値比較、算術式、数字ターゲットへの代入、算術ステートメント) で処理されます。

文字は、代入前にターゲット項目の表記に変換されます。

比較は、オペランドに適用できる照合シーケンス規則に基づいて行われます。 固有および非固有の英数字または DBCS 文字が比較される場合、比較は、有効になっている COLLSEQ オプションに基づいて行われます。

VALUE 節

VALUE 節は、データ項目の初期内容または条件名と関連付けられる値を指定します。 VALUE 節の用途は、それが DATA DIVISION のどのセクションで指定されているかに応じて異なります。

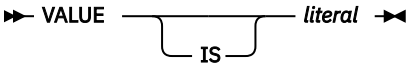
条件名項目以外の項目の FILE SECTION または LINKAGE SECTION で使用される VALUE 節は構文チェックされますが、プログラムの実行には何も影響しません。

WORKING-STORAGE SECTION および LOCAL-STORAGE SECTION では、VALUE 節は、条件名項目の中で使用することも、またはいずれかのデータ項目の初期値を指定するために使用することもできます。 その場合、そのデータ項目は、 プログラム実行の開始時点に指定された値を持つことになります。 初期値が明示的に指定されていない場合、その値は未確定です。

フォーマット 1

フォーマット 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 節や JUSTIFIED 節が指定されていても、それらとは無関係です。

フォーマット 1: リテラル値



OCCURS 節を含むか、または OCCURS 節に従属しているデータ記述項目の中に指定されているフォーマット 1 の VALUE 節は、 関連付けられたデータ項目が発生するたびに、指定された値を割り当てます。 OCCURS 節の DEPENDING ON 句を含む構造は、 VALUE による初期設定を目的として、それぞれ最大のオカレンス項目数を持っているものとみなされます。

VALUE 節は、EXTERNAL 節かまたは REDEFINES 節のいずれかを持つ項目を含むデータ記述項目か、 またはそれに従属しているデータ記述項目に対しては、指定することはできません。 この規則は、条件名項目には適用されません。

フォーマット 1 の VALUE 節は、基本データ項目またはグループ項目に対して指定することができます。 VALUE 節をグループ・レベルで指定する場合には、グループ域は、このグループ内にある従属項目を考慮せずに初期化されます。 さらに、このグループ内の従属項目に対して VALUE 文節を指定することはできません。

グループ項目の場合、従属項目が JUSTIFIED または SYNCHRONIZED VALUE 節を含む場合には、VALUE 節を指定してはなりません。

英数字グループに対して VALUE 節を指定する場合には、すべての従属項目は USAGE DISPLAY を使用して明示的または暗黙的に記述する必要があります。

VALUE 節は、データ記述項目の中にある他の節や、項目の階層のデータ記述の中にある他の節と矛盾するものであってはなりません。

記述された項目の初期値を判別する際には、PICTURE 節の編集文字の機能は無視されます。 ただし、編集文字は、項目サイズを判別する際には計算に入れられます。 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、リテラルが PICTURE +999.99 として定義され、 その値が +12.34 である場合、VALUE 節は、 VALUE "+012.34" と指定する必要があります。

VALUE 節を、外部浮動小数点項目に対して指定することはできません。

VALUE 節はタイプ名のデータ記述項目に指定できます。 このような VALUE 節は、このようなタイプ名を参照する TYPE 節を使用して定義されている任意のデータ名 (タイプ名ではない) を初期設定するために使用されます。 TYPE 節のサブジェクトのデータ記述に VALUE 節が指定されると、この項目に関しては、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 節はすべて無視されます。

先行するデータ項目が DEPENDING ON 句を持つ OCCURS 節を含む場合には、 それに続くデータ項目は VALUE 節を含むことはできません。

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 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 定義を突き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、当該項目の USAGE に PACKED-DECIMAL が指定されている場合は例外です。 その場合は、非数字リテラルはパック 10 進数に変換されます。

リテラルの値に関する規則

- リテラルを指定できるところであればどこでも、13 ページの『形象定数』 で指定されている規則に従って、 形象定数をその代わりに指定することができます。
- 項目が数字クラスである場合には、VALUE 節のリテラルは数字でなければなりません。 リテラルが WORKING-STORAGE 項目または LOCAL-STORAGE 項目の値を定義する場合、リテラルの位置合わせは数値の移動の規則に従って行われますが、追加の制約事項が 1 つあります。それは、このリテラルは、ゼロ以外の数字を切り捨てる必要のある値を持つことはできないということです。 そのリテラルが符号付きである場合は、 関連した PICTURE 文字ストリングには、符号記号 (S) が含まれなければなりません。
- 一部の例外を除いて、VALUE 節の数値リテラルは、その項目の PICTURE 節によって指定される値の 範囲内にある値でなければなりません。 例えば、PICTURE 99PPP の場合、リテラルは 0 であるか、または 1000 から 99000 の範囲内でなければなりません。 PICTURE PPP99 である場合、リテラルは 0.00000 から 0.00099 の範囲内でなければなりません。

  例外を以下に示します。

  – USAGE COMP-5 を使用して記述したデータ項目には、 その PICTURE 節にピクチャー記号 P はありません。
  – TRUNC(BIN) コンパイラー・オプションが有効であるとき、USAGE BINARY、COMP、または COMP-4 を使用して記述したデータ項目には、 その PICTURE 節に PICTURE 記号 P はありません。

    上記の項目の VALUE 節は、 本来のネイティブ・バイナリー表記の容量と同じ大きさまでの値を持つことができます。
- USAGE DISPLAY を指定して記述された英字、英数字、英数字編集、または数字編集項目に対して、VALUE 節を指定する場合、VALUE 節のリテラルは英数字リテラルまたは形象定数でなければなりません。 リテラルの位置合わせは英数字の位置合わせの規則に従って行われますが、追加の制約事項が 1 つあります。それは、 リテラル中の文字数が項目のサイズを超えてはならないことです。
- USAGE NATIONAL を指定して記述された基本国別、国別編集、または数字編集項目に対して、VALUE 節を指定する場合、VALUE 節のリテラルは国別リテラル、英数字リテラル、または 13 ページの『形象定数』に示すような形象定数でなければなりません。 英数字リテラルの値は、そのソース・コード表現から UTF-16 表現に変換されます。 リテラルの位置合わせは英数字の位置合わせの規則に従って行われますが、 追加の制約事項が 1 つあります。それは、リテラルの中の文字数は項目の文字位置のサイズを超えてはならないことです。
- 英数字グループに対して VALUE 節をグループ・レベルで指定する場合、リテラルは英数字リテラル、または 13 ページの『形象定数』 で示されているように、ALL 国別リテラル 以外の形象定数でなければなりません。 リテラルのサイズは、グループ項目のサイズを超えてはなりません。
- 国別グループに対して VALUE 節をグループ・レベルで指定する場合、リテラルは英数字リテラル、国別リテラル、または形象定数の ZERO、SPACE、 QUOTES、HIGH-VALUE、LOW-VALUE、シンボリック文字、ALL 国別リテラル、または ALL リテラルのうちのいずれか 1 つにすることができます。 英数字リテラルの値は、そのソース・コード表現から UTF-16 表現に変換されます。 形象定数は、それぞれ国別文字値を表します。 リテラルのサイズは、グループ項目のサイズを超えてはなりません。

- DBCS 項目と関連付けられた VALUE 節は、DBCS リテラル、形象定数 SPACE、または 形象定数 ALL *DBCS* リテラル を含まなければなりません。 リテラルの長さは、データ項目の PICTURE 節が示すサイズを超えてはなりません。
- 国別リテラルを指定する VALUE 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 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。
- COMPUTATIONAL-1 項目、 または COMPUTATIONAL-2 (内部浮動小数点) 項目と関連付けられた VALUE 節では、 浮動小数点リテラルを指定する必要があります。 さらに、形象定数 ZERO、および整数と 10 進数の両形式の ZERO リテラルは、 浮動小数点 VALUE 節の中で指定できます。

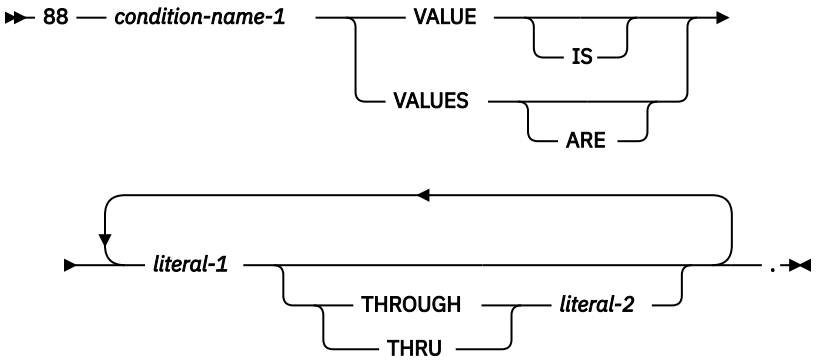
  浮動小数点フォーマットの数値リテラルを、固定小数点数値項目の VALUE 節に指定することはできません。

  浮動小数点リテラル値の詳細については、31 ページの『浮動小数点リテラルの値に関する規則』を参照してください。
- 項目がブールの場合、VALUE 節はブール・リテラルでなければなりません。

フォーマット 2

このフォーマットは、1 つの値、複数の値、値の範囲を条件名に関連付けます。 そのような条件名は、それぞれ別のレベル 88 の項目を必要とします。 レベル番号 88 と条件名は、フォーマット 2 の VALUE 節自体の一部ではありません。 これらは、表現を明確にするためにのみフォーマットに含めたものです。

フォーマット 2: 条件名の値



条件名-*1*
ある値を条件変数に関係付けるユーザーの指定した名前。 関係付けられた条件変数が添え字や指標を必要とする場合、 この条件名を手続き部で参照するたび、 必要に応じて条件変数に添え字または指標を付けなければなりません。

条件名は、プロシージャーとして条件名条件の中でテストされます (238 ページの『条件式』を参照)。

リテラル-*1*
条件名を単一の値に関連付けます。

リテラル-*1* のクラスは、 関係付けられた条件変数への割り当てにとって有効なクラスでなければなりません。

リテラル-*1* THROUGH リテラル-*2*
条件名を少なくとも 1 つの値の範囲に関連付けます。 THROUGH 句を使用する場合、関連データ項目が非年末尾型ウィンドウ表示日付フィールドである場合を除き、リテラル-*1* はリテラル-*2* よりも小さくなければなりません。詳しくは、223 ページの『条件名項目の規則』を参照してください。

リテラル-*1* およびリテラル-*2* は、同じクラスに属していなければなりません。 リテラル-*1* およびリテラル-*2* のクラスは、 関係付けられた条件変数への割り当てにとって有効なクラスでなければなりません。

THROUGH 句に指定する英数字リテラル、国別リテラル、または DBCS リテラルの範囲は、関連の条件変数に有効になっている照合シーケンスに基づいています。 照合シーケンスの詳細については、563 ページの『付録 H ロケールの考慮事項』を参照してください。

関連付けられている条件変数が DBCS クラスである場合には、リテラル-*1* および リテラル-*2* は DBCS リテラルでなければなりません。 形象定数 SPACE または形象定数 ALL *DBCS* リテラル を指定することができます。

関連付けられている条件変数が NATIONAL クラスである場合には、所定の条件名に対して、リテラル-*1* およびリテラル-*2* の両方が、国別リテラルまたは英数字リテラルでなければなりません。 形象定数 ZERO、SPACE、QUOTE、HIGH-VALUE、LOW-VALUE、シンボリック文字、ALL 国別リテラル、または ALL リテラル を指定することができます。

条件名項目の規則

条件名項目には一定の規則があります。

規則は以下のとおりです。

- VALUE 節は、条件名項目内で必須であると同時に、その項目内の唯一の節でなければなりません。 各条件名項目は、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 項目の前には必ず、条件変数に関する項目、または別のレベル 88 項目 (1 つの条件変数にいくつかの条件名が適用されるとき) がなければなりません。 そのようなレベル 88 項目は、 それぞれその条件変数の PICTURE 特性を暗黙のうちに持ちます。
- 連続したオペランドを区切るには、スペース、分離文字コンマ、または分離文字セミコロンが必要です。

  各項目は、分離文字ピリオドで終わらせる必要があります。
- キーワードの THROUGH と THRU は同じ意味です。
- 特定の条件変数に関係付けられる条件名は、 その条件変数の項目のすぐ後になければなりません。 条件変数は、以下のものを除く、任意の基本データ記述項目にすることができます。
  - 別の条件名。
  - RENAMES 節 (レベル 66 項目)。
  - USAGE IS INDEX で定義されているデータ項目。
  - USAGE POINTER、USAGE PROCEDURE-POINTER、または USAGE FUNCTION-POINTER を使用して記述された項目
- 条件名は、グループ・レベルおよび英数字グループ、または国別グループ内の従属レベルの両方で指定できます。
- 英数字グループ・データ記述項目に対して条件名を指定する場合
  - リテラル-*1* (または リテラル-*1* と リテラル-*2*) の値は、英数字リテラルまたは形象定数として指定する必要があります。
  - グループには、どの USAGE の項目でも含めることができます。
- 国別グループ・データ記述項目に対して条件名を指定する場合
  - リテラル-*1* (またはリテラル-*1* とリテラル-*2*) の値は、英数字リテラル、国別リテラル、または形象定数として指定する必要があります。
  - グループには、USAGE が NATIONAL の項目のみを含めることができます (このことは 169 ページの『GROUP-USAGE 節』に示してあります)。
- 条件名が英数字グループ・データ記述項目、国別グループ・データ記述項目、に関連付けられている場合には、以下のようになります。
  - それぞれのリテラルのサイズは、グループ内の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 節または SYNCHRONIZED 節を含めることはできません。

- 条件名の定義によって暗黙指定されている比較テストは、下記の表に示す規則に従って実行されます。

表 *23.* **条件名に関する比較テストの参照先**	
条件変数のタイプ	**比較条件の規則**
英数字グループ項目	249 ページの『グループの比較』
国別グループ項目 (国別クラスの基本データ項目として扱われる)	248 ページの『国別の比較』
英数字クラスの基本データ項目	247 ページの『英数字比較』
国別クラスの基本データ項目	248 ページの『国別の比較』
数字クラスの基本データ項目	249 ページの『数字の比較』
DBCS クラスの基本データ項目	248 ページの『DBCS 比較』

- 国別リテラルを指定する VALUE 節は、国別クラスのデータ項目に対してのみ定義されている条件名に関連付けることができます。
- DBCS リテラルを指定する VALUE 節は、DBCS クラスのデータ項目に対してのみ定義されている条件名と関連付けることができます。
- 国別クラスの基本データ項目または国別グループ項目の場合、条件名項目の中のリテラルは国別リテラルまたは英数字リテラルのいずれかであって、リテラル-*1* およびリテラル-*2* は同じクラスでなければなりません。 その他のクラスの英数字グループまた基本データ項目の場合には、リテラルのタイプは条件変数のデータ・タイプと一致していなければなりません。 次に例を示します。
  - CITY-COUNTY-INFO、COUNTY-NO、および CITY は、条件変数です。

    COUNTY-NO と関連付けられた PICTURE が、条件名の値を 2 桁の数字リテラルに限定します。

    CITY と関連付けられた PICTURE が、条件名の値を 3 桁の英数字リテラルに限定します。
  - 関連付けられた条件名は、レベル 88 の項目です。

    CITY-COUNTY-INFO と関連付けられた条件名の値は、どれも 5 文字以下でなければなりません。

    これは英数字グループ項目なので、リテラルは英数字リテラルでなければなりません。

```
05  CITY-COUNTY-INFO.
    88  BRONX                  VALUE "03NYC".
    88  BROOKLYN               VALUE "24NYC".
    88  MANHATTAN              VALUE "31NYC".
    88  QUEENS                 VALUE "41NYC".
    88  STATEN-ISLAND          VALUE "43NYC".
  10  COUNTY-NO                PICTURE 99.
      88  KINGS                 VALUE 24.
      88  NEW-YORK              VALUE 31.
      88  RICHMOND              VALUE 43.
  10  CITY                     PICTURE X(3).
      88  BUFFALO               VALUE "BUF".
      88  NEW-YORK-CITY         VALUE "NYC".
      88  POUGHKEEPSIE          VALUE "POK".
05  POPULATION...
```

- 条件名は、日付項目、時刻項目、またはタイム・スタンプ項目に関連付けることができます。 この場合は、次のようになります。
  - 条件名の値は非数字リテラルとして指定されなければなりません。
  - 暗黙的に条件名ごとに条件変数の FORMAT 特性があります。 したがって、この条件名を使用する比較テストはすべて、日時クラスの項目を比較する場合の規則に従って行われます。
  - 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できます。 この場合、リテラル-*1* の日時はリテラル-*2* の日時よりも前の日時でなければなりません。
- 項目がウィンドウ表示日付フィールドである場合には、以下の制約事項が適用されます。

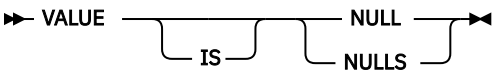
- 英数字の条件変数の場合:
  - リテラル-*1* およびリテラル-*2* (指定される場合) は、 ともに条件変数と同じ長さの英数字リテラルでなければなりません。
  - リテラルを形象定数として指定してはなりません。
  - リテラル-*2* を指定する場合は、 両方のリテラルに 10 進数字だけしか含めてはなりません。
- YEARWINDOW コンパイラー・オプションを負の整数として指定する場合は、リテラル-*2* を指定してはなりません。
- リテラル-*2* を指定する場合は、 YEARWINDOW コンパイラー・オプションで指定された世紀ウィンドウの適用後に、リテラル-*1* がリテラル-*2* よりも小さくなければなりません。 すなわち、リテラル-*1* の拡張日付値が、リテラル-*2* の拡張日付値よりも小さくなければなりません。

ウィンドウ表示日付フィールドで条件名を使用する詳しい方法については、 『条件名条件とウィンドウ表示日付フィールドの比較』 を参照してください。

フォーマット 3

このフォーマットでは、USAGE POINTER、USAGE PROCEDURE-POINTER、または USAGE FUNCTION-POINTER として定義された項目の初期値として無効なアドレスが割り当てられます。

フォーマット 3: NULL 値



VALUE IS NULL を指定できるのは、暗黙的または明示的に USAGE POINTER、USAGE PROCEDURE-POINTER、または USAGE FUNCTION-POINTER として定義されている基本項目のみです。

第 6 部 手続き部

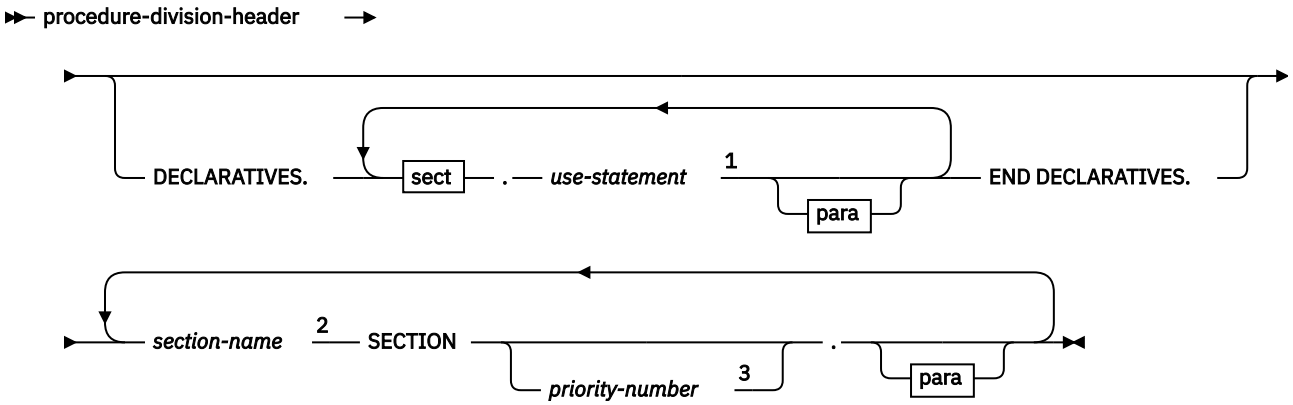
第 18 章 手続き部の構造

PROCEDURE DIVISION はオプションの部です。

プログラム手続き部

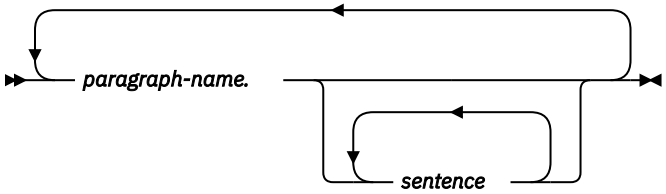
プログラムの手続き部は、オプションの宣言部分と、セクション、段落、文、およびステートメントを含むプロシージャーで構成されています。

フォーマット: 手続き部



セクション

段落



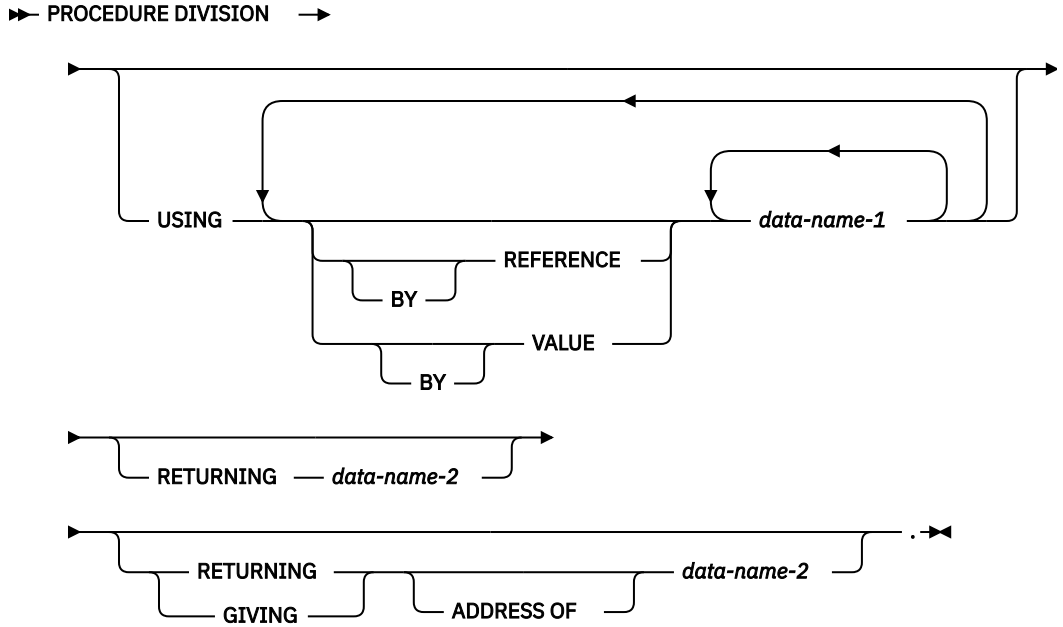
注:

[1] USE ステートメントについては、493 ページの『USE ステートメント』で説明しています。
[2] セクション名は省略することができます。 セクション名を省略する場合は、段落名も省略できます。
[3] 優先順位番号は、再帰的プログラムにおいては無効です。

PROCEDURE DIVISION ヘッダー

PROCEDURE DIVISION が指定される場合には、次のプログラム手続き部ヘッダーによって識別されます。

フォーマット: プログラム手続き部のヘッダー



USING 句

USING 句は、プログラムの呼び出し時にプログラムが受け取るパラメーターを指定します。

USING 句は、非宣言部分の先頭部分に入力される、呼び出されるサブプログラムの PROCEDURE DIVISION ヘッダーで有効です。 それぞれの USING ID は、呼 び出されるサブプログラムの LINKAGE SECTION で、レベル 01 またはレベル 77 の項目として定義する必要があります。

ENTRY ステートメントの後に続く最初の実行可能ステートメントで 入力された呼び出されるサブプログラムにおいて、USING 句は、ENTRY ステートメントの中で有効です。 それぞれの USING ID は、呼び出されるサブプログラムの LINKAGE SECTION で、レベル 01 またはレベル 77 の項目として 定義する必要があります。

ただし、CALL ステートメントの USING 句に指定されたデータ項目は、呼び出し側の COBOL プログラムの DATA DIVISION では、任意のレベルのデータ項目にすることができます。

ヘッダーの USING 句の中のデータ項目のデータ記述項目の中には、 REDEFINES 節を指定できます。

ユーザーは、COBOL 以外のプログラムから COBOL プログラムを呼び出したり、 システム・コマンドから COBOL メインプログラムにユーザー・パラメーターを渡したりすることができます。

コマンド行引数は、常にネイティブ・データ型として渡されます。 ホスト・データ型コンパイラー・オプションの CHAR(EBCDIC) または FLOAT(BE) を指定した場合は、引数の記述にそれらのコンパイラー・オプションで影響されるデータ型で NATIVE 句を指定しなければなりません。

呼び出す側と呼び出される側のサブプログラム内で、USING ID を指定する順序によ り、両方で使用可能な単一データ・セットの対応が決まります。 この対応付けは、位置関係によって決まるもので名前によるものではありません。 呼び出しサブプログラムと呼び出されるサブプログラムの場合、 対応する ID のバイト数は同じでなければならず、データ記述は同じである必要はありません。

指標名の場合、対応は確立されません。 呼び出し側プログラムと呼び出さ れるプログラムの中の 指標名は、常にそれぞれ個別の指標を参照します。

CALL USING ステートメントで指定した ID は、呼び出し側プログラム、あるいは呼び出されたプログラム内で参照できるプログラムが使用可能なデータ項目を指定します。 これらの項目は、どの DATA DIVISION セクションにも定義できます。

USING 句には、特定の ID を複数回指定できます。 CALL 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。

BY REFERENCE 句も BY VALUE 句も、 別の BY REFERENCE 句や BY VALUE 句で上書きされるまで、 それぞれの後に付くすべてのパラメーターに適用されます。

BY REFERENCE
BY CONTENT または BY REFERENCE によって引数を渡す場合は、 PROCEDURE または ENTRY USING 句の対応する仮パラメーターに対して BY REFERENCE を指定または暗黙指定する必要があります。

BY REFERENCE と BY VALUE を両方とも指定しないと、 BY REFERENCE がデフォルト値になります。

CALL ステートメント内の対応するデータ項目への参照で、BY REFERENCE によって (明示的または暗黙的に) 渡されるパラメーターが宣言されている場合は、呼び出されるサブプログラムの USING ID への各参照が、呼び出し側プログラムの対応する USING ID への参照によって置換されるようにプログラムが実行されます。

CALL ステートメント内の対応するデータ項目への参照で、BY CONTENT によっ て渡されるパラメーターが宣言されている場合、項目の値が移動するのは、CALL ステートメントが実行され、データ名-*1* の LINKAGE SECTION で宣言された属性を所有しているシステム定義ストレージ項目にそのステートメントが格納された場合で す。CALL ステートメントの BY CONTENT 句の中の各パラメーターのデータ記 述は、ヘッダーの USING 句の中の対応するパラメーターのデータ記述と 同じでなければなりません (変換、拡張、切り捨てがあってはなりません)。

BY VALUE
BY VALUE により引数が渡されるときは、送り出しデータ項目への参照ではなく、引数の値が渡されます。 受け取り側のサブプログラムは、送り出しデータ項目の 一時コピーにしかアクセスできません。 つまり、BY VALUE により渡された引数に対応する 仮パラメーターを変更しても、その引数には影響がありません。 これらの概念を表す例については、「*COBOL for Linux on x86* プログラミング・ガイド」内の『データの引き渡し』 を 参照してください。

データ名-*1*
LINKAGE SECTION では、データ名-*1* をレベル 01 項目またはレベル 77 項目にする必要があります。

GIVING/RETURNING 句

GIVING/RETURNING 句では、プログラムの結果を受け取るデータ項目を指定します。RETURNING と GIVING は同じ意味です。

データ名-*2*
データ名-*2* は、RETURNING データ項目です。 LINKAGE SECTION では、データ名-*2* をレベル 01 項目またはレベル 77 項目にする必要があります。

RETURNING データ項目は、出力専用のパラメーターです。

次のプログラムには、PROCEDURE DIVISION RETURNING 句を使わないでください。

- ENTRY ステートメントを含むプログラム
- ネストされたプログラム
- メインプログラム: メインプログラムに PROCEDURE DIVISION RETURNING を指定すると、結果は予測できません。呼び出されるサブプログラムにのみ PROCEDURE DIVISION RETURNING 句を指定する必要があります。メインプログラムの場合は、RETURN-CODE 特殊レジスターを使用して操作環境に値を戻してください。

ADDRESS OF 特殊レジスター
このレジスターに関する情報については16 ページの『ADDRESS OF』を参照してください。

LINKAGE SECTION の項目への参照

呼び出されるプログラムの LINKAGE SECTION に定義されたデータ項目は、それらがトピックに示された条件のうちの 1 つを満たしている場合にのみ、そのプログラムの PROCEDURE DIVISION 内で参照可能です。

- それらのデータ項目が ENTRY ステートメント、または PROCEDURE DIVISION のヘッダーの USING 句のオペランドである場合
- そのデータ項目が SET ADDRESS OF、または CALL ... BY REFERENCE ADDRESS OF のオペランドである場合
- それらのデータ項目が、REDEFINES 節または RENAMES 節を使用して定義され、 そのオブジェクトが上記 2 つの条件を満たす場合
- それらのデータ項目が、上記の規則に関する条件を満たすいずれかの項目に従属する項目である場合
- それらのデータ項目が、 上記の条件のいずれかを満たすデータ項目に関連付けられた条件名または指標名である場合

宣言部分

宣言部分に、例外条件が発生する際に実行される 1 つ以上の特定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場合は、それらのセクションを PROCEDURE DIVISION の冒頭部分にグループ化し、 その手続き部全体をいくつかのセクションに分ける必要があります。

各宣言セクションは、そのセクションの機能を識別する USE ステートメントで始まります。 その後に続く一連のプロシージャーには、例外条件が発生した場合に実行される処理を 指定します。 各宣言セクションは、USE ステートメントが続けて記述されている別のセクション名で終了しますが、 キーワード END DECLARATIVES によっても終了します。

宣言セクション・グループ全体の前には、キーワード DECLARATIVES を指定します。DECLARATIVES は、PROCEDURE DIVISION のヘッダーの後の行に指定します。 宣言セクション・グループの後には、キーワード END DECLARATIVES を指定します。 DECLARATIVES と END DECLARATIVES というキーワードは、両方とも領域 A で始め、後に分離文字ピリオドを付けなければなりません。 同じ行に他のテキストがあってはなりません。

PROCEDURE DIVISION の宣言部分では、各セクション・ヘッダーは 後に分離文字ピリオドを付け、その後に USE ステートメントを書かなければなりません。 このステートメントの後にも分離文字ピリオドを付けます。 同じ行に他のテキストがあってはなりません。

USE ステートメントのフォーマットは次のとおりです。

- 493 ページの『EXCEPTION/ERROR 宣言』
- 495 ページの『DEBUGGING 宣言 』

USE ステートメント自体が実行されることはありません。 その代わりに、USE ステートメントは、 その後にあるプロシージャー型段落 (行われる処置を指定している) が実行される条件を定義します。 そのプロシージャーの実行が終わると、このプロシージャーを活動化したルーチンに制御が戻されます。

宣言型プロシージャーを非宣言型プロシージャーから実行できます。

非宣言型プロシージャーを宣言型プロシージャーから実行できます。

宣言型プロシージャーは、宣言型プロシージャー中の GO TO ステートメント中で参照できます。

非宣言型プロシージャーは、宣言型プロシージャー中の GO TO ステートメント中で参照できます。

すでに呼び出されていて、まだ制御権を持っている 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も構いません。 ただし、無限ループを避けるため最終的な出口が最後にあることを確かめておく必要があります。

宣言型プロシージャーはその中の最後のステートメントが実行されると終了します。

プロシージャー

PROCEDURE DIVISION 内で、プロシージャー は、セクション またはセクションのグループ、および段落または段落のグループから構成されています。

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・ヘッダー の後ろには、必要に応じて、1 つ以上の段落を続けます。

セクション・ヘッダー
セクション名 の後ろには、キーワード SECTION、優先順位番号 (任意)、 および分離文字ピリオドを続けます。

キーワード END DECLARATIVES の後、または宣言部分がない場合、セクションのヘッダーはオプションです。

セクション名
セクションを識別するためのユーザー定義語です。 参照されるセクション名は、それが定義されているプログラムの中で固有でなければなりません。 セクション名は修飾することができないからです。

優先順位番号
整数 または正の符号付き数字リテラル 。0 から 99 の範囲の値。 優先順位番号 は固定セグメントまたは節を含む予定の独立セグメントを識別します。

宣言部分のセクションの優先順位番号は、 0 から 49 の範囲でなければなりません。

次のものには、優先順位番号を指定できません。

- RECURSIVE 属性を指定して宣言されているプログラム内

1 つのセクションは、次のセクション・ヘッダーの直前で終了するか、 PROCEDURE DIVISION の終わりで終了するか、または宣言部分の中でキーワードの END DECLARATIVES によって終了します。

セグメント
セグメントは、プログラム内の同じ優先順位番号を持つすべてのセクションから構成されます。 優先順位番号によって、実行時にセクションが固定セグメントまたは独立セグメントのいずれに保管されるかが決定されます。

優先順位番号が 0 から 49 のセグメントは固定セグメントです。 優先順位番号が 50 から 99 のセグメントは独立セグメントです。

セグメントのタイプ (固定または独立) はセグメンテーションの機能をコントロールします。

固定セグメントでは、プロシージャーは常に最後に使われた状態になります。 独立セグメントでは、GOBACK ステートメントまたは EXIT PROGRAM ステートメントを実行した結果として制御が渡される場合を除き、異なる優先順位番号を持つセグメントから制御を受け取るたびにプロシージャーは初期状態になります。 独立セグメントで ALTER、SORT、および MERGE ステートメントを使用する場合の制限事項がステートメントの下に記述されています。

COBOL for Linux は、85 COBOL 標準分割モジュールのオーバーレイ機能はサポートしていません。

段落
段落名 の後ろには、分離文字ピリオドを続けます。必要に応じて、1 つ以上の文をさらに 続ける場合もあります。

段落の前には必ずピリオドを付けます。段落が置かれるのは常に IDENTIFICATION DIVISION のヘッダー、セクション、または他の段落の後であり、 これらはすべてピリオドで終わらなければならないからです。

段落名
段落を識別するためのユーザー定義語です。 段落名は、修飾することが可能なので、固有である必要はありません。

宣言部分がない場合 (フォーマット-2)、PROCEDURE DIVISION の中に段落名は不要です。

1 つの段落は、次の段落名またはセクション・ヘッダーの直前で終了するか、 PROCEDURE DIVISION の終わりで終了するか、または宣言部分の中でキーワードの END DECLARATIVES によって終了します。

セクションの中に 1 つ以上の段落がある場合でも、すべての段落をセクションに入れる必要はありません。

文
1 つ以上のステートメント からなり、分離文字ピリオドで終わります。

ステートメント
COBOL ステートメントで始まる記号 (リテラルや比較演算子など) と *ID* の構文的に正しい組み合わせ。

ID
データ項目を一意に参照するために必要な 1 つまたは複数の語。 必要に応じて、修飾語、添え字、指標、および参照の修正を含めることができます。 PROCEDURE DIVISION の参照では (クラス・テストの場合を除き)、ID の内容は、PICTURE 節によって指定されたクラスに必ず合致していなければなりません。そうでない場合、結果は予測不可能になります。

実行は、宣言部分を除く PROCEDURE DIVISION の最初のステートメントから開始されます。 ステートメントは、コンパイルを行うために記述してある順序で実行されますが、 ステートメントの規則が別の実行順序を指示している場合はこの限りではありません。

PROCEDURE DIVISION の終わりは、次の項目のいずれかによって指示されます。

- IDENTIFICATION DIVISION のヘッダー。これはネストされたソース・プログラムの開始を示します。
- END PROGRAM のマーカー。
- プログラムの物理的な終わり。つまり、そこから後はソース・プログラム行がなくなるソース・プログラム中の物理的な位置。

算術式

算術式は、ある種の条件ステートメントや算術ステートメントのオペランドとして使われます。

算術式は、以下の項目のいずれかにより構成することができます。

1. 数字基本項目 (数字関数を含む) として記述された ID
2. 数字リテラル
3. 形象定数 ZERO
4. 項目 1、2、および 3 で定義して算術演算子で区切った ID とリテラル
5. 項目 1、2、3、または 4 で定義して算術演算子で区切った 2 つの算術式
6. 項目 1、2、3、4、または 5 で定義して括弧で囲んだ算術式

すべての算術式には、単項演算子を先行させることができます。

算術式の中で使われる ID やリテラルは、そこで算術計算が行える、数字基本項目または数字リテラルのどちらかを表していなくてはなりません。

指数式が正の数と負の数の両方で評価される場合、結果は常に正の値となります。 例えば、4 の平方根の場合を考えてみましょう。

```
4 ** 0.5
```

この場合、+2 と -2 が計算結果となります。 COBOL for Linux は常に +2 を戻します。

累乗される式 (指数の底) の値が 0 の場合、指数は 0 より大きな値にしなければなりません。 さもないと、サイズ・エラー条件になります。 式の計算の結果として、実数が存在しないような場合には、サイズ・エラー条件になります。

算術演算子

5 つの 2 項算術演算子および 2 つの単項算術演算子を、算術式で使用することができます。 これらの演算子は、その前後にスペースを伴う特定の文字で表されます。

これらの 2 項算術演算子および単項算術演算子は、235 ページの表 24 に示されています。

表 *24.* ***2*** **項演算子と単項演算子**

2 項演算子	**意味**	**単項演算子**	**意味**
+	加算	+	+1 による乗算
-	減算	-	-1 による乗算
*	乗算		
/	除算		
**	指数		

制限: 固定小数点指数式の指数は、9 桁を超えることはできません。 コンパイラーは、9 桁を超える指数を切り捨てます。 短縮形の場合、指数がリテラルまたは定数の場合、コンパイラーは診断メッセージを発行します。指数が変数またはデータ名の場合は、実行時に診断を発行します。

エレメントが評価される順序を指定するために、算術式の中に括弧を使用できます。

括弧の中の式が最初に計算されます。 式がネストした括弧の中にある場合、式の計算は、 最も包括的でない組 (最も内側) から、 最も包括的な組 (最も外側) へと行われます。

括弧を使用しない場合、または包括するレベルが 同じ括弧で囲んだ式の場合には、次の階層順序で計算が行われます。

1. 単項演算子
2. 指数
3. 乗算と除算
4. 加算と減算

括弧を使用することによって、同一の階層レベルで連続的に演算が行われる場合の論理的なあいまいさを除いたり、 通常の演算実行の階層シーケンスを必要に応じて変更したりできます。 同一の階層レベルにある連続する演算の順序が、 完全に指定されていない場合には、演算は左から右へと順番に行われます。

算術式は、左括弧、単項演算子、 またはオペランド (すなわち ID やリテラル) でのみ始めることができます。 また、右括弧またはオペランドでのみ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。

算術式の左括弧と右括弧の間には 1 対 1 の対応がなければならず、 それぞれの対応において、左括弧は対応する右括弧の左側に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であり、 その算術式が ID または別の算術式のすぐ後に続く場合には、そのすぐ前に左括弧が必要になります。

次の表では、使用可能な算術記号の対を示します。 対になる算術記号とは、そのような 2 つの記号が連続して現れることです。 この表では、「はい」と「いいえ」は次のような意味です。

はい
対にすることができます。

いいえ
対にすることができません。

表 25. 算術記号の有効な対					
	ID またはリテラルの 2 番目の記号	*** / ** + - 2 番目の記号**	**単項 + または単項 - 2 番目の記号**	**(2 番目の記号**	**) 2 番目の記号**
ID またはリテラルの最初の記号	いいえ	はい	いいえ	いいえ	はい
*** / ** + - 最初の記号**	はい	いいえ	はい	はい	いいえ
単項 + または単項 - 最初の記号	はい	いいえ	いいえ	はい	いいえ
(最初の記号	はい	いいえ	はい	はい	いいえ
) 最初の記号	いいえ	はい	いいえ	いいえ	はい

日付フィールドを使用する算術計算

日付フィールドを含む算術演算は、日付フィールドへの非日付の加算、日付フィールドからの非日付の減算、および互換性のある日付フィールドからの日付フィールドの減算に制限されています。

日付フィールド・オペランドは、年部分を除いて同じ日付フォーマットである場合に 互換性があります (年部分はウィンドウ表示西暦年でも拡張西暦年でも可能です)。

次の演算は許可されません。

- 互換性のない日付間の演算
- 2 つの日付フィールドの加算
- 非日付データからの日付フィールドの減算
- 日付フィールドに適用される単項の減算
- 日付フィールドの除算、指数、乗算
- 年末尾型日付フィールドを指定する算術式
- 年末尾型日付フィールドを指定する算術ステートメント。ただし、送り出しフィールドが非日付データである場合の受け取りデータ項目は除く。

サポートされる加算および減算で日付フィールドを使用した場合の結果を、以下のセクションに示します。

算術演算で日付フィールドを使用する方法については、以下を参照してください。

- 280 ページの『ADD ステートメント』
- 294 ページの『COMPUTE ステートメント』
- 389 ページの『SUBTRACT ステートメント』

使用上の注意

- 算術演算では、日付フィールドは数字項目として扱われます。 日付固有の内部構造であるとは認識されません。 例えば、値 991231 (アプリケーションで 1999 年 12 月 31 日を表すのに使用できる値) を内容とするウィンドウ表示日付フィールドに 1 を加算すると、値は 000101 ではなく 991232 になってしまいます。

- ウィンドウ表示日付フィールドを算術式または算術ステートメントでオペランドとして使用すると、 それは YEARWINDOW コンパイラー・オプションによって指定される世紀ウィンドウに応じて、自動的に拡張されます。を参照してください。

日付フィールドが関係する加算

次の表は、日付フィールドを使用した場合の結果および加算の互換オペランドを示しています。

表 *26.* **加算で日付フィールドを使用した場合の結果**		
	非日付データの第 2 オペランド	**日付フィールドの第 2 オペランド**
非日付データの第 1 オペランド	非日付データ	日付フィールド
日付フィールドの第 1 オペランド	日付フィールド	許可されない

結果が受け取りフィールドにどのように保管されるかについては、237 ページの『日付フィールドに関連する算術演算結果の保管』を参照してください。

日付フィールドが関係する減算

このセクションでは、減算で日付フィールドを使用した場合の結果について説明します。

次の表は、日付フィールドを使用した場合の結果および減算の互換オペランドを示しています。

第 *1* オペランド - 第 *2* オペランド

SUBTRACT ステートメントでは、 これらのオペランドは次のように逆順に現れます。

SUBTRACT 第 *2* オペランド FROM 第 *1* オペランド

表 *27.* **減算で日付フィールドを使用した場合の結果**		
	非日付データの第 2 オペランド	**日付フィールドの第 2 オペランド**
非日付データの第 1 オペランド	非日付データ	許可されない
日付フィールドの第 1 オペランド	日付フィールド	非日付データ

日付フィールドに関連する算術演算結果の保管

ADD、COMPUTE、DIVIDE、MULTIPY、および SUBTRACT の各ステートメントは、算術演算を実行し、その結果または送り出しフィールドを 1 つ以上の受け取りフィールドに保管します。

MULTIPLY ステートメントで日付フィールドにすることができるのは GIVING ID のみです。 DIVIDE ステートメントでは、GIVING ID または REMAINDER ID だけに日付フィールドを使用することができます。

算術式または算術ステートメントのオペランドであるウィンドウ表示日付フィールドは、162 ページの『ウィンドウ表示日付フィールドのセマンティクス』で説明されているように、使用前に拡張されたかのように扱われます。

送り出しフィールドが日付フィールドである場合、受け取りフィールドは互換日付フィールドでなければなりません。 すなわち、ウィンドウ表示西暦年か拡張西暦年の年部分を除けば、 2 つのフィールドの日付フォーマットは同じでなければなりません。

ON SIZE ERROR 節がステートメントに指定されなかった場合には、保管操作はそのステートメントの既存の COBOL 規則に従い、受け取りフィールドと送り出しフィールド (ウィンドウ表示日付フィールド・オペランド、 または結果の自動拡張後) の両方が非日付フィールドであるかのように処理されます。

238 ページの表 28 は、これらのステートメントが送り出しフィールドの値を受け取りフィールドに 保管する方法を示しています (どちらかのフィールドが日付フィールドである場合)。 このセクションでは、保管の実行方法を記述するのに以下の用語を使用します。

ウィンドウ表示なし
ステートメントは、266 ページの『SIZE ERROR 句』で説明しているように、特殊な日付依存サイズ・エラー処理を使わずに保管を実行します。

ウィンドウ表示非日付送り出しフィールド
非日付データ送り出しフィールドは、ウィンドウ表示日付受け取りフィールドと互換性のあるウィンドウ表示日付フィールドとして扱われます。ただし、年部分は 1900 年以降の年の値を表します (この表記は、年部分が 2 桁以下の正の値に限定されないことを除けば、基本年が 1900 であるウィンドウ表示日付フィールドに似ています)。 送り出しフィールドのこの想定年部分に 1900 を追加して拡張されたかのようにして、保管が行われます。

ウィンドウ表示日付送り出しフィールド
送り出しフィールドが拡張日付フィールドと互換性をもつように、すべてのウィンドウ表示日付フィールド・オペランドが必要に応じて拡張されたかのようにして、保管が行われます。

サイズ・エラー処理: 2 種類の送り出しフィールドのいずれについても、送り出しフィールドの想定または実際の年部分が世紀ウィンドウの範囲内であれば、 送り出しフィールドは、年部分の世紀コンポーネントを取り除いた後で受け取りフィールドに保管されます。 すなわち、拡張西暦年部分の下位または右端の 2 桁は保持され、上位または左端の 2 桁は破棄されます。

年部分が世紀ウィンドウの範囲内にない場合、 受け取りフィールドは変更されず、 残りの算術演算が完了した時点でサイズ・エラー命令ステートメントが実行されます。

次に例を示します。

```
77  DUE-DATE PICTURE 9(5) DATE FORMAT YYXXX.
77  IN-DATE  PICTURE 9(8) DATE FORMAT YYYYXXX VALUE 1995001.
    ...
    COMPUTE DUE-DATE = IN-DATE + 10000
      ON SIZE ERROR imperative-statement
    END-COMPUTE
```

送り出しフィールドは拡張日付フィールドで、2005 年 1 月 1 日を表します。 2005 が世紀ウィンドウの範囲内であるとすると、 DUE-DATE に保管される値は 05001 です (世紀コンポーネント 20 がない場合は 2005001 の送り出し値)。

表 *28.* ***ON SIZE ERROR*** **を指定した場合の、日付フィールドに関連する算術演算結果の保管**

	非日付データ送り出しフィールド	**日付フィールド送り出しフィールド**
非日付データの受け取りフィールド	ウィンドウ表示なし	許可されない
ウィンドウ表示日付フィールド受け取りフィールド	ウィンドウ表示	ウィンドウ表示
拡張日付フィールド受け取りフィールド	ウィンドウ表示なし	ウィンドウ表示なし

条件式

条件式によってオブジェクト・プログラムは、テスト結果の真の値に基づいて 代替制御パスを選択します。条件式は、EVALUATE、IF、PERFORM、および SEARCH の各ステートメントの中で指定できます。

条件式は、単純条件または複合条件のどちらかで指定することができます。 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めます。その 括弧は、条件が単純と複合のどちらでも変わりません。

単純条件

次に示すように 5 種類の単純条件があります。

単純条件は以下のとおりです。

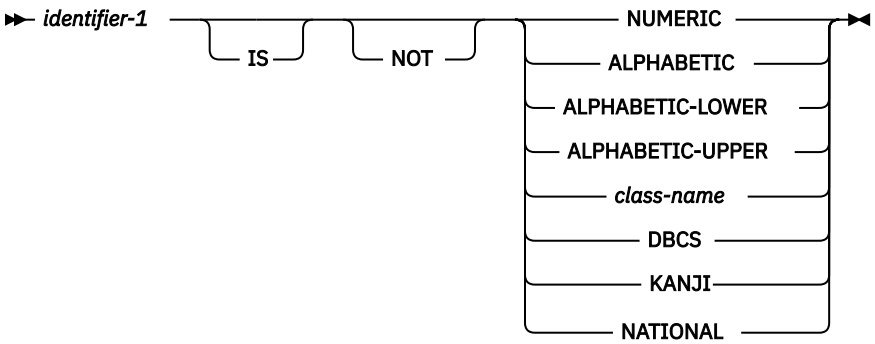
- クラス条件
- 条件名条件
- 比較条件
- 符号条件
- スイッチ状況条件

単純条件は、真か偽の真の値を持ちます。

クラス条件

クラス条件は、データ項目の内容が英字 (alphabetic) であるか、 小文字の英字 (alphabetic-lower) であるか、大文字の英字 (alphabetic-upper) であるか、数字 (numeric) であるか、 DBCS であるか、漢字 (KANJI) であるか、または ENVIRONMENT DIVISION の SPECIAL-NAMES 段落で定義されている CLASS 節で指定された文字セットの中の文字だけからなるかを判別します。

フォーマット



ID-1
以下のいずれかの USAGE を指定して記述されたデータ項目を参照します。

- DISPLAY、NATIONAL、COMPUTATIONAL-3、または PACKED-DECIMAL (NUMERIC が指定された場合)。
- DISPLAY-1 (DBCS または KANJI が 指定された場合)。
- DISPLAY または NATIONAL (ALPHABETIC、 ALPHABETIC-UPPER、 または ALPHABETIC-LOWER が指定された場合)。
- DISPLAY (クラス名が指定された場合)。

NUMERIC を指定した場合は、クラス英字であってはなりません。

ALPHABETIC、ALPHABETIC-UPPER、または ALPHABETIC-LOWER を指定した場合は、クラス数字であってはなりません。

240 ページの表 29 には ID のさまざまなタイプで有効なクラス条件の形式がリストされます。

ID-1 が関数 ID である場合、ID-1 は英数字関数、国別関数、または日時関数を参照する必要があります。

英数字グループ項目は、基本英数字項目を使用できるクラス条件で使用することができますが、そのグループに 1 つ以上の符号付き基本項目が含まれている場合は、NUMERIC クラス条件を使用できません。

NATIONAL
ID-1 は、以下の規則に従って NATIONAL 文字で全体が構成されます。

- NATIONAL データ項目の場合、テストされている ID は USAGE NATIONAL 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。

- NATIONAL 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。

NOT
これを指定すると、NOT と次のキーワードが、真の値を調べるために実行されるクラス・テストを定義します。 例えば、NOT NUMERIC は、NUMERIC クラス・テストの結果が偽 (項目に非数字データが含まれる) であるかどうかを 確認するテストです。

NUMERIC
ID-1 は、演算符号の付いた 0 から 9 の文字、または演算符号の付かない 0 から 9 の文字だけで構成されます。

PICTURE 節が演算符号を含まない場合は、内容が数字であり、かつ演算符号が存在しない場合に限って、 テストされている ID が数字であると判定されます。

PICTURE 節が演算符号を含む場合は、その項目が基本項目であり、 その内容が数字でしかも有効な演算符号が付いている場合に限って、 テストされている ID が数字であると判定されます。

ALPHABETIC
ID-1 全体が A から Z の小文字または大文字のラテン・アルファベットとスペースの任意の組み合わせで構成されます。

ALPHABETIC-LOWER
ID-1 全体が a から z の小文字のラテン・アルファベットとスペースの任意の組み合わせで構成されます。

ALPHABETIC-UPPER
ID-1 全体が A から Z の大文字のラテン・アルファベットとスペースの任意の組み合わせで構成されます。

クラス名
ID-1 は、SPECIAL-NAMES 段落内のクラス名の定義で掲げられている文字から全体が構成されています。

DBCS
ID-1 は、DBCS 文字だけで構成されます。 *ID-1* には、有効な EBCDIC DBCS 文字に対応する DBCS 文字が含まれます。

文字表示が有効であるかどうかを確認するため、項目に対して範囲検査が実行されます。 有効な範囲は、それぞれの DBCS 文字の 2 つのバイトとも、X'41' から X'FE' の範囲で、DBCS のブランクは X'4040' です。 (これらの範囲は、Enterprise COBOL for z/OS の同等の DBCS 文字表記に対応するもので、ワークステーション DBCS 文字の実際の DBCS 文字範囲ではありません。)

KANJI
ID-1 には、有効な EBCDIC DBCS 文字に対応する DBCS 文字が含まれます。

文字表示が有効であるかどうかを確認するため、項目に対して範囲検査が実行されます。 有効な値の範囲は、第 1 バイト目が X'41' から X'7E'、第 2 バイト目が X'41' から X'FE'、DBCS のブランクは X'4040' です。 (これらの範囲は、Enterprise COBOL for z/OS の同等の DBCS 文字表記に対応するもので、ワークステーション DBCS 文字の実際の DBCS 文字範囲ではありません。)

表 *29.* **データ項目のさまざまなタイプに対するクラス条件の有効な形式**

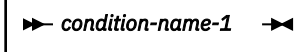
***ID-1* が示すデータ項目のタイプ**	**クラス条件の有効な形式**	
英字	ALPHABETIC ALPHABETIC-LOWER ALPHABETIC-UPPER *class-name*	NOT ALPHABETIC NOT ALPHABETIC-LOWER NOT ALPHABETIC-UPPER NOT クラス名

表 29. **データ項目のさまざまなタイプに対するクラス条件の有効な形式** (続き)		
***ID-1* が示すデータ項目のタイプ**	**クラス条件の有効な形式**	
英数字、 英数字編集、 または数字編集	ALPHABETIC ALPHABETIC-LOWER ALPHABETIC-UPPER NUMERIC *class-name*	NOT ALPHABETIC NOT ALPHABETIC-LOWER NOT ALPHABETIC-UPPER NOT NUMERIC NOT クラス名
外部 10 進数 または内部 10 進数	NUMERIC	NOT NUMERIC
DBCS	DBCS KANJI	NOT DBCS NOT KANJI
国別	NUMERIC ALPHABETIC ALPHABETIC-LOWER ALPHABETIC-UPPER NATIONAL	NOT NUMERIC NOT ALPHABETIC NOT ALPHABETIC-LOWER NOT ALPHABETIC-UPPER NOT NATIONAL
数字	NUMERIC *class-name*	NOT NUMERIC NOT クラス名
日時	NUMERIC *class-name*	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、 その値が条件名に関連付けられているいずれかの値に等しいかどうかを判別します。

フォーマット



条件名は、比較条件のための省略形として条件の中で使用されます。 条件変数を条件名の値と比較する際の規則は、 比較条件に関して指定されている規則と同じです。

条件名-*1* がある範囲の値と関係付けられている場合 (またはいくつかの範囲の値と関係付けられている場合)、 条件変数のテストは、値がその範囲内 (両端の値も含め) に収まっているかどうかを判別するテストになります。 条件名に対応した値の 1 つが、 それに関連付けられた条件変数の値に等しければ、テストの結果は真と判定されます。

条件名は、VALUE 節の条件名フォーマットに定義するように、英数字データ項目、ブール・データ項目、日時データ項目、DBCS データ項目、国別データ項目、および浮動小数点データ項目などで使用できます。

次の例は、条件変数と条件名の使用を具体的に示したものです。

```
01  AGE-GROUP         PIC  99.
    88  INFANT        VALUE 0.
    88  BABY          VALUE 1, 2.
    88  CHILD         VALUE 3 THRU 12.
    88  TEENAGER      VALUE 13 THRU 19.
```

AGE-GROUP は条件変数で、INFANT、BABY、CHILD、および TEENAGER は条件名です。 ファイルの中の個々のレコードに対して、 条件名項目の中に指定された値のうち 1 つだけが存在することができます。

上記の例に対して、以下のような IF ステートメントを加えることで、特定のレコードの年齢グループを判別することができます。

```
IF INFANT...         (値 0 のテスト)
IF BABY...           (値 1 と 2 のテスト)
IF CHILD...          (値 3 から 12 のテスト)
IF TEENAGER...       (値 13 から 19 のテスト)
```

条件名条件の評価結果に応じて、 オブジェクト・プログラムは代替の実行パスを選択します。

条件名条件とウィンドウ表示日付フィールドの比較

条件変数がウィンドウ表示日付フィールドである場合には、その条件名に関連する値は、ウィンドウ表示日付フィールドの値と同じように扱われます。 つまり、それらの値は拡張日付形式に変換されたかのように取り扱われます。

ウィンドウ表示日付フィールドのセマンティクスについて詳しくは、162 ページの『ウィンドウ表示日付フィールドのセマンティクス』を参照してください。

例えば、1945–2044 の世紀ウィンドウを指定した YEARWINDOW(1945) と次の定義が与えられた場合、

```
05  DATE-FIELD   PIC 9(6) DATE FORMAT YYXXXX.
    88  DATE-TARGET       VALUE 051220.
```

DATE-FIELD の 051220 の値は、次の条件を真にします。

```
IF DATE-TARGET...
```

DATE-TARGET に関連する値と DATE-FIELD の値は共に、 比較前に接頭部として「20」が付けられたかのように扱われるからです。

しかし、次の条件は偽になります。

```
IF DATE-FIELD = 051220...
```

ウィンドウ表示日付フィールドを使う比較では、世紀ウィンドウに関係なく リテラルは接頭部として「19」が付けられたかのように扱われるからです。 したがって、上記の条件は実際には次のようになります。

```
IF 20051220 = 19051220...
```

条件式でウィンドウ表示日付フィールドを使用する方法については、250 ページの『日付フィールドの比較』を参照してください。

比較条件

比較条件は、2 つのオペランドの比較を指定します。 2 つのオペランドを結合する比較演算子は、比較の種類を指定します。 指定された関係が 2 つのオペランド間に存在すれば比較条件は真であり、指定された関係が存在しなければ比較条件は偽になります。

比較は、次の場合に対して定義されます。

- 英字クラスの 2 つのオペランド
- 英数字クラスの 2 つのオペランド
- DBCS クラスの 2 つのオペランド
- 国別クラスの 2 つのオペランド

- 数字クラスの 2 つのオペランド
- 一方が数字の整数で、もう一方が英数字または国別クラスである 2 つのオペランド
- 一方が DBCS クラスで、もう一方が国別クラスである 2 つのオペランド
- 指標または指標データ項目を含む比較
- 2 つのデータ・ポインター・オペランド
- 2 つのプロシージャー・ポインター・オペランド
- 2 つの関数ポインター・オペランド
- 1 つの英数字グループと USAGE DISPLAY、DISPLAY-1、または NATIONAL の任意のオペランド
- 2 つの日時オペランド

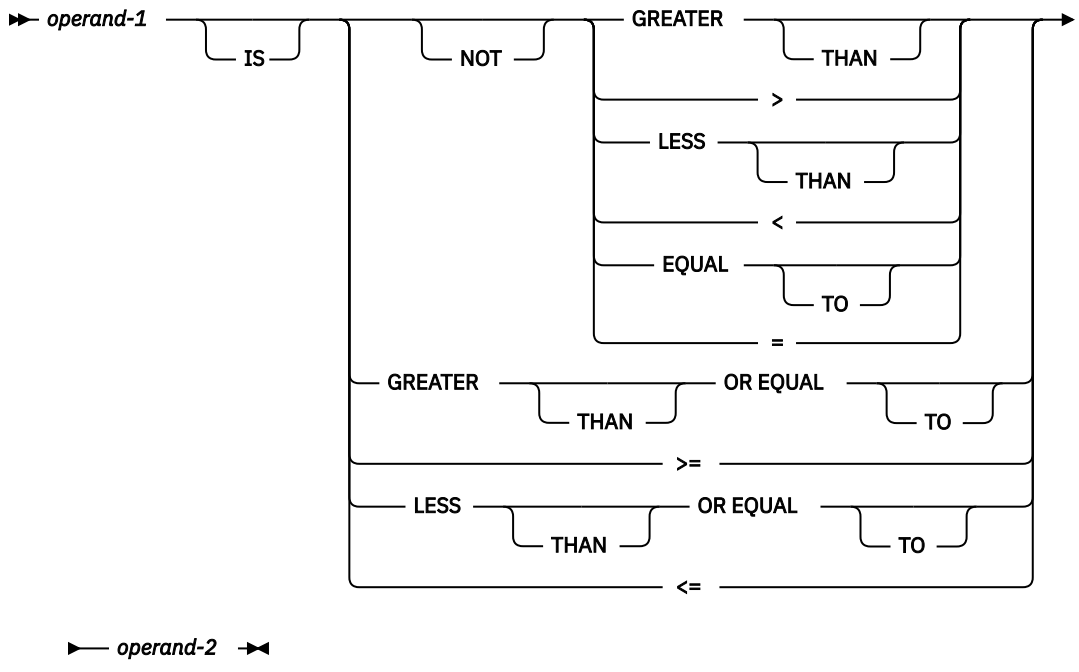
以下の比較条件形式が定義されます。

- 一般比較条件 (データ項目、リテラル、指標名、または指標データ項目のみを含む比較)。 詳細については、243 ページの『一般比較条件』を参照してください。
- データ・ポインター比較条件。 詳しくは、251 ページの『データ・ポインターの比較条件』を参照してください。
- プログラム・ポインター比較条件 (プロシージャー・ポインターまたは関数ポインターの比較)。 詳しくは、252 ページの『プロシージャー・ポインターと関数ポインターの比較条件』を参照してください。

一般比較条件

一般比較条件は 2 つのオペランドを比較します。そのオペランドはどちらも、ID、リテラル、算術式、または指標名のいずれかです。

フォーマット 1: 一般比較条件



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、または指標名であることができます。

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、または指標名であることができます。

英数字リテラルは、比較条件内で括弧で囲むことができます。

比較条件では、少なくとも 1 つの ID を参照していなければなりません。

比較演算子は、実行される比較の種類を指定します。244 ページの表 30 を参照してください。 各比較演算子は、前後にスペースを置かなければなりません。 比較演算子 >= と <= の 2 文字の間にはスペースを入れないでください。

表 *30.* **比較演算子とその意味**

比較演算子	別の表記法	意味
IS GREATER THAN	IS **>**	より大きい
IS NOT GREATER THAN	IS NOT **>**	より大きくない
IS LESS THAN	IS **<**	より小さい
IS NOT LESS THAN	IS NOT **<**	より小さくない
IS EQUAL TO	IS **=**	に等しい
IS NOT EQUAL TO	IS NOT **=**	に等しくない
IS GREATER THAN OR EQUAL TO	IS **>=**	より大きいかまたは等しい
IS LESS THAN OR EQUAL TO	IS **<=**	より小さいか等しい

一般比較条件では、クラスが英字、英数字、DBCS、国別、および数字のデータ項目、リテラル、および形象定数が、以下の比較の種類を使用して比較されます。

比較の種類	意味
英数字	2 つのオペランドの英数字文字の値の比較
ブール	2 つのオペランドのブール値の比較
日時	2 つのオペランドの日時値の比較
DBCS	2 つのオペランドの DBCS 文字の値の比較
国別	2 つのオペランドの国別文字の値の比較
数字	2 つのオペランドの代数値の比較
グループ	2 つのオペランドの英数字文字の値の比較 (オペランドの一方または両方は英数字グループ項目)

245 ページの表 31 および 246 ページの表 32 に、さまざまなタイプのオペランドとの比較が可能なペアを示します。可能な比較については、以下のキーワードを使用して、行と列が交差した部分に比較の種類を示しています。

Alph
英数字文字の比較 (247 ページの『英数字比較』 で詳述)

ブール
2 つのオペランドのブール値の比較 (248 ページの『ブール比較』 で詳述)

日時
日時値の比較 (248 ページの『日時比較』 で詳述)

DBCS
DBCS 文字の比較 (248 ページの『DBCS 比較』 で詳述)

Nat
国別文字の比較 (248 ページの『国別の比較』 で詳述)

Num
代数値の比較 (249 ページの『数字の比較』で詳述)

グループ
英数字グループを含む英数字文字の比較 (249 ページの『グループの比較』で詳述)

(Int)
整数項目のみ (タイプ Alph、Nat、Num、またはグループの比較と結合)

ブランク
比較はできない

年末尾型日付フィールドを含む比較に関する規則および制約事項については、250 ページの『日付フィールドの比較』を参照してください。

指標名および指標データ項目を含む比較に関する規則および制約事項については、250 ページの『指標名と指標データ項目の比較』を参照してください。

245 ページの表 31 の概要: この表は、次のように構成されています。

- 第 1 列の『データ項目またはリテラルのタイプ』では、各行はオペランドのタイプを示します。 場合によっては、オペランドのタイプは、比較に関して共通の特性を持つオペランドのグループを示します。 例えば、『英数字項目』の行は、以下のようにセル内にリストされているオペランドのタイプすべてを示します。
  - データ項目のカテゴリー
    - 英数字
    - 英数字編集
    - USAGE DISPLAY の数字編集
  - 英数字関数
- 以降の列見出しは、オペランドまたはオペランドのグループのタイプを示します。 例えば、列見出し『英字および英数字項目』は「英字データ項目」として識別されるオペランド、および「英数字項目」という名称を付けられたオペランドにグループ化されたオペランドのすべてのタイプを示します。
- リテラルは、第 1 列のみにオペランドのタイプとしてリストされます。 比較条件の両方のオペランドとして使用することはできないため、リテラルは列見出しには示されません。

表 *31.* **データ項目およびリテラルを含む比較**

データ項目またはリテラルのタイプ	英数字グループ項目	英字項目および英数字項目	ゾーン 10 進数項目	ネイティブ数字項目	英数字浮動小数点項目	国別文字項目	国別 10 進数項目	国別浮動小数点項目	DBCS 項目	ブール項目	日時項目
英数字グループ項目	グループ	グループ	グループ (Int)		グループ	グループ	グループ (Int)	グループ	グループ		
英字データ項目	グループ	Alph	Alph (Int)		Alph	Nat	Alph (Int)	Nat			
英数字項目: • データ項目のカテゴリー – 英数字 – 英数字編集 – USAGE DISPLAY の数字編集 • 英数字関数	グループ	Alph	Alph (Int)		Alph	Nat	Alph (Int)	Nat			
英数字リテラル	グループ	Alph	Alph (Int)		Alph	Nat	Alph (Int)	Nat			
数字リテラル	グループ (Int)	Alph (Int)	Num	Num	Num	Nat (Int)	Num	Num			
ゾーン 10 進数データ項目	グループ (Int)	Alph (Int)	Num	Num	Num	Nat (Int)	Num	Num			

表 *31.* **データ項目およびリテラルを含む比較** *(続き)*

データ項目またはリテラルのタイプ	英数字グループ項目	英字項目および英数字項目	ゾーン 10 進数項目	ネイティブ数字項目	英数字浮動小数点項目	国別文字項目	国別 10 進数項目	国別浮動小数点項目	DBCS 項目	ブール項目	日時項目
ネイティブ数字項目: • 2 進数 • 算術式 • 内部 10 進数 • 内部浮動小数点 **数字組み込み関数と整数組み込み関数**			Num	Num	Num		Num	Num			
display 浮動小数点項目	グループ	Alph	Num	Num	Num	Nat	Num	Num			
浮動小数点リテラル			Num	Num	Num		Num	Num			
国別文字項目: • データ項目のカテゴリー: – 国別 – 国別編集 – USAGE NATIONAL の数字編集 • 国別組み込み関数 • 国別グループ (基本項目として処理される)	グループ	Nat	Nat (Int)		Nat	Nat	Nat (Int)	Nat	Nat		
国別リテラル	グループ	Nat	Nat (Int)		Nat	Nat	Nat (Int)	Nat	Nat		
国別 10 進数項目	グループ (Int)	Alph (Int)	Num	Num	Num	Nat (Int)	Num	Num			
国別浮動小数点項目	グループ	Nat	Num	Num	Num	Nat	Num	Num			
DBCS データ項目	グループ					Nat			DBCS		
DBCS リテラル	グループ					Nat			DBCS		
ブール・データ項目										Alph	
ブール・リテラル										Alph	
日時データ項目	グループ		数値 (整数)	数値 (整数)	Alph		数値 (整数)				日時

表 *32.* **形象定数を含む比較**

形象定数	英数字グループ項目	英字項目および英数字項目	ゾーン 10 進数項目	ネイティブ数字項目	英数字浮動小数点項目	国別文字項目	国別 10 進数項目	国別 浮動小数点項目	DBCS 項目
ZERO	グループ	Alph	Num	Num	Num	Nat	Num	Num	
SPACE	グループ	Alph	Alph (Int)		Alph	Nat	Nat (Int)	Nat	DBCS
HIGH-VALUE、LOW-VALUE QUOTE	グループ	Alph	Alph (Int)		Alph	Nat	Nat (Int)	Nat	
シンボリック文字	グループ	Alph	Alph (Int)		Alph	Nat	Nat (Int)	Nat	
ALL 英数字リテラル	グループ	Alph	Alph (Int)		Alph	Nat	Nat (Int)	Nat	
ALL 国別リテラル	グループ	Nat	Nat (Int)		Nat	Nat	Nat (Int)	Nat	Nat
ALL DBCS リテラル	グループ					Nat			DBCS

英数字比較

英数字比較は、2 つのオペランドの 1 バイト文字の値の比較です。

一方のオペランドのクラスが英数字でも英字でもない場合、そのオペランドは以下のように処理されます。

- display 浮動小数点データ項目は、数値としてではなく、英数字カテゴリーのデータ項目であるかのように処理されます。
- ゾーン 10 進数の整数オペランドは、MOVE ステートメントの規則に従って、数値の総桁数と同じ長さの英数字カテゴリーの一時基本データ項目へ移動されるかのように処理されます。

  ZWB コンパイラー・オプションが有効なときには、整数オペランドの符号なし値は一時データ項目へ移動されます。 NOZWB コンパイラー・オプションが指定されているときは、符号付き値が一時データ項目へ移動されます。 ZWB (NOZWB) コンパイラー・オプションについて詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『*ZWB*』を参照してください。

  この後、比較は英数字カテゴリーの一時データ項目を使用して続行されます。

2 つの英数字オペランドの比較

比較に使用される照合シーケンスは、OBJECT-COMPUTER 段落の PROGRAM COLLATING SEQUENCE 節および COLLSEQ コンパイラー・オプションの設定によって決まります。

STANDARD-1、STANDARD-2、または EBCDIC の英字名で PROGRAM COLLATING SEQUENCE 節を指定した場合、COLLSEQ コンパイラー・オプションは無視されます。 指定した英字名で関連付けられた照合シーケンスが使用されます。 比較は、下の『標準比較』の説明のように行われます。

PROGRAM COLLATING SEQUENCE 節を指定しない場合、または NATIVE の英字名で指定した場合、比較方法は COLLSEQ コンパイラー・オプションによって決定されます。

- COLLSEQ(BINARY) コンパイラー・オプションが有効な場合、照合シーケンスは文字のバイナリー値によって決定されます。 比較は、下の『標準比較』の説明のように行われます。
- COLLSEQ(EBCDIC) コンパイラー・オプションが有効な場合、EBCDIC 照合シーケンスが使用されます。 比較は、下の『標準比較』の説明のように行われます。
- COLLSEQ(LOCALE) コンパイラー・オプションが有効な場合、照合シーケンスはランタイム・ロケールによって決定されます。 比較のため、末尾のスペースがオペランドから取り除かれます。オペランドがすべてのスペースから構成されていて、取り除かれると単一のスペースになる場合は除きます。 ロケール・ベースの比較は、必ずしも文字単位の比較ではありません。 非ロケール・ベースの比較で短いオペランドがスペースで拡張された場合、 結果が一般的に期待された結果にならない場合があります。 ロケールについては、563 ページの『付録 H ロケールの考慮事項』を参照してください。

PROGRAM COLLATING SEQUENCE 節を、リテラルによって定義された照合シーケンスを参照する英字名で指定した場合、照合シーケンスは、指定したリテラルの順序および COLLSEQ コンパイラー・オプションで示された順序によって決定されます。95 ページの『ALPHABET 節』を参照してください。 比較は、下の『標準比較』の説明のように行われます。

標準比較

標準比較とは、ロケールに基づかない比較のことをいいます。 標準比較方法は、比較されるオペランドが等しい長さであるか、異なる長さであるかに応じて決定されます。

2 つのオペランドの長さが異なる場合は、短い方のオペランドの右側に デフォルトの適切な数のスペース文字を埋め、両方のオペランドの長さが等しくなるようにして比較が行われます。 こうして比較は、長さが同じオペランドを比較するための規則に従って実行されます。

2 つのオペランドの長さが同じ場合は、オペランド内の同じ位置にある文字同士が比較されます。 比較は左端から開始され、途中で異なる文字が検出されるか、右端に到達するまで繰り返されます。 対応する文字がすべて同じであった場合は、2 つのオペランドが等しいと判別されます。

オペランド内で最初に検出された異なる文字は、2 つのオペランドの関係を判別するために比較されます。 より大きな照合値を持つ文字が入ったオペランドが、より大きなオペランドになります。

ブール比較

英数字比較は、2 つのオペランドの 1 バイト文字の値の比較です。

ブール・オペランドは、EQUAL TO または 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。

日時比較

日時比較は、2 つのオペランドの日付値や時刻値の比較です。

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。

DBCS 比較

DBCS 比較は、2 つの DBCS オペランドの比較です。

以下の規則が DBCS 比較に適用されます。

- DBCS オペランドの比較は、COLLSEQ コンパイラー・オプションによって指定された照合シーケンスに基づいています。
  - COLLSEQ(LOCALE) コンパイラー・オプションが有効な場合、照合シーケンスはランタイム・ロケールによって決定されます。 ロケールの詳細については、563 ページの『付録 H ロケールの考慮事項』を参照してください。
  - 有効でない場合は、照合シーケンスは DBCS 文字のバイナリー値によって決定されます。

国別の比較

国別比較は、国別クラスの 2 つのオペランドの国別文字値の比較です。

比較条件が国別クラスではないオペランドを指定しているときは、比較の前にそのオペランドが国別カテゴリーのデータ項目へ変換されます。 以下のリストで、オペランドの国別カテゴリーへの変換について説明します。

DBCS
DBCS オペランドは、DBCS オペランドと同じ長さの国別カテゴリーの一時データ項目へ移動されるかのように処理されます。 DBCS 文字は、対応する国別文字に変換されます。 変換で使用されるソース・コード・ページは、 は、DBCS オペランドが EBCDIC かまたは ASCII データ項目 かどうかに依存します。 DBCS オペランドが EBCDIC データ項目の場合 (コンパイラー・オプション CHAR(EBCDIC) が 有効で、NATIVE 句がそのオペランドに指定されていない場合)、そのオペランドに有効な EBCDIC コード・ページ が使用されます。そうでない場合は、そのオペランドに有効な、ロケールによって指定されたコード・ページ が使用されます。 詳細については、563 ページの『付録 H ロケールの考慮事項』を参照してください。

USAGE DISPLAY の英字、英数字、英数字編集、および数字編集
オペランドは、そのオペランド内の文字位置数を表すために必要な長さの国別カテゴリーの一時データ項目へ移動されるかのように処理されます。 英数字は、対応する国別文字に変換されます。 変換で使用されるソース・コード・ページは、英数字オペランドが EBCDIC かまたは ASCII データ項目 かどう

かに依存します。 英数字オペランドが EBCDIC データ項目の場合 (コンパイラー・オプション CHAR(EBCDIC) が 有効で、NATIVE 句がそのオペランドに指定されていない場合)、そのオペランドに有効な EBCDIC コード・ページ が使用されます。そうでない場合は、そのオペランドに有効な、ロケールによって指定されたコード・ページ が使用されます。詳細については、563 ページの『付録 H ロケールの考慮事項』を参照してください。

整数
整数オペランドは、整数の桁数と同じ長さの英数字カテゴリーの一時データ項目へ移動されるかのように処理されます。 符号なしの値が使用されます。 次に、この一時データ項目は英数字オペランドとして変換されます。

外部浮動小数点
display 浮動小数点項目は、数値としてではなく、英数字カテゴリーのデータ項目であるかのように処理されてから、英数字オペランドとして変換されます。

国別浮動小数点データ項目は、数値としてではなく、国別カテゴリーのデータ項目であるかのように処理されます。

変換に関する暗黙の移動は、MOVE ステートメントの規則に従って実行されます。

生成された国別カテゴリーのデータ項目は、2 つの国別オペランドの比較で使用されます。

2 つの国別オペランドの比較

比較に使用されるメソッドは、NCOLLSEQ コンパイラー・オプションの設定によって 判別されます。

NCOLLSEQ(BINARY) コンパイラー・オプションが有効な場合は、照合シーケンスは、国別文字 のバイナリー値によって判別されます。 比較は、次のようにして行われます。

- 2 つのオペランドの長さが異なる場合は、短い方のオペランドの右側に デフォルトの国別スペース文字 (NX'2000') を埋め、両方のオペランドの長さが等しくなるようにして比較が行われます。 こうして比較は、長さが同じオペランドを比較するための規則に従って実行されます。
- 2 つのオペランドの長さが同じ場合は、オペランド内の同じ位置にある国別文字同士が比較されます。比較は左端から開始され、途中で異なる国別文字が検出されるか、右端に到達するまで繰り返されます。対応する国別文字がすべて同じであった場合は、2 つのオペランドが等しいと判別されます。
- オペランド内で最初に検出された異なる国別文字は、2 つのオペランドの関係を判別するために比較されます。 より大きな照合値を持つ国別文字が入ったオペランドが、より大きなオペランドになります。

NCOLLSEQ(LOCALE) コンパイラー・オプションが有効な場合は、照合シーケンスは、ランタイム・ロケール によって判別されます。 比較を行うために、全桁スペースで構成されているオペランドがシングル・スペース に切り捨てられる場合を除き、オペランドの末尾のスペースは切り捨てられます。 ロケール・ベースの比較は、必ずしも文字単位の比較ではありません。 非ロケール・ベースの比較で短いオペランドがスペースで拡張された場合、 結果が一般的に期待された結果にならない場合があります。 ロケールについては、563 ページの『付録 H ロケールの考慮事項』を参照してください。

PROGRAM COLLATING SEQUENCE 節は、国別オペランドの比較には影響を及ぼしません。

数字の比較

数値比較は、数字クラスの 2 つのオペランドの代数値の比較です。

数字オペランドの代数値の場合に比較されます。

- オペランドの長さ (桁数) は意味を持ちません。
- オペランドの USAGE は意味を持ちません。
- 符号なしの数字オペランドは正とみなされます。
- すべてゼロの値の比較は等しく、符号の存在の有無は結果に影響しません。

グループの比較

グループ比較は、2 つのオペランドの英数字値の比較です。

比較演算では、各オペランドは、オペランドと同サイズ (バイト単位) の英数字カテゴリーの基本データ項目の場合と同様に処理されます。 比較は、247 ページの『英数字比較』で説明するように、英数字カテゴリーの 2 つの基本オペランドに関して進められます。

使用上の注意: グループ比較の場合、データの変換は行われません。 比較は、データ表現には関係なくデータの各バイトを処理します。 オペランドとグループ項目の内容のデータ表現が同じ場合には、基本項目またはリテラルのオペランドを英数字グループ項目と比較した結果は予測可能です。

指標名と指標データ項目の比較

指標名、指標データ項目、またはその両方の比較は、規則に準拠します。

比較の規則は以下のとおりです。

- 2 つの指標名の比較は、実際には対応するオカレンス番号の比較です。
- 指標名を (指標データ項目以外の) データ項目と比較する場合、または指標名をリテラルと比較する場合、指標名の値に対応したオカレンス番号が、 そのデータ項目やリテラルと比較されます。
- 指標名を算術式と比較する場合、 指標名の値に対応したオカレンス番号がその算術式と比較されます。

  算術式が使えるところでは整数関数が使用できるため、指標名を整数関数や数字関数と比較することができます。
- 指標データ項目を指標名または別の指標データ項目と比較する場合、 変換を行うことなく実際の値が比較されます。 指標データ項目が関係するその他の比較の結果は、定義されていません。

次の表に、指標名と指標データ項目の有効な比較を示します。

表 *33.* **指標名と指標データ項目に関する比較**

比較されるオペランド	指標名	指標データ項目	データ名 (数値整数のみ)	リテラル (数値整数のみ)	算術式
指標名	オカレンス番号を比較	変換せずに比較	オカレンス番号を参照データ項目の内容と比較	オカレンス番号をリテラルと比較	オカレンス番号を算術式と比較
指標データ項目	変換せずに比較	変換せずに比較	無効	無効	無効

日付フィールドの比較

日付フィールドは、英数字カテゴリー、ゾーン 10 進数、または内部 10 進数にすることができます。妥当性および比較タイプ (数字または英数字) に関する既存の規則が適用されます。

例えば、英数字の日付フィールドは、内部 10 進数日付フィールドと比較することはできません。 これらの規則に加えて、2 つの日付フィールドを それらに互換性がある場合だけ比較できます。すなわち、 ウィンドウ表示西暦年でも拡張西暦年でもよい年部分を除いて、同じ日付フォーマットでなければなりません。

年末尾型日付フィールドの場合、サポートされる唯一の比較は、 日付フォーマットが同じ 2 つの年末尾型日付フィールドの間の IS EQUAL TO および IS NOT EQUAL TO、または年末尾型日付フィールドと非日付データの間の比較だけです。

251 ページの表 34 は、非年末尾型日付フィールドについてサポートされる比較を示しています。 この表では、比較の実行方法を記述するのに以下の用語を使用しています。

ウィンドウ表示なし
オペランドが両方とも非日付データであるかのように、 ウィンドウ表示なしで比較が実行されます。

ウィンドウ表示
以下のように、比較が実行されます。

1. 比較の中のウィンドウ表示日付フィールドは、YEARWINDOW コンパイラー・オプションによって指定された世紀ウィンドウに従って 拡張されたかのように扱われます (162 ページの『ウィンドウ表示日付フィールドのセマンティクス』を参照)。
2. 反復する英数字の形象定数は、比較されるウィンドウ表示日付フィールド (英数字の非日付データの被比較数) のサイズまで拡張されたかのように扱われます。 反復の英数字の形象定数には、ZERO (英数字のコンテキストの場合)、SPACE、 LOW-VALUE、HIGH-VALUE、QUOTE、および ALL リテラル があります。
3. 非日付データオペランドは、日付フィールドと同じ日付フォーマットで、 1900 を基本年とするもののようにして扱われます。

その後、通常の COBOL 規則に従って比較が実行されます。 世紀値を接頭部として付けることによって英数字の比較が数字の比較に変更されることはありません。

表 *34.* **日付フィールドとの比較**

	非日付データ 第 2 オペランド	**ウィンドウ表示 日付フィールド 第 2 オペランド**	**拡張 日付フィールド 第 2 オペランド**
非日付データ 第 1 オペランド	ウィンドウ表示なし	ウィンドウ表示 [1]	ウィンドウ表示なし
ウィンドウ表示 日付フィールド 第 1 オペランド	ウィンドウ表示 [1]	ウィンドウ表示	ウィンドウ表示
拡張日付フィールド 第 1 オペランド	ウィンドウ表示なし	ウィンドウ表示	ウィンドウ表示なし

1. **ウィンドウ表示日付フィールドとの比較において非日付データは、1900 を基準とするウィンドウ表示西暦年を含むものと想定されます。 詳細については、「ウィンドウ表示」比較の定義の項目 3 を参照してください。**

比較条件に算術式を含めることができます。 算術式での日付フィールドの処理については、236 ページの『日付フィールドを使用する算術計算』を参照してください。

データ・ポインターの比較条件

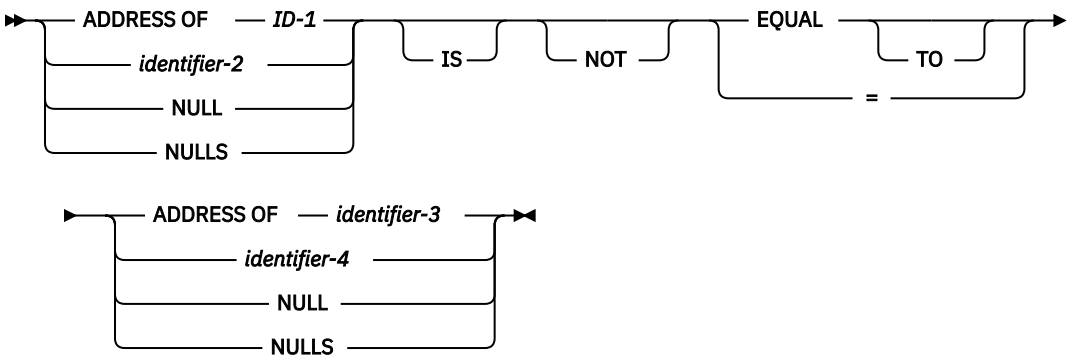
ポインター・データ項目を指定した場合は、 関係演算子として EQUAL および NOT EQUAL のみが使用可能です。

ポインター・データ項目は、USAGE POINTER として明示的に定義されている項目、 または USAGE POINTER として暗黙のうちに定義されている ADDRESS OF 特殊レジスターです。

比較に使われる 2 つのアドレスが結果的に同じ保管場所にあれば、 それらのオペランドは等しいことになります。

この比較条件は、IF、PERFORM、EVALUATE、および SEARCH のフォーマット 1 の各ステートメントの中で使用できます。 フォーマット 2 の SEARCH ステートメント (SEARCH ALL) の中では使用できません。 ポインター・データ項目に適用可能な意味のある順序付けは存在しないからです。

フォーマット 2: データ・ポインターの比較条件



ID-1*、*ID-3
レベル 66 とレベル 88 を除き、LINKAGE SECTION の中で定義されたどのレベルの項目でも指定することができます。

ID-2*、*ID-4
USAGE POINTER として記述されている必要があります。

NULL、 NULLS
もう一方のオペランドが、USAGE POINTER として定義されている場合に限り使用できます。 つまり、NULL=NULL になることはありえません。

以下の表は、USAGE POINTER、NULL、および ADDRESS OF に対して 可能な比較を要約したものです。

表 *35.* ***USAGE POINTER*、*NULL*、および *ADDRESS OF* に対して可能な比較**

可能な比較	**USAGE POINTER 第 2 オペランド**	**ADDRESS OF 第 2 オペランド**	**NULL または NULLS 第 2 オペランド**
USAGE POINTER 第 1 オペランド	はい	はい	はい
ADDRESS OF 第 1 オペランド	はい	はい	はい
NULL/NULLS 第 1 オペランド	はい	はい	いいえ

はい
EQUAL、NOT EQUAL に関してのみ可能な比較

いいえ
比較はできない

プロシージャー・ポインターと関数ポインターの比較条件

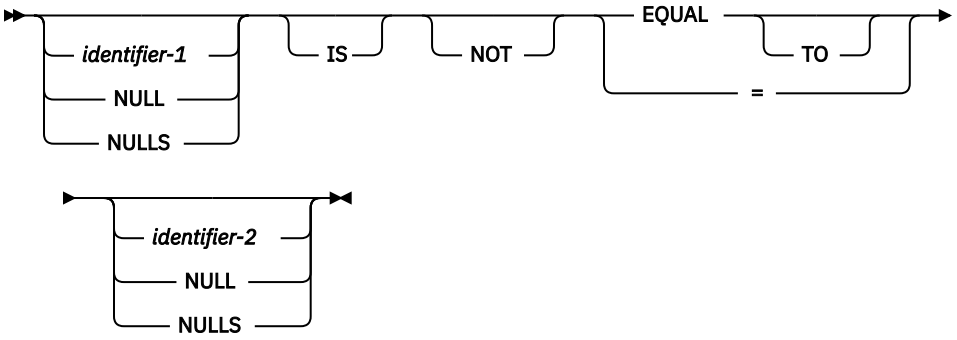
比較条件にプロシージャー・ポインターまたは関数ポインター・データ項目を指定した場合は、 関係演算子として EQUAL および NOT EQUAL のみが使用可能です。

プロシージャー・ポインター・データ項目は、USAGE PROCEDURE-POINTER として明示的に定義されます。 関数ポインター・データ項目は、USAGE FUNCTION-POINTER として明示的に定義されます。

比較に使われる 2 つのアドレスが結果的に同じ保管場所にあれば、 それらのオペランドは等しいことになります。

この比較条件は、IF、PERFORM、EVALUATE、および SEARCH のフォーマット 1 の各ステートメントの中で使用できます。 フォーマット 2 の SEARCH ステートメント (SEARCH ALL) の中では使用できません。 プロシージャー・ポインター・データ項目に適用可能な意味のある順序付けは存在しないからです。

フォーマット 3: プロシージャー・ポインターおよび関数ポインターの比較条件



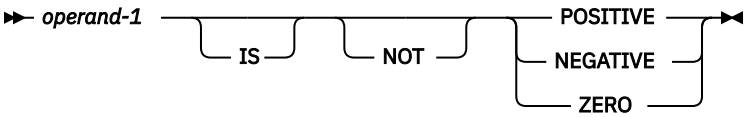
ID-1* 、 *ID-2
USAGE PROCEDURE-POINTER または USAGE FUNCTION-POINTER として記述します。 *ID-1* および *ID-2* を同じ内容にする必要はありません。

NULL、NULLS
もう一方のオペランドが、USAGE FUNCTION-POINTER または USAGE PROCEDURE-POINTER として定義されている場合に限り使用できます。 つまり、NULL=NULL になることはありえません。

符号条件

符号条件は、ある数字オペランドの代数値が、0 に比べてそれより大きいか、 小さいか、または等しいかを判別します。

フォーマット: 符号条件



オペランド*-1*
数字 ID、または変数への参照を少なくとも 1 つ含む算術式として定義されている必要があります。 オペランド*-1* は、浮動小数点 ID として定義することができます。

オペランドは次のように判別されます。

- その値が 0 より大きければ POSITIVE
- その値が 0 より小さければ NEGATIVE
- その値が 0 に等しければ ZERO

符号なしのオペランドは、POSITIVE かまたは ZERO のいずれかです。

NOT
符号条件の真の値に関して、1 回の代数テストが実行されます。 例えば、テストされたオペランドの値が正または負であるとき、 NOT ZERO は真とみなされます。

符号条件での日付フィールド

符号条件のオペランドとして日付フィールドも使用できますが、 符号条件テストでは非日付データとして処理されます。 したがって、オペランドがウィンドウ表示日付フィールドの ID である場合、 日付ウィンドウ表示は行われないため、符号条件を使用してウィンドウ表示日付フィールドの値がすべてゼロかどうかをテストできます。

ただし、オペランドが算術式である場合、式中のウィンドウ表示日付フィールドは符号条件テストの結果を使用する前に、 算術結果の計算中に拡張されることになります。

例えば、以下の状況を考えます:

- WIN-DATE がウィンドウ表示日付フィールドとして定義されていて、値が 0 である
- コンパイラー・オプション DATEPROC が有効である
- コンパイラー・オプション YEARWINDOW (開始年) が有効で、開始年 が 1900 以外である

以下の符号条件は真であると評価されます。

```
WIN-DATE IS ZERO
```

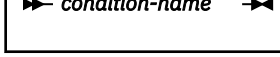
逆に、以下の符号条件は偽であると評価されます。

```
WIN-DATE + 0 IS ZERO
```

スイッチ状況条件

スイッチ状況条件は、UPSI スイッチがオン状況にあるかオフ状況にあるかを判別します。

フォーマット



条件名
UPSI スイッチのオン値またはオフ値に関連付けられている SPECIAL-NAMES 段落に 定義します。 (91 ページの『SPECIAL-NAMES 段落』を参照してください。).)

スイッチ状況条件は、条件名 に関連付けられている値をテストします (その値は英数字であるとします)。テストの実行結果は、UPSI スイッチが条件名 に対応した値 (0 か 1) に設定されていれば、真となります。詳しくは、*COBOL for Linux on x86* プログラミング・ガイド内の *UPSI* を参照してください。

複合条件

複合条件は、単純条件、複合条件、および論理演算子を伴う複合条件を組み合わせるか、またはそれらの各種条件を論理否定によって否定することにより形成されます。

各論理演算子は、前後にスペースを置かなければなりません。 次の表では、論理演算子とその意味を示します。

表 *36.* **論理演算子とその意味**

論理演算子	名前	意味
AND	論理積	両方の条件が真の場合に、真の値は真となります。
OR	包含論理和	両方もしくは一方の条件が真の場合に、真の値は真となります。
NOT	論理否定	真の値の逆 (条件が偽の場合、真の値は真となります)。

括弧によって変更しない限り、次のリストのような優先順位 (優先順位の高いものから低いものへの順で示してある) が適用されます。

1. 算術演算
2. 単純条件
3. NOT
4. AND
5. OR

複合条件の真の値 (括弧使用の有無に関係なく) は、以下に示す値のオプション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としての真の値です。

- 個々の単純条件の真の値
- 論理結合または論理否定された複数の条件の中間真の値

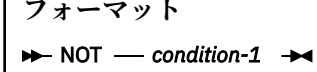
複合条件は、次のオプションのどちらかにすることができます。

- 単純否定条件
- 複合条件 (その否定形も可)

単純否定条件

単純条件は、論理演算子 NOT を使用することによって否定されます。

フォーマット



単純否定条件は、単純条件の真の値の反対の真の値を作ります。 つまり、単純条件の真の値が真ならば、同じ単純否定条件の真の値は偽であり、 この逆に単純条件の真の値が偽ならば、同じ単純否定条件の真の値は真です。

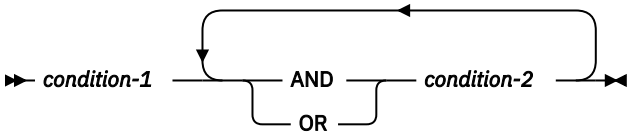
単純否定条件を括弧で囲んでも、その真の値は変わりません。 したがって、次の 2 つのステートメントは同じことです。

```
NOT A IS EQUAL TO B.
NOT (A IS EQUAL TO B).
```

複合条件

2 つ以上の条件を論理的に結合することによって、複合条件を形成することができます。

フォーマット



次のどの条件でも結合できます。

- 単純条件
- 単純否定条件
- 複合条件
- 複合否定条件 (論理演算子 NOT の後に括弧に入れた複合条件が続くもの)
- 上記のいくつかの条件の組み合わせを、次の表に示されている規則に応じて指定したもの

表 *37.* **複合条件—許されるエレメントのシーケンス**				
複合条件エレメント	**左端**	**左端にないときは、次のものが直前にある**	**右端**	**右端にないときは、次のものが直後にある**
単純条件	はい	OR NOT AND (	はい	OR AND)
OR AND	いいえ	単純条件)	いいえ	単純条件 NOT (
NOT	はい	OR AND (	いいえ	単純条件 (
(	はい	OR NOT AND (	いいえ	単純条件 NOT (
)	いいえ	単純条件)	はい	OR AND)

1 つの複合条件の中で AND か OR のどちらか一方だけしか使用されていない場合は、 括弧を付ける必要はありません。 ただし、演算子とオペランドの正しい論理関係を保持するために、 暗黙の優先順位規則に変更を加える場合には括弧を付けることもできます。

左括弧と右括弧には 1 対 1 の対応関係がなければなりません。 また左括弧は対応した右括弧の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 および条件 C2 の間の関係をわかりやすく示したものです。

表 *38.* **論理演算子と複合条件の評価結果**							
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

条件の評価順序

括弧は、明示による場合も暗黙による場合も、複合条件内の包含レベルを定義します。 同じ包含レベルで論理演算子 AND または OR だけで結合された 2 つ以上の条件は、複合条件内の 1 つの階層レベルを確立し

ます。 したがって、複合条件全体が複数のレベルからなる階層のネスト構造であり、 複合条件全体は、最も包括的な階層レベルを表します。

ここでは、複合条件全体の中での条件の評価は、条件の左側から始まります。 ある階層レベル内のコンポーネントである接続条件は左から右への順に評価され、 その階層レベルの評価は、その階層レベル内のコンポーネントである すべての接続条件が評価されたかどうかに関係なく、 その階層レベルに関する真の値が決定されると終了します。

算術式と算術関数に関する値は、それらを含む複合条件が評価される場合には、 その評価が行われたときに設定されます。 同様に否定条件は、それらの条件が表現している複合条件を評価する必要がある場合には、それが実行されたときに評価されます。 次に例を示します。

```
NOT A IS GREATER THAN B OR A + B IS EQUAL TO C AND D IS POSITIVE
```

上記の例は、括弧が次のように付いているかのように評価されます。

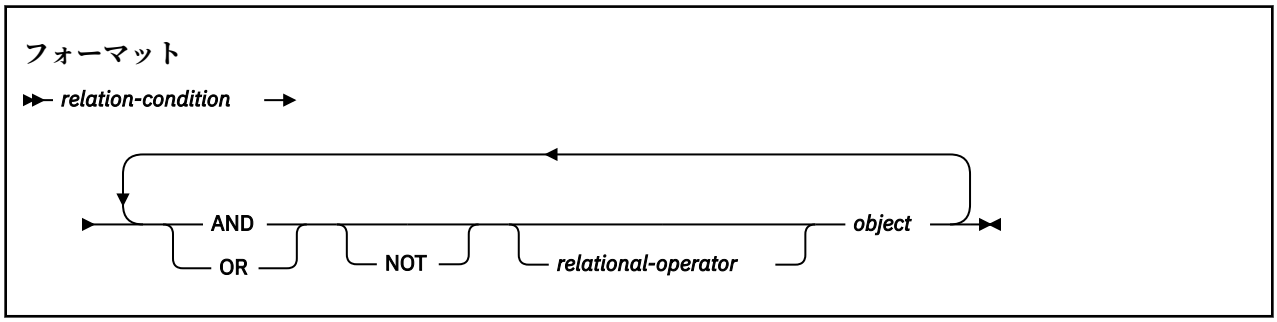
```
(NOT (A IS GREATER THAN B)) OR
(((A + B) IS EQUAL TO C) AND (D IS POSITIVE))
```

評価の順序

1. (NOT (A IS GREATER THAN B)) が評価され、ある中間的な真の値として、*t1* が得られます。 *t1* 真であれば複合条件が真となり、それ以上の評価は行われません。 *t1* が偽であれば、評価はさらに次のように継続されます。
2. (A + B) が評価され、ある中間的な結果として *x* が得られます。
3. (x IS EQUAL TO C) が評価され、ある中間的な真の値として *t2* が得られます。 *t2* が偽であれば、複合条件自体が偽となり、それ以上の評価は行われません。 *t2* が真ならば、評価はさらに次のように継続されます。
4. (D IS POSITIVE) が評価され、ある中間的な真の値として *t3* が得られます。 *t3* が偽ならば、複合条件は偽となります。 *t3* が真ならば、複合条件が真となります。

簡略複合比較条件

いくつかの比較条件を連続して記述する場合、2 番目以降の比較条件は、サブジェクトを省略するか、サブジェクトおよび関係演算子を省略することで、省略形で記述できます。



連続する一連の比較条件の中のどの部分でも、省略の 2 つの形式はどちらも使えます。 省略形の条件は、次のようにして評価されます。

1. 最後に記述されているサブジェクトが、省略されたサブジェクトとなります。
2. 最後に記述されている比較演算子が、省略された比較演算子となります。

結果としての複合条件は、複合条件の中のエレメント・シーケンスに関する規則に従っていなければなりません。255 ページの『複合条件』を参照してください。

GREATER THAN、>、LESS THAN、<、EQUAL TO、または = の直後に NOT がある場合、NOT は比較演算子の一部であるとみなされます。 他の位置にある NOT は、論理演算子であるとみなされます (したがって否定比較条件となります)。

括弧の使用

意図した演算順序を指定するために、結合した比較条件の中に括弧を使用できます。 括弧の使用は、条件式を読みやすくするためにも役立ちます。

簡略複合比較条件における括弧の使用には、次の規則が適用されます。

1. 括弧は、論理演算子の AND と OR の評価順序を変えるために使用することができます。
2. 語 NOT の直後に GREATER THAN、>、LESS THAN、<、EQUAL TO、または = が置かれているときは、NOT は比較演算子の一部であるとみなされます。
3. 他の位置にある NOT は、論理演算子であるとみなされるため、否定比較条件となります。 論理演算子として NOT を使用すると、NOT の直後の比較条件だけが否定されます。 否定は、同じサブジェクトと比較演算子を持つ簡略複合条件全体に伝搬するわけではありません。
4. 比較演算子の直後の括弧で囲まれた式の中に、論理演算子 NOT を含めることができます。
5. 左括弧が比較演算子の直後にある場合は、その比較演算子が括弧内のすべてのオブジェクトに分配されます。 比較演算子が「分配された」場合、その分配を終了させる右括弧以降も、 サブジェクトと比較演算子はそのまま残ります。 比較演算子が式の中で分配される場合、次の 3 つの制約事項が 適用されます。
   a. 分配の範囲内に単純条件は入れない。
   b. 分配の範囲内に別の比較演算子は入れない。
   c. 分配の範囲を定義する左括弧の直後に、論理演算子 NOT は入れない。
6. 評価は、最も内側の条件から最も外側の条件へと進められます。
7. 左括弧と右括弧には 1 対 1 の対応関係がなければなりません。 また左括弧は対応した右括弧の左側になければなりません。 括弧が片側しかない場合、コンパイラーがもう 1 つの括弧を挿入して、 E レベルのメッセージを出します。 ただし、コンパイラーが挿入した括弧によって式の切り捨てが生じた場合には、 S レベルの診断メッセージが出されます。
8. 最後に記述されたサブジェクトが、省略したサブジェクトの位置に挿入されます。
9. 最後に記述された比較演算子が、省略した比較演算子の位置に挿入されます。
10. 省略されていたサブジェクトまたは比較演算子の挿入処理は、次の場合に終了します。
    a. 他の単純条件が出てきたとき
    b. 条件名が出てきたとき
    c. サブジェクトの左側にある左括弧に対応する右括弧が出てきたとき
11. 連続する一連の比較条件では、 括弧を含むものと含まないものの両方の省略形の比較条件を使用できます。
12. 論理演算子 NOT が連続していると、互いに打ち消しあって、S レベルのメッセージが出されます。 ただし、2 番目の NOT が比較演算子の一部であるときは、 簡略複合比較条件に 2 つの NOT 演算子を連続して記入することができます。 例えば、下記の最初の条件を 2 番目の条件のように省略することができます。

```
A = B and not A not = C
A = B and not not = C
```

次の表は、簡略複合比較条件を記述するときの規則を要約しています。

表 *39.* **簡略複合条件: 許されるエレメントのシーケンス**

複合条件エレメント	左端	左端にないときは、次のものが直前にある	右端	右端にないときは、次のものが直後にある
サブジェクト	はい	NOT (	いいえ	比較演算子
オブジェクト	いいえ	比較演算子 AND OR NOT (	はい	AND OR)
比較演算子	いいえ	サブジェクト AND OR NOT	いいえ	オブジェクト (
AND OR	いいえ	オブジェクト)	いいえ	オブジェクト 比較演算子 NOT (
NOT	はい	AND OR (	いいえ	サブジェクト オブジェクト 比較演算子 (
(	はい	比較演算子 AND OR NOT (	いいえ	サブジェクト オブジェクト NOT (
)	いいえ	オブジェクト)	はい	AND OR)

次の表は、簡略複合比較条件の例 (括弧を使用した例と使用しない例を含む) と、それと同じ内容を簡略しないで表現した例を示しています。

表 *40.* **簡略複合条件とそれに対応する簡略化していない表現**

簡略複合比較条件	完全な形式での表現
A = B AND NOT < C OR D	((A = B) AND (A NOT < C)) OR (A NOT < D)
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))))

ステートメントのカテゴリー

COBOL ステートメントには、命令ステートメント、条件ステートメント、範囲区切りステートメント、およびコンパイラー指示ステートメントの 4 つのカテゴリーがあります。詳しくは、以下のトピックを参照してください。

命令ステートメント

命令ステートメント は、プログラムが行う無条件アクションを指定します。 あるいは、命令ステートメントは明示範囲終了符号で終了する条件ステートメントです。

1 つの命令ステートメントを指定できるときは、いつでも一連の命令ステートメントを指定できます。 明示的範囲終了符号によって終了する条件ステートメントも、命令ステートメントに分類されます。

明示的範囲終了符号について詳しくは、263 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。.

下の表は、COBOL の命令ステートメントを示しています。

算術演算

- ADD[1]
- COMPUTE[1]
- DIVIDE[1]
- MULTIPLY[1]
- SUBTRACT[1]

1. ON SIZE ERROR 句または NOT ON SIZE ERROR 句が指定されていないもの。

データの移動

- ACCEPT (DATE、DAY、DAY-OF-WEEK、TIME)
- INITIALIZE
- INSPECT
- MOVE
- SET
- STRING[2]
- UNSTRING[2]
- XML GENERATE[3]
- XML PARSE[3]

2. ON OVERFLOW 句または NOT ON OVERFLOW 句が指定されていないもの

3. ON EXCEPTION または NOT ON EXCEPTION 句が指定されていないもの

終了

- STOP RUN
- EXIT PROGRAM
- GOBACK

入出力

- ACCEPT *ID*
- CLOSE
- DELETE[4]

- DISPLAY
- OPEN
- READ[5]
- REWRITE[4]
- START[4]
- STOP リテラル
- UNLOCK
- WRITE[6]

4. INVALID KEY 句または NOT INVALID KEY 句が指定されていないもの。

5. AT END 句または NOT AT END 句が指定されていないもの、および INVALID KEY 句または NOT INVALID KEY 句が指定されていないもの。

6. INVALID KEY 句または NOT INVALID KEY 句が指定されていないもの、および END-OF-PAGE 句または NOT END-OF-PAGE 句が指定されていないもの。

順序付け

- フォーマット 1 SORT
- MERGE
- RELEASE
- RETURN[7]

7. AT END 句または NOT AT END 句が指定されていないもの。

プロシージャーのブランチ

- ALTER
- CONTINUE
- フォーマット 1 EXIT
- GO TO
- PERFORM

プログラムのリンケージ

- CALL[8]
- CANCEL

8. ON OVERFLOW 句が指定されていないもの、および ON EXCEPTION 句または NOT ON EXCEPTION 句が指定されていないもの。

テーブル操作

- フォーマット 2 SORT (テーブル SORT)
- SET

条件ステートメント

条件ステートメント は、ある条件の真の値を判別すること、またオブジェクト・プログラムの次の処置がこの真の値によって決まることを指定します。

条件式について詳しくは、238 ページの『条件式』を参照してください。

条件 (例えば、ON SIZE ERROR または ON OVERFLOW) が含まれるとき、およびステートメントが明示的範囲終了符号によって範囲を限定されないときに、条件ステートメントになる COBOL ステートメントを以下にリストします。

算術演算

- ADD ... ON SIZE ERROR
- ADD ... NOT ON SIZE ERROR
- 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

データの移動

- STRING ... ON OVERFLOW
- STRING...NOT ON OVERFLOW
- UNSTRING ... ON OVERFLOW
- UNSTRING...NOT ON OVERFLOW
- XML GENERATE ... ON EXCEPTION
- XML GENERATE ... NOT ON EXCEPTION
- XML PARSE ... ON EXCEPTION
- XML PARSE ... NOT ON EXCEPTION

判断

- IF
- EVALUATE

入出力

- DELETE ... INVALID KEY
- DELETE ... NOT INVALID KEY
- READ ... AT END
- READ...NOT AT END
- READ ... INVALID KEY
- READ...NOT INVALID KEY
- REWRITE ... INVALID KEY
- REWRITE ... NOT INVALID KEY
- START ... INVALID KEY
- START...NOT INVALID KEY
- WRITE ... AT END-OF-PAGE
- WRITE...NOT AT END-OF-PAGE
- WRITE ... INVALID KEY

- WRITE...NOT INVALID KEY

順序付け

- RETURN ... AT END
- RETURN...NOT AT END

プログラムのリンケージ

- CALL ... ON OVERFLOW
- CALL ... ON EXCEPTION
- CALL...NOT ON EXCEPTION

テーブル操作

- SEARCH

範囲区切りステートメント

通常は、DELIMITED SCOPE ステートメントは、明示的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します。

その後に、その命令ステートメントをネストすることができます。 明示的範囲終了符号は、命令ステートメントの範囲を限定するために使用することもできます。 明示的範囲終了符号は、条件句を持つことができるすべての COBOL ステートメントで利用できます。

特に明記されていない限り、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

明示的範囲終了符号 は、特定の PROCEDURE DIVISION ステートメントの終わりに マークを付けます。

明示的範囲終了符号によって区切られている条件ステートメントは、 命令ステートメントとみなされ、 命令ステートメントに関する規則に従わなければなりません。

明示的範囲終了符号は、次のとおりです。

- END-ADD
- END-CALL
- END-COMPUTE
- END-DELETE
- END-DIVIDE
- END-EVALUATE
- END-IF
- END-MULTIPLY
- END-PERFORM
- END-READ
- END-RETURN
- END-REWRITE
- END-SEARCH
- END-START
- END-STRING
- END-SUBTRACT

- END-UNSTRING
- END-WRITE
- END-XML

暗黙範囲終了符号

暗黙の範囲終了符号 とは、文の終わりにある分離文字ピリオドのことで、 これはその前に置かれていてまだ終了していないすべてのステートメントのスコープを終了させます。

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に含めることはできません。

IF ステートメント内のネストする条件ステートメントを除き、 ネストされた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に関する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ることはできません。

関連参照
RULES (*COBOL for Linux on x86* プログラミング・ガイド)

コンパイラー指示ステートメント

コンパイラー指示ステートメントとは、コンパイル時にコンパイラーに特定の処置を行わせるステートメントです。

指定したアクションを行うようにコンパイラーに指示するステートメントについて詳しくは、473 ページの『第 21 章 コンパイラー指示ステートメント』を参照してください。

ステートメント操作

このトピックでは、COBOL ステートメントで実行される操作のタイプについて説明します。

COBOL ステートメントは、次のような種類の操作を行います。

- 算術演算
- データ操作
- 入出力
- プロシージャーのブランチ

算術ステートメントとデータの処理ステートメントに共通する次のような句があります。

- CORRESPONDING 句
- GIVING 句
- ROUNDED 句
- SIZE ERROR 句

CORRESPONDING 句

CORRESPONDING (CORR) 句を使用すると、ADD、SUBTRACT、および MOVE の操作を同じ名前の基本データ項目に対して行うことができます。 ただし、その場合それらのデータ項目が属する英数字グループ項目または国別グループ項目が指定されている必要があります。

CORRESPONDING 句が使用されているときには、国別グループはグループ項目として処理されます。

キーワード CORRESPONDING の後に置かれる 2 つの ID は、グループ項目の名前を指名しなければなりません。 次の説明では、これらの ID は *ID-1* および *ID-2* として参照されます。 *ID-1* は送り出しグループ項目を参照します。 *ID-2* は受け取りグループ項目を参照します。

2 つの従属データ項目は、1 つは *ID-1* から、もう一方は *ID-2* からのもので、次に示す条件が真であれば対応します。

- ADD ステートメントまたは SUBTRACT ステートメントで、その 2 つのデータ項目は基本数字データ項目です。 それ以外の種類のデータ項目は無視されます。

- MOVE ステートメントでは、少なくともデータ項目の 1 つは基本項目であり、 データの移動は、移動規則に従って行えます。
- 2 つの従属項目が同じ名前と同じ修飾子を持ち、この修飾子は *ID-1* および *ID-2* を除き同じになります。
- 従属項目は、キーワード FILLER によっては識別されません。
- *ID-1* も *ID-2* もレベル 66、77、 または 88 の項目として記述されることはありません。 またどちらも指標データ項目として記述されることはありません。 *ID-1* も *ID-2* も、参照変更にすることはできません。
- *ID-1* も *ID-2* も、USAGE POINTER、USAGE FUNCTION-POINTER、または USAGE PROCEDURE-POINTER として記述されません。
- 従属項目でこれらの記述に含まれない節は、REDEFINES、RENAMES、OCCURS、USAGE INDEX、USAGE POINTER、USAGE PROCEDURE-POINTER、 または USAGE FUNCTION-POINTER です。

  ただし、*ID-1* および *ID-2* は、 REDEFINES 節や OCCURS 節をその記述中に含む項目を含んだり、またはそれに従属することはできます。
- これらの条件を満たすそれぞれの従属データ項目の名前は、暗黙の修飾子の適用後も固有です。

ID-1、*ID-2*、またはその両方は、FILLER 項目に従属することができます。

例えば、2 つのデータ階層が次のように定義されているものとします。

```
05 ITEM-1 OCCURS 6.
   10  ITEM-A PIC S9(3).
   10  ITEM-B PIC +99.9.
   10  ITEM-C PIC X(4).
   10  ITEM-D REDEFINES ITEM-C PIC 9(4).
   10  ITEM-E USAGE COMP-1.
   10  ITEM-F USAGE INDEX.
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.
```

ADD CORR ITEM-2 TO ITEM-1(x) が指定された場合、ITEM-A と ITEM-A(x)、ITEM-B と ITEM-B(x)、および ITEM-E と ITEM-E(x) は対応するものとみなされ、共に加算されます。 ITEM-C および ITEM-C(x) は、この処理に含まれません。これは数字ではないからです。 ITEM-D および ITEM-D(x) は含まれません。これは ITEM-D(x) が REDEFINES 節をそのデータ記述中に含むからです。 ITEM-F および ITEM-F(x) は、この処理に含まれません。これは指標データ項目だからです。 ITEM-1 が *ID-1* または *ID-2* どちらとして有効である点に注意してください。

ADD CORRESPONDING ステートメント内の個々の操作のいずれかにおいてサイズ・エラー条件が発生した場合、 ON SIZE ERROR 句の中の命令ステートメント*-1* は、個々の加算がすべて完了するまで実行されません。

GIVING 句

算術ステートメントの場合、GIVING という語の後に続く ID の値は算術演算の計算結果に等しい値に設定されます。この ID は計算の対象にならないので、数字編集項目にすることができます。

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。

結果として得られた小数部のサイズが それを記憶するために指定された桁数を超えているときは、ROUNDED 句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、結果 ID の最下位数字は、 超過分の最上位数字が 5 以上であるときはいつでも 1 だけ増加されます。

結果 ID が右端に P (複数個) を持つ PICTURE 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。

浮動小数点の算術演算では ROUNDED 句は関係ありません。 浮動小数点の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、「*COBOL for Linux on x86* プログラミング・ガイド」の『固定小数点演算と浮動小数点演算との対比』を参照してください。

ARITH(EXTEND) コンパイラー・オプションが指定されている場合は、 小数点の右側に 31 桁ある演算の受け取り側に対して ROUNDED 句はサポートされません。 例えば、次の X および Y のいずれも、ROUNDED 句では受け取り側としては有効ではありません。

```
01  X PIC V31.
01  Y PIC P(30)9(1).
    . . .
    COMPUTE X ROUNDED = A + B
    COMPUTE Y ROUNDED = A - B
```

これ以外の場合は、ROUNDED 句は拡張精度算術ステートメントに対して完全に サポートされます。

SIZE ERROR 句

サイズ・エラー条件は、さまざまな場合に発生する可能性があります。

- 小数点位置合わせが行われた後、算術計算の結果の絶対値が結果フィールドに収容できる最大の値を超えている場合
- 0 による除算が発生した場合
- 算術ステートメントの結果がウィンドウ表示日付フィールドに保管され、その結果の年が世紀ウィンドウの範囲外の場合。 例えば、(1940–2039 の世紀ウィンドウを指定する) YEARWINDOW(1940) が指定されている場合、次の SUBTRACT ステートメントはサイズ・エラーになります。

  ```
  01  WINDOWED-YEAR  DATE FORMAT YY PICTURE 99
                     VALUE IS 50.
      ...
      SUBTRACT 20 FROM WINDOWED-YEAR
               ON SIZE ERROR imperative-statement
  ```

  サイズ・エラーとなるのは、減算の結果 (ウィンドウ表示日付フィールド) が 1930 という年の有効値を持つことになり、 これが世紀ウィンドウの範囲外であるためです。 ウィンドウ表示日付フィールドが拡張日付フォーマットに変換されたかのように取り扱われる方法の詳細については、237 ページの『日付フィールドが関係する減算』を参照してください。

  日付フィールドを使用した場合のサイズ・エラーの発生の詳細については、237 ページの『日付フィールドに関連する算術演算結果の保管』を参照してください。
- 次の表で示されるような指数式の場合

表 *41.* **指数計算のサイズ・エラー条件**

サイズ・エラー	**SIZE ERROR 節が指定されているときに取られる処置**	**SIZE ERROR 節が指定されていないときに取られる処置**
0 が 0 乗された	SIZE ERROR 命令が実行されます。	値として 1 が戻され、メッセージが出されます。
0 が負数乗された	SIZE ERROR 命令が実行されます。	プログラムは終了します。
負数が小数部乗された	SIZE ERROR 命令が実行されます。	基数の絶対値が使用され、メッセージが出されます。

サイズ・エラー条件は最終結果にだけ適用され、中間結果には適用されません。

結果 ID が USAGE BINARY、COMPUTATIONAL、COMPUTATIONAL-4、または COMPUTATIONAL-5 を使用して定義されている場合、 TRUNC コンパイラー・オプションが有効であるかどうかにかかわらず、 結果のデータ項目に入れられる最大値は、その項目の 10 進 PICTURE 文字ストリングによって 暗黙に指定された値です。

ROUNDED 句が指定されている場合は、サイズ・エラー検査の前に丸めが実行されます。

サイズ・エラーが起きると、プログラムの以降の処置は 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 句が指定されていない場合にサイズ・エラー条件が起きると、切り捨て規則が適用されてから、 その影響を受ける結果の ID の値が計算されます。

ON SIZE ERROR 句が指定されている場合にサイズ・エラー条件が起きると、 そのサイズ・エラーによって影響を受ける結果の ID の値は 変更されません。つまり、エラー結果が受け取り側の ID に入れられることはありません。 算術演算の実行完了後に、ON SIZE ERROR 句で指定された命令ステートメントが実行され、 制御は算術ステートメントの終わりに移され、 NOT ON SIZE ERROR 句はその指定があっても無視されます。

ADD CORRESPONDING および SUBTRACT CORRESPONDING については、個々の算術演算がサイズ・エラー条件を起こした場合、ON SIZE ERROR 句の指定する命令ステートメントは、個々の加算や減算がすべて完了するまで実行されません。

NOT ON SIZE ERROR 句が指定されていた場合は、算術演算の実行後、 サイズ・エラー条件は存在せず、NOT ON SIZE ERROR 句が実行されます。

ON SIZE ERROR 句と NOT ON SIZE ERROR 句が両方とも指定されている場合で、実行される句の中のステートメントに明示的な制御の移動がなければ、 必要に応じてその句の実行後に算術ステートメントの終わりに暗黙の制御の移動が行われます。

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 の各ステートメントによって指定します。 これらの個々の演算は、COMPUTE ステートメントを使用する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、同じである必要はありません。 計算の間、コンパイラーは必要なデータ変換および小数点桁合わせを行います。

オペランドのサイズ

ARITH(COMPAT) コンパイラー・オプションが有効な場合は、各オペランドの最大サイズは 10 進数の 18 桁です。 ARITH(EXTEND) コンパイラー・オプションが有効な場合は、各オペランドの最大サイズは 10 進数の 31 桁です。

オペランドの合成 は、複数のオペランドを小数点の位置で合わせ、それらを互いに重ね合わせた結果 生成される仮想データ項目のことです。

ARITH(COMPAT) コンパイラー・オプションが有効な場合、オペランドの合成のサイズは、最大 30 桁となります。 ARITH(EXTEND) コンパイラー・オプションが有効な場合、オペランドの合成のサイズは、最大 31 桁となります。

次の表は、 各算術ステートメントにおいてオペランドの合成がどのようにして決められるかを示します。

表 *42.* **オペランド合成の決定方法**	
ステートメント	**オペランド合成の決め方**
SUBTRACT ADD	GIVING という語の後のオペランドを除き、所定のステートメント内のすべてのオペランドを重ね合わせる
MULTIPLY	受け取りデータ項目をすべてを重ね合わせる
DIVIDE	REMAINDER データ項目を除き、受け取りデータ項目をすべて重ね合わせる
COMPUTE	制約事項は適用されない

例えば、各項目が DATA DIVISION 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).
```

次のようなステートメントが実行されると、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になります。

```
COMPOSITE-OF-OPERANDS PICTURE 9(12)V9(5).
```

ADD ステートメントおよび SUBTRACT ステートメントでは、 オペランドの合成が 30 桁以下 (ARITH(COMPAT) コンパイラー・オプションを指定した場合)、 または 31 桁以下 (ARITH(EXTEND) コンパイラー・オプションを指定した場合) であれば、実行時に有効数字が切り落とされないように、コンパイラーで十分なスペースが確保されます。

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、必要な精度の最終結果が得られるように、十分な桁数と小数部を指定することが重要です。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『付録 *D.* 中間結果および算術精度』を参照してください。

オーバーラップしたオペランド

算術ステートメントのオペランドがそのストレージの一部を共用する場合 (すなわちオペランドが重なる場合)、こうしたステートメントを実行すると予測不能な結果が生じます。

複数の演算結果

1 つの算術ステートメントが複数の演算結果を持つとき、実行は概念的には次のようにして行われます。

1. ステートメントはすべての算術演算を行って複数の受け取り項目に入れる結果を出し、その結果を一時的な記憶場所に収めます。
2. 一連のステートメントで転送したり、この一時的な結果の値を それぞれの単一の受け取り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左から右の順序と同じ順序で記述されているものと考えることができます。

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、以下の一連のステートメントの実行と同じことです。

```
ADD A, B, C GIVING TEMP.
ADD TEMP TO C.
ADD TEMP TO D(C).
ADD TEMP TO E.
```

上記の例で TEMP とは、コンパイラーの提供する一時的な結果フィールドです。 D (C) に対して加算演算が行われると、添え字 C には、 C の新しい値が入ります。

データ操作ステートメント

次に示す COBOL ステートメントは、データの移動と検査を行います。ACCEPT、INITIALIZE、INSPECT、MOVE、READ、RELEASE、RETURN、REWRITE、SET、STRING、UNSTRING、WRITE、XML PARSE、および XML GENERATE。

オーバーラップしたオペランド

データ操作ステートメントの送り出しフィールドと受け取りフィールドがストレージの一部を共用している場合 (つまり、オペランドがオーバーラップしているとき)、そのようなステートメントを実行した結果は予測できません。

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たファイルへデータを転送し、また外部メディアに保管されたファイルからデータを転送します。また、入出力装置から 取得または入出力装置への送信が行われた少量のデータも制御します。

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です。 プログラマーはレコードに注意を払うだけで済みます。 バッファー、内部記憶域、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、およびボリューム切り替えプロシージャーへのデータの移動などの操作のためのプロビジョンは、自動的に行われます。

ENVIRONMENT DIVISION と DATA DIVISION でファイルがどのように記述されているかによって、PROCEDURE DIVISION で使用できる入出力ステートメントが決まります。 順次ファイルで使用可能なステートメントは 339 ページの表 52 に記載されています。 索引ファイルおよび相対ファイルで使用可能なステートメントは 339 ページの表 53 に記載されています。 行順次ファイルで使用可能なステートメントは、340 ページの表 54 に記載されています。

共通の処理機能

複数の入出力ステートメントに適用される共通の処理機能がいくつかあります。

共通の処理機能は次のとおりです。

- 269 ページの『ファイル状況キー』
- 274 ページの『無効キー条件』
- 274 ページの『INTO 句および FROM 句』
- 275 ページの『ファイル位置標識』

次のセクションでは、ボリューム およびリール という用語の使用について説明します。 ボリューム という用語は、ユニット・レコード入出力装置以外の入出力装置を指します。 リール はテープ装置にのみ適用されます。 順次アクセス・モードにおける直接アクセス装置の処理方法は、テープ装置の処理方法と論理的には同じです。

ファイル状況キー

ファイル制御項目で FILE STATUS 節を指定した場合は、そのファイルに対する要求の実行中、 指定したファイル状況キー (FILE STATUS 節で指定した 2 文字のデータ項目) の中に値が入れられます。 この値は、要求の状況を表します。

値がファイル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、INVALID KEY 句、または AT END 句が実行されます。

2 種類のファイル状況キー・データ名があります。1 つは、ファイル制御項目の FILE STATUS 節の中でデータ名-*1* によって記述されます。これは、ファイル状況キー 1 として認識される最初の 1 文字と、ファイル状況キー 2 として認識される 2 番目の文字を持つ 2 文字データ項目です。可能な値の組み合わせとその意味については、270 ページの表 43 に示しています。 .

もう 1 つのファイル状況キーは、ファイル制御項目の FILE STATUS 節の中にデータ名-*8* として記述されます。 *data-name-8* は、行順次ファイルには適用されません。 データ名-*8* について詳しくは、123 ページの『FILE STATUS 節』を参照してください。

表 *43.* **ファイル状況キーの値と意味**

上位数字	意味	下位数字	意味
0	正常終了	0	それ以上の情報はありません。
		2	このファイル状況値は、重複可能な代替キーのある指標ファイルのみに適用されます。 入出力ステートメントは正常に実行されましたが、複写するキーが見つかりました。 READ ステートメントの場合は、現行参照キーのためのキー値が、現行参照キー内の次のレコードにある同じキー値と等しい値でした。 REWRITE ステートメントまたは WRITE ステートメントの場合は、今書き込まれたレコードが、 重複が許される 1 つ以上の代替レコード・キーに対して複写キーを作成しました。
		4	READ ステートメントは正常に実行されましたが、処理されるレコードの長さがそのファイルの固定ファイル属性に一致しませんでした。
		5	OPEN ステートメントは正常に実行されましたが、参照されたオプション・ファイルが、OPEN ステートメントの実行時に使用可能ではありませんでした。 オープン・モードが I-O または EXTEND ならば、ファイルは作成済みです。
		7	NO REWIND 句、REEL/UNIT 句、 または FOR REMOVAL 句を持つ CLOSE ステートメントの場合、または NO REWIND 句を持つ OPEN ステートメントの場合、 参照されたファイルは非リール / ユニット・メディア上にありました。
1	AT END 条件	0	順次 READ ステートメントを試みましたが、ファイルの終わりに達したため次の論理レコードがファイル中に存在しませんでした。 または、最初の READ ステートメントをオプション入力ファイル上で試みましたが、そのファイルが使用可能でありませんでした。
		4	順次 READ ステートメントを相対ファイルで試みましたが、相対レコード番号の中の有効数字の桁数が、そのファイル用に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 *43.* **ファイル状況キーの値と意味** *(続き)*

上位数字	意味	下位数字	意味
2	無効キー条件	1	順次にアクセスされた索引付きファイルにシーケンス・エラーがあります。 READ ステートメントが正しく実行されてから、次の REWRITE ステートメントがそのファイルで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ました。 または、連続レコード・キー値に要求される昇順でなければならないという条件に違反していました。
		2	レコードを書き込む試みを行いましたが、相対ファイルに複写キーを作成するものでした。 または、レコードを書き込んだり再度書き込んだりする試みを行いましたが、そのレコードは DUPLICATES 句がないにもかかわらず、重複基本レコード・キーまたは重複代替レコード・キーを索引付きファイルに作成するものでした。
		3	ファイルに存在しないレコードに対しランダムにアクセスする試みを行いました。 または、START ステートメントまたはランダム READ ステートメントを、使用可能でないオプション入力ファイル上で試みようとしました。
		4	相対ファイル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もうとしました。 または、順次 WRITE ステートメントを相対ファイルで試みましたが、相対レコード番号の中の有効数字の桁数が、そのファイル用に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 *43.* **ファイル状況キーの値と意味** *(続き)*

上位数字	意味	下位数字	意味
3	永続エラー条件	3	OPEN ステートメントは、連結されたファイルが指定されていたが、連結の要件の 1 つ以上が満たされていなかったため、正常に実行されませんでした。 考えられる違反としては、次のものが挙げられます。 • ファイル ORGANIZATION が SEQUENTIAL または LINE SEQUENTIAL ではありませんでした。 • ACCESS MODE が SEQUENTIAL ではありませんでした。 • OPEN モードが INPUT ではありませんでした。 • 連結に指定されたファイル ID が多すぎたか、または指定されたファイル ID を記録するためのメモリーが取得できませんでした。 OPEN または READ ステートメントが、連結内の非オプション・ファイルが使用不可だったために正常に実行されませんでした。 例えば、ファイルが存在しなかったか、または十分なアクセス権限を持っていませんでした。 *Data-name-8* が指定されている場合は、オリジナルのファイル状況値、失敗したファイル ID、および非連結ファイル指定のために *data-name-8* に指定されている任意の追加情報を含みます。
		4	境界違反による永続エラーがあります。 順次ファイルの外部定義境界を超えて書き込もうとしました。
		5	INPUT 句、I-O 句、または EXTEND 句を指定した OPEN ステートメントを、使用可能でない非オプションのファイルに対して試みました。
		7	OPEN ステートメントで指定したオープン・モードをサポートしていないファイル上で OPEN ステートメントを試みました。 考えられる違反としては、次のものが挙げられます。 • EXTEND 句または OUTPUT 句を指定しましたが、そのファイルは書き込み操作をサポートしていません。 • I-O 句を指定しましたが、 そのファイルは許可される入出力操作をサポートしていません。 • INPUT 句を指定しましたが、そのファイルは読み取り操作をサポートしていません。
		8	OPEN ステートメントを、 以前にロック付きでクローズしたファイル上で試みました。
		9	固定ファイル属性とプログラムの中でそのファイルに対して指定した属性の間に不一致が検出されたため、 OPEN ステートメントが正しく実行されませんでした。 これらの属性には、ファイルの編成 (順次、相対、指標付き)、 基本レコード・キー、代替レコード・キー、コード・セット、 最大レコード・サイズ、レコード・タイプ (固定長、可変長)、 およびブロック化因数があります。

表 *43.* **ファイル状況キーの値と意味** *(続き)*

上位数字	意味	下位数字	意味
4	論理エラー条件	1	オープン・モードにあるファイルに対して OPEN ステートメントを試みました。
		2	オープン・モードにないファイルに対して CLOSE ステートメントを試みました。
		3	順次アクセス・モードにある大容量記憶ファイルの場合は、関連したファイルに対して REWRITE ステートメントの実行前に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・モードにある相対ファイルおよび索引付きファイルの場合には、関連したファイルに対して DELETE ステートメントまたは REWRITE ステートメントの実行前に実行された最後の 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	あるレコードをファイルに書き込む試みを行いましたが、そのレコードは書き換えようとするレコードと同じサイズではなかったために境界違反が起こりました。 またはあるレコードを書き込んだり、再度書き込んだりする試みを行いましたが、そのレコードは関連したファイル名の RECORD IS VARYING 節で許容されている最大レコードより大きかったか、 最小レコードより小さかったために、境界違反が起こりました。
		6	INPUT または I-O のオープン・モードにあるファイル上で順次 READ ステートメントを試みましたが、 有効な次のレコードが設定されていませんでした。 その理由としては、次のものが考えられます。 • 先行した READ ステートメントが正しく実行されなかったにもかかわらず、AT END 条件を引き起こしませんでした。 • 先行した READ ステートメントが AT END 条件を引き起こしました。
		7	INPUT や I-O のオープン・モードではないファイルに対して READ ステートメントの実行を試みました。
		8	I-O、OUTPUT、または EXTEND のオープン・モードではないファイルに対して WRITE ステートメントの実行を試みました。
		9	I-O のオープン・モードではないファイルに対して DELETE ステートメントまたは REWRITE ステートメントの実行を試みました。

表 *43.* **ファイル状況キーの値と意味** *(続き)*			
上位数字	**意味**	**下位数字**	**意味**
9	インプリメンター定義条件	0	それ以上の情報はありません。
		1	許可の失敗
		2	論理エラー
		3	リソースが利用できません。
		4	同時オープン・エラー
		5	ファイル情報が無効または不完全です。
		6	ファイル・システムが使用できません
		8	ファイルがロックされているため、オープンが失敗しました。
		9	レコードがロックされているため、レコードにアクセスできませんでした。

無効キー条件

無効キー条件が起こるのは、START、READ、WRITE、REWRITE、 または DELETE の各ステートメントのうちのいずれかの実行中です。 無効キー条件が発生した場合、 その条件を起こした入出力ステートメントは正しく実行されません。

無効キー条件が認識されると、次の順序で処置が取られます。

1. ファイル制御項目の中に FILE STATUS 節が指定されている場合は、キー条件が無効であることを示す値がファイル状況キーに入れられます。270 ページの表 43 を参照してください。
2. この条件を引き起こすステートメントの中に INVALID KEY 句が指定されていると、 制御は INVALID KEY 命令ステートメントに移ります。 このファイルに対して指定された EXCEPTION/ERROR 宣言型プロシージャーがあっても、それは実行されません。 その場合は、命令ステートメントで指定された各ステートメントの規則に従って実行が続けられます。
3. ファイルに対する入出力ステートメント内に INVALID KEY 句が指定されておらず、 利用可能な EXCEPTION/ERROR プロシージャーが存在する場合は、そのプロシージャーが実行されます。 NOT INVALID KEY 句は、たとえそれが指定されていても無視されます。

INVALID KEY 句も EXCEPTION/ERROR プロシージャーも両方とも省略することができます。

入出力操作の実行後に無効キー条件が存在しなければ、 INVALID KEY 句は指定されていても無視され、次の処置が取られます。

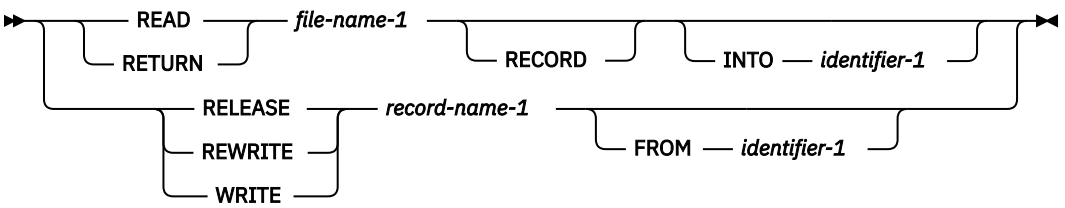
- 無効キー条件ではない例外条件が存在すれば、 USE AFTER EXCEPTION プロシージャーの実行に続く USE ステートメントの規則に従って、制御が移されます。
- 例外条件が何も存在しない場合には、制御は入出力ステートメントの終わりに移されるか、 または、もし NOT INVALID KEY 句が指定されていれば、制御はその句の中に指定されている命令ステートメントの終わりに移されます。

INTO 句および FROM 句

INTO 句および FROM 句は、READ、RETURN、RELEASE、REWRITE、および WRITE の各ステートメントに対して有効です。

WORKING-STORAGE SECTION または LINKAGE SECTION 内の項目の名前、またはすでにオープンされている別のファイル用のレコード記述を ID に 指定しなければなりません。

フォーマット: 入出力ステートメントの INTO および FROM 句



- レコード名-*1* および *ID-1* は、同じストレージ域を参照することはできません。
- レコード名-*1* または *ID-1* が国別グループ項目を参照する場合、この項目は国別カテゴリーの基本データ項目として処理されます。
- INTO 句は、READ ステートメントまたは RETURN ステートメントの中で指定することができます。

  INTO 句を伴う READ ステートメントまたは RETURN ステートメントの実行結果は、 次の規則を指定した順序で適用した結果と同じになります。

  – INTO 句を持たない同じ READ ステートメントまたは RETURN ステートメントを実行します。
  – 現行レコードを、CORRESPONDING 句のない MOVE ステートメントの規則に従って、そのレコード域から *ID-1* で指定された領域へ移動します。 現行レコードのサイズは、RECORD 節に指定された規則によって判別されます。 ファイル記述項目が RECORD IS VARYING 節を含む場合には、暗黙の移動はグループ移動になります。 READ ステートメントまたは RETURN ステートメントの実行が正しく行われなかった場合には、 暗黙の MOVE ステートメントの実行は行われません。 *ID-1* に関連付けられた添え字付けまたは参照変更は、 レコードが読み取られたか、または戻された後、 それがデータ項目に移動される直前に評価されます。 レコードは、そのレコード域と *ID-1* によって参照されるデータ項目の両方で使用可能です。
- FROM 句は RELEASE、REWRITE、 または WRITE の各ステートメントで指定することができます。

  FROM 句を伴う RELEASE ステートメント、REWRITE ステートメント、または WRITE ステートメントの実行結果は、 次のステートメントを指定した順序で実行した結果と同じになります。

  1. MOVE *ID-1* TO レコード名-*1*
  2. FROM 句を伴わない同じ RELEASE ステートメント、REWRITE ステートメント、または WRITE ステートメント

  RELEASE、REWRITE、または WRITE の各ステートメントの実行を完了した後は、*ID-1* によって参照される領域にある情報は、レコード名-*1* によって参照される領域の情報が使用可能でない場合でも使用可能です。 ただし、SAME RECORD AREA 節によって指定されたものを除きます。

ファイル位置標識

ファイル位置標識は、本書で使用される概念上のエンティティーですが、特定のシーケンスでの入出力操作中に、特定のファイル内で次にアクセスされるレコード (または前のレコード)の正確な指定を容易にします。

ファイル位置標識の設定値は、OPEN、 CLOSE、READ、および START の各ステートメントによってのみ影響を受けます。 ファイル位置標識の概念は、OUTPUT または EXTEND のモードでオープンされたファイルでは何も意味を持ちません。

第 19 章 PROCEDURE DIVISION ステートメント

PROCEDURE DIVISION 内のステートメント、文、および段落は連続して実行されますが、 EXIT、 GO TO、 PERFORM、 GOBACK、 または STOP などのプロシージャー・ブランチ・ステートメントが使用されている場合は、 連続して実行されません。

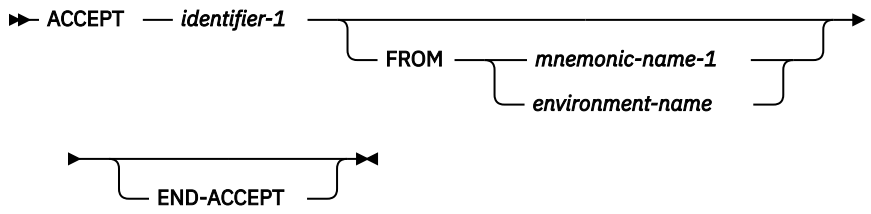
ACCEPT ステートメント

ACCEPT ステートメントは、指定された ID によって参照されるデータ域へデータまたはシステムの 関連情報を転送します。 転送されてくるデータの編集やエラー・チェックは行われません。

データ転送

フォーマット 1 では、データは入力ソースから *ID-1* が参照するデータ項目 (受け取り領域) へ転送されます。 FROM 句を省略すると、システム入力装置が想定されます。

フォーマット 1: データ転送



プログラム中にオペレーターの介入が (特定のメッセージ、コード、 または例外標識を提供するために) 必要である例外状況では、フォーマット 1 が役立ちます。 オペレーターには、応答に使用する、該当するメッセージを提供する必要があります。

入力ファイルは、バイト・ストリーム・ファイル (例えば、レコード終了文字で区切られたレコードを持つテキスト・データから構成されるファイル) でなければなりません。 COBOL プログラムでは、行順次ファイル入出力または DISPLAY ステートメントを使用してバイト・ストリーム・ファイルを作成できます。 (通常のテキスト・エディター を使用してバイト・ストリーム・ファイルを作成することもできます)。

SdU、SFS、または STL ファイル (順次、相対、または索引ファイルを含む) を、入力ファイルとして使用できません。

ACCEPT ステートメントのソースがファイルで、受け取り領域がレコード終止符で完全に区切られたレコードで埋められていない場合、 そのファイルの次の ACCEPT ステートメントで、入力レコードの残りが使用されます。 入力レコードを受け取り領域に移動する前に、 レコード区切り文字が入力データから除去されます。

ACCEPT ステートメントのソースがキーボードの場合、Enter キーの前にキーボードから入力したデータが、 入力データとして取り扱われます。 入力データが受け取り領域より短い場合、その領域は受け取り領域に 適切な表現のスペースで埋められます。

ID-1
受け取り領域。 リテラル-3 は、下記のものになります。

- 英数字グループ項目

- 国別グループ項目
- USAGE DISPLAY、DISPLAY-1、または NATIONAL の基本データ項目

国別グループ項目は、国別カテゴリーの基本データ項目として処理されます。

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 の使用法が NATIONAL で、入力データの入力元が キーボードである場合は、入力がネイティブ・コード・ページ (ランタイムのロケールで 指定されたコード・ページ) から国別文字表現に変換されます。

簡略名-*1*
入力装置を指定します。 簡略名-*1* は、SPECIAL-NAMES 段落内で環境名と関連付ける必要があります。 91 ページの『SPECIAL-NAMES 段落』を参照してください。 .

環境名
入力データのソースを識別します。 94 ページの表 5 に 示された名前の中から環境名 を指定できます。

環境名付き ACCEPT は、操作環境の環境変数割り当てによってその環境名に関連付けされた ソースを使用します。 COBOL 環境名に対応する環境変数が設定されていない場合は、 SYSIN、または SYSIPT からの ACCEPT は、システム論理入力装置からの ACCEPT になります。 CONSOLE からの ACCEPT は、ユーザーのキーボードからの ACCEPT です。 環境変数について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『例*:* 環境変数の設定とアクセス』 を参照してください。

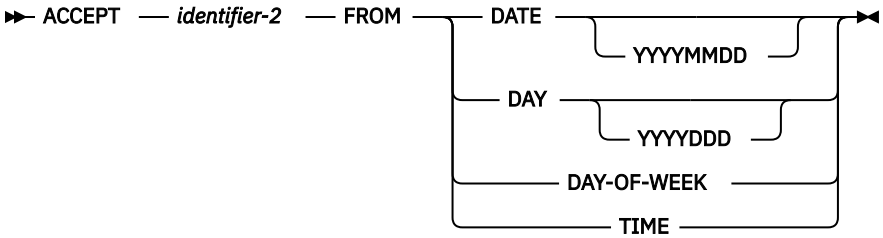
装置が、LINE SEQUENTIAL ファイルの READ ステートメントで使われる装置と同じである場合、結果は予想できません。

システム日付関連情報の転送

指定された概念上のデータ項目 DATE、DATE YYYYMMDD、DAY、DAY YYYYDDD、DAY-OF-WEEK、または TIME に含まれているシステム情報は、*ID-2* によって参照されるデータ項目へ転送することができます。 この転送は、CORRESPONDING 句を伴わない MOVE ステートメントの規則に従わなければなりません。

詳しくは、327 ページの『MOVE ステートメント』を参照してください。.

フォーマット 2: システム情報転送



ID-2
受け取り領域。 リテラル-3 は、下記のものになります。

- 英数字グループ項目
- 国別グループ項目
- 以下のいずれかのカテゴリーの基本データ項目
  - 英数字
  - 英数字編集
  - 数字編集 (USAGE DISPLAY または NATIONAL)
  - 国別
  - 国別編集

- 数字
- 内部浮動小数点
- 外部浮動小数点 (USAGE DISPLAY または NATIONAL)

国別グループ項目は、国別カテゴリーの基本データ項目として処理されます。

フォーマット 2 では 2 つのフォーマットの現在日付、すなわち、 システムによって随時更新される曜日または時刻を使用します。これは、 あるオブジェクト・プログラムがいつ実行されたのかを識別するのに役立ちます。 また、フォーマット 2 は、ヘッダーやフッターに日付を付けるために使用することもできます。

現在の日付や時刻は、 日時の組み込み関数 CURRENT-DATE を使用してもアクセスできます。 CURRENT-DATE は 4 桁の年の値をもサポートしており、 追加情報を提供します (438 ページの『CURRENT-DATE』を参照)。

DATE、DATE YYYYMMDD、DAY、DAY YYYYDDD、DAY-OF-WEEK、および TIME

概念上のデータ項目 DATE、DATE YYYYMMDD、DAY、DAY YYYYDDD、 DAY-OF-WEEK、および TIME には、 USAGE DISPLAY が暗黙的に指定されます。 これらは概念上のデータ項目であるため、COBOL プログラムで記述すること はできません。

概念上のデータ項目の内容は、MOVE ステートメントの規則を使用して受け取り領域へ移動されます。 受け取り領域が USAGE NATIONAL の場合は、データは国別文字表現に変換されます。

DATE

暗黙の指定として PICTURE 9(6) になります。 DATEPROC コンパイラー・オプションが有効な場合は、戻り値は暗黙の DATE FORMAT YYXXXX になり、*ID-2* はこの日付フォーマットで定義されていなければなりません。

データ・エレメントのシーケンスは (左から右へ) 次のような内容になっています。

```
Two digits for the year
Two digits for the month
Two digits for the day
```

したがって、2003 年 4 月 27 日は 030427 となります。

DATE YYYYMMDD

暗黙の指定として PICTURE 9(8) になります。 DATEPROC コンパイラー・オプションが有効な場合は、戻される値は暗黙の DATE FORMAT YYYYXXXX を持ち、*ID-2* はこの日付フォーマットで定義されていなければなりません。

データ・エレメントのシーケンスは (左から右へ) 次のような内容になっています。

```
Four digits for the year
Two digits for the month
Two digits for the day
```

したがって、2003 年 4 月 27 日は 20030427 となります。

DAY

暗黙の指定として PICTURE 9(5) になります。 DATEPROC コンパイラー・オプションが有効な場合は、戻り値は暗黙の DATE FORMAT YYXXX になり、*ID-2* はこの日付フォーマットで定義されていなければなりません。

データ・エレメントのシーケンス (左から右へ) は次のような内容になっています。

```
Two digits for the year
Three digits for the day
```

したがって、2003 年 4 月 27 日は 03117 となります。

DAY YYYYDDD
これは暗黙の指定として PICTURE 9(7) になります。 DATEPROC コンパイラー・オプションが有効な場合は、 戻される値は暗黙の DATE FORMAT YYYYXXX を持ち、*ID-2* はこの日付フォーマットで定義されていなければなりません。

データ・エレメントのシーケンスは (左から右へ) 次のような内容になっています。

```
Four digits for the year
Three digits for the day
```

したがって、2003 年 4 月 27 日は 2003117 となります。

DAY-OF-WEEK
これは暗黙の指定として PICTURE 9(1) になります。

単一のデータ・エレメントは、次のような値で曜日を表します。

```
1 represents Monday           5 represents Friday
2 represents Tuesday          6 represents Saturday
3 represents Wednesday        7 represents Sunday
4 represents Thursday
```

したがって、水曜日は 3 となります。

TIME
これは暗黙の指定として PICTURE 9(8) になります。

データ・エレメントのシーケンス (左から右へ) は次のような内容になっています。

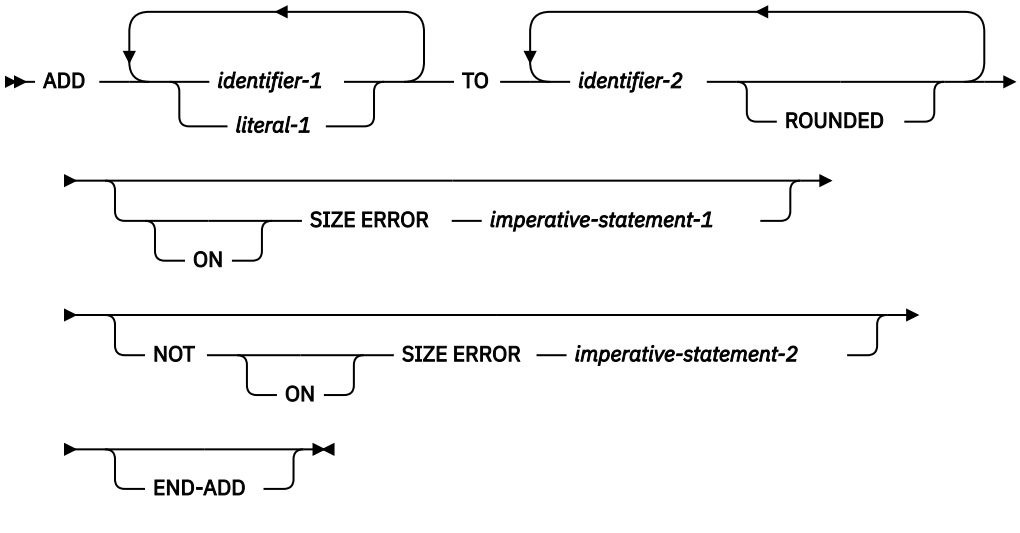
```
Two digits for hour of day
Two digits for minute of hour
Two digits for second of minute
Two digits for hundredths of second
```

したがって、2:41 PM は 14410000 となります。

ADD ステートメント

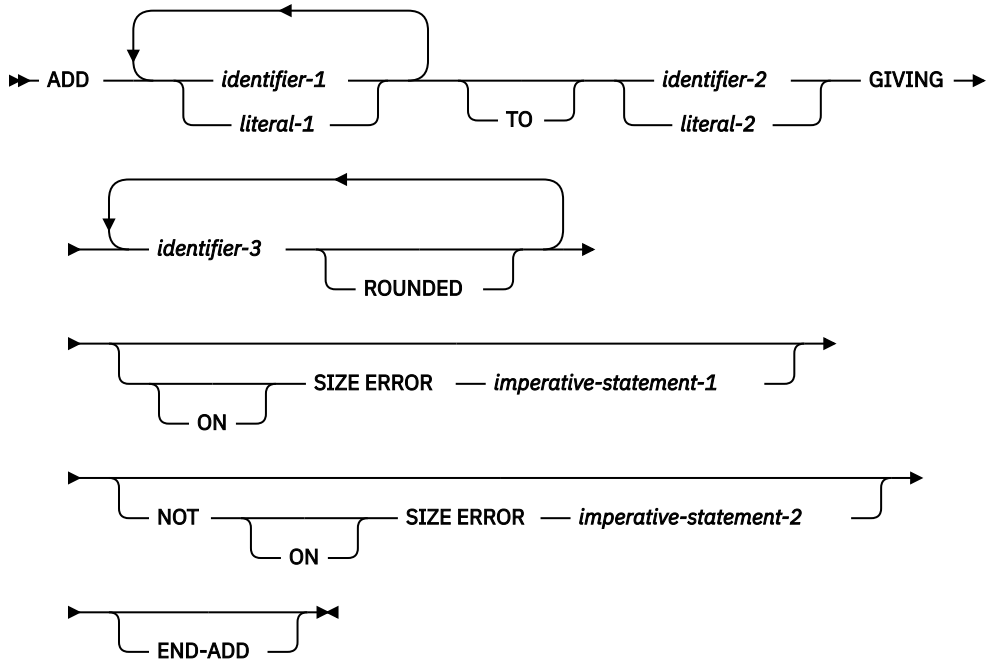
ADD ステートメントは、2 つ以上の数字オペランドを合計し、 その結果を保管します。

フォーマット 1: ADD ステートメント



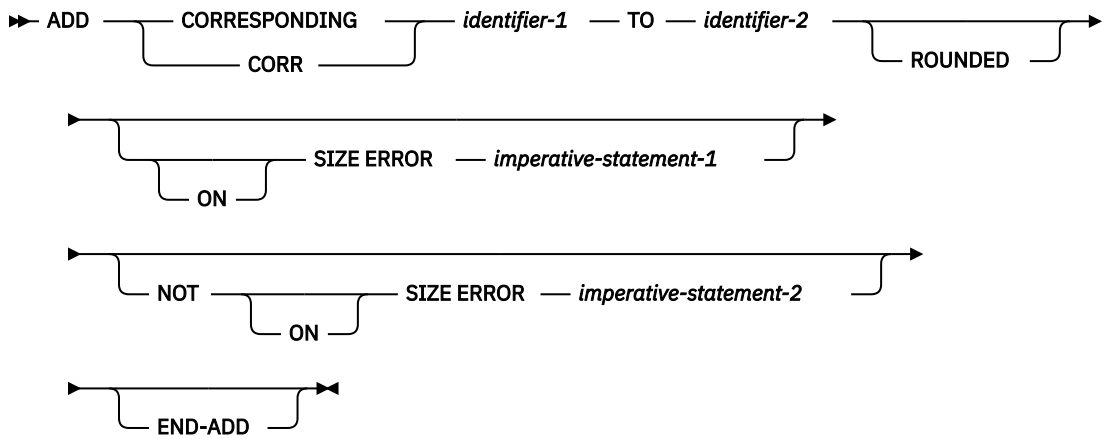
キーワード TO の前にあるすべての ID やリテラルが加算され、 この合計が、*ID-2* に加算され保管されます。 この処理は、*ID-2* が連続する場合、それぞれの ID-2 ごとに、*ID-2* が指定されている 順序で左から右へと繰り返されます。

フォーマット 2: GIVING 句を含む ADD ステートメント



キーワード GIVING の前にあるオペランドの値は互いに加算され、 その合計は、*ID-3* 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。

フォーマット 3: CORRESPONDING 句を含む ADD ステートメント



ID-1 内の基本データ項目が加算されて、対応する *ID-2* 内の基本項目に保管されます。

すべてのフォーマットに関して次のことが言えます。

ID-1*、 *ID-2

フォーマット 1 では、基本数字項目を指定しなければなりません。

フォーマット 2 では、ワード GIVING の後にあるもの以外は、基本数字項目でなければなりません。 ワード GIVING に続く各 ID には、 基本数字項目または数字編集項目を指名しなければなりません。

フォーマット 3 では、英数字グループ項目または国別グループ項目を 指定する必要があります。

以下の制約事項は、日付フィールドに適用されます。

- フォーマット 1 では、*ID-2* は 1 つ以上の日付フィールドを指定することができます。 *ID-1* では、日付フィールドを指定する必要はありません。
- フォーマット 2 では、*ID-1* または *ID-2* のどちらか (両方ではない) が 1 つの日付フィールドを指定することができます。 *ID-1* または *ID-2* が日付フィールドを指定する場合は、*ID-3* のすべてのインスタンスでは、*ID-1* または *ID-2* で指定された日付フィールドと互換性のある日付フィールドを指定しなければなりません。 すなわち、ウィンドウ表示西暦年でも拡張西暦年でもよい年部分を除いて、 同じ日付フォーマットでなければなりません。

  ID-1 または *ID-2* のいずれも日付フィールドを指定しない場合は、*ID-3* に 1 つ以上の日付フィールドを指定することができ、日付フォーマットに関する制約事項はありません。
- フォーマット 3 では、*ID-2* 内の対応する基本項目だけを日付フィールドにすることができます。 これらの日付フィールドのフォーマットに関する制約事項はありません。
- 年末尾型日付フィールドを ADD ステートメントに指定できるのは、*ID-1* としてのみ、および加算の結果が非日付データである場合だけです。

1 つ以上の日付フィールドに関連する ADD ステートメントの結果を判別するには、次の 2 つのステップがあります。

1. 加算: 237 ページの『日付フィールドが関係する加算』のようにして、加算の結果を判別します。
2. 保管: その結果が受け取りフィールドにどのように保管されるかを判別します。 (フォーマット 1 と 3 では、受け取りフィールドは *ID-2* です。フォーマット 3 では、 受け取りフィールドは GIVING *ID-3* です。) 詳しくは、237 ページの『日付フィールドに関連する算術演算結果の保管』を参照してください。

リテラル
これは、数字リテラルでなければなりません。

数字データ項目または数字リテラルを指定できる場所であればどこでも、 浮動小数点データ項目および浮動小数点リテラルも使用できます。

ARITH(COMPAT) コンパイラー・オプションが有効な場合は、オペランドの合成が最大 30 桁になります。 ARITH(EXTEND) コンパイラー・オプションが有効な場合は、オペランドの合成が最大 31 桁になります。 詳しくは、267 ページの『算術ステートメントのオペランド』、および「*COBOL for Linux on x86* プログラミング・ガイド」の『付録 *A.* 中間結果および算術精度』の算術計算中間結果についての説明を参照してください。

ROUNDED 句

フォーマット 1、2、および 3 については、265 ページの『ROUNDED 句』を参照してください。

SIZE ERROR 句

フォーマット 1、2、および 3 については、266 ページの『SIZE ERROR 句』を参照してください。

CORRESPONDING 句 (フォーマット 3)

264 ページの『CORRESPONDING 句』を参照してください。

END-ADD 句

この明示的範囲終了符号は、ADD ステートメントの 範囲を区切るために使用されます。 END-ADD を使用すると、 条件付き ADD ステートメントを別の条件ステートメントにネストさせることができます。 END-ADD は、命令の ADD ステートメントと共に使用することもできます。

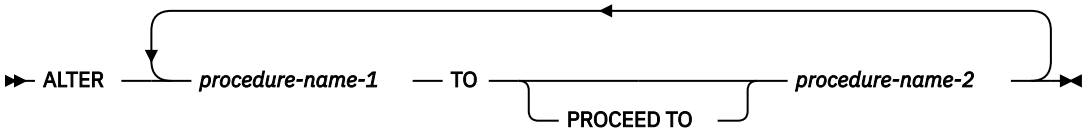
詳しくは、263 ページの『範囲区切りステートメント』を参照してください。

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。

ALTER ステートメントは、非構造化プログラミングの使用を助長します。 EVALUATE ステートメントには ALTER ステートメントと同じ機能がありますが、 プログラムの構造化に役立ちます。

フォーマット



ALTER ステートメントは、プロシージャー名-*1* によって指定された段落中の GO TO ステートメントを 修正します。 この修正された GO TO ステートメントが以降に実行されると (場合によっては複数)、制御はプロシージャー名-*2* に移されます。

プロシージャー名-*1*
文が 1 つだけ、つまり、DEPENDING ON 句を指定しない GO TO ステートメントが入った PROCEDURE DIVISION でなければなりません。

プロシージャー名-*2*
PROCEDURE DIVISION のセクションまたは段落を指定しなければなりません。

ALTER ステートメントの実行前に、プロシージャー名-*1* で指定した段落に制御が達すると、GO TO ステートメントは、その GO TO ステートメント中に指定されている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は、次に、プロシージャー名-*1* に指定された段落に制御が達するとき、GO TO ステートメントはプロシージャー名-*2* に指定された段落に制御を移します。

ALTER ステートメントは、プログラム・スイッチとして動作します。例えば、 初期設定時にはある一連のステートメントを実行し、 大量のファイル処理の実行中は別の一連のステートメントを実行するといったことが可能です。

INITIAL 属性を持つプログラム内で修正された GO TO ステートメントは、 プログラムの実行が開始されるたびに初期状態に戻されます。

ALTER ステートメントを、RECURSIVE 属性が指定されたプログラムでは使用しないでください。

セグメント化に関する考慮事項

独立セグメント内にコーディングされた GO TO ステートメントは、優先順位番号が異なるセグメントの ALTER ステートメントによって参照されないようにしなければなりません。 ALTER ステートメントの他の使用方法はすべて有効であり、ALTER が参照する GO TO ステートメントが固定セグメント中にある場合でも実行されます。

異なる優先順位番号 を持った独立セグメントにより ALTER で変更された GO TO を含む別の独立セグメントに制御が移されるとき、独立セグメント中にある変更された GO TO ステートメントは、 それ自体の初期状態に戻ります。

このような制御の移動が起こりうるのは、次の理由からです。

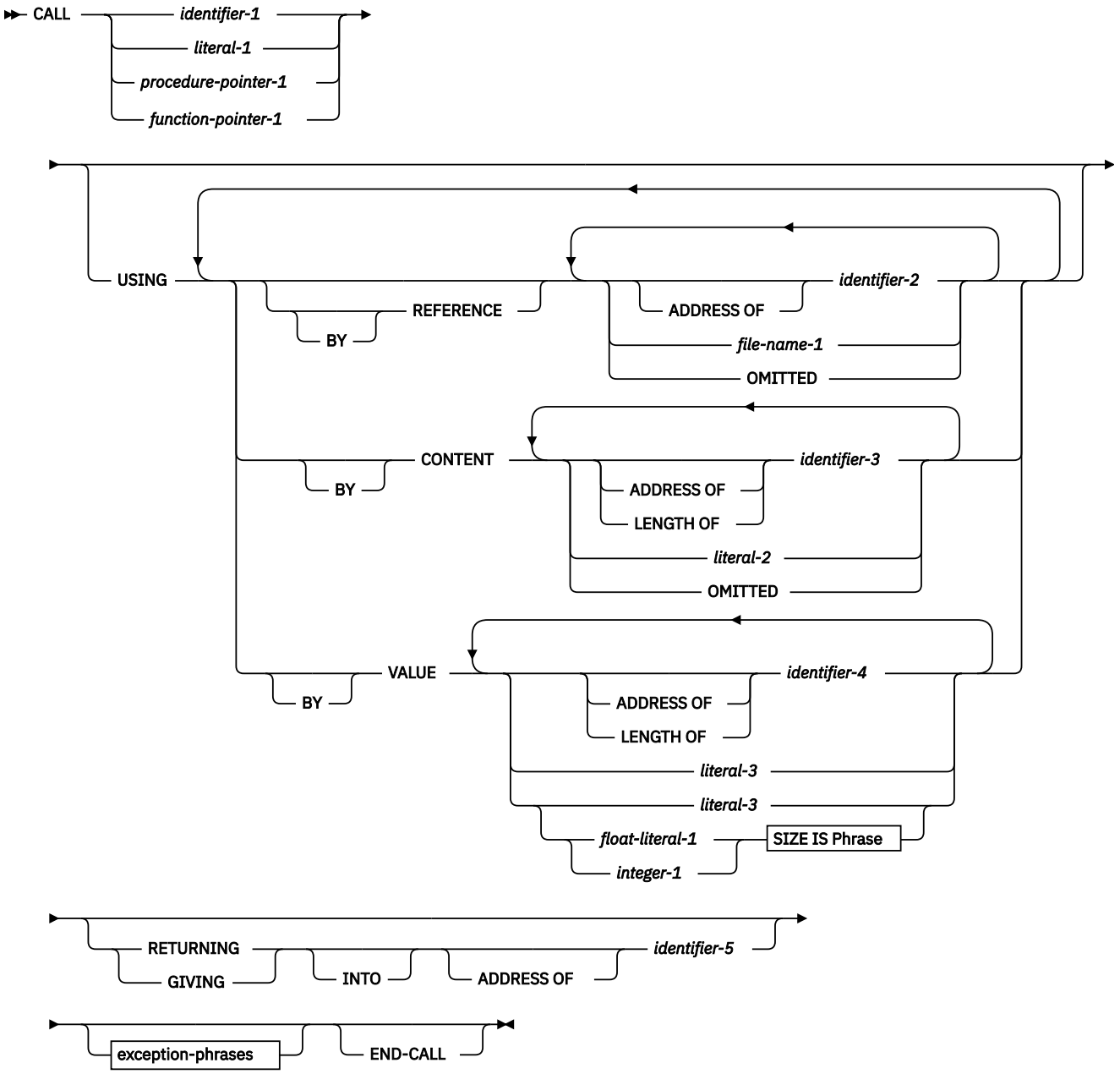
- 以前のステートメントによる影響。
- PERFORM ステートメントまたは GO TO ステートメントによる明示的な制御の移動。
- INPUT 句または OUTPUT 句が指定されたソート・ステートメントまたはマージ・ステートメント。

CALL ステートメント

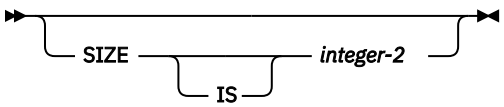
CALL ステートメントは、あるオブジェクト・プログラムからその実行単位内の別のオブジェクト・プログラムに制御を移します。

CALL ステートメントが入ったプログラムは呼び出し側プログラムであり、 CALL ステートメント内で識別されるプログラムは、呼び出されるサブプログラムです。 呼び出されるプログラムに CALL ステートメントを入れることはできますが、直接的または間接的にそれ自体 を呼び出す CALL ステートメントを実行できるのは、RECURSIVE 節で定義されたプログラムだけです。

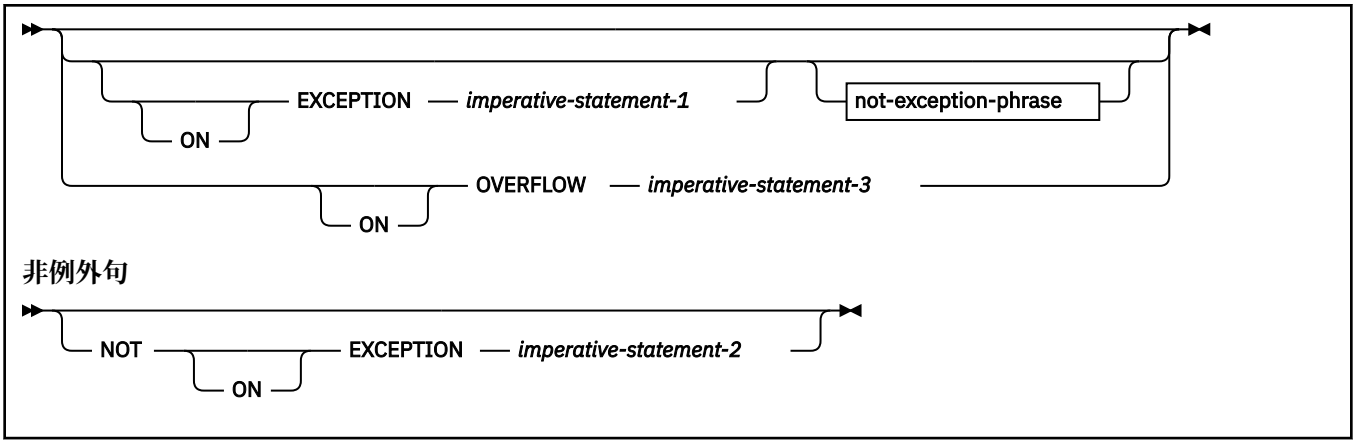
フォーマット



SIZE IS Phrase



例外句



ID-1*、リテラル-*1
リテラル-*1* は英数字リテラルでなければなりません。*ID-1* は、その値をプログラム名にできる USAGE DISPLAY を指定して記述された英数字、英字、または数字データ項目でなければなりません。

プログラム名の形成の規則は、PGMNAME コンパイラー・オプションによって異なります。詳しくは、84 ページの『PROGRAM-ID 段落』のプログラム名の説明を参照してください。また、の説明も参照してください。 *COBOL for Linux on x86* プログラミング・ガイド内の *PGMNAME* を参照してください。

ID-1 をウィンドウ表示日付フィールドにすることはできません。

使用上の注意: CALL ステートメントには、クラスの名前を指定しないでください。

プロシージャー・ポインター-*1*
USAGE IS PROCEDURE-POINTER で定義し、有効なプログラムの入り口点に設定する必要があります。そのようにしないと、CALL ステートメントの結果は未定義となります。

プログラムが COBOL によって取り消されたか、PL/I または C で解放されたか、またはアセンブラーによって削除された後は、 そのプログラムの入り口点に設定されていたプロシージャー・ポインターは、無効になります。

関数ポインター-*1*
USAGE IS FUNCTION-POINTER で定義し、有効な関数またはプログラムの入り口点に設定する必要があります。 そのようにしないと、CALL ステートメントの結果は未定義となります。

プログラムが COBOL でキャンセルされるか、PL/I または C で解放されるか、またはアセンブラーで削除されると、 その関数またはプログラムの入り口点に設定されていた関数ポインターは、すべて無効になります。

呼び出されるサブプログラムに入るときに、PROCEDURE DIVISION の最初から入る場合は、リテラル-*1* または *ID-1* の内容には、呼び出されるサブプログラムのプログラム名を指定しなければなりません。

呼び出されるサブプログラムに入る際に、ENTRY ステートメントから入るときには、リテラル-*1* または *ID-1* の内容は、呼び出されるサブプログラムの ENTRY ステートメント中に指定された名前と同じにしなければなりません。

USING 句

USING 句は、ターゲット・プログラムに渡される引数を指定します。

USING 句を CALL ステートメントに入れるのは、PROCEDURE DIVISION のヘッダー、または呼び出されるプログラムが実行される ENTRY ステートメント内に USING 句がある場合だけにしてください。各 USING 句内のオペランドの数は同じでなければなりません。

USING 句の詳細については、230 ページの『PROCEDURE DIVISION ヘッダー』を参照してください。

CALL ステートメントの USING 句でオペランドを指定する順序、および 呼び出されるサブプログラムの PROCEDURE DIVISION のヘッダーまたは ENTRY ステートメント内での対応する USING 句でオペランドを 指定する順序によって、呼び出し側プログラムと呼び出されるプログラムで使用されるオペランドの間の対応関係が決まります。この対応は、位置によるものです。

CALL ステートメントの USING 句で参照されるパラメーターの値は、 その CALL ステートメントが実行された時点で、 呼び出されるサブプログラムに対して使用可能になります。 呼び出されるプログラム中のデータ項目の記述は、 呼び出し側プログラム中の対応するデータ項目の記述と同じ文字位置の数で記述しなければなりません。

BY CONTENT 句、BY REFERENCE 句、および BY VALUE 句は、別の BY CONTENT 句、BY REFERENCE 句、または BY VALUE 句が現れるまで、それぞれの後に続くパラメーターに適用されます。また、BY CONTENT 句、BY REFERENCE 句、または BY VALUE 句を最初のパラメーターより前に指定しないと、BY REFERENCE 句が想定されます。

BY REFERENCE 句

パラメーターに対して BY REFERENCE 句が明示的または暗黙的に指定された場合、呼び出し側プログラム内の対応するデータ項目は、呼び出されるプログラムのデータ項目と同じストレージ域を占めます。

ID-2
DATA DIVISION 内の任意のレベルのデータ項目にできます。*ID-2* を関数 ID にすることはできません。

LINKAGE SECTION または FILE SECTION で定義する場合は、 CALL ステートメントを呼び出す前に、*ID-2* をアドレス可能にしておく必要があります。これは、`SET ADDRESS OF identifier-2 TO pointer` または `PROCEDURE/ENTRY USING` のいずれかをコーディングすることで行うことができます。

ファイル名-1
順次編成ファイルのファイル名。CALL ステートメントでのファイル名の使用について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『データの受け渡し』を参照してください。

ADDRESS OF *identifier-2*
ID-2 は、LINKAGE SECTION の中に定義されたレベル 01 またはレベル 77 の項目である必要があります。

OMITTED
引数がなにも渡されないことを示します。

BY CONTENT 句

パラメーターに対して BY CONTENT 句が明示的または暗黙的に指定された場合、 CALL ステートメントの USING 句で参照されたとき、呼び出されるプログラムはこのパラメーターの値を変更することはできません。しかし、 呼び出されるプログラムは、その PROCEDURE DIVISION のヘッダーにある対応するデータ名によって 参照されるデータ項目の値を変更することは可能です。呼び出されるプログラム内のパラメーターを変更しても、 呼び出し側プログラム内の対応する引数には影響ありません。

ID-3
DATA DIVISION 内の任意のレベルのデータ項目にできます。*ID-3* を関数 ID にすることはできません。

LINKAGE SECTION または FILE SECTION で定義されている場合は、 すでに CALL ステートメントを呼び出す前に、*ID-3* をアドレス可能にしておく必要があります。それには、 以下の句のいずれかをコーディングします。

- `SET ADDRESS OF` *ID-3* `TO pointer`
- `PROCEDURE DIVISION USING`
- `ENTRY USING`

リテラル-2
リテラル-3 は、下記のものになります。

- 英数字リテラル
- ブール・リテラル
- 形象定数 (ALL リテラル または NULL/NULLS を除く)
- DBCS リテラル

- 国別リテラル

LENGTH OF 特殊レジスター
LENGTH OF 特殊レジスターの詳細については、18 ページの『LENGTH OF』を参照してください。.

ADDRESS OF*ID-3*
ID-3 は、LINKAGE SECTION、WORKING-STORAGE SECTION、または LOCAL-STORAGE SECTION で定義された 66 または 88 を除いたレベルのデータ項目である必要があります。

OMITTED
引数がなにも渡されないことを示します。

英数字リテラルの場合、呼び出されるサブプログラムでは、パラメーターを PIC X(*n*) USAGE DISPLAY と記述するようにします。この場合、*n* は、リテラル内の文字の数です。

DBCS リテラルの場合、呼び出されるサブプログラムでは、USAGE DISPLAY-1 を暗黙的または明示的に指定して、パラメーターを PIC G(*n*) USAGE DISPLAY-1、または PIC N(*n*) と記述するようにします。この場合、*n* はリテラルの長さです。

国別リテラルの場合、呼び出されるサブプログラムでは、USAGE NATIONAL を暗黙的または明示的に指定して、 パラメーターを PIC N(*n*) と記述するようにします。 この場合、*n* はリテラルの長さです。

BY VALUE 句

BY VALUE 句は、別の BY REFERENCE または BY CONTENT 句によってオーバーライドされるまで、後続のすべての引数に適用されます。

ある引数に BY VALUE 句が指定されているか、 または暗黙指定されている場合は、送り出しデータ項目への参照ではなく、 その引数の値が渡されます。 呼び出されるプログラムは BY VALUE 引数に対応する仮パラメーターを修正できますが、 呼び出される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権を持っているため、 そのような変更は、この引数には適用されません。

BY VALUE 引数は主として非 COBOL プログラム (C など) との通信向けですが、COBOL 相互間の呼び出しにも使用できます。 その場合は、CALL USING 句および、PROCEDURE DIVISION USING 句内の対応する仮パラメーター内の両方の引数に、BY VALUE を指定または暗黙指定しなければなりません。

ID-4
DATA DIVISION 内の基本データ項目にしなければなりません。 これは、以下の項目のいずれかでなければなりません。

- バイナリー (USAGE BINARY、COMP、COMP-4、または COMP-5)
- 浮動小数点 (USAGE COMP-1 または COMP-2)
- 関数ポインター (USAGE FUNCTION-POINTER)
- ポインター (USAGE POINTER)
- プロシージャー・ポインター (USAGE PROCEDURE-POINTER)
- 1 つの 1 バイトの英数字文字 (PIC X や PIC A など)
- 1 つの国別文字 (PIC N)、国別カテゴリーの基本データ項目として記述されます。

以下の項目も、BY VALUE によって渡されます。

- USAGE DISPLAY の参照変更項目および長さ 1
- USAGE NATIONAL の参照変更項目および長さ 1
- SHIFT-IN および SHIFT-OUT 特殊レジスター
- LINAGE-COUNTER 特殊レジスター (USAGE BINARY の場合)

ADDRESS OF *ID-4*
ID-4 は、LINKAGE SECTION、WORKING-STORAGE SECTION、または LOCAL-STORAGE SECTION で定義された 66 または 88 を除いたレベルのデータ項目である必要があります。

LENGTH OF 特殊レジスター
BY VALUE で渡される LENGTH OF 特殊レジスターは、PIC 9(9) バイナリーとして扱われます。
LENGTH OF 特殊レジスターの詳細については、18 ページの『LENGTH OF』を参照してください。.

リテラル-*3*
以下のいずれかのタイプでなければなりません。

- ブール・リテラル
- 数字リテラル
- 形象定数 ZERO
- 1 文字の英数字リテラル
- 1 文字の国別リテラル
- シンボリック文字
- 1 バイトの形象定数
  - SPACE
  - QUOTE
  - HIGH-VALUE
  - LOW-VALUE

ZERO は数値として扱われます。フルワード・バイナリーの 0 が渡されます。

ID-3 は、固定点数字リテラルである場合、9 以下の桁の精度でなければなりません。 その場合は、フルワードのバイナリー表記のリテラル値が渡されます。

リテラル-*3* が浮動小数点数字リテラルである場合は、8 バイトの内部浮動小数点 (COMP-2) 表記の値が渡されます。

リテラル-*3* は DBCS リテラルであってはいけません。

浮動小数点リテラル-*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、バイナリー値として渡されます。 整数-*2* を指定しないと、整数-*1* が 4 バイトのバイナリー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。

RETURNING 句

RETURNING と GIVING は同じ意味です。

ADDRESS OF *ID-5*
ID-5 は、LINKAGE SECTION の中に定義されたレベル 01 またはレベル 77 の項目である必要があります。

ID-5
DATA DIVISION に定義された任意のデータ項目を指定できる RETURNING データ項目です。呼び出されるプログラムの戻り値は暗黙的に *ID-5* に保管されます。

COBOL、C、または C のリンケージ規約を使用するその他のプログラミング言語で作成した関数への呼び出しとして、 RETURNING 句を指定することができます。 COBOL サブプログラムへの CALL に RETURNING 句を指定する場合は、次のようになります。

- 呼び出されるサブプログラムでは、その PROCEDURE DIVISION のヘッダーに RETURNING 句を指定していなければなりません。
- *ID-5* およびそのターゲット・プログラム内での対応する PROCEDURE DIVISION RETURNING ID には、同じ PICTURE、USAGE、SIGN、SYNCHRONIZE、JUSTIFIED、 および BLANK WHEN ZERO 句が指定されていなければなりません (ただし、DECIMAL POINT IS COMMA 節により、PICTURE 節の通貨記号は違う指定が可能であり、 またピリオドとコンマは交換可能という点は除きます)。

ターゲットが戻されるときは、*ID-6* が INDEX、POINTER、FUNCTION-POINTER または PROCEDURE-POINTER の場合、SET ステートメントの規則を使用して、その戻り値が *ID-5* に割り当てられま す。 *ID-5* がその他の USAGE の場合、MOVE ステートメントの規則が使用されます。

CALL... RETURNING データ項目は、出力専用パラメーターです。 呼び出されるプログラムに入った時点で、PROCEDURE DIVISION RETURNING データ項目の初期状態の値は未定義であり予測不可能です。 呼び出されるプログラムの PROCEDURE DIVISION RETURNING データ項目を初期化してから、 値を参照するようにしてください。 呼び出されるプログラムから戻る時点で呼び出し側プログラムに戻される値は、PROCEDURE DIVISION RETURNING の最終的な値です。

例外またはオーバーフローが起きても、*ID-5* は変更されません。 *ID-5* は、参照変更にすることはできません。

RETURN-CODE 特殊レジスターは、RETURNING 句が入った CALL ステートメントを実行しても設定されません。

CALL ステートメントの RETURNING 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。

ON EXCEPTION 句

例外条件は、呼び出されたサブプログラムを使用可能にできないときに起こります。 そのときは、次の 2 つの処置のどちらかが行われます。

1. ON EXCEPTION 句が指定されている場合は、制御は 命令ステートメント-*1* に移ります。 次いで、命令ステートメント-*1* に指定された各ステートメントの規則に従って、 実行が継続されます。 明示的な制御の移動を起こすプロシージャーのブランチ・ステートメントや条件ステートメントが 実行された場合は、制御はそのステートメントの規則に従って移されます。 それ以外の場合は、命令ステートメント-*1* の実行が完了すると、制御は CALL ステートメントの終わりに移され、NOT ON EXCEPTION 句が指定されている場合はそれが無視されます。
2. ON EXCEPTION 句が CALL ステートメント内に指定されていないと、NOT ON EXCEPTION 句が指定されている場合はそれが無視されます。

NOT ON EXCEPTION 句

例外条件が起こらなければ (呼び出されたサブプログラムを使用可能にできれば)、制御は呼び出されたプログラムに移されます。 呼び出されるプログラムから制御が戻ると、次のものに制御が移されます。

- 命令ステートメント-*2*。ただし、NOT ON EXCEPTION 句が指定されている場合。
- その他の場合は、CALL ステートメントの終わり (ただし ON EXCEPTION 句が指定されていれば、それは無視されます)。

制御が命令ステートメント-*2* に移された場合、命令ステートメント-*2* に指定された各ステートメントの規則に従って、実行が継続されます。 明示的な制御の移動を起こすプロシージャーのブランチ・ステートメントや条件ステートメントが 実行された場合は、制御はそのステートメントの規則に従って移されます。 それ以外の場合は、命令ステートメント-*2* の実行が完了すると、 制御は CALL ステートメントの終わりに移されます。

ON OVERFLOW 句

ON OVERFLOW 句を使うと、ON EXCEPTION 句と同じ結果が得られます。

END-CALL 句

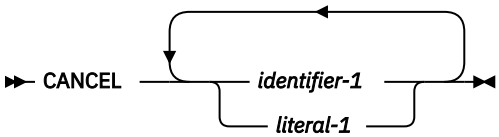
この明示的範囲終了符号は、CALL ステートメントの有効範囲を区切るために使用されます。 END-CALL を使用すると、 条件付き CALL ステートメントを別の条件付きステートメントにネストすることができます。 END-CALL は命令 CALL ステートメントでも使用できます。

詳しくは、263 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。

CANCEL ステートメント

CANCEL ステートメントは、参照されるサブプログラムが次に呼び出されるとき、初期状態になるようにします。

フォーマット



***ID-1*、リテラル-1**

リテラル-1 は英数字リテラルでなければなりません。*ID-1* は、その値をプログラム名にできる英数字、英字、またはゾーン 10 進数データ項目でなければなりません。 プログラム名の形成の規則は、PGMNAME コンパイラー・オプションによって異なります。詳しくは、84 ページの『PROGRAM-ID 段落』のプログラム名の説明、および「*COBOL for Linux on x86* プログラミング・ガイド」の『*PGMNAME*』の説明を参照してください。

ID-1 をウィンドウ表示日付フィールドにすることはできません。

リテラル-1 または *ID-1* の内容は、 関連する CALL ステートメント内に指定される ID のリテラルまたは内容と同じでなければなりません。

CANCEL ステートメントには、クラスまたはメソッドの名前を指定しないでください。

呼び出されたサブプログラムに対して CANCEL ステートメントを実行した後、 そのサブプログラムは、当該プログラムに対して論理的関連を失います。 サブプログラムによって記述された外部データ・レコード中のデータ項目の内容は、 サブプログラムが取り消されても変更されることはありません。 実行単位内でいずれかのプログラムが、同じサブプログラムを指名して CALL ステートメントを実行した場合、 そのサブプログラムは、その初期状態で実行されます。

CANCEL ステートメントが実行されるとき、その CANCEL ステートメントに参照されるプログラムに入っている他のすべてのプログラムも、 取り消されます。 個別にコンパイルされたプログラム内のプログラムが 現れる順番とは逆の順序で、それらの各包含プログラムに対して、 有効な CANCEL ステートメントが実行された場合も、その結果は同じになります。

CANCEL ステートメントは、明示的な CANCEL ステートメントに指定されたプログラム内の、 内部ファイル結合子と関連するすべてのオープン・ファイルをクローズします。 それらのファイルと関連する USE プロシージャーは実行されません。

以下のいずれかの方法で、呼び出されたサブプログラムを取り消すことができます。

- CANCEL ステートメントのオペランドとしてそのサブプログラムを参照する
- そのサブプログラムがメンバーである実行単位を終了する
- サブプログラムが初期属性を持つ場合にその呼び出されたサブプログラム内で EXIT PROGRAM ステートメント または GOBACK ステートメントを実行する

次のどちらかのプログラムを指定した CANCEL ステートメントが実行されたときは、 何の処置も取られません。

- この実行単位内で、別の IBM COBOL プログラムによって動的に呼び出されていないもの。
- 呼び出されたが、それ以降取り消されたプログラム。

マルチスレッド環境の場合、プログラムは、スレッド上のアクティブなプログラムを 指定して CANCEL ステートメントを実行することはできません。 指定するプログラムは、完全に非アクティブでなければなりません。

呼び出されるサブプログラムには、CANCEL ステートメントを入れることができます。 ただし、呼び出されるサブプログラムは、 呼び出し側プログラム自体を直接または間接に取り消す CANCEL ステートメントを実行することはできません。 または、呼び出しの階層内でそのプログラム自体より上位にある他のプロ

グラムを取り消す CANCEL ステートメントを 実行することもできません。 それを行うと、実行単位が終了します。

CANCEL ステートメント内で指名されるプログラムは、呼び出されて EXIT PROGRAM ステートメント または GOBACK ステートメントを実行しているプログラムでなければなりません。

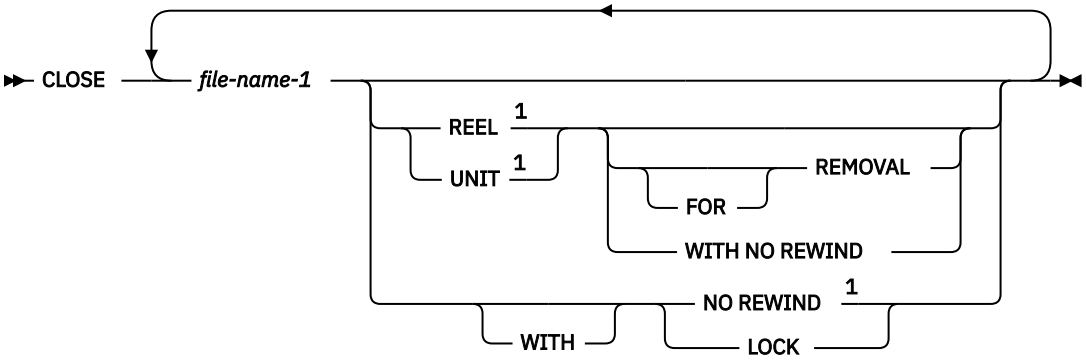
ただし、CANCEL ステートメントを実行するプログラムが呼び出し階層において取り消すプログラムより上位または等しいレベルであれば、 取り消し側プログラムは呼び出していないプログラムを取り消すことができます。 次に例を示します。

```
A calls B and B calls C    (When A receives control, it can cancel C.)
A calls B and A calls C    (When C receives control, it can cancel B.)
```

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します。

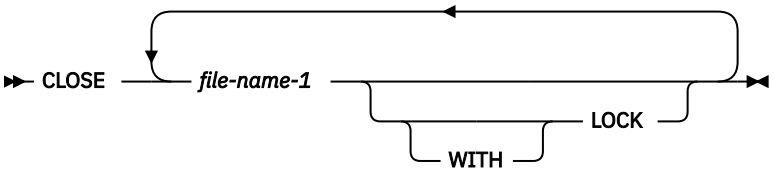
フォーマット 1: 順次ファイルの CLOSE ステートメント



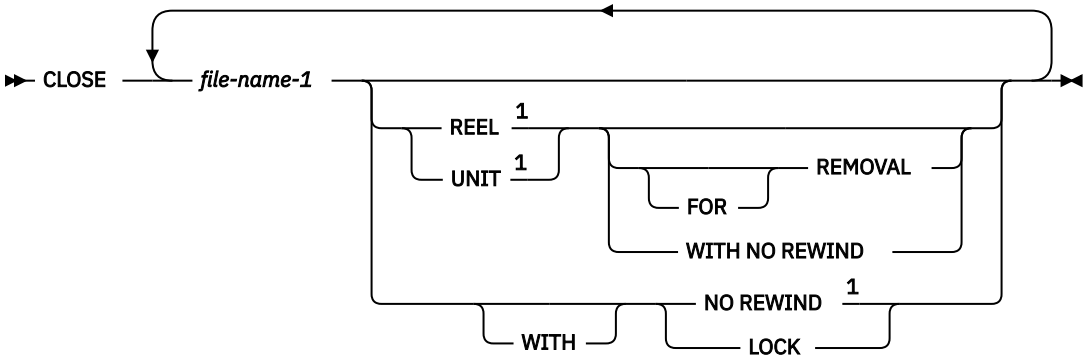
注:

[1] UNIT、REEL、および NO REWIND 句はコメントとして扱われます。それでも、ファイル状況は 07 に設定されて、非リール / ユニット・メディアの CLOSE が正常に完了したことを示します。

フォーマット 2: 索引付きおよび相対ファイルの CLOSE ステートメント



フォーマット 3: 行順次ファイルの CLOSE ステートメント



注:

[1] UNIT、REEL、および NO REWIND 句はコメントとして扱われます。それでも、ファイル状況は 07 に設定されて、非リール / ユニット・メディアの CLOSE が正常に完了したことを示します。

ファイル名-*1*
CLOSE ステートメントの操作対象となるファイルを指定します。 複数のファイル名を指定する場合、それらのファイルは同じ編成や同じアクセス方式である必要はありません。 ファイル名-*1* はソート・ファイルまたはマージ・ファイルにはできません。

REEL および UNIT
REEL および UNIT は構文チェックされますが、プログラムの実行には何も影響しません。

WITH NO REWIND および FOR REMOVAL
WITH NO REWIND および FOR REMOVAL は構文チェックされますが、プログラムの実行には何も影響しません。

CLOSE ステートメントは、オープン・モードのファイルに対してのみ実行することができます。 CLOSE ステートメント (フォーマット 1 を使用する場合は、 REEL/UNIT 句を使用しないもの) が正常に実行すると、次のようになります。

- ファイル名に関連付けられたレコード域は、使用不能になります。 CLOSE ステートメントの実行が失敗した場合は、 レコード・データが使用可能かどうかはわかりません。
- ファイルに対して他の入出力ステートメントを実行する前、およびそのファイルに関連したレコード記述項目にデータを移動する前に、ファイルに対して OPEN ステートメントを実行しなければなりません。
- クローズしたファイルのファイル結合子によって保持されていたレコード・ロックおよびファイル・ロックは、 すべて解放されます。

ファイル制御項目内に FILE STATUS 節が指定されている場合は、関連するファイル状況キーが、CLOSE ステートメントの実行時に更新されます。

ファイルがオープン状態にあり、CLOSE ステートメントの実行が正常に実行しない場合は、 そのファイルに対して EXCEPTION/ERROR プロシージャーが (指定した場合) 実行されます。

ファイル・タイプへの CLOSE ステートメントの効果

ファイルのファイル制御項目で SELECT OPTIONAL 節が指定され、実行時にファイルが使用可能でない場合、標準のファイル終了処理は実行されません。

ファイルは、以下のタイプに分けられます。

リール・ファイル/ユニット以外
その入力メディアまたは出力メディアに対して、REWIND、REEL、 および UNIT の指定が意味を持たないようなファイル。 すべてのファイルは、非リール/ユニット・ファイル・タイプです。

順次単一ボリューム
全体が 1 つのファイルに入っている順次ファイル。 複数のファイルをこのボリュームに入れることができます。 すべてのファイルは、単一ボリュームです。

順次マルチボリューム
複数のボリュームに入っている順次ファイル。 ボリュームの概念は、意味がありません。

CLOSE ステートメント句の許可される組み合わせについては、以下のテーブルを参照してください。

- 順次ファイルの場合: 順次ファイルおよび CLOSE ステートメント句
- 索引付きおよび相対ファイルの場合: 293 ページの表 45
- 行順次ファイルの場合: 293 ページの表 46

各キー文字の意味は、293 ページの表 47 に示されています。

表 *44.* **順次ファイルおよび *CLOSE* ステートメント句**

CLOSE ステートメント句	**リール/ユニット以外**	**順次単一ボリューム**	**順次マルチボリューム**
CLOSE	C	C、G	A、C、G
CLOSE WITH LOCK	C、E	C、E、G	A、C、E、G

表 *45.* **索引付きファイル・タイプと相対ファイル・タイプおよび *CLOSE* ステートメント句**

CLOSE ステートメント句	**アクション**
CLOSE	C
CLOSE WITH LOCK	C、E

表 *46.* **行順次ファイル・タイプおよび *CLOSE* ステートメント句**

CLOSE ステートメント句	**アクション**
CLOSE	C
CLOSE WITH LOCK	C、E

表 *47.* **順次ファイル・タイプのキー文字の意味**

キー	**取られる処置**
A	**前のボリュームは影響を受けない** **入力ファイルおよび入出力ファイル**: 標準のボリューム切り替え処理が、 前のすべてのボリュームに対して行われる (先行する CLOSE REEL/UNIT ステートメントにより 制御されるものを除く)。 後続のボリュームはいずれも処理されない。 **出力ファイル**: 標準のボリューム切り替え処理が、 前のすべてのボリュームに対して行われる (先行する CLOSE REEL/UNIT ステートメントにより制御されるものを除く)。

表 47. **順次ファイル・タイプのキー文字の意味** (続き)	
キー	**取られる処置**
C	**ファイルをクローズする** **入力ファイルおよび入出力ファイル**: ファイルがその終了位置にあり、 ラベル・レコードが指定されていれば、標準の終了ラベル・プロシージャーが実行される。 ついで標準のシステム・クローズ・プロシージャーが行われる。 ファイルがその終了位置にあっても、ラベル・レコードが指定されていない場合は、 ラベル処理は行われないが、標準のシステム・クローズ・プロシージャーが行われる。 ファイルがその終了位置にない場合、標準のシステム・クローズ・プロシージャーが実行されるが、 終了のラベル処理は行われない。 **出力ファイル**: ラベル・レコードが指定されていると、 標準の終了ラベル・プロシージャーが実行される。 ついで標準のシステム・クローズ・プロシージャーが行われる。 ラベル・レコードが指定されていないと、終了ラベル・プロシージャーは実行されないが、 標準のシステム・クローズ・プロシージャーは実行される。
E	**ファイルをロックする**: コンパイラーは、 オブジェクト・プログラムの実行中にこのファイルが再びオープンできないようにする。 ファイルが磁気テープ装置である場合は、巻き戻しおよびアンロードが行われる。
G	**巻き戻す**: 現在のボリュームは物理的な先頭に位置付けられる。

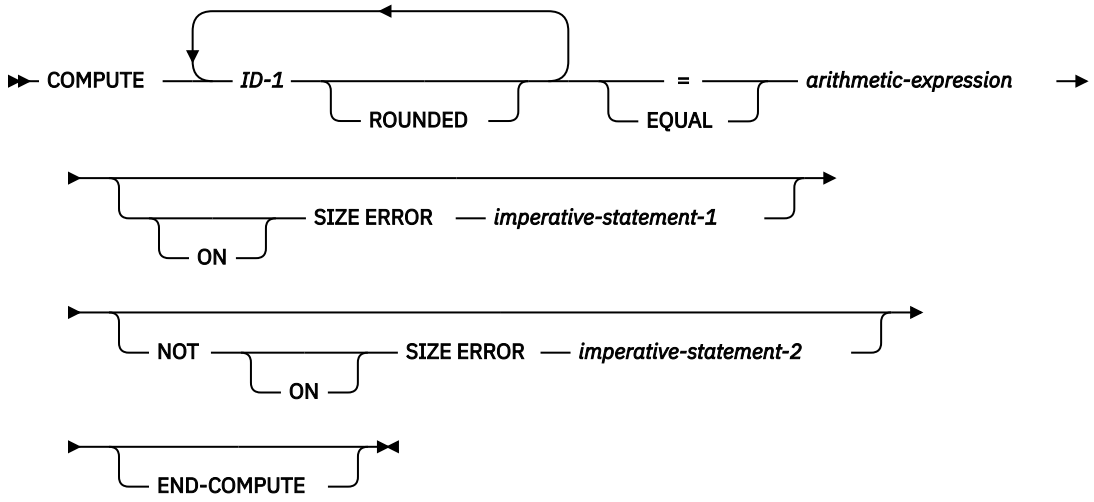
COMPUTE ステートメント

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 ステートメントでは、算術演算を組み合わせることができますが、 その際、ADD、SUBTRACT、MULTIPLY、および DIVIDE ステートメントの規則による、 データ項目受け取りに関する制限はありません。

算術演算を組み合わせる場合、個々の算術ステートメントを列挙して記述するよりも、 COMPUTE ステートメントを使用するほうが効率的です。

フォーマット



ID-1

基本数字項目または 基本数字編集項目を 指定する必要があります。

基本浮動小数点データ項目を指定することもできます。

ID-1 または算術式 の結果 (あるいはその両方) が日付フィールドである場合、結果が *ID-1* にどのように保管されるかについては、237 ページの『日付フィールドに関連する算術演算結果の保管』を参照してください。 *ID-1* として年末尾型日付フィールドを指定した場合、算術式 の結果は非日付データにならなければなりません。

算術式

234 ページの『算術式』に定義されている任意の算術式にできます。

COMPUTE ステートメントが実行されると、算術式 の値が計算され、*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。

1 つの ID、数字関数、またはリテラルから構成される算術式を使用すると、*ID-1* によって参照されるデータ項目 (単数または複数) の値をその ID、関数、またはリテラルの値に等しく設定することができます。

算術式の中に年末尾型日付フィールドを指定することはできません。

ROUNDED 句

ROUNDED 句の説明については、265 ページの『ROUNDED 句』を参照してください。

SIZE ERROR 句

SIZE ERROR 句の説明については、266 ページの『SIZE ERROR 句』を参照してください。

END-COMPUTE 句

この明示的範囲終了符号は、COMPUTE ステートメントの範囲を区切るために使用されます。 END-COMPUTE 句を使用することによって、 条件付き COMPUTE 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-COMPUTE 句は、命令 COMPUTE ステートメントと共に使用することもできます。

詳しくは、263 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

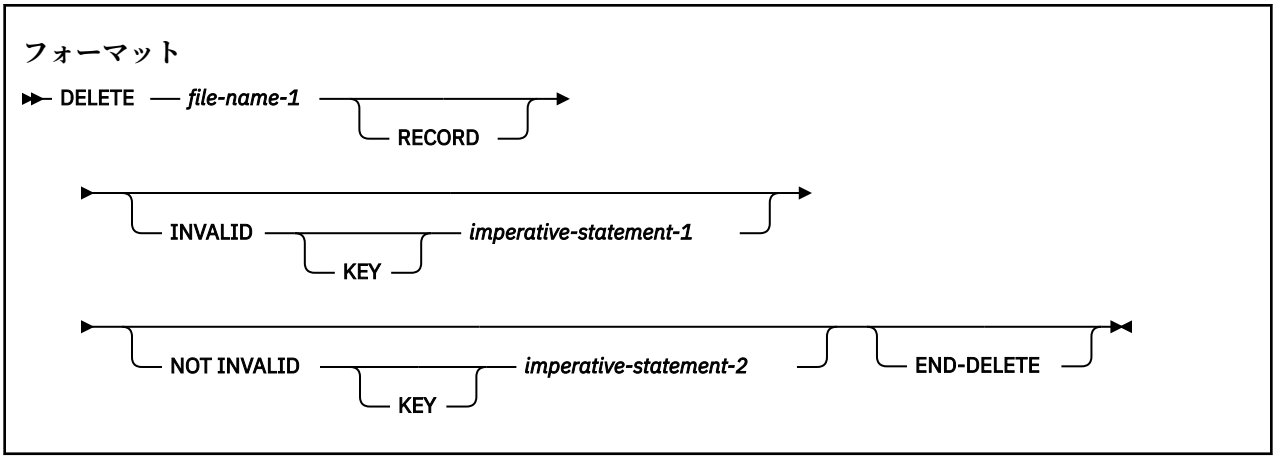
CONTINUE ステートメントは、オペレーションのないステートメントです。 CONTINUE ステートメントは、実行可能な命令が存在しないことを示します。

フォーマット

DELETE ステートメント

DELETE ステートメントは、索引付きファイルまたは相対ファイルからレコードを削除します。 索引付きファイルの場合、削除されたそのレコードのキーは、 新たに追加されるレコードで再使用することができます。 相対ファイルの場合、削除されたそのレコードのスペースは、 同じ RELATIVE KEY 値を持つ新しいレコードで使用可能です。

DELETE ステートメントを実行するときには、その対象となるファイルは、I-O モードでオープンされている必要があります。



ファイル名-*1*
DATA DIVISION の FD 項目において定義されていなければなりません。 また索引付きファイルまたは相対ファイルの名前でなければなりません。

DELETE ステートメントが正しく実行された後は、 そのレコードはファイルから削除され、以降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-*1* に関連するレコード域の内容には影響しません。 また、ファイル名-*1* に関連する RECORD 節の DEPENDING ON 句に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にも影響しません。

ファイル制御項目内に FILE STATUS 節が指定されている場合は、 関連するファイル状況キーが、DELETE ステートメントの実行時に更新されます。

ファイル位置標識は、DELETE ステートメントの実行によって影響を受けることはありません。

レコード・ロックが有効な場合は、以下の処理が行われます。

- 単一レコード・ロックが、ファイル名-*1* に関連付けられているファイル結合子に対して指定された場合:
  - DELETE ステートメントの実行が正常に完了したときに、削除されたレコードに対して当該ファイル結合子が保持するロックが解放されます。
  - DELETE ステートメントの実行が開始されたときに、別のレコードに対して当該ファイル結合子が保持するロックが解放されます。
- ファイル名-*1* に関連付けられているファイル結合子に対して複数のレコード・ロックが指定された場合、DELETE ステートメントの実行が正常に完了したときに、削除されたレコードに対して保持されているロックがすべて解放されます。

順次アクセス・モード

順次アクセス・モードのファイルの場合、前回の入出力ステートメントは、正しく実行された READ ステートメントである必要があります。 DELETE ステートメントが実行されると、システムは READ ステートメントによって取り出されたレコードを削除します。

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。 EXCEPTION/ERROR プロシージャーを指定することはできます。

ランダムまたは動的アクセス・モード

ランダム・アクセス・モードまたは動的アクセス・モードでは、 DELETE ステートメント実行の結果は、ファイル編成 (索引付きファイルであるか相対ファイルであるか) によって異なります。

DELETE ステートメントが実行されると、システムは、 索引ファイルの基本 RECORD KEY データ項目の内容によって示されるレコード、 または相対ファイルの RELATIVE KEY データ項目によって示されるレコードを除去します。 ファイルにその種のレコードがない場合は、無効キー条件が存在しています (274 ページの『無効キー条件』を参照してください。.)

INVALID KEY 句

INVALID KEY 句および該当する EXCEPTION/ERROR プロシージャーは、両方とも省略することができます。

NOT INVALID KEY 句の指定がある DELETE ステートメントが正しく実行された後、 制御は、その句と関連付けられた命令ステートメントに移動します。

END-DELETE 句

この明示的範囲終了符号は、DELETE ステートメントの範囲を区切るために使用されます。 END-DELETE 句を使用することによって条件的な DELETE ステートメントを他の条件的なステートメントの中にネストすることができます。 END-DELETE 句は、命令の DELETE ステートメントと共に使用することもできます。

詳しくは、263 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。.

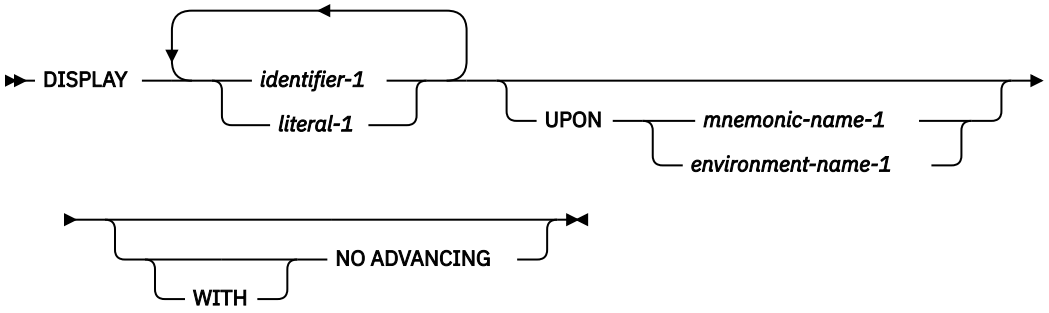
DISPLAY ステートメント

DISPLAY ステートメントは、各オペランドの内容を出力装置に転送します。 その内容はオペランドのリストであり、左から右の順に出力装置上に表示されます。

ターゲット・ファイルは、COBOL 環境名 (CONSOLE、SYSIN、SYSIPT、SYSOUT、SYSLIST、SYSLST、SYSPUNCH、および SYSPCH) を確認することによって判別します。 環境変数が COBOL 環境名に対応して定義された場合は、環境変数の値は システム・ファイル ID として使用されます。 COBOL 環境名に対応する環境変数が設定されていない場合は、 SYSOUT、SYSLIST、 または SYSLST 上の DISPLAY はシステム論理出力装置 (stdout) になります。

SYSPUNCH および SYSPCH では、対応する環境変数が有効な宛先を示すように設定されていないと、DISPLAY ステートメントが失敗します。 環境変数について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『例: 環境変数の設定とアクセス』を参照してください。

フォーマット



ID-1

ID-1 は表示されるデータを参照します。 *ID-1* は、USAGE が PROCEDURE-POINTER、FUNCTION-POINTER、または INDEX の項目以外のすべてのデータ項目を参照できます。 *ID-1* を索引名にすることはできません。

ID-1 が 2 進数、内部 10 進数、または内部浮動小数点のデータ項目の場合は、 *ID-1* は以下のように外部フォーマットに自動的に変換されます。

- 2 進数項目および内部 10 進数項目は、 ゾーン 10 進数に変換されます。 負符号の付いた値では、最下位桁に符号が上重ねされます。
- 内部浮動小数点数は外部浮動小数点数に変換され、以下のように表示されます。
  - COMP-1 項目は、-.9(8)E-99 という外部浮動小数点 PICTURE 節を持つものとして表示されます。
  - COMP-2 項目は、-.9(17)E-99 という外部浮動小数点 PICTURE 節を持つものとして表示されます。

USAGE POINTER を指定して定義されたデータ項目は、PIC 9(10) という暗黙的な PICTURE 節を持つゾーン 10 進数に変換されます。

USAGE NATIONAL を指定して記述されたデータ項目は、現行ロケールに関連付けされた コード・ページに変換されます。

その他のデータ・カテゴリーは変換が不要です。

日付フィールドは、DISPLAY ステートメントで指定されたときは、非日付データとして扱われます。すなわち、DATE FORMAT は無視され、データ項目の内容はそのまま出力装置に転送されます。

明示的または暗黙的に USAGE DISPLAY-1 として定義された DBCS データ項目は、出力装置の送り出しフィールドに転送されます。

DBCS と非 DBCS の両方のオペランドを単一の DISPLAY ステートメントに指定することができます。

リテラル-1
任意のリテラル、または 13 ページの『形象定数』に示す任意の形象定数にすることができます。 形象定数を指定した場合、その形象定数の 1 回のオカレンスだけが表示されます。

UPON
環境名-*1* または簡略名-*1* に関連した環境名を出力装置に関連付ける必要があります。 91 ページの『SPECIAL-NAMES 段落』を参照してください。

UPON 句が省略されている場合、システム論理出力装置が想定されます。 DISPLAY ステートメントにおける有効な環境名のリストは、 94 ページの表 5 にあります。

WITH NO ADVANCING
これが指定されると、出力装置の文字位置は、最後のオペランドが表示された後、 いかなる場合も変更されることはありません。 出力装置が特定の文字位置に位置決めが可能な場合、表示された最終オペランドの最終文字 に続く文字位置に位置づけされます。 出力装置が特定の文字位置に位置決めができない場合、 垂直位置のみ (可能な場合) に影響します。 これは上重ね印刷の原因になります。

WITH NO ADVANCING 句の指定がない場合、最後のオペランドが出力装置に転送された後、 出力装置の文字位置は装置の次の行の左端位置にリセットされます。

DISPLAY ステートメントは、送り出しフィールドのデータを出力装置に転送します。 送り出しフィールドのサイズは、リストされた全オペランドのバイト数の合計です。 出力装置が転送されるデータ項目と同じサイズのデータを受け取ることができれば、 データ項目が転送されます。 出力装置が転送されるデータ項目と同じサイズのデータを受け取ることができなければ、 次のどちらかが適用されます。

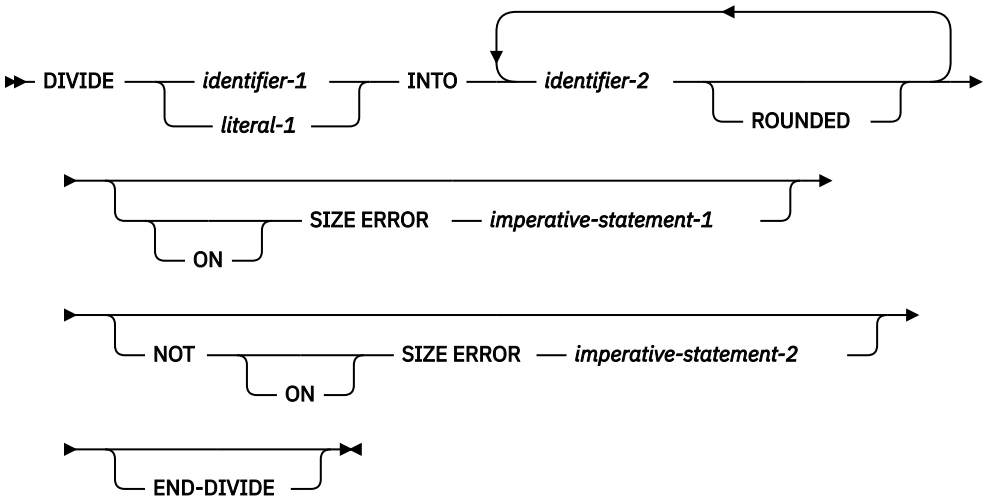
- 合計数が装置の最大文字数より小さければ、残りの右側の文字位置にスペースが埋め込まれます。
- 合計数が装置の最大文字数を超えていれば、 すべてのオペランドを表示するのに必要なだけの複数のレコードが書き出されます。 1 つのレコードの終わりに達すると、印刷中かまたは表示中のオペランドは、次のレコードに継続されます。

DBCS オペランドを複数のレコードに分割する必要がある場合は、2 バイトの境界でのみ分割されます。

DIVIDE ステートメント

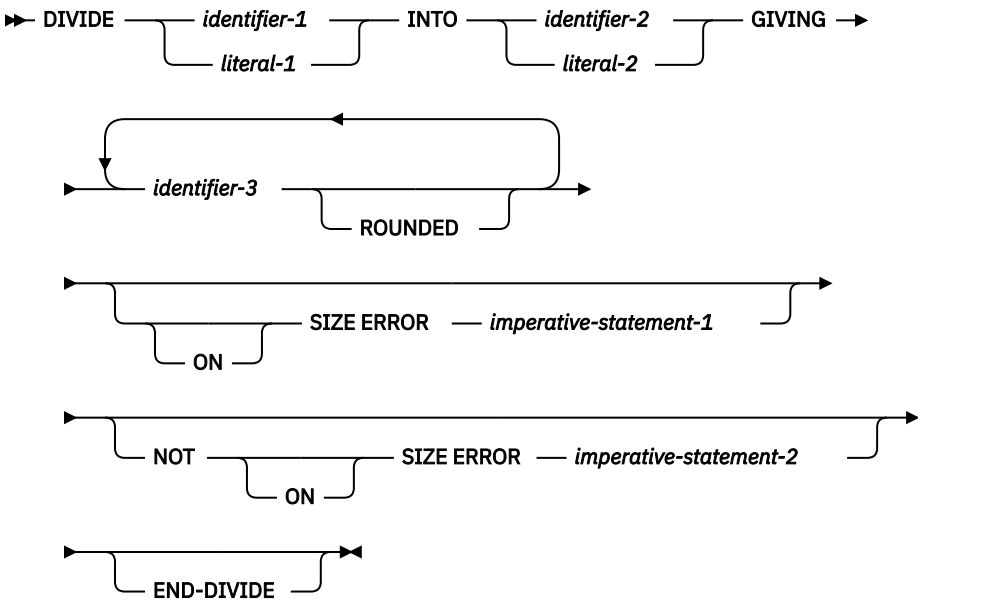
DIVIDE ステートメントは、1 つの数字データ項目と他の数字データ項目 (複数可) との間で 除算を行い、商と剰余をデータ項目に保管します。

フォーマット 1: DIVIDE ステートメント



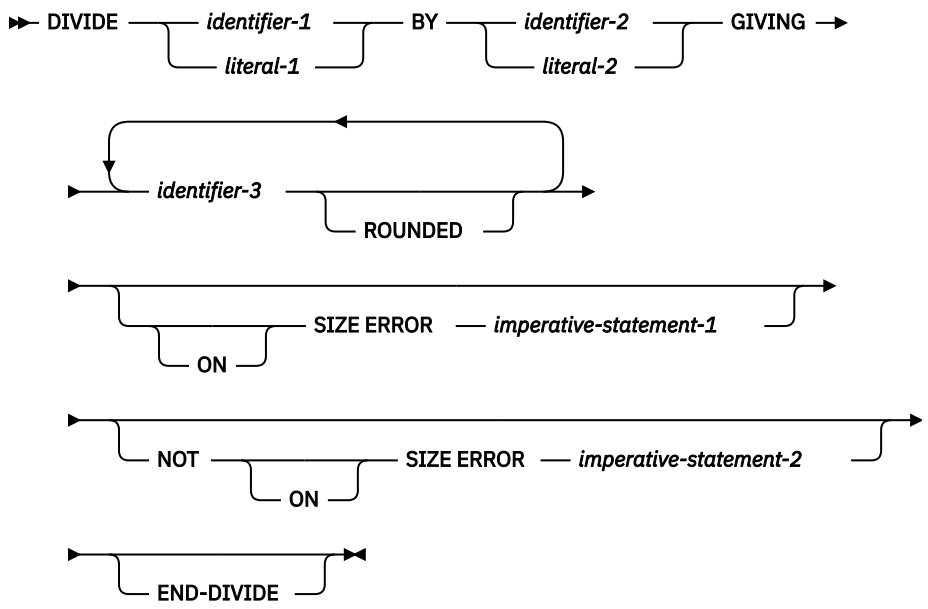
フォーマット 1 では、*ID-1* またはリテラル-*1* の値を *ID-2* の値 で割り、その商を *ID-2* に保管します。 *ID-2* は複数指定可能で、左から右へ順に 除算が行われ、商はそれぞれの *ID-2* に保管されます。

フォーマット 2: INTO および GIVING 句を含む DIVIDE ステートメント



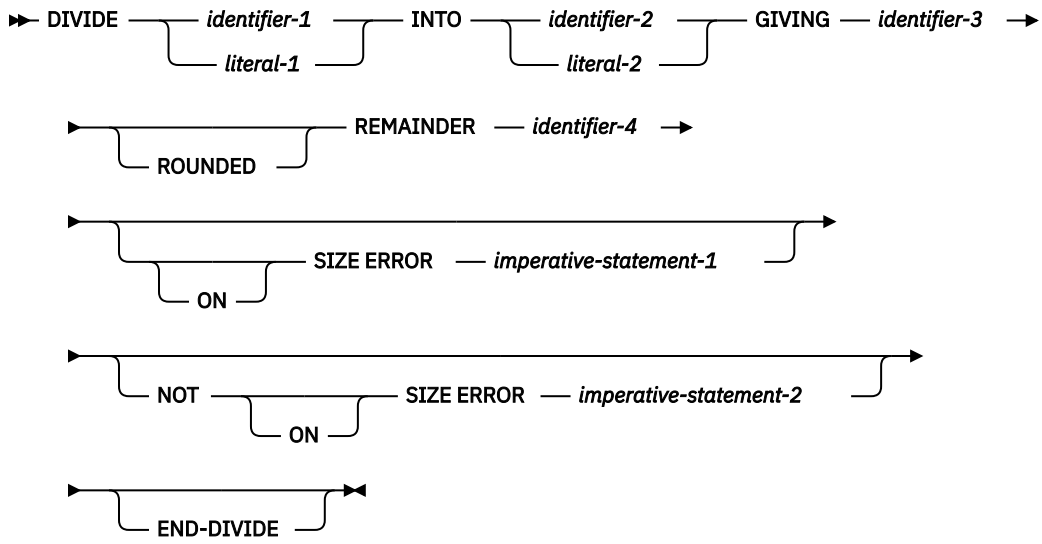
フォーマット 2 では、*ID-1* またはリテラル-*1* の値を *ID-2* または リテラル-*2* の値で割り、その商の値は、*ID-3* で参照される各データ項目に 保管されます。

フォーマット 3: BY および GIVING 句を含む DIVIDE ステートメント



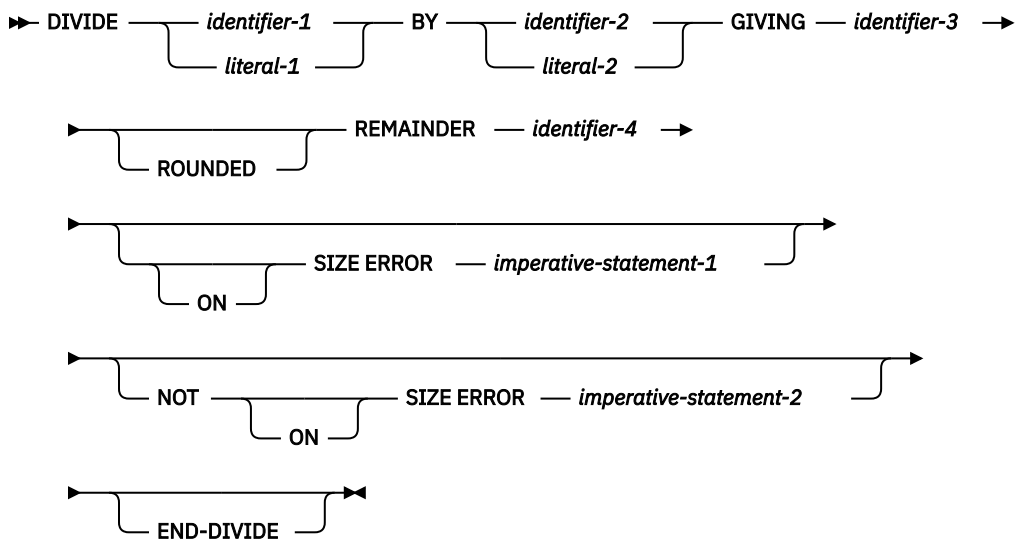
フォーマット 3 では、*ID-1* またはリテラル-*1* の値を *ID-2* または リテラル-*2* の値で割り、その商の値は、*ID-3* で参照される各データ項目 に保管されます。

フォーマット 4: INTO および REMAINDER 句を含む DIVIDE ステートメント



フォーマット 4 では、*ID-1* またはリテラル-*1* の値を *ID-2* または リテラル-*2* の値で割ります。 その商の値は *ID-3* に保管され、剰余の値は *ID-4* に保管されます。

フォーマット 5: BY および REMAINDER 句を含む DIVIDE ステートメント



フォーマット 5 では、*ID-1* またはリテラル*-1* の値を *ID-2* または リテラル*-2* の値で割ります。 その商の値は *ID-3* に保管され、剰余の値は *ID-4* に保管されます。

すべてのフォーマットに関して次のことが言えます。

ID-1*、*ID-2
基本数字データ項目である必要があります。 *ID-1* および *ID-2* は日付フィールドにはできません。

ID-3*、 *ID-4
基本数字項目または数字編集項目を指定する必要があります。

ID-3 または *ID-4* が日付フィールドである場合、商または剰余が *ID-3* にどのように保管されるかについては、237 ページの『日付フィールドに関連する算術演算結果の保管』を参照してください。

リテラル*-1*、 リテラル*-2*
これは、数字リテラルでなければなりません。

フォーマット 1、2、および 3 の場合、数字データ項目またはリテラルを指定できる場所であれば、 浮動小数点データ項目およびリテラルを使用することができます。

フォーマット 4 および 5 では、浮動小数点データ項目またはリテラルは使用できません。

ROUNDED 句

フォーマット 1、2、および 3 については、265 ページの『ROUNDED 句』を参照してください。

フォーマット 4 および 5 の場合は、剰余を計算するために使用される商は、 中間フィールドにあります。 中間フィールドの値は、丸めが行われるの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じた結果が、*ID-4* に保管されます。 *ID-3* が、数字編集項目であれば、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。

REMAINDER 句は、受け取りフィールドまたはオペランドのいずれかが浮動小数点項目の場合は無効です。

REMAINDER 句の中で *ID-4* に添え字が付いていると、その添え字は、 除算結果が GIVING 句の *ID-3* に入れられた後で評価されます。

SIZE ERROR 句

フォーマット 1、2、および 3 については、266 ページの『SIZE ERROR 句』を参照してください。

フォーマット 4 および 5 の場合、商にサイズ・エラーが起こると、剰余の計算は意味がありません。 したがって、商フィールド (*ID-3*) と剰余フィールド (*ID-4*) の内容は変わりません。

剰余にサイズ・エラーが起こると、剰余フィールド (*ID-4*) の内容は変わりません。

上記のいずれの場合も、実際にどの状態が起こったかを判別するために、結果を分析する必要があります。

NOT ON SIZE ERROR 句の詳細については、266 ページの『SIZE ERROR 句』を参照してください。

END-DIVIDE 句

この明示的範囲終了符号は、DIVIDE ステートメントの範囲を区切るために使用されます。 END-DIVIDE では、 条件付き DIVIDE ステートメントを命令ステートメントにして、 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-DIVIDE は、命令 DIVIDE ステートメントと共に使用できます。

詳しくは、263 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。

ENTRY ステートメント

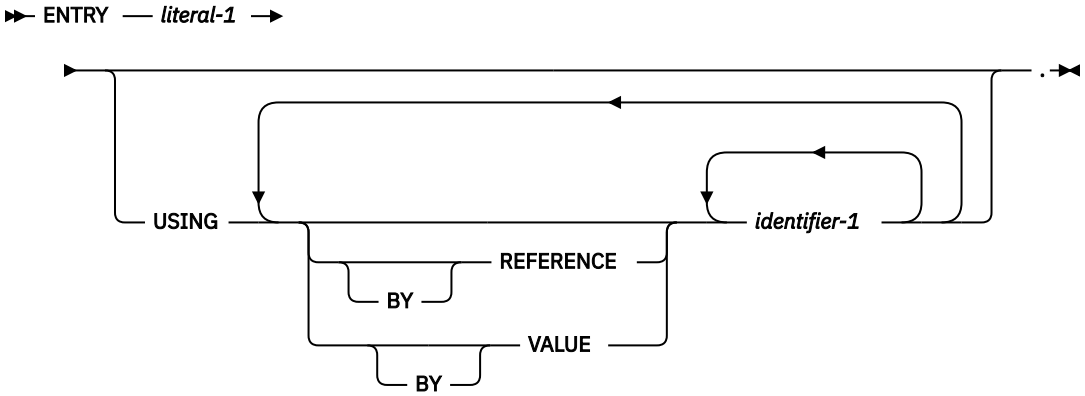
ENTRY ステートメントは、呼び出されたサブプログラムの代替入り口点を設定します。

ENTRY ステートメントは、以下で使用できません。

- PROCEDURE DIVISION RETURNING 句を使用する戻り値を指定するプログラム。 詳細については、230 ページの『PROCEDURE DIVISION ヘッダー』の RETURNING 句の説明を参照してください。
- ネストされたプログラム。 ネストされたプログラムについて詳しくは、77 ページの『ネストされたプログラム』を参照してください。

代替入り口点を指定した CALL ステートメントが呼び出し側プログラムの中で実行されると、 ENTRY ステートメントの後ろにある次の実行可能ステートメントに制御が移ります。

フォーマット



リテラル-1

英数字リテラルでなければならず、最外部プログラムのプログラム名形成 の規則に従っている必要があります (84 ページの『PROGRAM-ID 段落』を参照)。

プログラム ID、またはこのプログラムの中の他の ENTRY リテラルと同じにすることはできません。

形象定数にすることはできません。

呼び出されたプログラムの実行は、CALL ステートメントで指定されたリテラルまたは ID に対応するリテラルを持つ ENTRY ステートメントの後にある最初の実行可能ステートメントから開始されます。

ENTRY ステートメント上の入り口点名は、 PGMNAME コンパイラー・オプションによって影響を受ける可能性があります。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『*PGMNAME*』を参照してください。

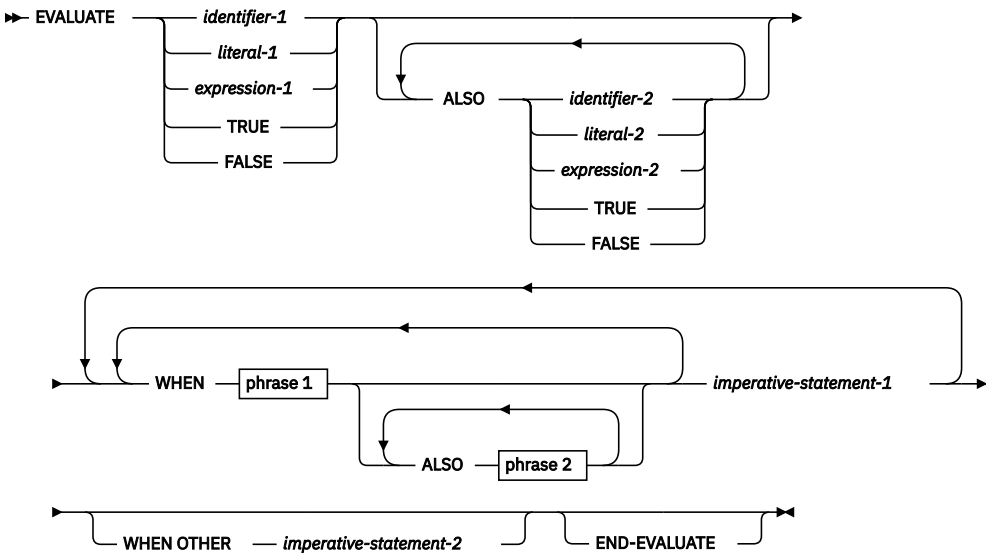
USING 句

USING 句の説明については、230 ページの『PROCEDURE DIVISION ヘッダー』を参照してください。 .

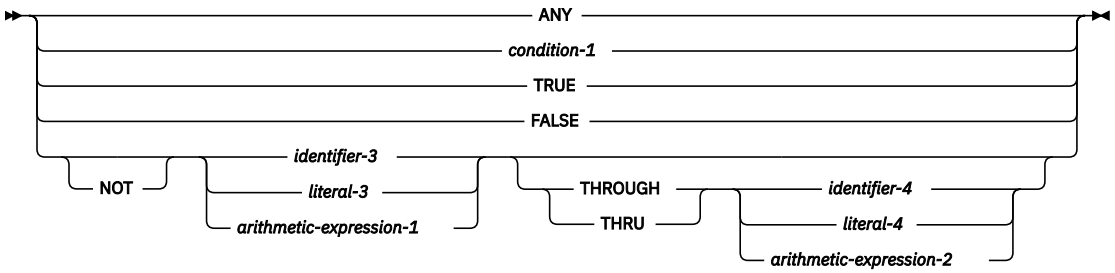
EVALUATE ステートメント

EVALUATE ステートメントは、一連のネストされた IF ステートメントの省略表現を提供します。 EVALUATE ステートメントは、複数の条件を評価することができます。 以降の処置は、これらの評価の結果次第です。

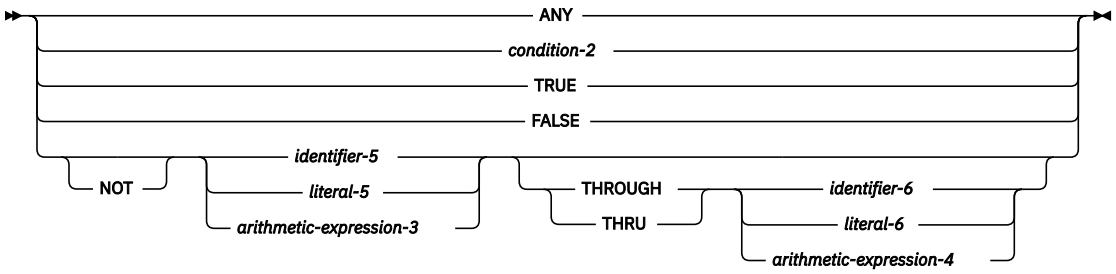
フォーマット



句 1



句 2



WHEN 句の前にあるオペランド

これらのオペランドは、2 つの方法のいずれかにより解釈されます。 その解釈は、それらが指定される方法に応じて異なります。

- 個別に解釈されます。それらは選択サブジェクト と呼ばれます。

- まとめて解釈されます。それらは、選択サブジェクトの集合 と呼ばれます。

WHEN 句の中のオペランド
これらのオペランドは、2 つの方法のいずれかにより解釈されます。 その解釈は、それらが指定される方法に応じて異なります。

- 個別に解釈されます。それらは選択オブジェクト と呼ばれます。
- まとめて解釈されます。それらは選択オブジェクトの集合 と呼ばれます。

ALSO
一連の選択サブジェクト内の選択サブジェクトを分離し、 一連の選択オブジェクト内の選択オブジェクトを分離します。

THROUGH と THRU
これらのキーワードは同じ意味です。

THRU 句によって結合されている 2 つのオペランドは、同じクラスに属していなければなりません。 このようにして結合された 2 つのオペランドは、1 つの選択オブジェクトを構成します。

選択オブジェクトの集合内にある選択オブジェクトの個数は、選択サブジェクトの個数と一致しなければなりません。

選択オブジェクトの集合内にあるそれぞれの選択オブジェクトは、次に示す規則に従って、 選択サブジェクトの集合内にある同じ順序位置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集合の中にある対応したオペランドと比較して有効なオペランドである必要があります。 日付フィールドに関する比較については、250 ページの『日付フィールドの比較』を参照してください。 .
- 選択オブジェクトとして現れる条件-*1*、条件-*2*、またはキーワード TRUE / FALSE は、選択サブジェクトの集合の中の条件式またはキーワード TRUE / FALSE と対応していなければなりません。
- ワード ANY は、どのタイプの選択サブジェクトとでも対応することができます。

END-EVALUATE 句

この明示的範囲終了符号は、EVALUATE ステートメントの範囲を区切るために使用されます。 END-EVALUATE 句を使うことによって、 条件的な EVALUATE ステートメントを別の条件ステートメントの中にネストすることができます。

詳しくは、263 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 ステートメントを実行すると、それぞれの選択サブジェクトと選択オブジェクトが評価され、数字、英数字、DBCS、または国別文字の値、数字、英数字、DBCS、または国別文字の値の範囲、 または真の値が割り当てられるように演算が実行されます。

これらの値は、次のようにして決定されます。

- *ID-1*、*ID-2*、... によって指定される選択サブジェクト、 および NOT 句または THRU 句を伴わない *ID-3* または *ID-5* によって指定される選択オブジェクトには、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*、... によって指定される選択サブジェクト、 および NOT 句または THRU 句を伴わない リテラル-*3* またはリテラル-*5* によって指定される選択オブジェクトには、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* が形象定数の ZERO、QUOTE、または SPACE である場合は、形象定数には対応する選択サブジェクトのクラスが割り当てられます。
- 算術 式として式-*1*、式-*2*、... が指定された選択サブジェクトと算術式-*1* または算術式-*3* が指定された選択オブジェクト (NOT 句も THRU 句も伴わないもの) には、算術式を評価する際の規則に従って、数字が割り当てられます。 (234 ページの『算術式』を参照してください。)
- 条件 式として式-*1*、式-*2*、... が指定された選択サブジェクトと、条件-*1* または条件-*2* が指定された選択オブジェクトには、 条件式を評価する際の規則に従って、真の値が割り当てられます。 (238 ページの『条件式』を参照してください。)

- ワード TRUE または FALSE によって指定された選択サブジェクトまたは選択オブジェクトにはいずれも、真の値が割り当てられます。 真の値「TRUE」はワード TRUE で指定された項目に割り当てられ、真の値「FALSE」はワード FALSE で指定された項目に割り当てられます。
- キーワード ANY を付けて指定された選択オブジェクトは、どれもそれ以上評価されることはありません。
- THRU 句が、NOT 句を持たずに選択オブジェクトに対して指定されている場合、 選択サブジェクトと比較するとき、比較される値の範囲は、 比較の規則に従って第 1 オペランド以上で第 2 オペランド以下にあるすべての値を含みます。 第 1 オペランドが第 2 オペランドより大きい場合には、範囲の中に該当する値は存在しません。
- NOT 句が選択オブジェクトに指定されている場合、その項目に割り当てられる値は、NOT 句が省略されたときに項目に割り当てられる値または値の範囲に等しくない、すべての値になります。

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、いずれかの WHEN 句が選択サブジェクトのセットの条件を満たすかどうかを判別するために、選択サブジェクトと選択オブジェクトに割り当てられた値が比較されたかのように行われます。

この比較は、次のようにして行われます。

1. 最初の WHEN 句の選択オブジェクトのセット内にある各選択オブジェクトが、選択サブジェクトのセット内にある同じ順序位置を持つ選択サブジェクトと比較されます。 比較が条件を満たすためには、以下の条件のうち 1 つを満たす必要があります。
   a. 比較される項目に、数字、英数字、DBCS、または国別文字の値、または数字、英数字、DBCS、または国別文字の値の範囲が割り当てられている場合、選択オブジェクトに割り当てられた値、または値の範囲内の 1 つの値が、比較の規則に従って、選択サブジェクトに割り当てられた値と等しい場合に、比較条件を満足したことになります。
   b. 比較される項目に真の値が割り当てられている場合は、その項目に同じ真の値が割り当てられると比較が実行されます。
   c. 比較される選択オブジェクトが、ワード ANY によって指定されている場合、選択サブジェクトの値とは無関係に、比較は常に条件を満たします。
2. 比較され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ついて、上記の比較が条件を満たした場合は、その選択オブジェクトのセットを含む WHEN 句が、選択サブジェクトのセットの条件を満たすものとして選択されます。
3. 比較され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ついて、上記の比較が条件を満たさないものがある場合は、その選択オブジェクトのセットは、選択サブジェクトのセットの条件を満たしていません。
4. このプロシージャーは、以降の選択オブジェクトの集合について、ソース・テキスト中にそれらが現れる順番に繰り返され、 選択サブジェクトの集合を満たすいずれかの WHEN 句が選ばれるか、または、選択オブジェクトの全集合がなくなるまで行われます。

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進められます。

- ある WHEN 句が選ばれると、その選択された WHEN 句に続く最初の命令ステートメント*-1* から実行が継続されます。 1 つの命令ステートメント*-1* に対して、複数の WHEN 句を指定することができる点に注意してください。
- 何も WHEN 句が選択されないが、WHEN OTHER 句を指定してある場合には、実行は命令ステートメント*-2* から継続されます。
- WHEN 句が選択されず、WHEN OTHER 句も指定されていない場合は、範囲終了文字に続く次の実行可能ステートメントから実行が継続されます。
- EVALUATE ステートメントの実行の有効範囲は、選択された WHEN 句または WHEN OTHER 句の有効範囲の終わりに実行が達したとき、あるいは WHEN 句が選択されず、WHEN OTHER 句も指定されていないときに終了します。

EXIT ステートメント

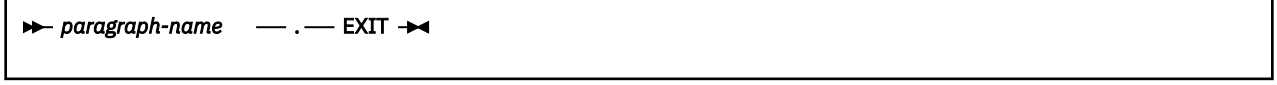
EXIT ステートメントは、一連のプロシージャーに共通の終了点を提供します。 また、セクション、段落、または行内 PERFORM ステートメントから出る方法も提供します。

注 : 形式 4 EXIT ステートメント EXIT FUNCTION は、まだサポートされていません。

形式 1 (シンプル)

形式 1 EXIT ステートメントは、一連のプロシージャーに共通の終了点を提供します。

形式 1



形式 1 EXIT ステートメントを使用すると、プログラム内の所定の点にプロシージャー名を割り当てることができます。

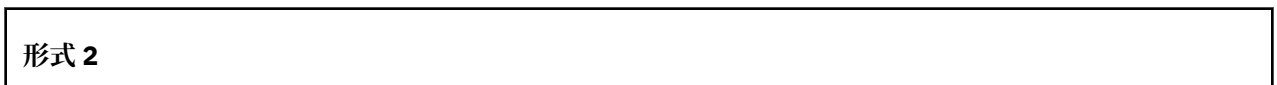
形式 1 EXIT ステートメントは、CONTINUE ステートメントとして扱われます。 EXIT ステートメントの後続のステートメントはすべて実行されます。

形式 2 (プログラム)

EXIT PROGRAM ステートメントは、呼び出されたプログラムの終わりを指定し、 制御を呼び出し側プログラムに戻します。

EXIT PROGRAM はプログラムの PROCEDURE DIVISION にしか指定できません。 EXIT PROGRAM は、GLOBAL 句が指定されている宣言型プロシージャーの中で指定することはできません。

形式 2



CALL ステートメントの制御のもとで (すなわち、 CALL ステートメントがアクティブである) 操作が行われているときに、 制御が INITIAL 属性を持たないプログラムの中の EXIT PROGRAM ステートメントに達すると、呼び出し側ルーチン (プログラム) の中の CALL ステートメントのすぐ後の地点に制御が戻されます。 このとき、呼び出し側ルーチンの状態は、 そのプログラムが CALL ステートメントを実行したときに存在していたものと同じです。 ただし、データ項目の内容、 および呼び出し側ルーチンと呼び出されたプログラム間で共用されたデータ・ファイルの内容は、変更されている可能性があります。 呼び出されたプログラムの状態は無変更です。 ただし、実行されたすべての PERFORM ステートメントの範囲の終わりに達したと みなされる場合は除きます。

INITIAL 属性を持つ呼び出されたプログラム内で EXIT PROGRAM ステートメントを実行するのは、そのプログラムを参照して CANCEL ステートメントを実行するのと同じことになります。

制御が EXIT PROGRAM ステートメントに到達したときに、CALL ステートメントがアクティブでなければ、制御はこの出口点を通過し、 次の実行可能なステートメントに渡されます。

サブプログラムに PROCEDURE DIVISION RETURNING 句が指定されている場合、 その RETURNING 句によって参照されるデータ項目内の値が、サブプログラム呼び出しの結果になります。

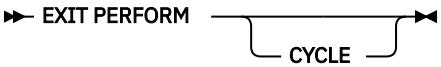
EXIT PROGRAM ステートメントは、一連の命令ステートメントの最後に指定する必要があります。 指定しないと、CALL ステートメントがアクティブの場合に、 EXIT PROGRAM ステートメントの後のステートメントが実行されません。

呼び出されたプログラムに次の実行可能ステートメントがない場合は、暗黙の EXIT PROGRAM ステートメントが実行されます。

形式 5 (インライン実行)

EXIT PERFORM ステートメントは、GO TO ステートメントまたは PERFORM ... THROUGH ステートメントを使用しない、行内 PERFORM ステートメントの終了を制御します。

形式 5



行内 PERFORM ステートメントの外部に EXIT PERFORM ステートメントを指定すると、EXIT PERFORM は無視されます。

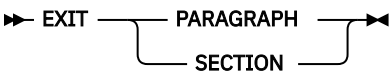
CYCLE 句のない EXIT PERFORM ステートメントが実行されると、暗黙的 CONTINUE ステートメントに制御が渡されます。 この暗黙的 CONTINUE ステートメントは、最も近い先行する未終了の行内 PERFORM ステートメントに対応する END-PERFORM 句の直後に続きます。

CYCLE 句のある EXIT PERFORM ステートメントが実行されると、暗黙的 CONTINUE ステートメントに制御が渡されます。 この暗黙的 CONTINUE ステートメントは、最も近い先行する未終了の行内 PERFORM ステートメントに対応する END-PERFORM 句の直前に置かれます。

形式 6 (プロシージャー)

EXIT PARAGRAPH ステートメントは、段落内の後続のどのステートメントも実行せずに、その段落の途中からの終了を制御します。 EXIT SECTION ステートメントは、セクション内の後続のどのステートメントも実行せずに、そのセクションからの終了を制御します。

形式 6



EXIT PARAGRAPH

EXIT PARAGRAPH ステートメントが実行されると、現在の段落の最後の明示的ステートメントの直後にある暗黙的 CONTINUE ステートメントに制御が渡されます。 この戻りのメカニズムは、言語エレメント (その段落の PERFORM、SORT、USE など) に関連付けられている他のあらゆる戻りメカニズムに優先します。

EXIT SECTION

EXIT SECTION ステートメントは、セクション内にのみ指定できます。

EXIT SECTION ステートメントが実行されると、現行セクションの最後の段落の直後にある名前のない空の段落に制御が渡されます。 この戻りのメカニズムは、言語エレメント (そのセクションの PERFORM、SORT、USE など) に関連付けられている他のあらゆる戻りメカニズムに優先します。

GOBACK ステートメント

GOBACK ステートメントは、EXIT PROGRAM ステートメントが呼び出されるプログラム の一部としてコーディングされている場合と同じように機能し、さ らに STOP RUN ステートメントが主プログラム内でコーディングされている場合と同様に 機能します。

GOBACK ステートメントは、呼び出されるプログラムの論理的終わりを指定します。

フォーマット

GOBACK ステートメントは、1 つの文の中の唯一のステートメント、または、 1 つの文の中の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現れなければなりません。 これは、GOBACK ステートメントの後にあるステートメントが実行されることがないからです。 GOBACK は、GLOBAL 句が指定されている宣言型プロシージャーの中で指定することはできません。

CALL ステートメントがアクティブであるときに、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じように、 呼び出し側プログラム中の CALL ステートメントのすぐ後の点に制御が戻されます。

さらに、INITIAL 属性を持つ呼び出されたプログラム内で GOBACK ステートメントを実行することは、 そのプログラムを指定して CANCEL ステートメントを実行する場合と同じになります。

次の表は、メインプログラムまたはサブプログラムで GOBACK ステートメントに対して実行されるアクションを示したものです。

終了 ステートメント	メインプログラム	サブプログラム
GOBACK	呼び出し側プログラムへ戻る。 (それがシステムの場合は、アプリケーションを終了する。)	呼び出し側プログラムへ戻る。

GO TO ステートメント

GO TO ステートメントは、PROCEDURE DIVISION のある部分から別の部分へ制御を移します。

GO TO ステートメントには、次のような種類があります。

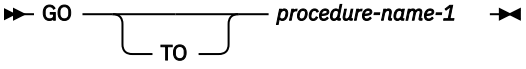
- 無条件
- 条件付き
- 変更される

無条件 GO TO

無条件 GO TO ステートメントは、プロシージャー名によって識別された段落またはセクションの最初のステートメントに制御権を移動します。 ただし、ALTER ステートメントによって GO TO ステートメントが変更された場合を除きます。

詳しくは、283 ページの『ALTER ステートメント』を参照してください。.

フォーマット 1: 無条件 GO TO ステートメント



プロシージャー名-1
GO TO ステートメントと同じ PROCEDURE DIVISION のプロシージャーまたはセクションの名前でなければなりません。

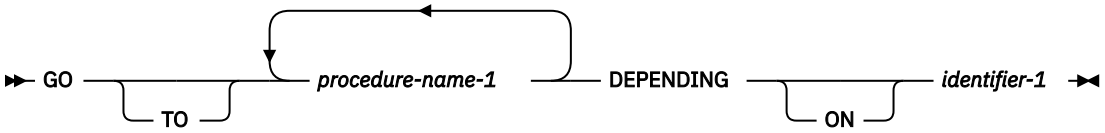
無条件 GO TO ステートメントが一連の命令ステートメントの先頭または途中に指定されていると、その後のステートメントが実行されません。

段落が ALTER ステートメントによって参照されている場合、 その段落は無条件 GO TO ステートメントまたは変更された GO TO ステートメントに続く段落名である必要があります。

条件付き GO TO

条件付き GO TO ステートメントは、*ID-1* によって参照されるデータ項目の値に応じて、複数のプロシージャーのうちの 1 つのプロシージャーに制御を移します。

フォーマット 2: 条件付き GO TO ステートメント



プロシージャー名-*1*

GO TO ステートメントと同じ PROCEDURE DIVISION のプロシージャーまたはセクションでなければなりません。プロシージャー名の個数は 255 以下でなければなりません。

ID-1

これは、整数からなる数字基本データ項目でなければなりません。 *ID-1* をウィンドウ表示日付フィールドにすることはできません。

1 を指定すると、プロシージャー名-*1* の最初のオカレンスによって指名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移ります。

2 を指定すると、プロシージャー名-*1* の 2 番目のオカレンスによって指名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移ります。以下同様です。

ID の値が、1 から n (n はこの GO TO ステートメント中で指定されたプロシージャー名の数) の範囲内にない場合、制御の移動は起きません。 代わりに、制御は通常の実行順序で次のステートメントに渡されます。

変更される GO TO

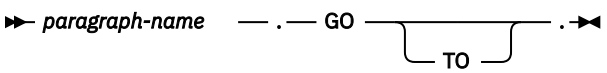
変更される GO TO ステートメントは、ALTER ステートメント中に指定された段落の最初のステートメントに制御を移します。

変更される GO TO ステートメントは、以下の場合には指定できません。

- RECURSIVE 属性を持つプログラム

変更される GO TO ステートメントが入った段落を参照している ALTER ステートメントが実行されていないと、 この GO TO ステートメントを実行することはできません。 それ以外の場合、GO TO ステートメントは CONTINUE ステートメントと同じように機能します。

フォーマット 3: 変更される GO TO ステートメント

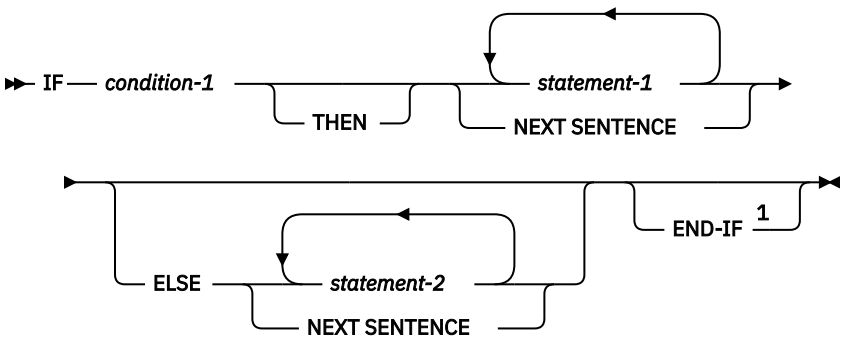


ALTER ステートメントがある段落を参照している場合、その段落は無条件 GO TO ステートメントまたは変更される GO TO ステートメントを後に付けた段落名のみで構成することもできます。

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。

フォーマット



注:

[1] END-IF はステートメント-*2* または NEXT SENTENCE と共に指定することができます。

条件-*1*
これは、単純条件にも複合条件にもすることができます (238 ページの『条件式』を参照)。

ステートメント-*1*、ステートメント-*2*
以下のいずれかのオプションを指定できます。

- 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が後につく命令ステートメント

NEXT SENTENCE
NEXT SENTENCE 句を指定すると、次の分離文字ピリオド の直後にある暗黙の CONTINUE ステートメントに制御が渡されます。

NEXT SENTENCE を END-IF と共に指定すると、END-IF の後のステートメントに制御が 渡されるのではなく、 最も近い後続のピリオドの後のステートメントに制御が渡されます。

END-IF 句

この明示的範囲終了符号は、IF ステートメントの範囲を区切るために使用されます。 END-IF 句を使うことによって、 条件付き IF ステートメントを他の条件ステートメント中にネストすることができます。 明示的範囲終了符号の詳細については、263 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。

IF ステートメントの範囲は以下のオプションのいずれかによって区切ることができます。

- ネスト構造で同じレベルにある END-IF 句。
- 分離文字ピリオド。
- ネストされている場合は、より高いネスト・レベルにある IF ステートメントに関連付けられている ELSE 句。

制御の移動

このトピックでは、テストされた条件が真または偽であるときに取る処置について説明します。

テストされた条件が、真であれば、次に示す処置の 1 つが取られます。

- ステートメント-*1* が指定されていれば、ステートメント-*1* が実行されます。 ステートメント-*1* に、 プロシージャー・ブランチ・ステートメントまたは条件ステートメントが入っている場合、 そのステートメントの規則に従って制御が移ります。 ステートメント-*1* にプロシージャー・ブランチ・ステートメントが含まれていない場合は、ELSE 句が指定されていても無視され、対応する END-IF 句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に制御が渡されます。
- NEXT SENTENCE が指定されている場合、制御は、次の分離文字ピリオドの直後にある暗黙の CONTINUE ステートメントに渡されます。

テストされた条件が、偽であれば、次に示す処置の 1 つが取られます。

- ELSE ステートメント-*2* が指定されていれば、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・ブランチ・ステートメントまたは条件ステートメントが含まれている場合は、そのステートメントの規則に従って制御が移ります。 ステートメント-*2* にプロシージャー・ブランチ・ステートメントまたは条件ステートメントが入っていない場合、制御は、対応する END-IF 句の後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に渡されます。
- ELSE NEXT SENTENCE が指定されている場合、制御は、次の分離文字ピリオドの直後にある暗黙の CONTINUE STATEMENT に渡されます。
- ELSE ステートメント-*2* も ELSE NEXT SENTENCE も指定されていない場合、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に渡されます。

ELSE 句が省略されている場合には、この条件の後に続く、 対応する END-IF 句、または、分離文字ピリオドの前までにあるすべてのステートメントは、ステートメント-*1* の一部とみなされます。

ネストされた IF ステートメント

IF ステートメントがステートメント-*1* またはステートメント-*2* として現れるか、またはステートメント-*1* またはステートメント-*2* の一部として現れる場合、 この IF ステートメントはネスト された IF ステートメントです。

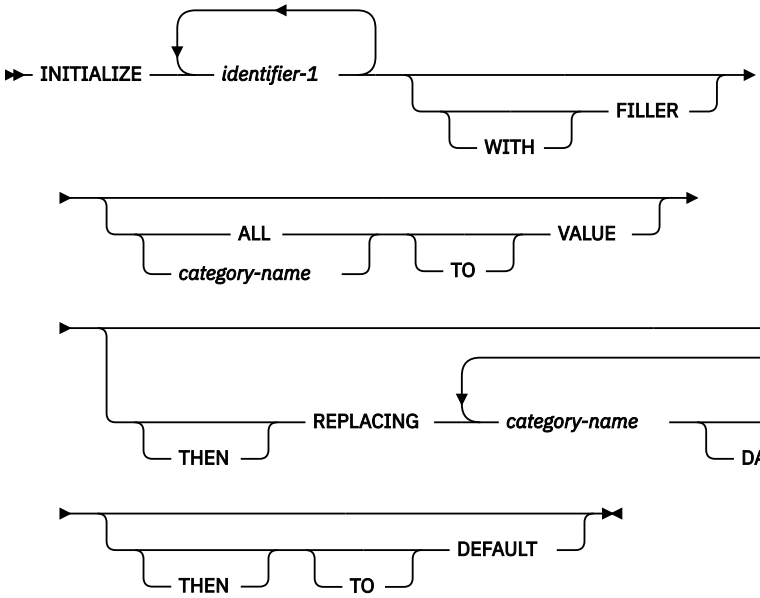
IF ステートメントがステートメント-*1* またはステートメント-*2* として現れるか、またはステートメント-*1* またはステートメント-*2* の一部として現れる場合、 この IF ステートメントはネスト された IF ステートメントです。

ネストされた IF ステートメントは、左から右に処理される、一致した IF、ELSE、および END-IF の組み合わせとみなされます。 したがって、検出される ELSE はすべて、まだ ELSE と一致していないか、暗黙的または明示的に終了されていない、最も近い先行 IF に一致します。 検出される END-IF はすべて、暗黙的または明示的に終了されていない、 先行の最も近い場所にある IF と一致します。

INITIALIZE ステートメント

INITIALIZE ステートメントは、選ばれたカテゴリーのデータ・フィールドを あらかじめ定義されている値に設定します。 INITIALIZE ステートメントは、1 つ以上の MOVE ステートメントを実行するのと機能的に同等です。

フォーマット



ここで、カテゴリー名は以下のいずれかです。

- ALPHABETIC
- ALPHANUMERIC
- ALPHANUMERIC-EDITED
- DBCS
- EGCS
- NATIONAL
- NATIONAL-EDITED
- NUMERIC
- NUMERIC-EDITED

ID-1

受け取り領域。

identifier-1 は、以下の項目のいずれかを参照する必要があります。

- 英数字グループ項目
- 国別グループ項目
- 以下のいずれかのカテゴリーの基本データ項目
  - 英字
  - 英数字
  - 英数字編集
  - DBCS
  - 外部浮動小数点
  - 内部浮動小数点

- 国別
- 国別編集
- 数字
- 数字編集

• 送信オペランドとして *ID-2* またはリテラル *-1* が指定されている MOVE ステートメントの中で 受信オペランドとして有効な特殊レジスター。

identifier-1 は基本項目またはグループ項目を参照します。 INITIALIZE ステートメントの実行の効果は、一連の暗黙 MOVE ステートメント (それぞれが、その受信オペランドとして基本データ項目を持っている) が実行された場合と同じになります。

ID-1 が国別グループ項目を参照する場合は、*ID-1* はグループ項目として処理されます。

ID-2*、リテラル*-1
送り出し領域。

ID-2 が国別グループ項目を参照する場合は、*ID-2* は 国別カテゴリーの基本データ項目として処理されます。

ID-2 は、*ID-1* を受け取りオペランドとして指定した MOVE ステートメントの送り出しオペランドとして有効な基本データ項目 (または基本項目として扱われる国別グループ項目) を参照する必要があります。

リテラル*-1* は、*ID-1* を受け取りオペランドとして指定した MOVE ステートメントの送り出しオペランドとして有効なリテラルでなければなりません。

添え字付きの項目を *ID-1* に指定することができます。 完全なテーブルを含むグループを *ID-1* として指定することによってのみ、テーブル全体を初期設定することができます。

使用上の注意: *ID-1* のデータ記述項目には、OCCURS 節の DEPENDING 句を入れることができます。 ただし、INITIALIZE ステートメントを使用しても、可変位置項目または可変長項目を初期化することはできません。

ID-1 のデータ記述項目に、RENAMES 節を入れることはできません。

特殊レジスターは、暗黙の MOVE ステートメントの有効な受け取りフィールドまたは送り出しフィールドである場合のみ、 それぞれ *ID-1* および *ID-2* として指定することができます。

FILLER 句

FILLER 句が指定されている場合、明示 FILLER 節または暗黙 FILLER 節を持つ受信基本データ項目が初期化されます。

VALUE 句

VALUE 句が指定される場合

• ALL が VALUE 句に指定されている場合、カテゴリー名にリストされているカテゴリーすべてが実行された場合と同じになります。
• VALUE 句の中では、同じカテゴリーを繰り返すことはできません。

REPLACING 句

REPLACING 句が指定されている場合:

• *ID-2* は、REPLACING 句で指定された対応するカテゴリーの項目への MOVE ステートメントの送信オペランドとして有効なカテゴリーの項目を参照する必要があります。
• リテラル*-1* は、REPLACING 句で指定された対応するカテゴリーの項目への MOVE ステートメントの送信オペランドとして有効なカテゴリーでなければなりません。
• 浮動小数点リテラル、内部浮動小数点カテゴリーのデータ項目、または外部浮動小数点カテゴリーのデータ項目は、NUMERIC カテゴリー内にあるかのように扱われます。

- REPLACING 句の中では、同じカテゴリーを繰り返すことはできません。

EGCS を例外として、語 REPLACING に続くキーワードは、137 ページの『データのクラスとカテゴリー』に示されたデータのカテゴリーに対応しています。

REPLACING 句内の EGCS は DBCS と同義です。

INITIALIZE ステートメントの規則

INITIALIZE ステートメントの実行の効果は、一連の暗黙 MOVE ステートメント (それぞれが、その受信オペランドとして基本データ項目を持っている) が実行された場合と同じになります。 これらの暗黙ステートメントの受信オペランドは『規則 1』に、送信オペランドは『規則 2』に定義されています。

1. 各暗黙 MOVE ステートメントにおける受信オペランドは、以下の順番に規則 a、b、および c を適用することによって決定されます。 ただし、特定の規則によって受け取り側として除外されていないデータ項目であっても、後続の規則が適用されたときに受け取り側として除外されることがあります。 例えば、規則 a によって除外されないデータ項目であっても、規則 b や規則 c によって除外されることがあります。

   a. まず、以下のデータ項目が受信オペランドの対象から除外されます。

      - MOVE ステートメントの有効な受信オペランドではない ID すべて。
      - FILLER 句が指定されていない場合、明示 FILLER 節または暗黙 FILLER 節を持つ基本データ項目。
      - データ記述項目に REDEFINES 節または RENAMES 節が含まれている、またはデータ記述項目に REDEFINES 節が含まれているデータ項目に従属している、*ID-1* に従属する基本データ項目のすべて。 ただし、*ID-1* それ自体が REDEFINES 節を持っているか、または REDEFINES 節を持つデータ項目に従属している可能性があります。

   b. 次に、以下のいずれかに該当する場合、基本データ項目は受け取り項目である可能性があります。

      - *ID-1* によって明示的に参照されている。
      - *ID-1* によって参照されているグループ・データ項目の中に含まれている。 基本データ項目がテーブル・エレメントであれば、その基本データ項目のオカレンスそれぞれが、受信オペランドである可能性があります。

   c. 最後に、受信オペランドの可能性があるオペランドはそれぞれ、以下の条件の少なくとも 1 つが真である場合、受信オペランドです。

      - VALUE 句が指定されていて、基本データ項目のカテゴリーが、その VALUE 句に指定 (または暗黙指定) されているカテゴリーのいずれかで、以下の条件のどちらかが真である。
        - データ項目フォーマット VALUE 節が、基本データ項目のデータ記述項目に指定されている。
        - テーブル・フォーマット VALUE 節が、基本項目のデータ記述項目に指定されていて、その VALUE 節が、基本データ項目の特定のオカレンスの値を指定している。
      - REPLACING 句が指定されていて、基本データ項目のカテゴリーが、その REPLACING 句に指定されているカテゴリーのいずれかである。
      - DEFAULT 句が指定されている。
      - REPLACING 句と VALUE 句がどちらも指定されていない。

2. それぞれの暗黙 MOVE ステートメントの送信オペランドは、以下のようにして決定されます。

   - VALUE 句が原因でデータ項目が受信オペランドとして適格であれば、送信オペランドは、データ項目のデータ記述項目に指定されている VALUE 節のリテラルによって決定されます。 データ項目がテーブル・エレメントであれば、初期化されるオカレンスに対応する VALUE 節のリテラルが、送信オペランドを決定します。 実際の送信オペランドは、MOVE ステートメントで受信オペランドに移動されるときに、VALUE 節の適用によって作成されるデータ項目の初期値と同じ結果をもたらすリテラルです。
   - データ項目が、VALUE 句が原因で受信オペランドとして適格ではなくても、REPLACING 句が原因で適格である場合、送信オペランドは、その REPLACING 句に指定されたカテゴリーに関連付けられているリテラル-*1* または *ID-2* です。

- 前述の 2 つの規則によってデータ項目が適格ではない場合、使用される送信オペランドは以下のように、受信オペランドのカテゴリーに応じて異なります。
  - カテゴリーが英字、英数字、英数字編集、DBCS、EGCS、国別、または国別編集の受け取り項目については、SPACE が暗黙の送り出し項目になります。
  - カテゴリーが数字または数字編集の受け取り項目については、ZERO が暗黙の送り出し項目になります。

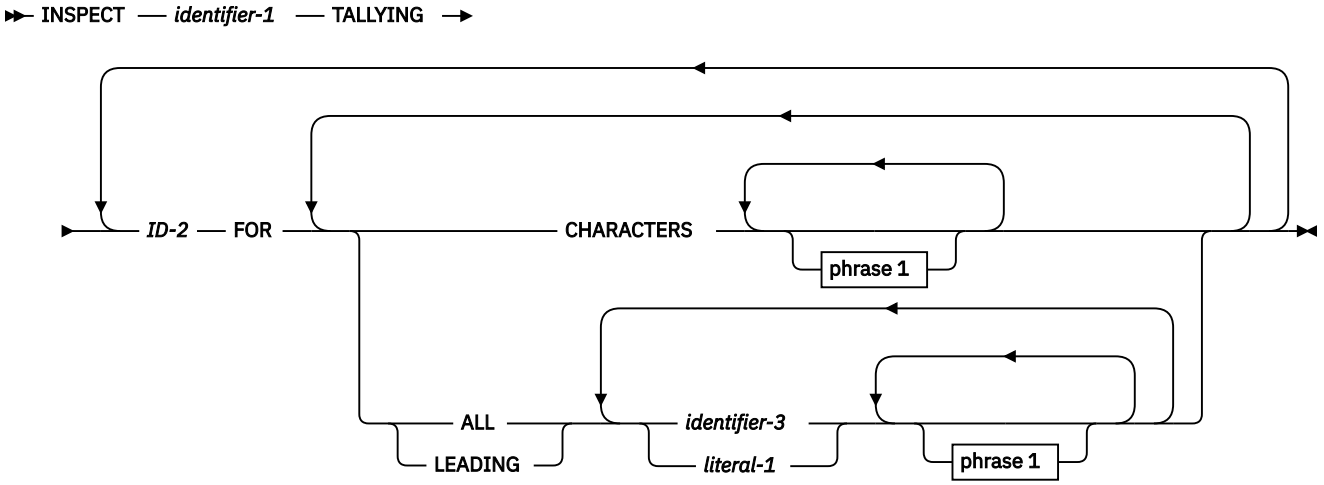
INSPECT ステートメント

INSPECT ステートメントは、データ項目の文字または文字のグループを検査します。

INSPECT ステートメントは、以下のタスクを実行します。

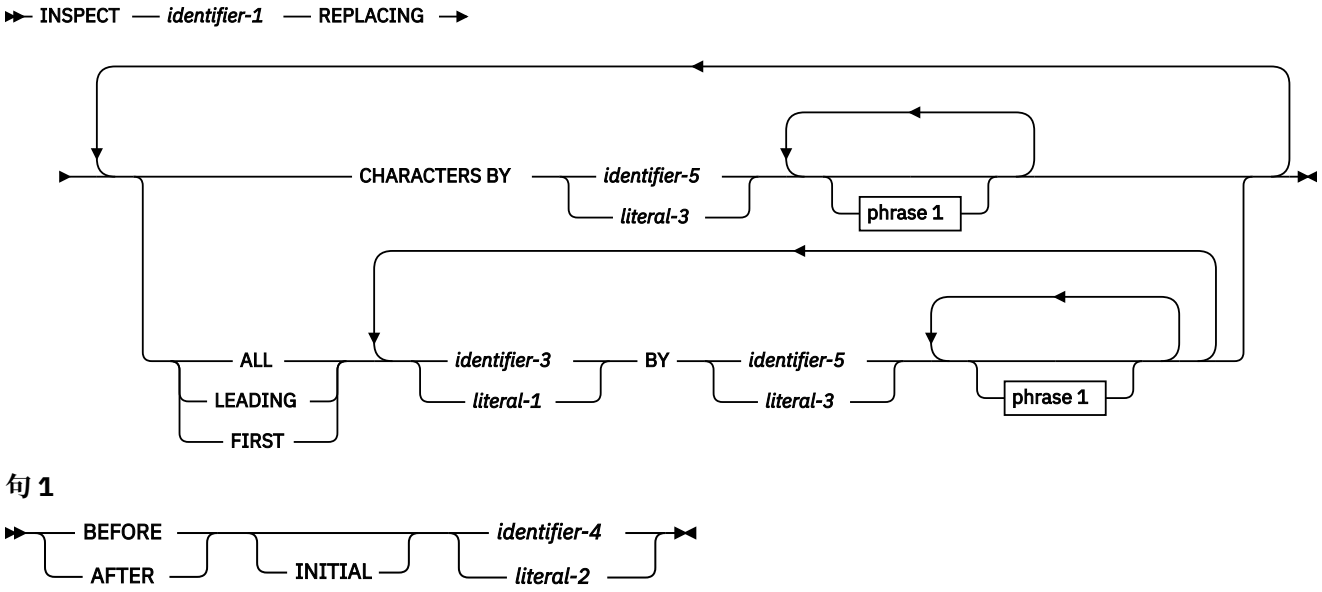
- データ項目内で特定の文字 (英数字、DBCS、または国別) のオカレンスを数えます (フォーマット 1 および 3)。
- 特定の文字のオカレンスを数えたり、データ項目の一部または全部をスペースや 0 のような指定した文字で 満たします (フォーマット 2 および 3)。
- データ項目内で特定の文字のすべてのオカレンスをユーザー指定の置き換え文字に変換します (フォーマット 4)。

フォーマット 1: TALLYING 句を含む INSPECT ステートメント



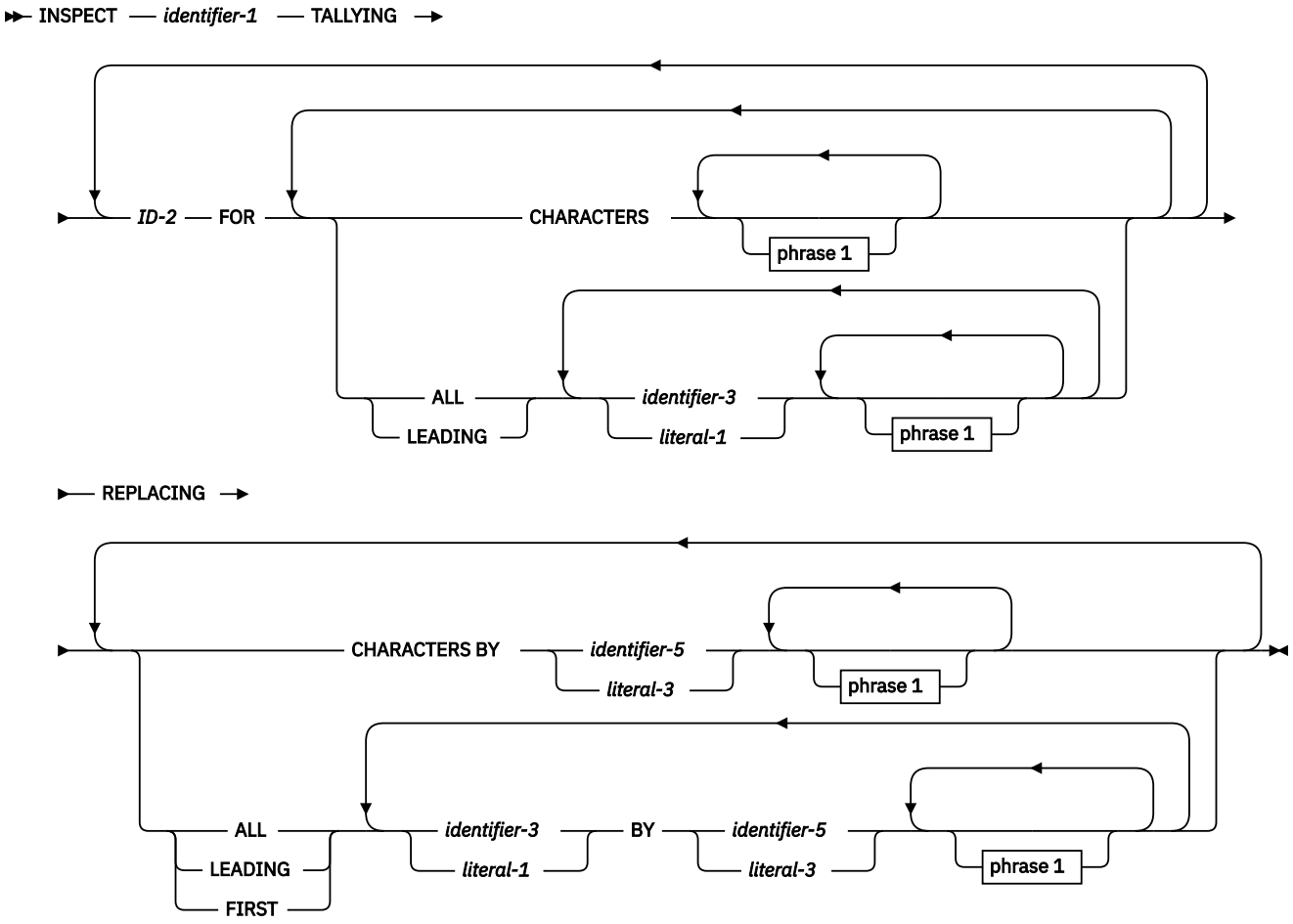
句 1

フォーマット 2: REPLACING 句を含む INSPECT ステートメント

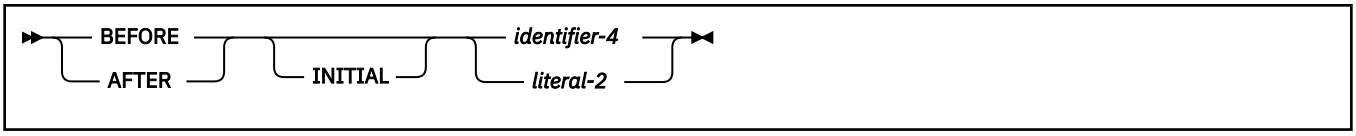


句 1

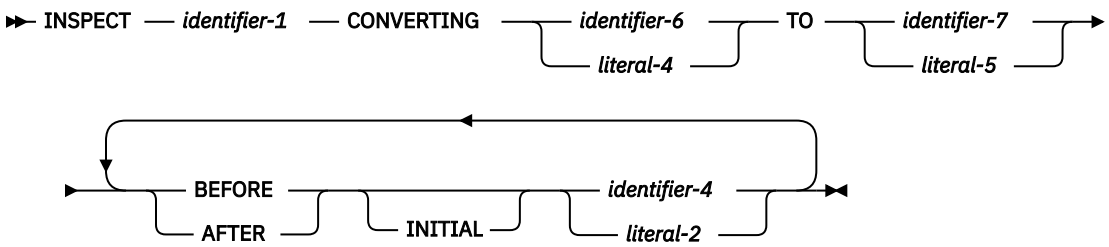
フォーマット 3: TALLYING および REPLACING 句を含む INSPECT ステートメント



句 1



フォーマット 4: CONVERTING 句を含む INSPECT ステートメント



ID-1
これは検査項目 であり、以下の項目のいずれかにできます。

- 英数字グループ項目または国別グループ項目
- USAGE DISPLAY、DISPLAY-1、または NATIONAL により明示的または暗黙的に記述されている基本データ項目。 この項目は、選択された使用法に有効であればどのカテゴリーを持つこともできます。

ID-3*、*ID-4*、*ID-5*、*ID-6*、*ID-7
USAGE DISPLAY、DISPLAY-1、または NATIONAL により明示的または暗黙的に記述されている基本データ項目を参照しなければなりません。

リテラル-1*、*リテラル-2*、*リテラル-3*、*リテラル-4
英数字、DBCS、または国別カテゴリーでなければなりません。

ID-1 が USAGE NATIONAL の場合、リテラルは国別カテゴリーでなければなりません。

ID-1 が USAGE DISPLAY-1 の場合、リテラルは DBCS カテゴリーでなければなりません。

ID-1 が USAGE DISPLAY の場合、リテラルは英数字カテゴリーでなければなりません。

ID-1 が USAGE DISPLAY-1 (DBCS) の場合、リテラルは形象定数 SPACE にすることができます。

ID-1 が USAGE DISPLAY または NATIONAL の場合、リテラルは、13 ページの『形象定数』で示すように、ALL というワードで始まらない任意の形象定数にすることができます。 . 形象定数は、*ID-1* が USAGE DISPLAY のときは 1 文字英数字リテラルとして、*ID-1* が USAGE NATIONAL のときは 1 文字国別リテラルとして扱われます。

ID (*ID-2* を除く) はすべて、*ID-1* と同じ使用法でなければなりません。 リテラルはすべて、*ID-1* の使用法が DISPLAY、 DISPLAY-1、または NATIONAL のときには、それぞれ英数字、DBCS 、または国別のカテゴリーでなければなりません。

INSPECT ステートメントのどの ID もウィンドウ表示日付フィールドにはできません。

TALLYING 句 (フォーマット 1 および 3)

この句は、特定の文字または特殊文字のデータ項目内でのオカレンスを数えます。

ID-1 が DBCS データ項目の場合は、DBCS 文字が数えられます。*ID-1* が USAGE NATIONAL である場合は、国別文字 (エンコード・ユニット) が数えられます。それ以外の場合は、英数字文字 (バイト数) が数えられます。

ID-2
カウント・フィールド であり、 その PICTURE 文字ストリング内に記号 P を使用せずに定義された、基本整数項目でなければなりません。

ID-2 は外部浮動小数点カテゴリーにすることはできません。

INSPECT ステートメントの実行を開始する前に、*ID-2* を初期設定しておく必要があります。

使用上の注意: カウント・フィールドは、USAGE NATIONAL を指定して定義された整数データ項目にすることができます。

ID-3* またはリテラル-*1
これは計算フィールド (オカレンスが累計される項目) です。

CHARACTERS
CHARACTERS が指定されていても、BEFORE も AFTER 句も指定されていない場合は、 被検査項目 (*ID-1*) 内で、カウント・フィールド (*ID-2*) が、スペース文字を含む 1 文字ごとに 1 ずつ増加されます。したがって、TALLYING 句を指定した INSPECT ステートメントが実行されると、カウント・フィールドの値は、被検査項目内の文字位置数だけ増加します。

ALL
ALL が指定されていても BEFORE 句または AFTER 句が指定されていない場合は、被検査項目 (*ID-1*) 内の一致する被比較数 (*ID-3* または リテラル-*1*) のオーバーラップしないオカレンスごとに、 カウント・フィールド (*ID-2*) が 1 ずつ増加されます。 増加は左端の文字位置から右端に続きます。

LEADING
LEADING が指定されていても BEFORE または AFTER 句が指定されていない場合は、 被検査項目 (*ID-1*) 内の累計被比較数のオーバーラップしない連続した各オカレンスごとに、カウント・フィールド (*ID-2*) が 1 ずつ増加されます。その場合、左端のこのようなオカレンスは、この累計被比較数が関与する最初の比較サイクルにおいて、比較が開始される点に位置します。

FIRST (フォーマット 3 のみ)
FIRST が指定されていても、BEFORE も AFTER 句も 指定されていない場合、置換フィールドは被検査項目 (*ID-1*) 内の サブジェクト・フィールドの左端のオカレンスを置換します。

REPLACING 句 (フォーマット 2 および 3)

この句は、データ項目の一部または全部をスペースや 0 のような指定した文字で満たします。

ID-3* またはリテラル-*1
サブジェクト・フィールド です。置換する文字を識別します。

ID-5* またはリテラル-*3
これは、置換フィールド (サブジェクト・フィールドを置換する項目) です。

置換対象フィールドと置換文字フィールドは、同じ長さでなければなりません。

CHARACTERS BY
CHARACTERS BY 句を使用する場合、置換フィールドは 1 文字位置の長さでなければなりません。

CHARACTERS BY を指定しても BEFORE または AFTER 句を指定しないと、 置換フィールドは被検査項目 (*ID-1*) 内の各文字を置換します。 置換は左端の文字位置から始まり右端へと続きます。

ALL
ALL が指定されていても BEFORE または AFTER 句が指定されていない場合、 置換フィールドは被検査項目 (*ID-1*) 内のサブジェクト・フィールドのオーバーラップしないオカレンスをそれぞれ置換します。置換は左端の文字位置から右端に続きます。

LEADING
LEADING を指定しても BEFORE または AFTER 句を指定しないと、 置換フィールドは、被検査項目 (*ID-1*) 内のサブジェクト・フィールドの、オーバーラップしない連続したオカレンスをそれぞれ置換します。その場合、 左端のこのようなオカレンスは、この置換フィールドが関与する最初の比較サイクルにおいて、 比較が開始される点に位置します。

FIRST
FIRST が指定されていても、BEFORE も AFTER 句も 指定されていない場合、置換フィールドは被検査項目 (*ID-1*) 内の サブジェクト・フィールドの左端のオカレンスを置換します。

TALLYING 句と REPLACING 句を両方とも指定している場合 (フォーマット 3)、INSPECT ステートメントの実行は、INSPECT TALLYING ステートメント (フォーマット 1) と そのすぐ後に INSPECT REPLACING ステートメント (フォーマット 2) が続いて指定されている場合と 同様に行われます。

次のような置換文字規則が適用されます。

- サブジェクト・フィールドが形象定数の場合、1 文字置換フィールドが、検査項目内の各文字を、 形象定数と等価に置換します。
- 置換フィールドが形象定数の場合、置換フィールドは、 検査項目内のサブジェクト・フィールドのオーバーラップしないオカレンスをそれぞれ置換します。
- サブジェクト・フィールドと置換フィールドが文字ストリングであるときは、 置換フィールド内に指定された文字ストリングが、 被検査項目内のサブジェクト・フィールドのオーバーラップしない各オカレンスを置換します。
- 検査項目内の所定の文字位置で置換が行われると、それ以降、 INSPECT ステートメントの実行中にはその文字位置の置換は行われなくなります。

BEFORE および AFTER 句 (すべてのフォーマット)

この句は、カウントする項目または置き換える項目の集合を制限します。

ALL、LEADING、CHARACTERS、FIRST あるいは CONVERTING の各句には、 BEFORE 句と AFTER 句は 1 つしか指定できません。

ID-4* およびリテラル-*2
これは、区切り文字 です。

区切り文字はカウントまたは置換は行われません。

INITIAL
指定の項目の最初のオカレンス。

BEFORE および AFTER 句はカウントおよび置換が行われる方法を変更します。

- BEFORE 句が指定されている場合、被検査項目 (*ID-1*) のカウントまたは置換は、左端の文字位置から始まり、区切り文字が最初に現れるまで続けられます。 被検査項目の中に区切り文字がなければ、カウントまたは置換は、右端の文字位置まで続けられます。
- AFTER 句が指定されている場合、被検査項目 (*ID-1*) のカウントまたは置換は、区切り文字の右側にある最初の文字位置から始まり、被検査項目の右端の文字位置まで続けられます。 被検査項目に区切り文字がなければ、カウントや置換文字は行われません。

CONVERTING 句 (フォーマット 4)

この句は、データ項目 (*ID-1*) 内にある特定の文字、または文字ストリングをすべて、 ユーザー指定の置換文字に変換します。

ID-6* またはリテラル-*4
置換される 文字ストリングを指定します。

同じ文字が リテラル-*4* または *ID-6* に複数回現れてはいけません。

ID-7* またはリテラル-*5
置換する 文字ストリングを指定します。

置換する文字ストリング (*ID-7* またはリテラル-*5*) は、置換される文字ストリング (*ID-6* またはリテラル-*4*) と同じサイズでなければなりません。

フォーマット 4 の INSPECT ステートメントは、同じ *ID-1* を指定している ALL 句 (リテラル-*4* の各文字ごとに 1 つ) を並べて記述したフォーマット 2 の INSPECT ステートメントであるかのように解釈され実行されます。 その効果は、リテラル-*4* の 1 文字が、それぞれリテラル-*1* として参照され、 それに対応してリテラル-*5* の 1 文字がリテラル-*3* として参照された場合と同じです。 リテラル-*4* の文字とリテラル-*5* の文字とは、 データ項目内の順序位置によって対応付けられます。

ID-4、*ID-6*、または *ID-7* が *ID-1* と同じストレージ域を占める場合、これらが同じデータ記述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です。

以下の表では、INSPECT ステートメントのオペランドとして使用可能なデータ項目の処理方法について説明しています。

表 *48.* **データ項目の内容の扱い**	
***ID-2* 以外の ID によって参照される際のカテゴリーの各項目の内容...**	**処理**
英数字または英字	英数字文字ストリングとして処理される。
DBCS	DBCS 文字ストリングとして処理される。
国別	国別文字ストリングとして処理される。
英数字編集、USAGE DISPLAY の数字編集、または USAGE DISPLAY の数字 (符号なし、外部 10 進数)	英数字文字ストリングを参照する INSPECT ステートメントで、英数字カテゴリーとして再定義される。
国別編集、USAGE NATIONAL の数字編集、または USAGE NATIONAL の数字 (符号なし、外部 10 進数)	国別文字ストリングを参照する INSPECT ステートメントで、国別カテゴリーとして再定義される。
USAGE DISPLAY の数字 (符号付き、外部 10 進数)	ID と同じ長さを持つ USAGE DISPLAY の符号なし外部 10 進数項目に移動されて、英数字文字ストリングを参照する INSPECT ステートメントで英数字カテゴリーとして再定義される。 記号が区切り文字の場合は、記号の入っているバイトは検査されず、したがって、 その文字の置き換えも行われない。 参照された項目が *ID-1* の場合は、置換または変換操作の結果生じたストリングは、*ID-1* へコピーされる。
USAGE NATIONAL の数字 (符号付き、外部 10 進数)	ID と同じ長さで USAGE NATIONAL の符号なし外部 10 進数項目に移動され、国別文字ストリングを参照する INSPECT ステートメントで、国別カテゴリーとして再定義される。 記号が区切り文字の場合は、記号の入っているバイトは検査されず、したがって、 その文字の置き換えも行われない。 参照された項目が *ID-1* の場合は、置換または変換操作の結果生じたストリングは、*ID-1* へコピーされる。
USAGE DISPLAY の外部浮動小数点	英数字文字ストリングを参照する INSPECT ステートメントで、英数字カテゴリーとして再定義される。
USAGE NATIONAL の外部浮動小数点	国別文字ストリングを参照する INSPECT ステートメントで、国別カテゴリーとして再定義される。

データ・フロー

BEFORE 句または AFTER 句を指定した場合を除き、検査は被検査項目 (*ID-1*) の左端の文字位置から始まり、右端まで 1 文字ずつ進められます。

以降の句の被比較数は、INSPECT ステートメントに指定されている順序で、 左から右へ比較されます。

- TALLYING (リテラル-*1* または *ID-3*、...)
- REPLACING (リテラル-*3* または *ID-5*、...)

ID が、添え字付けされていたり、参照変更であったり、 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、 INSPECT ステートメントの実行の中で最初の操作として 1 回だけ評価されます。

TALLYING および REPLACING の例については、「*COBOL for Linux on x86* プログラミング・ガイド」の『データ項目の計算および置換 *(INSPECT)*』を参照してください。

比較の周期

比較の周期は、このトピックで説明されているアクションから構成されます。

1. 最初の被比較項目が、被検査項目の左端から連続する同じ桁数の文字位置と比較されます。 両方が文字ごとに等しい場合にのみ、被比較項目は被検査文字に一致したことになります。

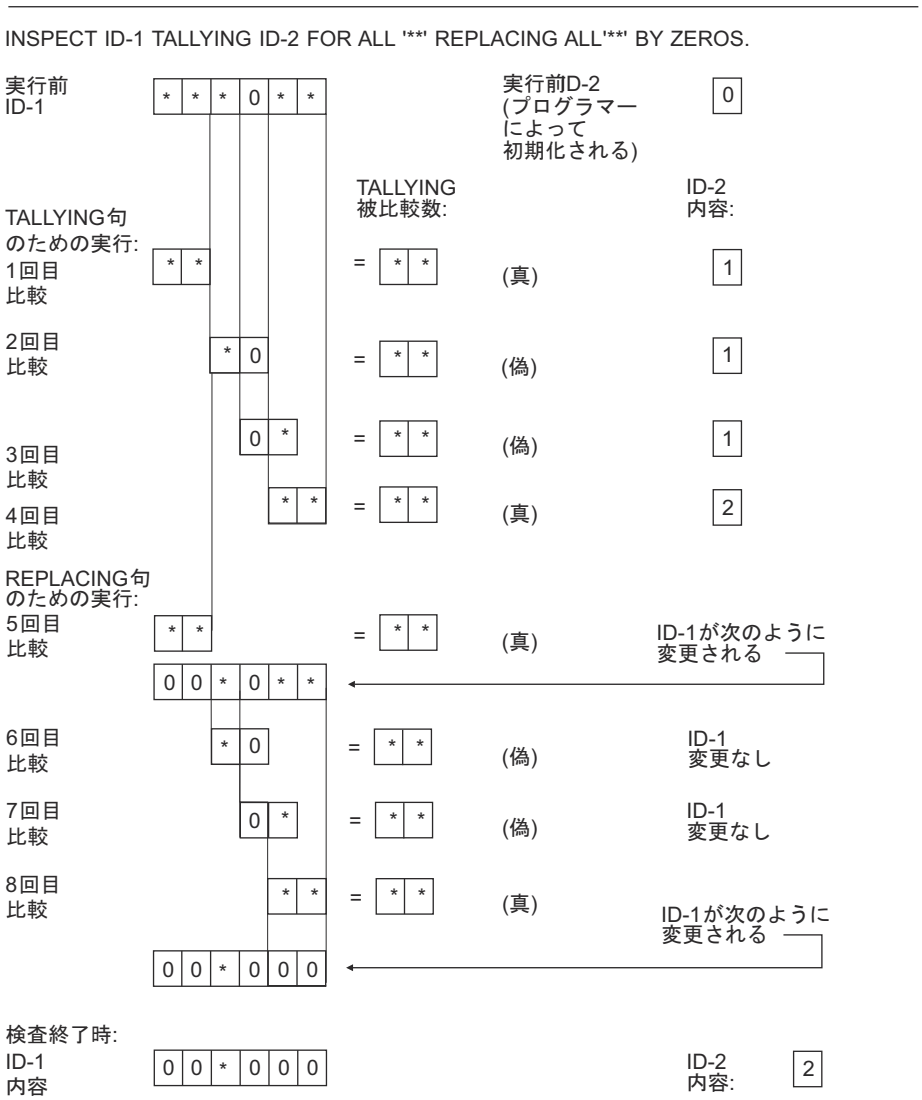
   CHARACTERS 句を指定した場合は、暗黙の 1 文字の被比較項目が使用されます。 暗黙の文字は、被検査項目の中の被検査文字と常に一致するものとみなされます。
2. 最初の被比較数に関して不一致となった場合、一致するものが見つかるか、 またはすべての被比較数が処理されるまで、 後続のそれぞれの被比較数について比較が繰り返されます。
3. 一致が検出されたかどうかによって、これらの処置が取られます。
   - 一致が検出される場合、TALLYING および REPLACING 句の記述内で説明されているとおりの累計または置換が行われます。

     被検査項目内により多くの文字位置がある場合は、 右端の一致する文字に続く最初の文字位置が、左端の文字位置であるとみなされます。 アクション 1 および 2 で述べた処理が繰り返されます。
   - 一致するものが見つからず、被検査項目に文字位置がさらにある場合は、 検査された文字の左端の後にある最初の文字位置が、今度は左端の文字位置であるとみなされます。 アクション 1 および 2 で述べた処理が繰り返されます。
4. 被検査項目内の右端の文字位置が、一致するかまたは左端の文字位置にあるとみなされるまで、 アクション 1 から 3 が繰り返されます。

BEFORE または AFTER 句が指定されている場合、319 ページの『BEFORE および AFTER 句 (すべてのフォーマット)』で説明したとおり、比較の周期は修正されます。 .

INSPECT ステートメントの例

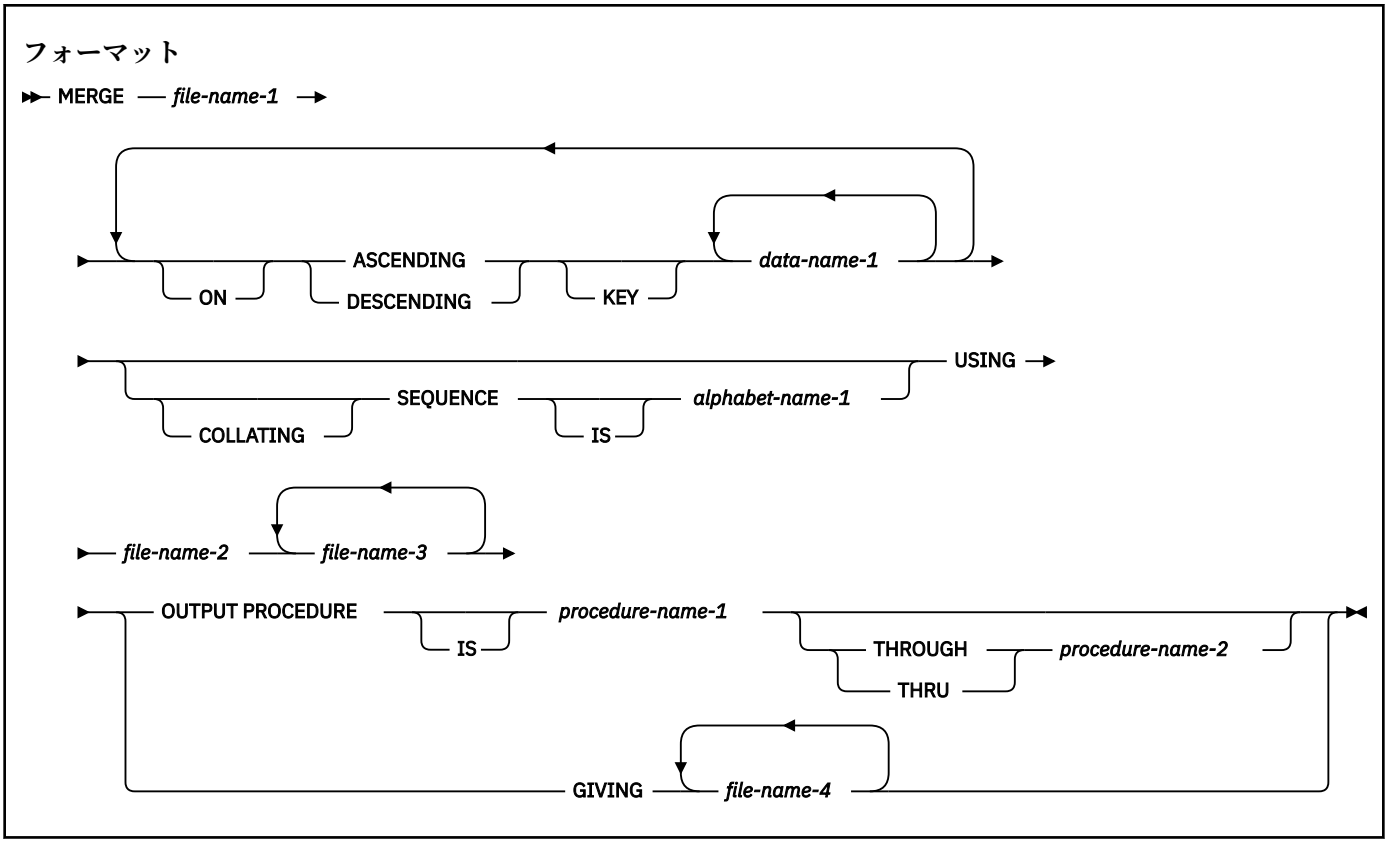
トピックは、INSPECT ステートメントの結果の例を示しています。



MERGE ステートメント

MERGE ステートメントは、1 つ以上のキーに基づいて、複数の同じシーケンスで並べられたファイル (すなわち、同じ 1 組の昇順または降順キーに従って既にソートされているファイル) を結合して、出力プロシージャーまたは出力ファイルでレコードをマージした順序で使用できるようにします。

MERGE ステートメントは、宣言セクション以外ならば、PROCEDURE DIVISION のどこにでも置くことができます。



ファイル名-1
マージするレコードを記述する SD 項目の中で指定された名前。

MERGE ステートメントの中で同じファイル名を繰り返すことはできません。

MERGE ステートメントの中で対になるどのファイル名も、同じ SAME AREA 節、 SAME SORT AREA 節、 または SAME SORT-MERGE AREA 節の中で指定することはできません。 同じ SAME RECORD AREA 節内では、MERGE ステートメントに任意のファイル名を指定できます。

MERGE ステートメントが実行されると、ファイル名-*2*、ファイル名-*3* などに含まれているすべてのレコードがマージ・プログラムに受け取られ、その後指定されたキー (複数のキーも可) に従ってマージされます。

ASCENDING/DESCENDING KEY 句

この句は、指定されたマージ・キーに基づいて、 レコードを昇順または降順 (どちらになるかは指定された句次第) で処理することを指定します。

データ名-1
マージを行う際に基準となる KEY データ項目を指定します。 そのようなデータ名はそれぞれ、ファイル名-*1* に関連するレコードの中のデータ項目を識別する必要があります。 KEY という語の後に置かれるデータ名は、これらがどのように KEY 句に分割されるかに関係なく、 キーのレベルが高いものから順に MERGE ステートメントの中で左から右へ列挙されていきます。 左端のデータ名が最もレベルの高いキーとなり、次のデータ名が 2 番目のレベルのキーとなる、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、各入力ファイルの中で、物理的に同じ位置になければならず、 また同じデータ・フォーマットを持っていなければなりません。 しかし、同じデータ名を持っている必要はありません。
- ファイル名-*1* が 2 つ以上のレコード記述を持つ場合には、KEY データ項目は、 どちらか一方のレコード記述の中にのみ記述されている必要があります。

- ファイル名-*1* が可変長レコードを含んでいる場合には、KEY データ項目はすべて、 レコードの最初の *n* 個の文字位置内に入っていなければなりません (*n* はファイル名-*1* で指定されている最小レコード・サイズ)。
- KEY データ項目は、OCCURS 節を含んでいたり、OCCURS 節を含む項目に従属していたりすることはできません。
- KEY データ項目には、以下を指定できません。
  - 可変位置項目
  - 可変オカレンス・データ項目を含むグループ項目
  - ウィンドウ表示日付フィールド
  - USAGE NATIONAL で記述された数字カテゴリー (国別 10 進数タイプ)
  - USAGE NATIONAL で記述された外部浮動小数点カテゴリー (国別浮動小数点)
  - DBCS カテゴリー
- KEY データ項目は、 修飾することができます。
- KEY データ項目は、以下のデータ・カテゴリーのいずれかにすることができます。
  - 英字、英数字、英数字編集
  - 数字 (USAGE NATIONAL の数字を除く)
  - 数字編集 (USAGE DISPLAY または NATIONAL)
  - 内部浮動小数点または display 浮動小数点
  - NCOLLSEQ(BINARY) コンパイラー・オプションが有効な場合は、国別または国別編集。 バイナリー照合シーケンスが国別キーに適用されます。

ファイル名-*4* が、基本レコード・キーがレコード・キー名ではない索引付きファイルを参照している場合、データ名-*1* の最初の指定は ASCENDING 句に関連付けられていなければならず、そのデータ名-*1* によって参照されるデータ項目は、そのファイルの基本レコード・キーに関連付けられているデータ項目と同じ文字位置をこのレコード内で占めていなければなりません。

ファイル名-*4* が索引付きファイルを参照しており、その索引付きファイルの基本レコード・キーがレコード・キー名 である場合は、レコード・キー名 の SOURCE 句内の各データ名に対応するデータ名-*1* を指定する必要があります。各データ名-*1* は ASCENDING 句に関連付けられていなければならず、各データ名-*1* は SOURCE 句内の対応するデータ名と同じ文字位置を占めていなければなりません。

レコード・キー名について詳しくは、103 ページの『FILE-CONTROL 段落』を参照してください。

マージ処理の方向は、次に示すように ASCENDING または DESCENDING のどちらのキーワードを指定するかによって異なります。

- ASCENDING を指定すると、最低のキー値から最高のキー値へのシーケンスとなります。
- DESCENDING を指定すると、最高のキー値から最低のキー値へのシーケンスとなります。

KEY データ項目が英字、英数字、英数字編集、または数字編集の場合は、キー値のシーケンスは使用されている照合シーケンスによって決まります (以下の 325 ページの『COLLATING SEQUENCE 句』を参照)。

KEY データ項目が USAGE NATIONAL で記述されている場合、KEY 値のシーケンスは国別文字の 2 進値に基づきます。

KEY が USAGE DISPLAY を使用した外部浮動小数点項目の場合、キーは英数字カテゴリーとみなされます。レコードがマージされるシーケンスは、使用している照合シーケンスによって決まります。

KEY が USAGE NATIONAL を使用した外部浮動小数点項目の場合、キーは NATIONAL カテゴリーとみなされます。

KEY が内部浮動小数点項目の場合、キー値のシーケンスは数値順になります。

COLLATING SEQUENCE 句が指定されていない場合は、 比較条件のオペランド比較規則に従ってキーが比較されます。 詳細については、243 ページの『一般比較条件』を参照してください。

COLLATING SEQUENCE 句が指定されている場合は、英字、英数字、英数字編集、外部浮動小数点、および数字編集の各カテゴリーのキー・データ項目に対して、指定された照合シーケンスが使用されます。 その他すべてのキー・データ項目に対しては、比較条件でのオペランドの比較規則に従って比較が行われます。

COLLATING SEQUENCE 句

この句によって、このマージ処理で KEY データ項目に対して行われる英数字比較で 使用する照合シーケンスを指定します。

COLLATING SEQUENCE 句は、英字または英数字以外のキーには影響を与えません。

COLLATING SEQUENCE 句は、1 バイト ASCII コード・ページが有効な場合にのみ有効です。

英字名-*1*
これは SPECIAL-NAMES 段落の ALPHABET 節で指定されている必要があります。 英字名節の句のうちいずれか 1 つを指定することができ、次のようになります。

STANDARD-1
照合シーケンスは、文字の 16 進値の順序に基づきます。

STANDARD-2
照合シーケンスは、文字の 16 進値の順序に基づきます。

NATIVE
ランタイム・ロケールによって指示された照合シーケンスが選択されます。

EBCDIC
EBCDIC 照合シーケンスがすべての英数字比較のために使用されます。 (EBCDIC 照合シーケンスは、535 ページの『EBCDIC 照合シーケンス』に示されています。)

リテラル (literal)
ALPHABET-NAME 節でリテラルを指定したことにより設定された照合シーケンスが、すべての英数字比較のために使用されます。

COLLATING SEQUENCE 句を省略した場合は、 OBJECT-COMPUTER 段落の PROGRAM COLLATING SEQUENCE 節 (指定されている場合) で使用したい照合シーケンスを識別します。 MERGE ステートメントの COLLATING SEQUENCE 句および OBJECT-COMPUTER 段落の PROGRAM COLLATING SEQUENCE 節を両方とも省略した場合には、使用される照合シーケンスは COLLSEQ コンパイラー・オプションで指定されます。 COLLSEQ(EBCDIC) が指定された場合は、EBCDIC 照合シーケンスが使用されます。
COLLSEQ(LOCALE) が指定された場合は、ロケールによって指示された照合シーケンスが使用されます。 ロケールの詳細については、563 ページの『付録 H ロケールの考慮事項』を参照してください。

USING 句

ファイル名-*2* 、 ファイル名-*3* 、 ...
これは入力ファイルを指定します。

MERGE 操作中、ファイル名-*2*、ファイル名-*3*、... (入力ファイル) にあるレコードはすべてファイル名-*1* に転送されます。 MERGE ステートメントの実行時に、これらのファイルがオープンされていてはなりません。 入力ファイルは自動的にオープンされ、読み取られ、クローズされます。 これらのファイルの入力操作に DECLARATIVE プロシージャーが指定されていると、エラーが起きた場合には宣言はエラーとなります。

入力ファイルはいずれも、順次アクセス・モードまたは動的アクセス・モードを指定し、 DATA DIVISION の中の FD 項目に記述され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、入力ファイル (ファイル名-*2*、ファイル名-*3*、...) に含まれるレコードのサイズは、ファイル名-*1* に記述されている最小レコード以上かつ最大レコード以下でなければなりません。 ファイル名-*1* に固定長レコードが含まれている場合、入力ファイルに含まれるレコ

ードのサイズは、ファイル名-*1* に対して記述されている最大レコード以下でなければなりません。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『ファイルのソートおよびマージ』を参照してください。

GIVING 句

ファイル名-*4* 、 ...
これは出力ファイルを指定します。

GIVING 句が指定されたとき、ファイル名-*1* にあるマージされたレコードはすべて、 自動的に出力ファイル (ファイル名-*4*, ...) に転送されます。

出力ファイルはいずれも、順次アクセス・モードまたは動的アクセス・モードを指定し、 DATA DIVISION の中の FD 項目に記述されていなければなりません。

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、ファイル名-*1* に含まれるレコードのサイズは、その出力ファイルについて記述された最小レコード以上かつ最大レコード以下でなければなりません。 出力ファイルに固定長レコードが含まれている場合、ファイル名-*1* に含まれるレコードのサイズは、出力ファイルについて記述された最大レコードを超えてはいけません。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『ファイルのソートおよびマージ』を参照してください。

MERGE ステートメントの実行時に、出力ファイル (ファイル名-*4*、...) がオープンされてはなりません。 出力ファイルは自動的にオープンされ、その出力ファイルに書き込みが行われて、その出力ファイルがクローズされます。 これらのファイルの出力操作に DECLARATIVE プロシージャーが指定されていると、エラーが起きた場合には宣言はエラーとなります。

OUTPUT PROCEDURE 句

この句は、マージ処理から得られた出力レコードを選択または変更する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*
OUTPUT PROCEDURE の中の最初の (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 の中の最後のセクションまたは段落を指定します。

OUTPUT PROCEDURE は、ファイル名-*1* によって参照されたファイルから RETURN ステートメントによって 1 つずつ使用可能にされるレコードをマージ順に選択、修正、またはコピーするために必要な、任意のプロシージャーから構成することができます。 この範囲には、出力プロシージャーの範囲内で CALL、EXIT、GO TO、PERFORM、および XML PARSE ステートメントによって制御が移動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、この範囲には、出力プロシージャーの範囲内のステートメントが実行されると実行される宣言型プロシージャーの中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内で、MERGE ステートメント、RELEASE ステートメント、SORT ステートメントを実行することはできません。

出力プロシージャーが指定されていると、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されてから、制御はこの出力プロシージャーに渡されます。 コンパイラーは、出力プロシージャー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。 制御がこの出力プロシージャーの中の最後のステートメントに移ると、 この戻りメカニズムによってマージ処理が終了し、ついで制御を MERGE ステートメントの後ろにある実行可能ステートメントに渡します。 出力プロシージャーに入る前に、マージ・プロシージャーは要求されたとき、 次のレコードをマージされた順に選択できる段階になっています。 出力プロシージャーの中の RETURN ステートメントは、次のレコードを要求することになります。

OUTPUT PROCEDURE 句は、基本的な PERFORM ステートメントと似ています。 例えば、OUTPUT PROCEDURE 句の中でプロシージャー名を指定すると、そのプロシージャーは、 それが PERFORM ステートメントで指定された場合と同様にして、マージ操作時に実行されます。 PERFORM ステートメントによる場合と同様に、プロシージャーの実行は、 最後のステートメントがその実行を終えると終了します。 OUTPUT PROCEDURE 句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 (306 ページの『EXIT ステートメント』を参照).

MERGE 特殊レジスター

このトピックでは、MERGE ステートメントの特殊レジスターについて説明します。

SORT-CONTROL 特殊レジスター
ソート制御ファイル (これによりソート/マージ機能に追加を指定できます) は、SORT-CONTROL 特殊レジスターで識別します。

ソート制御ファイルを使用して制御ステートメントを指定する場合は、 ソート制御ファイルの中に指定されている値がその他の SORT 特殊レジスターにある値に優先します。

詳しくは、21 ページの『SORT-CONTROL』を参照してください。

SORT-MESSAGE 特殊レジスター
詳細については、21 ページの『SORT-MESSAGE』を参照してください。 特殊レジスターの SORT-MESSAGE は、ソート制御ファイルの中にある制御ステートメント用のオプションのキーワードと同じ働きをします。

SORT-RETURN 特殊レジスター
詳細については、22 ページの『SORT-RETURN』を参照してください。

セグメント化に関する考慮事項

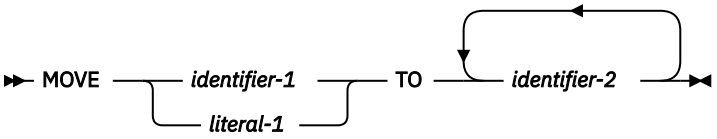
固定セグメントで MERGE ステートメントがコード化された場合、この MERGE ステートメントが参照するすべての出力プロシージャーは完全に固定セグメントの範囲内にあるか、プロシージャー全体が単一の独立セグメントに含まれている必要があります。

独立セグメントで MERGE ステートメントがコード化された場合、この MERGE ステートメントが参照するすべての出力プロシージャーは完全に固定セグメントの範囲内にあるか、プロシージャー全体が MERGE ステートメントと同じ独立セグメントに含まれている必要があります。

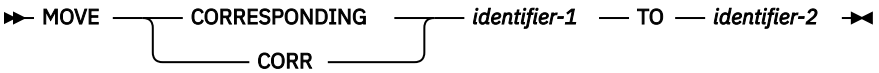
MOVE ステートメント

MOVE ステートメントは、データをあるストレージ域から別の 1 つまたは 複数のストレージ域に転送します。

フォーマット 1: MOVE ステートメント



フォーマット 2: CORRESPONDING 句を含む MOVE ステートメント



CORR は、CORRESPONDING の省略形で、意味は同じです。

***ID-1*、リテラル-1**
送り出し領域。

ID-2
受け取り領域。 *ID-2* は組み込み関数を参照してはなりません。

フォーマット 1 を指定する場合:

- *ID-1* または *ID-2* の一方が国別グループ項目を参照し、もう一方のオペランドが英数字グループ項目を参照するときには、国別グループは 1 つのグループ項目として処理されます。それ以外の場合には、国別グループ項目は国別カテゴリーの基本データ項目として処理されます。

- 送り出し領域のデータは、*ID-2* データ項目が MOVE ステートメントに指定された順に *ID-2* のそれぞれによって参照されるデータ項目の中に移動されます。 以下の 328 ページの『基本移動』および 333 ページの『グループ移動』を参照してください。

フォーマット 2 を指定する場合:

- ID は両方ともグループ項目でなければなりません。
- 国別グループ項目はグループ項目として (国別カテゴリーの基本データ項目としてではなく) 処理されます。
- *ID-1* で選択された項目が、264 ページの『CORRESPONDING 句』の規則に従って、*ID-2* へ移動されます。 結果は、CORRESPONDING ID のそれぞれの対が、 別々の MOVE ステートメントで参照された場合と同じです。

以下のタイプの USAGE で記述されたデータ項目は、MOVE ステートメントに指定できません。

- INDEX
- POINTER
- FUNCTION-POINTER
- PROCEDURE-POINTER

USAGE INDEX、POINTER、FUNCTION-POINTER、または PROCEDURE-POINTER で定義されたデータ項目は、MOVE CORRESPONDING ステートメント内で参照される英数字グループ項目に含めることができます。ただし、これらのデ ータ項目からデータが移動されることはありません。

送り出し領域または受け取り領域の長さの評価は、OCCURS 節の DEPENDING ON 句の影響を受けます (173 ページの『OCCURS 節』を参照)。

送り出しフィールド (*identifier-1*) が参照による変更、添え字付き、もしくは英数字、数字、整数、または国別関数 ID、参照修飾子、添え字、または関数は、データが受け取りオペランドの 先頭に移動される直前に一度だけ評価されます。

受け取りフィールド (*ID-2*) に関連する長さの計算、添え字付け、または参照による修正は、 データがその受け取りフィールドに移動される直前に評価されます。

例えば、

```
MOVE A(B) TO B, C(B).
```

のステートメントは、次のステートメントと同じ結果になります。

```
MOVE A(B) TO TEMP.
MOVE TEMP TO B.
MOVE TEMP TO C(B).
```

ここで TEMP は、中間結果項目として定義されています。 添え字 B は、 最初の移動が行われた時と C(B) の移動が最後に実行された時とでは、 値が変わっています。

中間結果について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『付録 *A.* 中間結果および算術精度』を参照してください。

MOVE ステートメントを実行した後も、送り出しフィールドには、実行前と同じデータが入っています。

使用上の注意: MOVE ステートメント内にオーバーラップ・オペランドがあると、予測できない結果が生じる可能性があります。

基本移動

基本移動とは、受け取り項目が基本データ項目であり、送り出し項目が基本データ項目またはリテラルである移動のことです。

有効なオペランドは、以下のいずれかのカテゴリーに属します。

- **英字**: 英字カテゴリーのデータ項目および形象定数 SPACE が含まれます。

- **英数字**: 以下の項目が含まれます。
  - 英数字カテゴリーのデータ項目
  - 英数字関数
  - 英数字リテラル
  - 形象定数 ALL 英数字リテラルおよび NULL を除くその他すべての形象定数 (英数字送り出し項目を必要とする文脈で使用される場合)
- **英字数字編集**: 英数字編集カテゴリーのデータ項目が含まれます。
- **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。
- **日時**: 日時クラスの日付データ項目、時刻データ項目、およびタイム・スタンプ・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。
- **DBCS**: DBCS カテゴリーのデータ項目、DBCS リテラル、および形象定数 ALL DBCS リテラルが含まれます。
- **外部浮動小数点**: 外部浮動小数点カテゴリーのデータ項目 (USAGE DISPLAY または USAGE NATIONAL を指定して記述) および浮動小数点リテラルが含まれます。
- **内部浮動小数点**: 内部浮動小数点カテゴリーのデータ項目 (USAGE COMP-1 または USAGE COMP-2 として定義) が含まれます。
- **国別**: 以下の項目が含まれます。
  - 国別グループ項目 (国別カテゴリーの基本項目として扱われる)
  - 国別カテゴリーのデータ項目
  - 国別リテラル
  - 国別関数
  - 形象定数 ZERO、SPACE、QUOTE、および ALL 国別リテラル (国別送り出し項目を必要とする文脈で使用される場合)
- **国別編集**: 国別編集カテゴリーのデータ項目が含まれます。
- **数字**: 以下の項目が含まれます。
  - 数字カテゴリーのデータ項目
  - 数字リテラル
  - 形象定数 ZERO (ZERO が数字または数字編集項目へ移動される場合)
- **数字編集**: 数字編集カテゴリーのデータ項目が含まれます。

基本移動の規則

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 (必要な場合)。 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。 英数字文字へ、または英数字文字からの変換で使用されるコード・ページは、実行時に特定のデータ項目に適用可能なコード・ページです。

次の規則は、有効な基本移動がどのように実行されるかを示します。 受け取りフィールドは、以下のとおりです。

英字:

- 位置合わせと必要なスペースの埋め込みまたは切り捨ては、142 ページの『位置合わせの規則』の説明のようにして行われます。.
- 送り出し項目のサイズが、受け取り項目のサイズより大きければ、受け取り項目がいっぱいになった後は、 右端の余分の文字が切り捨てられます。

英数字または**英数字編集**:

- 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。

- 位置合わせと必要なスペースの埋め込みまたは切り捨ては、142 ページの『位置合わせの規則』の説明のようにして行われます。
- 送り出し項目のサイズが、受け取り項目のサイズより大きければ、受け取り項目がいっぱいになった後は、 右端の余分の文字が切り捨てられます。
- 最初の送り出し項目が演算符号を持つ場合は、符号なしの値が使用されます。 演算符号が別の 1 文字を占有している場合には、その文字は移動されず、送り出し項目のサイズは実際のサイズよりも 1 文字分だけ小さいとみなされます。
- 送り出し項目がブールの場合は、長さ 1 の英数字項目として送り出し項目が記述されたかのように、データが移動されます。
- 送り出し項目が日時の場合、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取り項目に移動されます。 送り出し日時項目が PACKED-DECIMAL の USAGE を持つ場合、この項目は最初に DISPLAY の USAGE に変換されます。

ブール:

- ブール受け取り項目の場合、送り出し項目の最初のバイトのみが移動されます。
- 送り出し項目が英数字の場合は、送り出し項目の最初の文字が移動されます。 文字「0」および「1」はそれぞれ、ブール値 B"0" および B"1" に相当します。
- 送り出し項目が ZERO の場合は、ブール・リテラル B"0" として扱われます。

DBCS:

- 送り出し項目と受け取り項目のサイズが異なる場合、送り出しデータは右側で切り捨てられるか、右側が DBCS スペースで埋められます。 埋め込み要求が 2 バイト文字で重複整合がない場合、1 バイト文字が使用されます (例えば、英数字グループ項目に DBCS データを移動した場合など)。

日時:

- 送り出し項目が日時の場合、日時送り出し項目は、その形式がまず受け取り項目の形式に変換され、次に項目が移動されます。 送り出し項目がタイム・スタンプで、受け取り項目が日付項目または時刻項目の場合は、タイム・スタンプ項目の日付または時刻の部分のみが受け取り項目へ移動されます。 送り出し項目が日付項目または時刻項目で、受け取り項目がタイム・スタンプの場合は、タイム・スタンプの日付または時刻の部分のみが置き換えられます。
- 送り出し項目が数値の場合、各受け取り項目の数値変換指定子は、送り出し項目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の数字から始まります。 英数字変換指定子はすべて、デフォルト値を持ちます。
- 送り出し項目が数字編集の場合、数字編集項目は編集解除されます。 結果の数値は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の場合、日時受け取り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。

外部浮動小数点:

- 浮動小数点の送り出し項目の場合、浮動小数点値は受け取り側の外部浮動小数点項目の USAGE に変換されます (送り出し項目の表現と異なる場合)。
- その他の送り出し項目の場合は、値が内部浮動小数点に変換されてから、受け取り側の外部浮動小数点項目の USAGE に変換されるように、数値が処理されます。

内部浮動小数点:

- 送り出しオペランドのカテゴリーが内部浮動小数点ではない場合、送り出し項目の数値は内部浮動小数点フォーマットに変換されます。

国別 または **国別編集**:

- 送り出し項目の表現が国別文字ではない場合、その送り出しデータは国別文字に変換され、切り捨てや埋め込みが行われることのない長さの、国別カテゴリーの一時データ項目に移動されるように扱われます。生成された国別カテゴリーのデータ項目は、送り出しデータ項目として扱われます。
- 送り出し項目の表現が国別文字の場合は、送り出しデータが変換されずに使用されます。

- 位置合わせと必要なスペースの埋め込みまたは切り捨ては、142 ページの『位置合わせの規則』の説明のようにして行われます。 プログラマーは、1 つの図形文字を形成する複数のエンコード・ユニットが切り捨てによって分離されることのないようにする必要があります。
- 送り出し項目が演算符号を持つ場合には、符号なしの値が使われます。 演算符号が別の 1 文字を占有している場合には、その文字は移動されず、送り出し項目のサイズは実際のサイズよりも 1 文字分だけ小さいとみなされます。

数字または**数字編集**:

- 編集上の必要により 0 が置き換えられる場合を除き、小数点による位置合わせと 0 による埋め込み (必要な場合) が行われます。詳しくは、142 ページの『位置合わせの規則』を参照してください。
- 受け取り項目に記号が付いていれば、送り出し項目の記号は記号変換されてから (必要な場合)、 受け取り項目に入れられます。 送り出し項目が符号なしであるときは、受け取り項目に対して正の演算符号が生成されます。
- 受け取り項目が符号なしのときは、受け取り項目に対して演算符号は生成されず、移動では送り出し項目の絶対値が使用されます。
- 送り出し項目のカテゴリーが英数字、英数字編集、国別、または国別編集のときは、送り出し項目が符号なし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。
- 送り出し項目が浮動小数点であるときは、データはまず 2 進数かまたは内部 10 進表記に変換されてから移動されます。
- 受け取り項目が数字編集のときは、その受け取り項目に関連付けられた PICTURE 文字ストリングまたは BLANK WHEN ZERO 節で定義されているように編集が行われます。
- 送り出し項目が数字編集のときは、コンパイラーは送り出しデータを編集解除して、数字編集項目の未編集の値を設定します (この値は符号付きにすることができます)。 数字または数字編集の受け取りデータ項目への移動では、未編集の数値が使用されます。

使用上の注意:

1. 受け取り項目のカテゴリーが英数字、英数字編集、数字編集、国別、または国別編集のときに、送り出しフィールドが数字の場合は、ピクチャー記号 P を指定して記述された送り出し項目内のすべての桁位置はゼロの値を持つとみなされます。 それぞれの P は、送り出し項目サイズの計算に入れられます。
2. 受け取り項目が数字で、送り出しフィールドが英数字リテラル、国別リテラル、または ALL リテラルの場合は、そのリテラルのすべての文字は数字でなければなりません。

有効な基本移動と無効な基本移動

表は、それぞれのカテゴリーごとに有効な基本移動と無効な基本移動を 示したものです。

この表では、「はい」と「いいえ」は次のような意味です。

- はい = 移動が有効である。
- いいえ = 移動は無効である。
- 列見出しは受け取り項目のカテゴリーを示します。行見出しは送り出し項目のカテゴリーを示します。

表 *49.* **有効な基本移動と無効な基本移動**

送り出し項目カテゴリー	受け取り項目カテゴリー												
	英字	英数字	英数字編集	ブール	日付	時刻	タイム・スタンプ	数字	数字編集	外部浮動小数点	内部浮動小数点	DBCS [1]	国別、国別編集
英字および SPACE	はい	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい
英数字 [2]	はい	はい	はい	はい [10]	はい	はい	はい	はい [3]	はい [3]	はい [8]	はい [8]	いいえ	はい
英数字編集	はい	はい	はい	いいえ	はい	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	はい
ブール [11]	いいえ	はい	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ

表 49. **有効な基本移動と無効な基本移動** (続き)

送り出し項目カテゴリー	受け取り項目カテゴリー												
	英字	英数字	英数字編集	ブール	日付	時刻	タイム・スタンプ	数字	数字編集	外部浮動小数点	内部浮動小数点	DBCS [1]	国別、国別編集
日付	いいえ	はい	はい	いいえ	はい	いいえ	はい	はい	はい	いいえ	いいえ	いいえ	いいえ
時刻	いいえ	はい	はい	いいえ	いいえ	はい	はい	はい	はい	いいえ	いいえ	いいえ	いいえ
タイム・スタンプ	いいえ	はい	はい	いいえ	はい	はい	はい	はい	はい	いいえ	いいえ	いいえ	いいえ
整数 [4]	いいえ	はい	はい	いいえ	はい	はい	はい	はい	はい	はい	はい	いいえ	はい
ゼロ [4]	いいえ	はい	はい	はい	いいえ	いいえ	いいえ	はい	はい	はい	はい	いいえ	はい
非整数 [5]	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい	はい	はい	はい	いいえ	いいえ
数字編集	いいえ	はい	はい	いいえ	はい	はい	はい	はい	はい	はい	はい	いいえ	はい
浮動小数点 [6]	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい	はい	はい	はい	いいえ	いいえ
DBCS[7]	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい	はい
国別 [9]	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい	はい	はい	はい	いいえ	はい
国別編集	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい

1. DBCS データ項目を含む。
2. 英数字リテラルを含む。
3. 形象定数と英数字リテラルは、数字だけで構成しなければならず、数字整数フィールドとして扱われる。
4. 整数の数字リテラルを含む。
5. 非整数の数字リテラルを含む。
6. 浮動小数点リテラル、外部浮動小数点データ項目 (USAGE DISPLAY または USAGE NATIONAL)、 および内部浮動小数点データ項目 (USAGE COMP-1 または USAGE COMP-2) を含む。
7. DBCS データ項目、DBCS リテラル、および形象定数 SPACE を含む。
8. 形象定数と英数字リテラルは、数字だけで構成しなければならず、数字整数フィールドとして扱われる。 リテラル ALL は、送り出し項目として使用することはできない。
9. 国別データ項目、国別リテラル、国別関数、および形象定数 ZERO、SPACE、QUOTE、および ALL 国別リテラルを含む。
10. 送り出し項目の先頭文字は、その値に関係なく移動される。
11. ブール・リテラルが含まれる。

日付フィールドが関係する移動

送り出し項目を年末尾型日付フィールドとして指定した場合は、 受け取りフィールドもすべて年末尾型日付フィールドにし、 その日付フォーマットを送り出し項目と同じにしなければなりません。また、年末尾型日付フィールドを受け取り項目として指定した場合は、 送り出し項目を非日付データまたは年末尾型日付フィールドのいずれかにし、 その日付フォーマットを受け取り項目と同じにしなければなりません。そのようにするなら、どちらの場合も移動が実行され、 項目がすべて非日付データになっている場合と同じ結果になります。

333 ページの表 50 は、非年末尾型日付フィールドが関係する移動の動作を示しています。 送り出し項目が日付フィールドである場合、受け取りフィールドは互換日付フィールドでなければなりません。送り出し項目と受け取り項目の両方が日付フィールドである場合は、 それらに互換性がなければなりません。すなわち、 ウィンドウ表示西暦年でも拡張西暦年でも可能な年部分を除いて、同じ日付フォーマットにする必要があります。

この表では、移動を記述するのに以下の用語を使用しています。

通常
送り出し項目と受け取り項目の両方が非日付データであるかのように、 日付感知動作なしで移動が実行されます。

拡張
ウィンドウ表示日付フィールドの送り出し項目は、162 ページの『ウィンドウ表示日付フィールドのセマンティクス』で説明されているように、 最初に拡張形式に変換されたかのように扱われます。

無効
移動は許可されません。

表 *50.* **日付フィールドが関係する移動**

	非日付データ受け取り項目	ウィンドウ表示日付フィールド受け取り項目	拡張日付フィールド受け取り項目
非日付データ送り出し項目	通常	通常	通常
ウィンドウ表示日付フィールド送り出し項目	無効	通常	拡張
拡張日付フィールド送り出し項目	無効	通常 [1]	通常

1. 拡張日付フィールドからウィンドウ表示日付フィールドに移動すると、拡張日付フィールドの世紀部分が切り捨てられることになるため、 それは実際には「ウィンドウ表示」移動になります。 移動が英数字の場合は、受け取りウィンドウ表示日付フィールドはそのデータ記述に JUSTIFIED RIGHT を指定した場合と同じ方法で処理されます。これは、 受け取りウィンドウ表示日付フィールドがグループ項目 (JUSTIFIED 節を指定できない) である場合も同様です。

ファイル・レコード域が関係する移動

あるファイルに対して OPEN ステートメントが正常に実行されると、そのファイルのレコード域が使用可能になります。 ファイルに関連したレコード記述項目との間でデータをやり取りできるのは、そのファイルがオープン状態になっている場合のみです。

暗黙的または明示的な CLOSE ステートメントを実行すると、ファイルがオープン状態から除去され、レコード域が使用不可になります。

グループ移動

グループ移動とは、送り出し項目または受け取り項目、あるいはその両方が英数字グループ項目であるような移動のことです。

グループ移動:

- 以下の項目のいずれかから英数字グループ項目への移動
  - MOVE ステートメント内で送り出し項目として有効な任意の基本データ項目
  - 国別グループ項目
  - リテラル
  - 形象定数
- 英数字グループ項目から以下の項目のいずれかへの移動
  - MOVE ステートメント内で受け取り項目として有効な任意の基本データ項目
  - 国別グループ項目
  - 英数字グループ項目

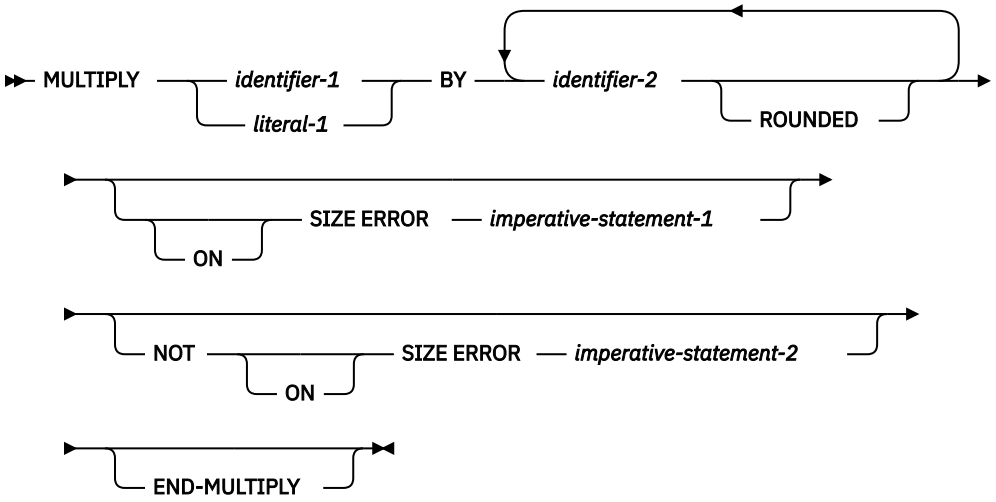
グループ移動は、ある内部表現の形式から別の形式へのデータ変換を行わないことを除けば、英数字から英数字への基本移動と同じように扱われます。 グループ移動では、送り出し領域または受け取り領域に含

まれている個々の基本項目には関係なく、受け取り領域にデータが入れられます。 ただし、OCCURS 節で注記されている場合を除きます。 (173 ページの『OCCURS 節』を参照してください。).)

MULTIPLY ステートメント

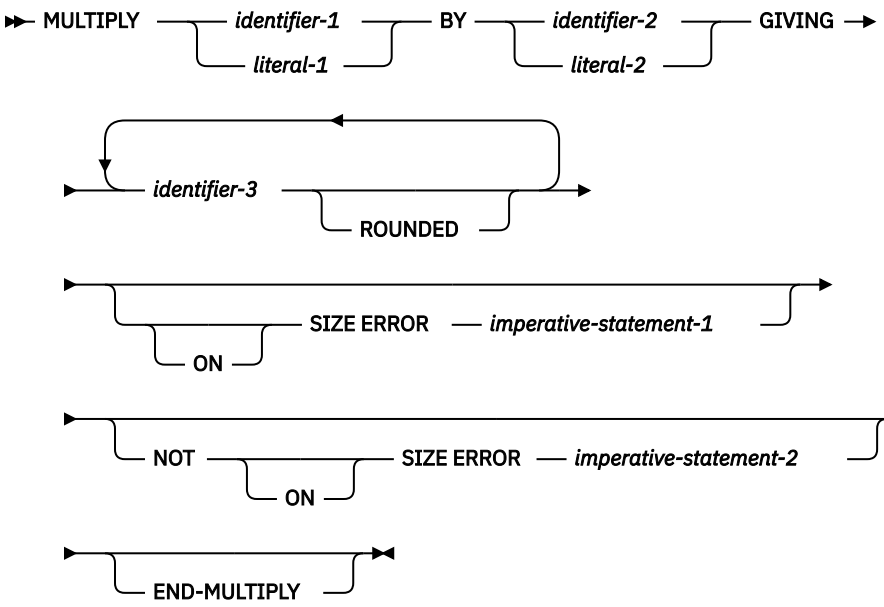
MULTIPLY ステートメントは、数字項目を乗算し、その結果をデータ項目の値として 設定します。

フォーマット 1: MULTIPLY ステートメント



フォーマット 1 では、*ID-1* またはリテラル-*1* の値は、*ID-2* の値によって乗算され、その積は *ID-2* に入れられます。 *ID-2* が連続して現れるたびに、*ID-2* が指定されている順に左から右へと乗算が行われます。

フォーマット 2: GIVING 句を含む MULTIPLY ステートメント



フォーマット 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* の値で乗算されます。 その後その積は、*ID-3* によって参照されるデータ項目の中に保管されます。

すべてのフォーマットに関して次のことが言えます。

ID-1*、*ID-2
基本数字項目である必要があります。 *ID-1* および *ID-2* は日付フィールドにはできません。

literal-1*、*literal-2
これは、数字リテラルでなければなりません。

フォーマット 2 について

ID-3
基本数字項目または数字編集項目を指定する必要があります。

GIVING 句の ID である *ID-3* は、 MULTIPLY ステートメントの中で日付フィールドを使用することができる唯一の ID です。

ID-3 名が日付フィールドである場合、*ID-3* に積がどのように保管されるかについては、237 ページの『日付フィールドに関連する算術演算結果の保管』を参照してください。

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。

ARITH(COMPAT) コンパイラー・オプションが有効な場合は、オペランドの合成が最大 30 桁になります。 ARITH(EXTEND) コンパイラー・オプションが有効な場合は、オペランドの合成が最大 31 桁になります。 詳しくは、267 ページの『算術ステートメントのオペランド』、および「*COBOL for Linux on x86* プログラミング・ガイド」の『付録 *A.* 中間結果および算術精度』の算術計算中間結果についての説明を参照してください。

ROUNDED 句

フォーマット 1 および 2 については、265 ページの『ROUNDED 句』を参照してください。

SIZE ERROR 句

フォーマット 1 および 2 については、266 ページの『SIZE ERROR 句』を参照してください。

END-MULTIPLY 句

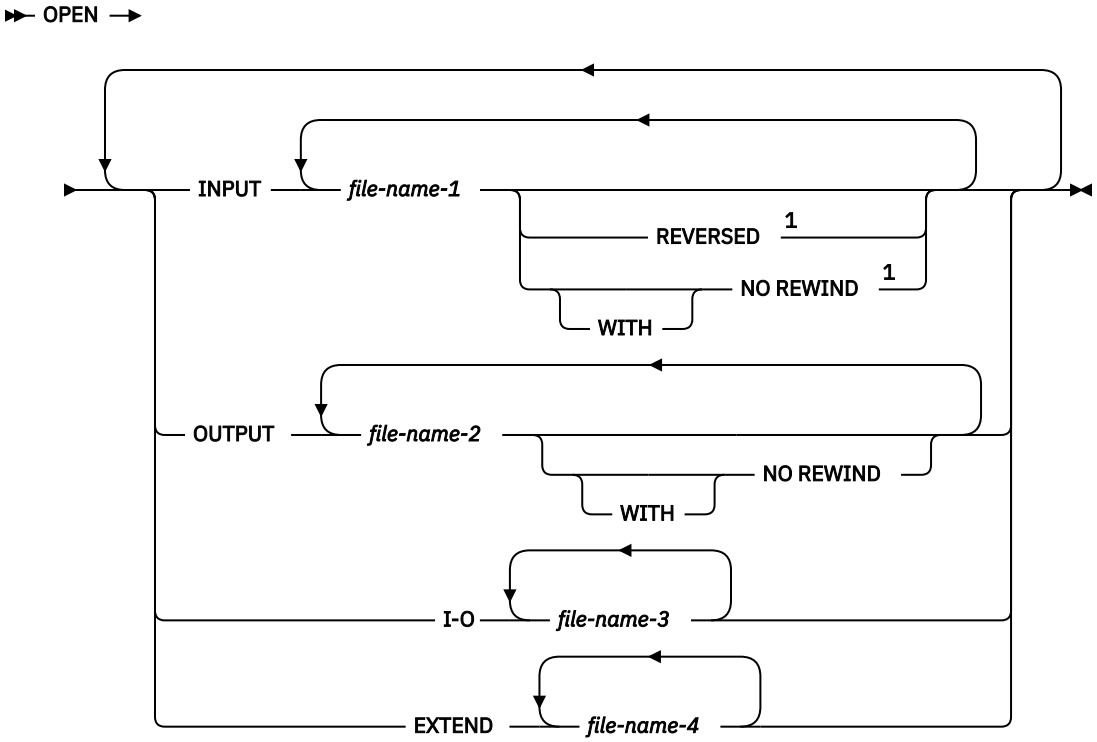
この明示的範囲終了符号は、MULTIPLY ステートメントの範囲を区切るために使用されます。 END-MULTIPLY 句を使用することによって、 条件的な MULTIPLY ステートメントを他の条件ステートメント内にネストすることができます。 END-MULTIPLY は、命令 MULTIPLY ステートメントと共に使用することもできます。

詳しくは、263 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ます。 ラベルのチェックまたは書き込み、あるいはその両方を行います。

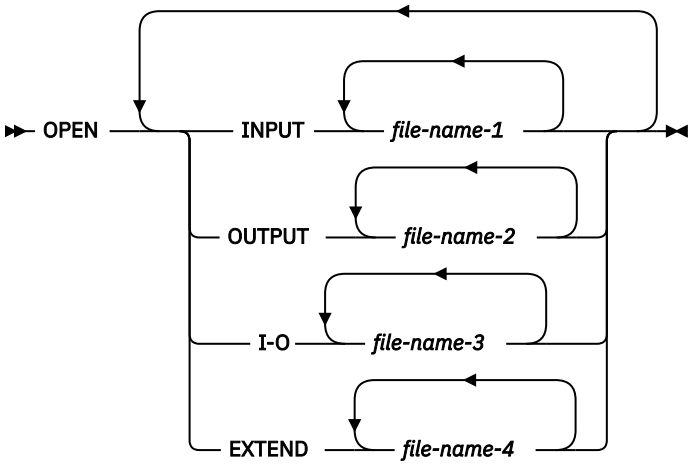
フォーマット 1: 順次ファイルの OPEN ステートメント



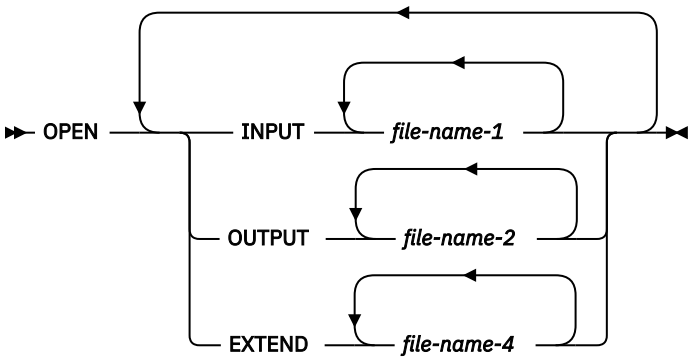
注:

[1] REVERSED および WITH NO REWIND 句は構文チェックされますが、プログラムの実行には何も影響しません。

フォーマット 2: 索引付きおよび相対ファイルの OPEN ステートメント



フォーマット 3: 行順次ファイルの OPEN ステートメント



INPUT、OUTPUT、I-O、および EXTEND 句は、ファイルのオープンに使用するモ ードを指定します。 キーワード OPEN と共に、INPUT、OUTPUT、I-O、または EXTEND の各句のうち少なくとも 1 つを指定しなければなりません。 INPUT、 OUTPUT、I-O、および EXTEND の句は、任意の順序で指定できます。

INPUT
入力操作を実行できます。

OUTPUT
出力操作を実行できます。 この句は、ファイルの作成時に指定することができます。

OUTPUT はレコードを含むファイルには指定しないでください。 このファイルは新規データに置き換えられます。

I-O
入力操作と出力操作の両方を実行できます。 I-O 句は、直接アクセス装置に割り当てられているファイルに対してのみ指定することができます。

I-O 句は行順次ファイルには無効です。

EXTEND
ファイルを追加またはファイルを作成する出力操作を実行できます。

EXTEND 句を順次アクセス・ファイルで使用できるのは、新規データが昇順で作成されている場合だけです。 EXTEND 句は、LINAGE 節が指定されたファイルで使用可能です。

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 ステートメントによる処理の対象となるファイルを指定します。 複数のファイルを指定する場合、それらのファイルは同一の編成やアクセス・モードを持つ必要はありません。 各ファイル名は、DATA DIVISION の FD 項目に定義されていなければならず、また、 ソート・ファイルやマージ・ファイルにすることはできません。FD 項目は、ファイルが定義されるときに提供された情報と同じでなければなりません。

REVERSED
REVERSED 句は、構文チェックされますが、プログラムの実行には何も影響しません。

NO REWIND
NO REWIND 句は 構文チェックされますが、プログラムの実行には何も影響しません。です。

ファイル・サイズの情報については、523 ページの『付録 B コンパイラー限界値』を参照してください。.

一般規則

このトピックでは、OPEN ステートメントの一般規則について説明します。

- INPUT 句を指定してオープンしたファイルがオプション・ファイルであり、それが使用可能でない場合、OPEN ステートメントは、 オプショナル入力ファイルが使用可能でないことを示すようにファイル位置標識を設定します。
- OPEN INPUT ステートメントや OPEN I-O ステートメントが実行されると、 ファイル位置標識は、次のように設定されます。

  - 索引ファイルについては、そのファイルに関連する照合シーケンスにおける、 最低の順序位置を持つ文字。
  - 順次ファイルおよび相対ファイルの場合は、1。
- EXTEND 句を指定する場合は、OPEN ステートメントは、ファイル位置標識をそのファイルに書き込まれた最後のレコードの直後に位置付けます (最高の第 1 レコード・キー値を持つ (索引ファイルの場合) か、または相対キー値を持つ (相対ファイルの場合) レコードは、最後のレコードとみなされます)。 それ以後に実行される WRITE は、 そのファイルが OUTPUT としてオープンされているかのように、レコードを追加します。 EXTEND 句を指定できるのは、ファイル作成時ですが、 その他にもレコードが入ったファイル、 または削除されたレコードが入っていたファイルに指定することもできます。 詳しくは、338 ページの『OPEN ステートメントに関する注意事項』 の注意事項 1 および 108 ページの『SELECT 節』 の SELECT OPTIONAL を参照してください。
- EXTEND 句が指定されなかった場合は、OPEN ステートメントはファイルをその先頭に位置付けます。

複数のファイル名が OPEN ステートメントに指定されている場合、その OPEN ステートメントの実行結果は、個別の OPEN ステートメントがファイル名ごとに、OPEN ステートメントに指定されている順序と同じ順序で書き込まれた場合と同じです。 これら個別の OPEN ステートメントはそれぞれ、OPEN ステートメントで指定されたとおりの同じオープン・モード指定、および REWIND 句を持ちます。OPEN ステートメントが暗黙指定された結果として宣言型プロシージャーが実行されて、NEXT STATEMENT 句が指定された RESUME ステートメントが実行された場合、次の暗黙 OPEN ステートメントがあれば、その OPEN ステートメントで処理が再開されます。

ラベル・レコード

ラベル処理はサポートされていません。

以下のいずれかの言語エレメントが起こった場合、警告メッセージが出ます。

- LABEL RECORDS IS データ名
- USE...AFTER...LABEL PROCEDURE

OPEN ステートメントに関する注意事項

このトピックには、OPEN ステートメントに関する注意事項があります。

注意事項は以下のとおりです。

1. OPEN ステートメントが正常に実行されると、そのファイルは使用可能であると判別され、 オープン・モードになります。 ファイルが使用できる のは、 そのファイルが物理的に存在し、入出力制御システムによって認識されるものである場合です。 下の表に、使用可能なファイルと使用可能でないファイルのオープン結果を示します。

表 *51.* **ファイルの使用可能性**

OPEN の形式	**ファイルが 使用可能**	**ファイルが使用可能でない**
INPUT	通常のオープン	オープンに失敗する (ファイル状況 35)
INPUT (オプション・ファイル)	通常のオープン	通常のオープン。最初の読み取りにより、終了条件または無効キー条件が生じる (ファイル状況 05)
I-O	通常のオープン	オープンに失敗する (ファイル状況 35)
I-O (オプション・ファイル)	通常のオープン	オープンするとファイルが作成される (ファイル状況 05)
OUTPUT	通常のオープン。ファイルにレコードが入っていない。	オープンするとファイルが作成される
EXTEND	通常のオープン	オープンに失敗する (ファイル状況 35)

表 *51.* **ファイルの使用可能性** *(続き)*		
OPEN の形式	**ファイルが 使用可能**	**ファイルが使用可能でない**
EXTEND (オプション・ファイル)	通常のオープン	オープンするとファイルが作成される (ファイル状況 05)

2. OPEN ステートメントが正常に実行されると、ファイルがオープン状態になり、関連するレコード域はプログラムに対して使用可能になります。
3. OPEN ステートメントは最初のデータ・レコードを取得または解放しません。
4. レコード域との間でデータをやり取りできるのは、ファイルがオープン状態になっている場合のみです。
5. 使用可能な任意の入出力ステートメントを使用する場合も、その実行前に OPEN ステートメントが正しく実行されていなければなりません (USING 句または GIVING 句付きの SORT や MERGE ステートメントを除く)。 以下の表では、'X' はステートメントが、最上段に示されているオープン・モードで 使用できることを意味しています。

表 *52.* **順次ファイルで使用可能なステートメント**				
ステートメント	**入力オープン・モード**	**出力オープン・モード**	**I-O オープン・モード**	**拡張オープン・モード**
READ	X		X	
WRITE		X		X
REWRITE			X	

以下の表では、'X' は、その行に示されているアクセス・モードで使用されると、 上段に示されたオープン・モードで使用できることを意味します。

表 *53.* **索引付きファイルと相対ファイルで使用可能なステートメント**					
ファイル・アクセス・モード	**ステートメント**	**入力オープン・モード**	**出力オープン・モード**	**I-O オープン・モード**	**拡張オープン・モード**
順次	READ	X		X	
	WRITE		X		X
	REWRITE			X	
	START	X		X	
	DELETE			X	
ランダム	READ	X		X	
	WRITE		X	X	
	REWRITE			X	
	START				
	DELETE			X	

表 53. 索引付きファイルと相対ファイルで使用可能なステートメント (続き)					
ファイル・アクセス・モード	**ステートメント**	**入力オープン・モード**	**出力オープン・モード**	**I-O オープン・モード**	**拡張オープン・モード**
動的	READ	X		X	
	WRITE		X	X	
	REWRITE			X	
	START	X		X	
	DELETE			X	

以下の表では、'X' はステートメントが、最上段に示されているオープン・モードで 使用できることを意味しています。

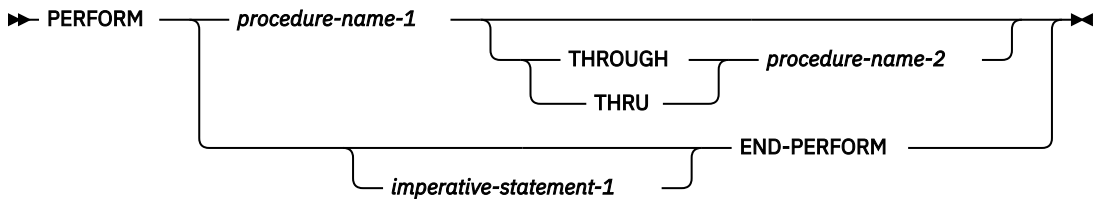
表 54. 行順次ファイルで使用可能なステートメント				
ステートメント	**入力オープン・モード**	**出力オープン・モード**	**I-O オープン・モード**	**拡張オープン・モード**
READ	X			
WRITE		X		X
REWRITE				

1. 同一プログラムの中で、1 つのファイルを INPUT、OUTPUT、I-O、または EXTEND (順次ファイルおよび行順次ファイルのみ) としてオープンすることができます。 あるファイルに対して最初の OPEN ステートメントを実行した後は、それ以降の OPEN ステートメントを実行する度に、その前に、LOCK 句の指定のない CLOSE ファイル・ステートメントをそのファイル に対して正しく実行しておく必要があります。
2. ファイル制御項目内に FILE STATUS 節が指定されている場合は、 関連するファイル状況キーが、OPEN ステートメントの実行時に更新されます。
3. すでにオープン状態になっているファイルに対して OPEN ステートメントを実行すると、そのファイルに対して EXCEPTION/ERROR プロシージャーが指定されていれば、それが実行されます。

PERFORM ステートメント

PERFORM ステートメントは、1 つまたは複数のプロシージャーに明示的に制御を 移し、指定されたプロシージャーの実行終了後に、PERFORM ステートメントの次 の実行可能ステートメントに暗黙的に制御を戻します。

フォーマット 1: 基本 PERFORM ステートメント



プロシージャー名-1 、プロシージャー名-2
手続き部の中のセクションまたは段落を指名し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 を両方指定する場合、 いずれか一方が宣言型プロシージャーの中のプロシージャー名であれば、 両方が、同じ宣言型プロシージャーの中のプロシージャー名でなければなりません。

プロシージャー名-*1* を指定した場合、命令ステートメント-*1* と END-PERFORM 句を指定することはできません。

プロシージャー名-*1* を省略した場合、命令ステートメント-*1* と END-PERFORM 句を指定する必要があります。

命令ステートメント-*1*
行内 PERFORM ステートメントで実行されるステートメント

行内および行外の PERFORM ステートメント

プロシージャー名-*1* が省略されている場合、PERFORM ステートメントは行内 PERFORM ステートメントです。

プロシージャー名-*1* が指定されている場合、PERFORM ステートメントは行外 PERFORM ステートメントです。

行内 PERFORM ステートメントは、END-PERFORM 句によって区切る必要があります。

行内フォーマットと行外フォーマットを合わせて使用することはできません。例えば、プロシージャー名-*1* を指定した場合、 命令ステートメントと END-PERFORM 句は指定できません。

EXIT PERFORM ステートメントを使用して、GO TO ステートメントまたは PERFORM ... THROUGH ステートメントを使用せずに行内 PERFORM ステートメントを終了することができます。 詳しくは、307 ページの『形式 5 (インライン実行)』を参照してください。

END-PERFORM

行内 PERFORM ステートメントの範囲を区切ります。 この範囲内の最後のステートメントが実行されると、行内 PERFORM の実行が完了します。

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 (単数または複数) が 1 回実行されると、 制御は PERFORM ステートメントの後続の次の実行可能ステートメントに渡されます。

注: PERFORM ステートメントではそれ自体を実行することはできません。 再帰的 PERFORM ステートメントでは、予測できない結果が生じる可能性があります。

行内 PERFORM ステートメントは、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が、行内 PERFORM ステートメントでは行内 PERFORM に 含まれるステートメントが、プロシージャー名-*1* (プロシージャー名-*2* が指定され ている場合は、プロシージャー名-1 からプロシージャー名-2) の範囲内にあるステ ートメントの代わりに実行されるところだけが異なります。行内 または行外 という語が特に明示されていない限り、 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 ステートメントにも適用されます。

行外 PERFORM ステートメントが実行される度に、プロシージャー名-*1* という名 前を持つプロシージャーの最初のステートメントに制御が移されます。 そして必ず PERFORM ステートメントの次にあるステートメントに制御は戻されます。 制御がどの地点で戻されるかは、次のようにして決定されます。

- 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、プロシージャー名-*1* の段落の最後のステートメントの実行後に制御の戻りが行われます。
- 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、プロシージャー名-*1* のセクションにある最後の段落の最後のステートメントの実行後に制御の戻りが行われます。
- プロシージャー名-*2* が指定されており、それが段落名である場合、プロシージャー名-*2* の段落の最後のステートメントの実行後に制御の戻りが行われます。
- プロシージャー名-*2* が指定されており、それがセクション名である場合、プロシージャー名-*2* のセクションにある最後の段落の最後のステートメントの実行後に制御の戻りが行われます。

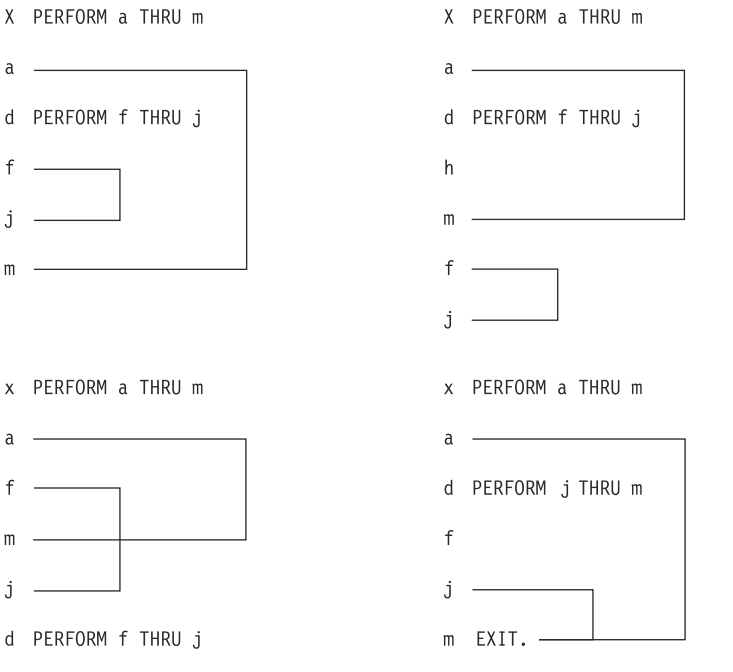
プロシージャー名-*1* とプロシージャー名-*2* との間で保持しなければならない唯一の関係は、連続する一連の処理が、プロシージャー名-*1* によって指名されるプロシージャーから始まり、プロシージャー名-*2* によって指名されるプロシージャーの実行によって終了するということだけです。

PERFORM ステートメントを、実行されるプロシージャーの中で指定することがで きます。戻る地点への論理パスが 2 つ以上ある場合、 EXIT ステートメントだけからなる段落の名前をプロシージャー名-*2* として指定することができます。 その場合、戻り点へのすべてのパスは、この段落に導かれます。

実行されるプロシージャー中に別の PERFORM ステートメントが含まれているとき、 組み込まれた PERFORM ステートメントに関連する一連のプロシージャーは、 最初の PERFORM ステートメントにより実行されるプロシージャーの中に完全に含まれているか、 または完全に外側になければなりません。 すなわち、実行開始点が、 別のアクティブな PERFORM ステートメントにより実行されるプロシージャーの範囲内にある PERFORM ステートメントは、 別のアクティブな PERFORM ステートメントの出口点を通りすぎて制御を渡すことはできません。 ただし、2 つ以上のアクティブな PERFORM ステートメントには、共通出口を持たせることができます。

PERFORM ステートメント以外の手段によって一連のプロシージャーに制御が渡さ れる場合、それらのプロシージャーを参照する PERFORM ステートメントが何もな いかのように、制御は出口点を通りすぎて次の実行可能ステートメントに渡されま す。

以下の図は、PERFORM ステートメントの有効な実行シーケンスを示したものです。



この図は、有効な PERFORM ステートメントの実行シーケンスを示したものです。 英字の文字は、プロシージャーを表すために使用されます。次の例が示されます。

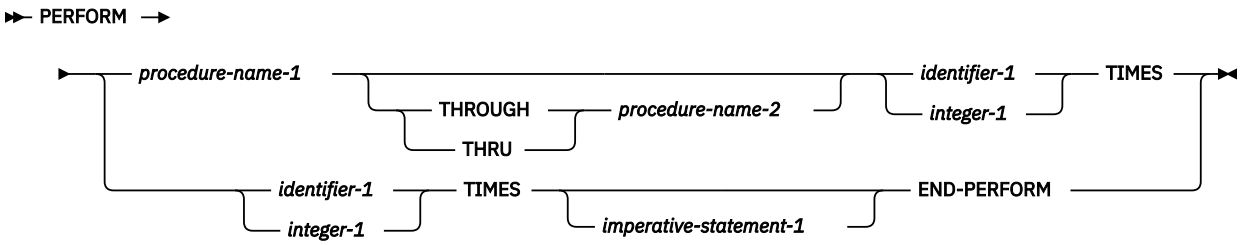
1. PERFORM a THRU m。プロシージャーのシーケンスは、a、d、f、j、および m です。プロシージャー d は、PERFORM f THRU j を含みます。このシーケンスでは、プロシージャー f THRU j は、プロシージャー a THRU m の範囲内でネストされます。
2. PERFORM a THRU m。プロシージャーのシーケンスは、a、d、h、m、f、および j です。プロシージャー d は、PERFORM f THRU j を含みます。このシーケンスでは、プロシージャー f THRU j は、プロシージャー a THRU m の範囲外です。
3. PERFORM a THRU m。プロシージャーのシーケンスは、a、f、m、j、および d です。プロシージャー d は、PERFORM f THRU j を含みます。このシーケンスでは、2 つの PERFORM ステートメントがオーバーラップする範囲を持ちます。f thru j が a thru m とオーバーラップします。

4. PERFORM a THRU m。プロシージャーのシーケンスは、a、d、f、j、および m です。プロシージャー m は、EXIT ステートメントで終了します。 プロシージャー d は、PERFORM d THRU m を含みます。このシーケンスでは、両方の PERFORM ステートメントが同じ出口点を共有します。

TIMES 句を指定した PERFORM

PERFORM ステートメントの TIMES 句の中で参照されたプロシージャーは、 最大 999,999,999 回まで、*ID-1* または整数-*1* の中の値によって指定した回数だけ実行されます。 ついで、PERFORM ステートメントの後にある次の実行可能ステートメントに渡されます。

フォーマット 2: TIMES 句を含む PERFORM ステートメント



プロシージャー名-*1* を指定した場合、命令ステートメント-*1* と END-PERFORM 句を指定することはできません。

ID-1

ここには整数項目を指定しなければなりません。 *ID-1* をウィンドウ表示日付フィールドにすることはできません。

PERFORM ステートメントが開始されたときに、*ID-1* が 0 または負数である場合、 制御は PERFORM ステートメントの次のステートメントに移されます。

PERFORM ステートメントが開始された後で *ID-1* が変更されても、 そのプロシージャーを実行する回数は変更されません。

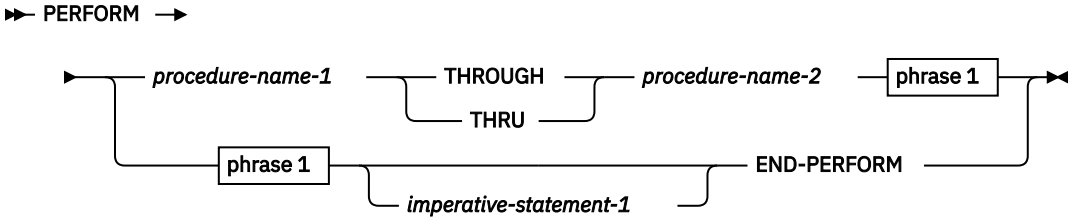
整数-*1*

正の符号付き整数にできます。

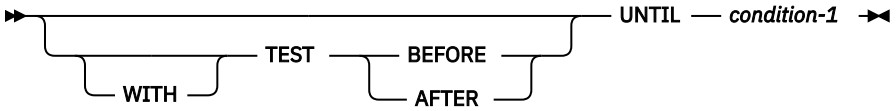
UNTIL 句を指定した PERFORM

UNTIL 句形式では、参照されるプロシージャーは、UNTIL 句によって指定された条件が真になるまで 実行されます。 条件が真になると、制御は、 PERFORM ステートメントの後にある次の実行可能ステートメントに渡されます。

フォーマット 3: UNTIL 句を含む PERFORM ステートメント



句 1



プロシージャー名-*1* を指定した場合、命令ステートメント-*1* と END-PERFORM 句を指定することはできません。

条件-*1*

238 ページの『条件式』に説明してある条件ならばどの条件でも使用できます。 PERFORM ステートメントが開始される時点で条件が真であれば、 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付けられた添え字がある場合には、 条件がテストされるたびに評価されます。

TEST BEFORE 句が指定されているかまたは想定されている場合、 条件がテストされない限り、 ステートメントは何も実行されません (DO WHILE に対応したもの)。

TEST AFTER 句が指定されている場合、実行されるステートメントは、 条件がテストされる前に少なくとも 1 回は実行されます (DO UNTIL に対応したもの)。

どちらの場合も、条件が真の場合は、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が移ります。 TEST BEFORE 句も TEST AFTER 句も指定されていない場合は、TEST BEFORE 句が想定されます。

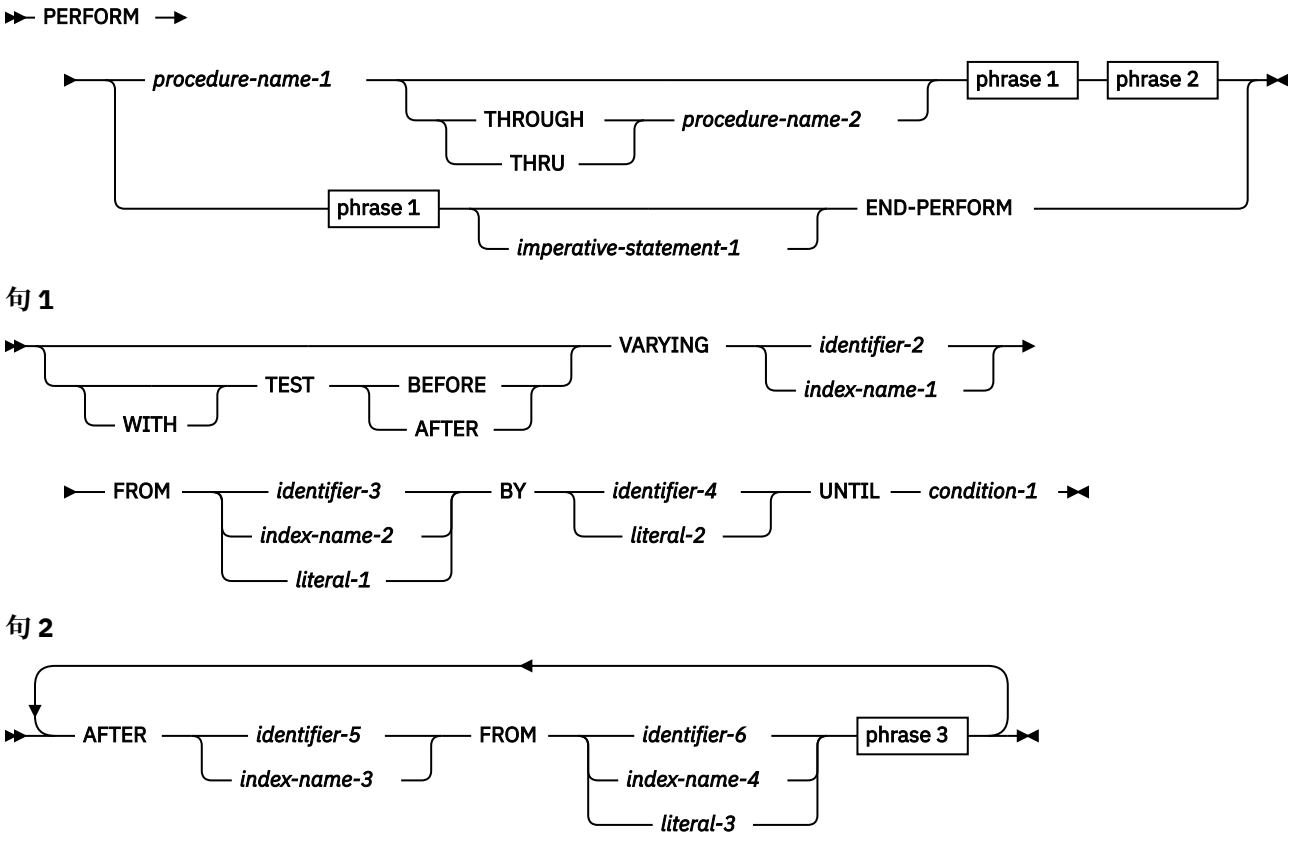
VARYING 句を指定した PERFORM

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。

詳しくは、348 ページの『VARYING 句の規則』を参照してください。

フォーマット 4 の VARYING 句指定の PERFORM ステートメントは、7 次元のテーブル全体をシリアル検索することができます。

フォーマット 4: VARYING 句を含む PERFORM ステートメント



プロシージャー名-*1* を指定した場合、命令ステートメント-*1* と END-PERFORM 句を指定することはできません。 プロシージャー名-*1* を省略した場合、AFTER 句を指定することはできません。

ID-2* から *ID-7
これらは数字基本項目を指名する必要があります。 これらの ID は、ウィンドウ表示日付フィールドにはできません。

リテラル-*1* からリテラル-*4*
これらは数字リテラルを表さなくてはなりません。

条件-*1*、条件-*2*
238 ページの『条件式』に説明してある条件ならばどの条件でも使用できます。 PERFORM ステートメントが開始される時点で条件が真であれば、 指定されたプロシージャーは実行されません。

UNTIL 句で指定した条件が満たされると、 制御は PERFORM ステートメントの後にある次の実行可能ステートメントに渡されます。

条件-*1* または条件-*2* に指定されたオペランドのいずれかが、添え字付き、参照変更、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。

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。

TEST BEFORE 句が指定されていると、指定されているすべての条件がテストされない限り、最初の実行は行われず、しかも指定した条件がすべて 満たされないときに限り、 実行されるステートメントが実行されます。 TEST AFTER 句が指定されていると、実行されるステートメントは、 条件がテストされる前に少なくとも 1 回は実行されます。

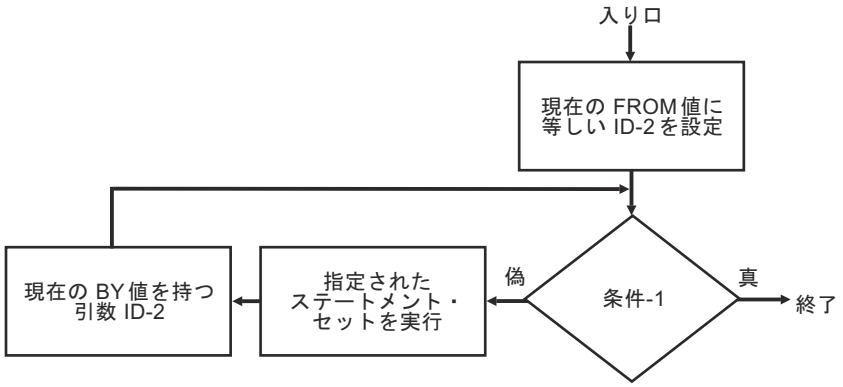
TEST BEFORE 句も TEST AFTER 句も指定されていない場合は、TEST BEFORE 句が想定されます。

ID の変更

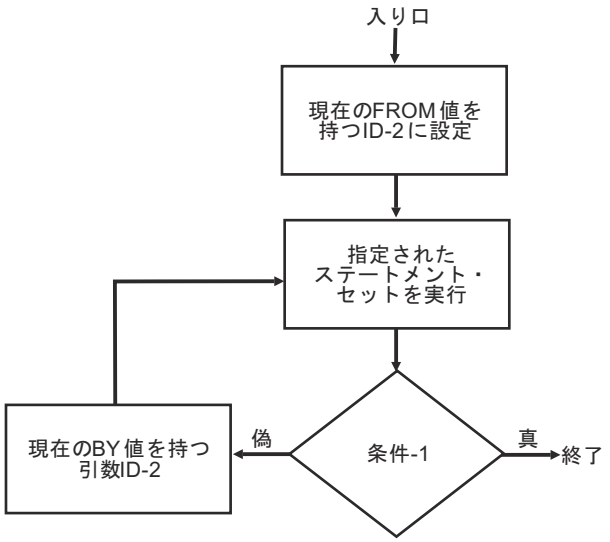
オペランドがどのようにして増加したり減少したりするかは、 指定する変数の個数によって異なります。 説明では、*ID-n* についての説明はすべて指標名-*n* にも当てはまります (ただし、*ID-n* が BY 句のオブジェクトであるときは別です)。

ID-2 または *ID-5* に添え字が付いている場合、その添え字は ID によって参照されたデータ項目の内容が設定されるか、増えていくたびに評価されます。 *ID-3*、*ID-4*、*ID-6*、または *ID-7* に添え字が付いている場合、その添え字は ID によって参照されたデータ項目の内容が設定操作または増加操作で使用されるたびに評価されます。

以下の図に、ID を TEST BEFORE 句を使用して変更するときの PERFORM ステートメントの論理を示します。



以下の図に、ID を TEST AFTER 句を使用して変更するときの PERFORM ステートメントの論理を示します。



2 つの ID の変更

このトピックでは、2 つの ID を変更するステップについて説明します。

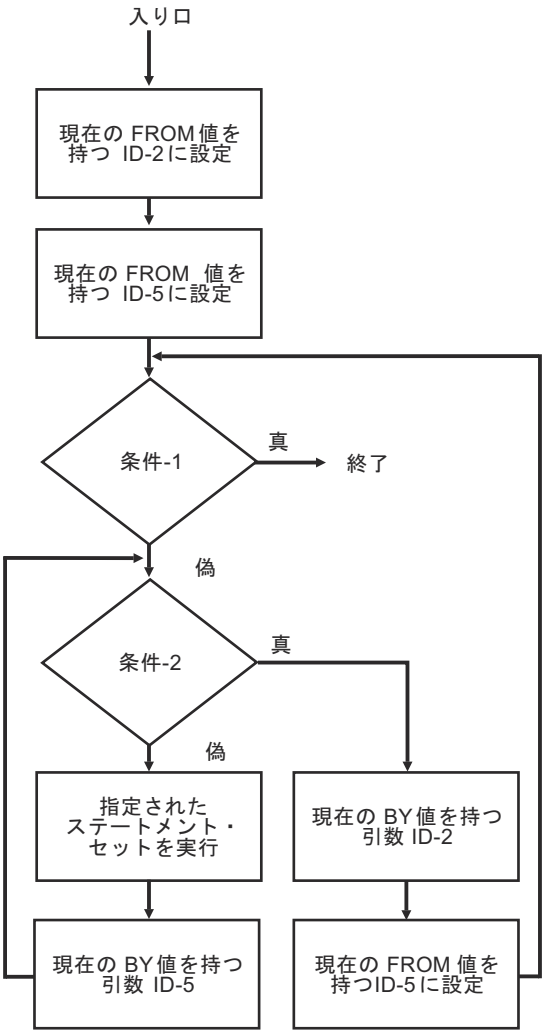
```
PERFORM PROCEDURE-NAME-1 THROUGH PROCEDURE-NAME-2
      VARYING ID-2 FROM ID-3
            BY ID-4 UNTIL CONDITION-1
         AFTER ID-5 FROM ID-6
            BY ID-7 UNTIL CONDITION-2
```

1. *ID-2* および *ID-5* が、その初期値である *ID-3* および *ID-6* にそれぞれ設定されます。
2. 条件*-1* が、次のようにして評価されます。
   a. 偽であれば、ステップ 3 から 7 が実行されます。
   b. 真であれば、制御は、直接 PERFORM ステートメントの後にあるステートメントに渡されます。
3. 条件*-2* が、次のようにして評価されます。
   a. 偽であれば、ステップ 4 から 6 が実行されます。
   b. 真であれば、*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* およびプロシージャー名*-2* が 指定されていれば、1 回だけ実行されます。
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、ステップの 3 から 5 を繰り返します。
7. 条件*-1* が真になるまで、ステップの 2 から 6 を繰り返します。

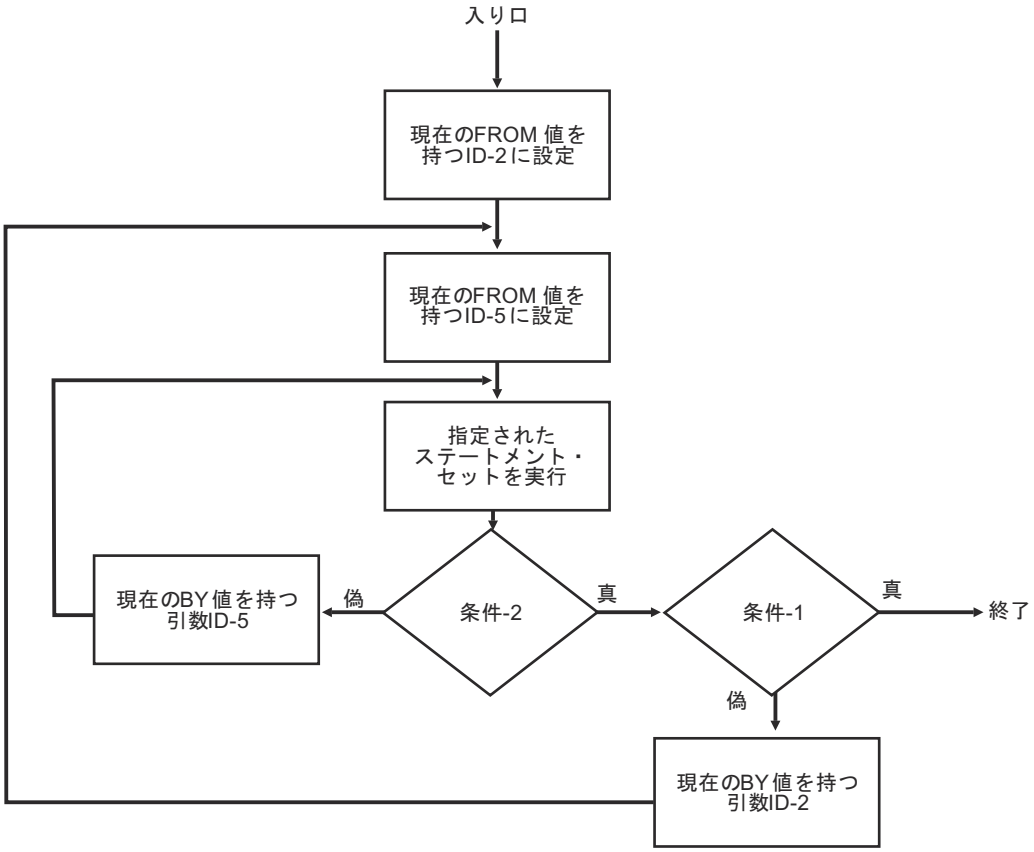
PERFORM ステートメントの実行終了時には、次のようになっています。

- *ID-5* には、*ID-6* の現行値が入っています。
- *ID-2* には、最後に使用された設定値を、増分値だけ超えた値または減分値だけ減らした値が入っています (PERFORM ステートメントの実行開始時に条件*-1* が真である場合以外。 条件*-1* が真である場合、*ID-2* には *ID-3* の現行値が入っています)。

以下の図に、2 つの ID を TEST BEFORE 句を使用して変更するときの PERFORM ステートメントの論理を示します。



以下の図に、2 つの ID を TEST AFTER 句を使用して変更するときの PERFORM ステートメントの論理を示します。



3 つの *ID* の変更

このトピックでは、3 つの ID を変更するステップについて説明します。

```
PERFORM PROCEDURE-NAME-1 THROUGH PROCEDURE-NAME-2
      VARYING ID-2 FROM ID-3
            BY ID-4 UNTIL CONDITION-1
          AFTER ID-5 FROM ID-6
            BY ID-7 UNTIL CONDITION-2
          AFTER ID-8 FROM ID-9
            BY ID-10 UNTIL CONDITION-3
```

この場合の動作も、次の点を除けば 2 つの ID の場合と同じです。すなわち、*ID-8* は、*ID-5* が *ID-7* だけ増やされるたびに完全なサイクルで処理され、一方、*ID-2* が変更されるたびに完全なサイクルで処理されます。

PERFORM ステートメントの実行終了時には、次のようになっています。

- *ID-5* および *ID-8* には、それぞれ *ID-6* および *ID-9* の現行値が入っています。
- *ID-2* には、最後に使用された設定値を、1 つの増分値だけ超えた値または減分値だけ減らした値が入っています (PERFORM ステートメントの実行開始時に条件-*1* が真でない場合。 条件-*1* が真である場合、*ID-2* には *ID-3* の現行値が入ります)。

4 つ以上の *ID* の変更

最大 4 つの AFTER 句を追加して、前述の例に似た PERFORM ステートメントによる処理を行うことができます。

VARYING 句の規則

この句には、指定された変数の数とは無関係に、一定の規則が適用されます。

規則は以下のとおりです。

- VARYING 句または AFTER 句の中に指標名が指定されている場合

- 指標名は、217 ページの『INDEX 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。 (366 ページの『SET ステートメント』も参照してください。)
- 関連する FROM 句では、ID を整数として記述し、 正の値を持つ必要があります。 リテラルは正の整数でなければなりません。
- 関連する BY 句内では、ID は整数として記述する必要があります。 リテラルはゼロ以外の整数でなければなりません。

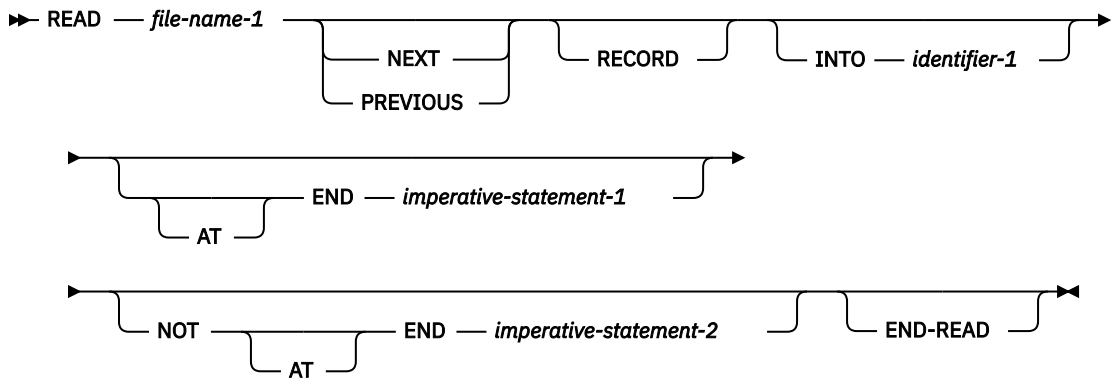
- FROM 句の中に指標名が指定されている場合
  - 関連する VARYING 句または AFTER 句の中では、ID は整数として記述する必要があります。 これは、SET ステートメントの中で記述されたように初期設定されます。
  - 関連する BY 句は、 ID を整数として記述し、正の値を持たせなければなりません。 リテラルはゼロ以外の整数でなければなりません。
- BY 句の中で、ID とリテラルはゼロ以外の値を持たねばなりません。
- VARYING、FROM、および BY 句の中で ID または指標名の値を変更することは、 プロシージャーの実行回数を変更することになります。

READ ステートメント

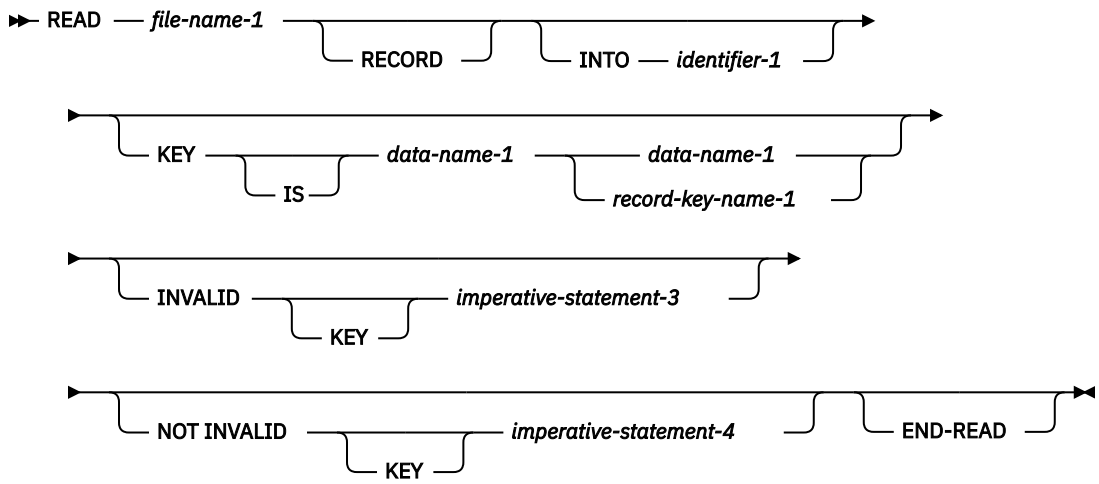
順次アクセスの場合、READ ステートメントはファイル内の次の論理レコードをオ ブジェクト・プログラムが使用できるようにします。 ランダム・アクセスの場合には、READ ステートメントは、オブジェクト・プログラムが直接アクセス・ファイル内の指定したレコードを使用可能にします。

READ ステートメントを実行するときには、関連するファイルを INPUT モードま たは I-O モードでオープンしておく必要があります。

フォーマット 1: 順次検索の READ ステートメント



フォーマット 2: ランダム検索の READ ステートメント



制約事項 : レコード・キー名 は、STL ファイル・システムでのみサポートされます。

ファイル名-1
DATA DIVISION の FD 項目に定義されている必要があります。

NEXT RECORD
レコードの論理的なシーケンスの中で次の位置にあるレコードを読み取ります。 NEXT は、アクセス・モードが順次である場合のオプショナ ルです。READ ステートメントの実行には影響がありません。

動的アクセス・モードのファイルから、レコードを順次検索するためには、NEXT 句または PREVIOUS 句のいずれかを指定する必要があります。

PREVIOUS RECORD
レコードの論理的なシーケンスの中で前の位置にあるレコードを読み取ります。 PREVIOUS は DYNAMIC アクセス・モードの索引付きファイルおよび相対ファイルに適用します。

レコードを順次検索するためには、動的アクセス・モードのファイルに、NEXT 句または PREVIOUS 句のいずれかを指定する必要があります。

READ...PREVIOUS 指定して前の論理レコードが存在しなかった場合は、AT END 条件となり、READ ステートメントは失敗します。

READ...PREVIOUS を指定すると、以下の規則に従って、どのレコードを使用可能にするか判別するために、ファイル位置標識の設定が使用されます。

- ファイル位置標識が有効な前のレコードが設定されていないことを示している場合は、READ は失敗します。
- ファイル位置標識が OPEN ステートメントの実行によって位置付けられている場合は、AT END 条件が起こります。
- ファイル位置標識が前の START ステートメントによって設定されている場合は、相対レコード番号 (相対ファイルの場合) またはキー値 (索引付きファイルの場合) がファイル位置標識より小さいか等しいファイルの最初に存在するレコードが選択されます。
- ファイル位置標識が前の READ ステートメントによって設定されている場合は、相対レコード番号 (相対ファイルの場合) またはキー値 (索引付きファイルの場合) がファイル位置標識より小さいファイルの最初に存在するレコードが選択されます。

INTO *ID-1*
ID-1 は受け取りフィールドです。

ID-1 ID-1 は、選択された送り出しレコード記述項目に対して、MOVE ステートメントの規則に従う有効な受け取りフィールドでなければなりません。

ファイル名-*1* に関連付けられたレコード域と *ID-1* は、 同じストレージ域を占めることはできません。

ファイル名-*1* に関連付けられたレコード記述が 1 つだけしかない場合、または *ID-1* によって 参照されるすべてのレコードとデータ項目に基本英数字項目または英数字グループ項目が記述されている場合、INTO 句を指定した READ ステートメントの実行結果は、指定された順序で以下の規則を適用するのと同じことになります。

- INTO 句を指定しないだけであとは同じ READ ステートメントを実行します。
- 現行レコードを、CORRESPONDING 句のない MOVE ステートメントの規則に従って、そのレコード域から *ID-1* で指定された領域へ移動します。 現在のレコードのサイズは、RECORD 節で指定された規則によって決定されます。 ファイル記述項目が RECORD IS VARYING 節を含む場合には、暗黙の移動はグループ移動になります。 READ ステートメントの実行が正しく行われなかった場合には、暗黙の MOVE ステートメントの実行は行われません。 *ID-1* に関連する添え字付けまたは参照変更があれば、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、それは評価されます。レコードは、そのレコード域と *ID-1* によって参照されるデータ項目の両方で使用可能です。

  ID-1 が日付フィールドである場合は、332 ページの『日付フィールドが関係する移動』で説明された動作に従って、 暗黙の MOVE ステートメントが実行されます。

ファイル名-*1* に関連付けられたレコード記述が複数あり、それらのすべてに英数字グループ項目または基本英数字項目が記述されていない場合は、以下の規則が適用されます。

1. ファイル名-*1* で参照したファイルが、可変長レコードを含んでいるとして記述され ている場合、グループ移動が行われます。
2. ファイル名-*1* で参照したファイルが、固定長レコードを含んでいるとして記述されている場合は、最大数の文字位置を指定しているレコードを送り出しフィールド記述として使用して、MOVE ステートメントの規則に従って移動が行われます。 そのようなレコードが複数存在する場合には、選択される送り出しフィールド・レコードは、該当するレコードのうちファイル名-*1* の記述のもとで最初に現れるレコードとなります。

KEY IS 句

KEY IS 句は、索引ファイルに対してのみ指定できます。

データ名-*1* または レコード・キー名-*1* は、ファイル名-*1* に関連付けられた RECORD KEY 節または ALTERNATE RECORD KEY 節に指定する必要があります。データ名-*1* または レコード・キー名-*1* は修飾することができます。

AT END 句

順次アクセスの場合、AT END 句および該当する EXCEPTION/ERROR プロシージャーは、両方とも省略することができます。

AT END 条件の処理については、『AT END 条件』を参照してください。

INVALID KEY 句

INVALID KEY 句および該当する EXCEPTION/ERROR プロシージャーは、両方とも省略することができます。

INVALID KEY 句の処理に関する情報は、274 ページの『無効キー条件』を参照してください。

END-READ 句

この明示的範囲終了符号は、READ ステートメントの範囲を区切るために使用されます。 END-READ 句を使用することによって、 条件的な READ ステートメントを他の条件ステートメント内にネストすることができます。 END-READ 句は、READ 命令ステートメントと共に使用することもできます。 詳しくは、263 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。.

複数レコードの処理

複数のレコード記述項目が ファイル名-*1* に関連付けられている場合、それらのレコードは自動的に同じストレージ域を共有します。 つまり、それらは暗黙的に再定義されます。 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にあるデータ項目のみが置換されます。 この範囲を超えて格納されているデータ項目は定義されません。 以下の例では、この概念を説明しています。

現行レコードの範囲がファイル名-*1* のレコード記述項目を超えている場合、そのレコードは最大サイズまで右側が切り捨てられます。

上記のどちらの場合も、READ ステートメントは正常に実行され、入出力状況が 04 に設定されてレコード長に矛盾があったことを示します。

次の例では、FD 内の違うサイズの 2 つのレコード域を示しています。 短い方のレコードが読み取られると、残りのレコード域の内容は未定義となります。

```
FD INPUT-FILE LABEL RECORD OMITTED.
01    RECORD-1 PICTURE X(30).
01    RECORD-2 PICTURE X(20).
```

READ ステートメントが実行された際の入力域の内容

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ1234
```

(RECORD-2) で読み取られるレコードの内容

```
01234567890123456789
```

READ ステートメント実行後の入力域の内容

```
0123456789012345678?????????
```

"?" 文字は、入力域では未定義の文字です。

順次アクセス・モード

順次アクセス・モードの全ファイルについてフォーマット 1 を使用しなければなり ません。

フォーマット-1 READ ステートメントを実行すると、ファイルから次の論理レコードが取り出されます。 アクセスされる次のレコードは、ファイル編成によって決定されます。

順次ファイル

NEXT RECORD とは、レコードの論理的なシーケンスの中で次の位置にあるレコードのことです。 NEXT 句を指定する必要はありません。 READ ステートメントの実行には影響がありません。

このファイルのファイル制御項目の中に SELECT OPTIONAL が指定してあり、 オブジェクト・プログラムのこの実行中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で AT END 条件が生じます。ただし、 ファイルが使用可能でないので、システム定義のファイル終了処理は実行されません。

AT END 条件

ファイル位置標識が、次の論理レコードが存在しないということ、またはオプションの入力ファイルが使用可能でないということを示している場合は、AT END 条件の処理が特定の順序で行われます。

順序は以下のとおりです。

1. ファイル位置標識の設定値から得られた値が、ファイル名-*1* 関連する入出力状況の中に入れられ、AT END 条件が生じたことを示します。

2. AT END 条件を起こすステートメントの中に AT END 句が指定されている場合、 制御はその AT END 句の中にある命令ステートメント-*1* に移ります。 ファイル名-*1* に関連して USE AFTER STANDARD EXCEPTION プロシージャーが指定されていても、それは実行されません。
3. AT END 句が指定されておらず、利用可能な USE AFTER STANDARD EXCEPTION プロシージャーが存在する場合は、 そのプロシージャーが実行されます。 そのプロシージャーからの戻りは、 READ ステートメントの終わりの後にある次の実行可能ステートメントになります。

   AT END 句および利用可能な EXCEPTION/ERROR プロシージャーは、両方とも省略できます。

   AT END 条件が起こると、READ ステートメントの実行は正しく行われません。 関連付けられたレコード域の内容は未定義のままであり、 ファイル位置標識は有効な次のレコードが設定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中に AT END 条件が起こらなければ、 AT END 句は指定されていても無視され、以下の処置が行われます。

1. ファイル位置標識が設定され、ファイル名-*1* に関連付けられた入出力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、ファイル名-*1* に対して適用可能な USE AFTER STANDARD EXCEPTION プロシージャーの実行後、READ ステートメントの終わりに制御が移されます。

   USE AFTER STANDARD EXCEPTION プロシージャーが指定されていなければ、制御は READ ステートメントの終わりか、または命令ステートメント-*2* が指定されていればそのステートメントに移されます。
3. 例外条件が起こらなければ、レコード域にあるレコードが使用可能になり、INTO 句の存在による暗黙の移動が実行されます。 制御は、READ ステートメントの終わりか、または命令ステートメント-*2* が指定していればそのステートメントに移されます。 後者の場合には、命令ステートメント-*2* の中に指定してある各ステートメントの規則に従って、実行は継続されます。 プロシージャー・ブランチまたは明示的な制御の移動を引き起こす条件ステートメントが実行される場合、 制御は、それを起こすステートメントの規則に従って移されます。 条件ステートメントが実行されない場合、命令ステートメント-*2* の実行が完了するとすぐに、 制御が READ ステートメントの終了へと移されます。

READ ステートメントの実行が失敗した後、関連するレコード域の内容は未定義であり、ファイル位置標識は、有効な次のレコードが設定されていないことを示すように設定されます。 読み取りの失敗の後、データにアクセスしたり、デー タをレコード域へ移動しようとすると、セグメンテーション違反という結果になる可能性があります。

索引付きファイルまたは相対ファイル

NEXT RECORD とは、キー・シーケンスで次に続く論理レコードです。

PREVIOUS RECORD とは、キー・シーケンスで先行する論理レコードです。

索引付きファイルでは、キー・シーケンスは、現行参照キーの昇順となる値のシーケンスです。 相対ファイルでは、キー・シーケンスは、 ファイル内に存在するレコードが持つ相対レコード番号の昇順となる値のシーケンスです。

READ ステートメントを実行する場合は、OPEN、START、または READ ステート メントを正常に実行して、ファイル位置標識を事前に設定しておく必要がありま す。 READ ステートメントが実行されると、ファイル位置標識によって示されるレコードがそのファイル位置標識によって示されるパスを通してアクセス可能であれば、そのレコードは使用可能になります。

レコードがすでにアクセス可能でなくなっている場合(例えば削除されてしまった ために)、ファイル位置標識はファイル内の次の (または前の) 既存レコードを指し示すように更新 され、そのレコードが使用可能にされます。

順次アクセス・モードのファイルの場合、NEXT 句を指定する必要はありません。

動的アクセス・モードのファイルの場合、レコードを順次検索するためには、NEXT 句 (または PREVIOUS 句) を指定しなければなりません。

AT END 条件

この条件が存在するのは、ファイル位置標識が、次の論理レコードが存在しないと いうこと (または前のレコードが存在しないということ)、またはオプショナル入力ファイルが使用可能でないということを示している 場合です。 上述の PREVIOUS RECORD の説明を参照してください。

READ ステートメントの実行中に、AT END 条件も無効キー条件も発生しなければ、AT END 句または INVALID KEY 句は指定されていても無視されます。 順次ファイルで AT END 条件が起こらなかった場合と同じアクションが実行されます (『AT END 条件』を参照)。

順次にアクセスされる索引付きファイル

DUPLICATES の指定のある ALTERNATE RECORD KEY が参照キーである場合には、重複するキー値を持つファイル・レコードは、 それらがファイルに入れられた際の順序で使用可能にされます。

順次にアクセスされる相対ファイル

ファイルに対して RELATIVE KEY 節が指定されている場合は、 READ ステートメントが実行されると、使用可能なレコードの相対レコード番号を示すために、 RELATIVE KEY データ項目が更新されます。

ランダム・アクセス・モード

ランダム・アクセス・モードの索引付きファイルおよび相対ファイルに対しては、 フォーマット 2 を指定する必要があります。また、レコードの取り出しがランダム であるときの動的アクセス・モードのファイルの場合にも、フォーマット 2 を指定 する必要があります。

READ ステートメントの実行は、以下のセクションで説明されているように、ファイル編成によって異なります。

索引付きファイル

フォーマット 2 の READ ステートメントが実行されると、参照キーの値が、 ファイル・レコードの中にある対応するキー・データ項目の値と比較されます。 この比較は、一致した値を持つ最初のレコードが見つかるまで行われます。 見つかったレコードはファイル位置標識が位置付けられ、 そしてそのレコードが使用可能になります。 値の一致するレコードが見つからない場合には、INVALID KEY 条件が起こり、 READ ステートメントの実行は失敗に終わります。 (無効キー条件の詳細については、274 ページの『無効キー条件』を参照してください。)

KEY 句が指定されていなければ、基本 RECORD KEY が、 この要求のために使用され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、基本 RECORD KEY は、 別の参照キーが設定されるまで、 後続の順次 READ ステートメントの実行のための参照キーとしても使用されます。

KEY 句が指定されている場合には、データ名-*1* がこの要求のために使用される参照キーになります。 動的アクセスが指定されている場合、別の参照キーが設定されるまで、 この参照キーが後続の順次 READ ステートメントの実行のために使用されます。

特定のファイル結合子を介してアクセスされる索引付きファイルでは、KEY 句が指定されている場合は、データ名-*1* またはレコード・キー名-*1* がこの検索のための参照キーになります。動的アクセスが指定された場合、この参照キーは、ファイル結合子を介した、ファイルに対する後続の順次 READ ステートメントの実行にも使用されます。 これは、別の参照キーがそのファイル結合子を介してファイルに設定されるまで行われます。

相対ファイル

フォーマット-2 の READ ステートメントを実行すると、RELATIVE KEY データ項目に含まれている相対レコード番号を持つレコードを指すようにファイル位置標識ポインターが設定され、そのレコードが使用可能になります。

ファイルに該当するレコードが含まれていなければ、INVALID KEY 条件が起こり、 READ ステートメントの実行は失敗に終わります。 (無効キー条件の詳細については、274 ページの『無効キー条件』を参照してください。)

KEY 句を相対ファイルに対して指定することはできません。

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は、ファイル制御項目内で動的アクセス・モードを指定できます。 動的アクセス・モードでは、使用するフォーマットに応じて、順次またはランダムのどちらかのレコード検索を使用できます。

順次レコード検索のときには、NEXT 句を指定したフォーマット 1 を使用する必要があります。 順次アクセスに関するその他すべての規則がここでも適用されます。

READ ステートメントに関する注意事項

このトピックには、READ ステートメントに関する注意事項が記載されています。

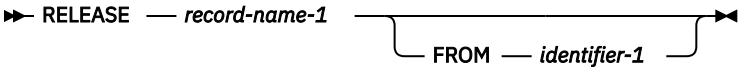
- ファイル制御項目に FILE-STATUS 節の指定がある場合は、関連するファイル状況キーが READ ステートメントの実行で更新されます。
- READ ステートメントの実行が失敗した後では、関連するレコード域の内容とファイル位置標識の値は未定義です。 読み取りの失敗の後、データにアクセスしたり、データをレコード域へ移動しようとすると、セグメンテーション違反という結果になる可能性があります。

RELEASE ステートメント

RELEASE ステートメントは、レコードを入出力域からソート処理の初期フェーズ へ渡します。

RELEASE ステートメントが使用できるのは、SORT ステートメントに関連する INPUT PROCEDURE 句の範囲内のみです。

フォーマット: RELEASE



INPUT PROCEDURE 句の中には、少なくとも 1 つの RELEASE ステートメントを指定する必要があります。

RELEASE ステートメントを実行すると、レコード名-*1* の現在の内容は、 ソート・ファイルに配置されます。 これによって、ソート操作の初期フェーズでレコードが使用可能になります。

レコード名-*1*
ソート・マージ・ファイル記述項目 (SD) にある論理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* は修飾することができます。

FROM 句
FROM *ID-1* 句を指定した RELEASE ステートメントの実行結果は、次のステートメントを指定した順序で実行した場合と同じになります。

```
MOVE ID-1 to record-name-1.
RELEASE record-name-1.
```

MOVE は、CORRESPONDING 句を指定しない MOVE ステートメントの規則に従って行われます。

ID-1
identifier-1 は、以下の項目のいずれかを参照する必要があります。

- WORKING-STORAGE SECTION、LOCAL-STORAGE SECTION、または LINKAGE SECTION 内の項目
- すでにオープンされた別のファイルのレコード記述
- 英数字関数、または国別関数

ID-1 は、受け取り項目としてレコード名-*1* が指定された、 MOVE ステートメントの規則に従う有効な送り出し項目でなければなりません。

ID-1 およびレコード名-*1* は、同じストレージ域を参照することはできません。

RELEASE ステートメントの実行後も、*ID-1* の中の情報は使用可能です (『共通の処理機能』にある 274 ページの『INTO 句および FROM 句』を参照してください)。

ファイル名-*1* に対する SD 項目を SAME RECORD AREA 節で指定せずに RELEASE ステートメントを実行した場合、レコード名-*1* の中に入っている情報は、使用できません。

SD 項目を SAME RECORD AREA 節の中で指定した場合は、レコード名-*1* は、その節で指定された他のファイルのレコードとして、依然として使用可能です。

FROM *ID-1* が指定されていると、*ID-1* の情報は依然として使用可能です。

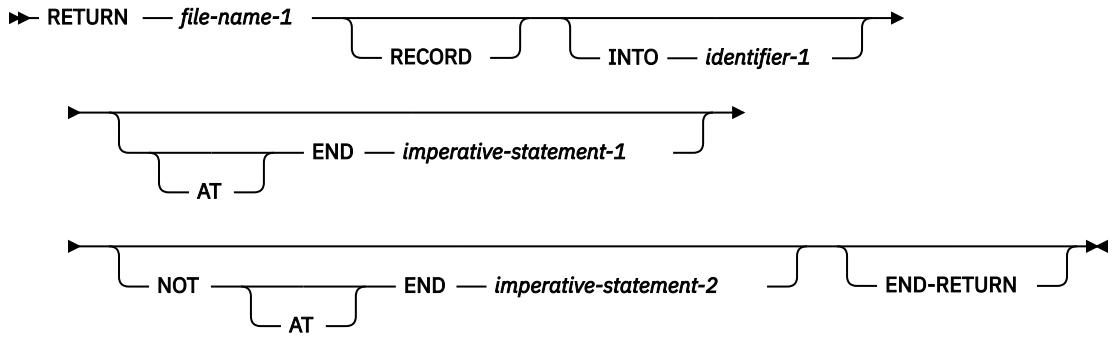
INPUT PROCEDURE から制御を渡されるとき、ソート・ファイルは、 RELEASE ステートメントの実行によりその中に入れられたすべてのレコードから構成されています。

RETURN ステートメント

RETURN ステートメントは、ソート処理またはマージ処理の最終フェーズから OUTPUT PROCEDURE へレコードを渡します。

RETURN ステートメントは、SORT ステートメントまたは MERGE ステートメン トと関連付けられた OUTPUT PROCEDURE 句の範囲内でのみ使用することができ ます。

フォーマット: RETURN ステートメント



OUTPUT PROCEDURE 句の中では、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* の次の位置にあるレコードが OUTPUT PROCEDURE 句による処理のため使用可能になります。

ファイル名-*1*
DATA DIVISION の SD 項目に記述されていなければなりません。

ファイル名-*1* に関連付けられた複数のレコード記述がある場合、それらのレコードは自動的に同じストレージを共用します。つまり、 そのストレージは暗黙に再定義されます。 RETURN ステートメントを実行した後は、現行レコードの内容のみが使用可能です。 現行レコードの長さを超え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 句
ファイル名-*1* に関連付けられたレコード記述が 1 つだけしかない場合、または *ID-1* によって 参照されるすべてのレコードとデータ項目に基本英数字項目または英数字グループ項目が記述されている場合、INTO 句を指定した RETURN ステートメントの実行結果は、指定された順序で以下の規則を適用するのと同じことになります。

- INTO 句を指定しないだけであり、同じ RETURN ステートメントを実行します。
- 現行のレコードを CORRESPONDING 句を伴わない MOVE ステートメントの規則に従って、そのレコード域から *ID-1* によって指定された領域へ移動します。 現在のレコードのサイズは、RECORD 節で指定された規則によって決定されます。 ファイル記述項目が RECORD IS VARYING 節を含む場合に

は、暗黙の移動はグループ移動になります。 RETURN ステートメントの実行が正しく行われなかった場合には、 暗黙の MOVE ステートメントの実行は行われません。 *ID-1* に関連する添え字付けまたは参照変更があれば、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、それは評価されます。 レコードは、そのレコード域と *ID-1* によって参照されるデータ項目の両方で使用可能です。

ファイル名-*1* に関連付けられたレコード記述が複数あり、それらのすべてに英数字グループ項目または基本英数字項目が記述されていない場合は、以下の規則が適用されます。

1. ファイル名-*1* によって参照されるファイルに可変長レコードが含まれている場合 は、グループ移動が行われます。
2. ファイル名-*1* で参照したファイルが固定長レコードを含んでいる場合は、最大数の文字位置を指定しているレコードを送り出しフィールド記述として使用して、MOVE ステートメントの規則に従って移動が行われます。 そのようなレコードが複数存在する場合には、選択される送り出しフィールド・レコードは、該当するレコードのうちファイル名-*1* の記述のもとで最初に現れるレコードとなります。

ID-1 ID-1 は、選択された送り出しレコード記述項目に対して、MOVE ステートメントの規則に従う有効な受け取りフィールドでなければなりません。

ファイル名-*1* に関連付けられたレコード域と *ID-1* は、 同じストレージ域を占めることはできません。

AT END 句

AT END 句で指定された命令ステートメントは、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。 これが実行されると、それ以上の RETURN ステートメントを現在の出力プロシージャーとして実行することはできません。

RETURN ステートメントの実行中に AT END 条件が発生しなかった場合は、レコ ードが使用可能にされた後、および INTO 句を指定したことで生じた暗黙の MOVE の実行後に、NOT AT END 句で指定された命令ステートメントに制御が移 されます。 AT END 条件が発生した場合は、制御は RETURN ステートメントの終わりに移されます。

END-RETURN 句

この明示的範囲終了符号は、RETURN ステートメントの範囲を区切るために使用されます。 END-RETURN 句を使用することによって、条件的な RETURN ステートメントを他の条件ステートメントの中にネストすることができます。 END-RETURN 句は、命令の RETURN ステートメントと共に使用することもできます。

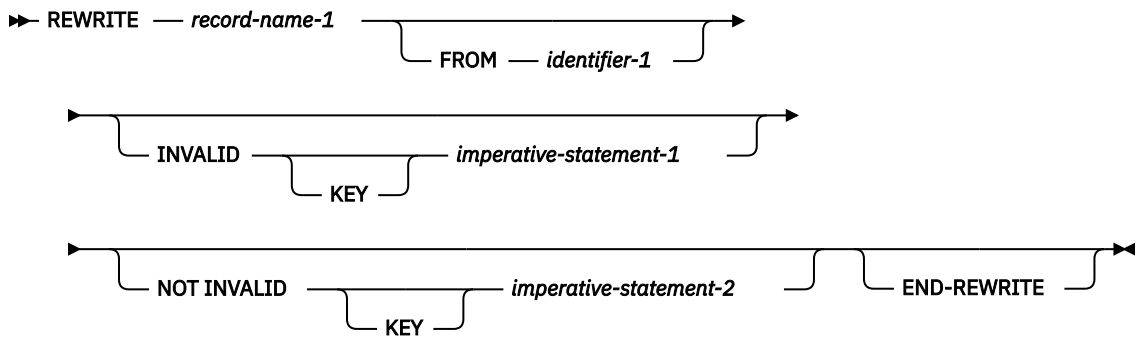
詳しくは、263 ページの『範囲区切りステートメント』を参照してください。

REWRITE ステートメント

REWRITE ステートメントは、直接アクセス・ファイル内にある既存のレコードを 論理的に置き換えます。 REWRITE ステートメントを実行するときは、関連する直接アクセス・ファイルは 入出力モードでオープンされていなければなりません。

REWRITE ステートメントは、行順次ファイルについてはサポートされていません。

フォーマット: REWRITE ステートメント



レコード名-*1*

DATA DIVISION の FD 項目内にある論理レコードの名前でなければなりません。レコード名は修飾することができます。

FROM 句

FROM *ID-1* 句を指定した REWRITE ステートメントの実行結果は、 次のステートメントを指定した順序で実行した場合と同じになります。

```
MOVE identifier-1 TO record-name-1.
REWRITE record-name-1
```

MOVE は、CORRESPONDING 句を指定しない MOVE ステートメントの規則に従って行われます。

ID-1

ID-1 は、以下の項目のいずれかを参照できます。

- すでにオープンされた別のファイルのレコード記述
- 英数字関数、または国別関数
- WORKING-STORAGE SECTION、LOCAL-STORAGE SECTION、または LINKAGE SECTION に定義されたデータ項目

ID-1 は、受け取り項目としてレコード名-*1* が指定された、 MOVE ステートメントの規則に従う有効な送り出し項目でなければなりません。

ID-1 およびレコード名-*1* は、同じストレージ域を参照することはできません。

REWRITE ステートメントの実行後も、*ID-1* の中の情報は使用可能です (『共通の処理機能』にある 274 ページの『INTO 句および FROM 句』を参照)。

INVALID KEY 句

INVALID KEY 条件は、次のいずれか場合に起こります。

- 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 まれている値が、このファイルから最後に取り出されたレコードの基本 RECORD KEY データ項目に等しくない場合
- 基本 RECORD KEY に含まれる値が、ファイル内のどのレコードの基本 RECORD KEY とも等しくない場合
- DUPLICATES の指定されていない ALTERNATE RECORD KEY データ項目の値が、ファイルの中にすでにあるレコードの値と等しい場合

無効キーの処理について詳しくは、『無効キー条件』を参照してください。

END-REWRITE 句

この明示的範囲終了符号は、REWRITE ステートメントの範囲を区切るために使用されます。 END-REWRITE 句を使用することによって、条件的な REWRITE ステートメントを他の条件ステートメントの中にネストすることができます。 END-REWRITE 句は、命令の REWRITE ステートメントと共に使用することもできます。

詳しくは、263 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。.

論理レコードの再使用

REWRITE ステートメントが正常に実行されると、関連付けられたファイルが SAME RECORD AREA 節の中で指定されていない限り、レコード名-*1* の中の論 理レコードはもはや使用可能ではありません (SAME RECORD AREA 節で指定 されている場合、レコードは、その SAME RECORD AREA 節で指定されている 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って影響を受けることはありません。

ファイル制御項目内に FILE STATUS 節が指定されている場合は、関連するファイル状況キーが、REWRITE ステートメントの実行時に更新されます。

順次ファイル

ファイルが順次アクセス・モードの場合、このファイルに対して最後に実行された 前回の入出力ステートメントは、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。 REWRITE ステートメントを実行すると、 READ ステートメントによって取り出されたレコードが論理的に置き換えられます。

レコード名-*1* 内の文字位置の数は、置き換えられるレコード内の文字位置の数と等しくなければなりません。

順次編成のファイルに対しては、INVALID KEY 句を指定することはできません。 EXCEPTION/ERROR プロシージャーを指定することはできます。

索引付きファイル

レコード名-*1* の中の文字位置の数は、置き換えられるレコードの中の文字位置の数 と異なる数にすることができます。

順次アクセス・モードの場合、 置き換えられるレコードは基本 RECORD KEY の中にある値によって指定されます。 REWRITE ステートメントを実行するときには、この値は、 このファイルから読み込まれた最後のレコードの中の基本 RECORD KEY データ項目の値と等しくなければなりません。

INVALID KEY 句および該当する EXCEPTION/ERROR プロシージャーは、両方とも省略することができます。

アクセス・モードがランダムかまたは動的であるとき、置き換えられるレコードは、 基本 RECORD KEY の中にある値によって指定されます。

書き直されるレコードの中の ALTERNATE RECORD KEY データ項目の値は、 置き換えられるレコードの中の値と異なるものにすることができます。 システムは、 後でそのレコードにアクセスする際に、どのレコード・キーでも可能にします。

無効キー条件が生じると、REWRITE ステートメントの実行は失敗し、 更新処理は行われません。この場合、レコード名-*1* の中のデータは影響を受けません (『共通の処理機能』で『無効キー条件』を参照してください)。

相対ファイル

レコード名-*1* の中の文字位置の数は、置き換えられるレコードの中の文字位置の数 と異なる数にすることができます。

順次アクセス・モードの相対ファイルの場合、INVALID KEY 句を指定することはできません。 EXCEPTION/ERROR プロシージャーを指定することはできます。

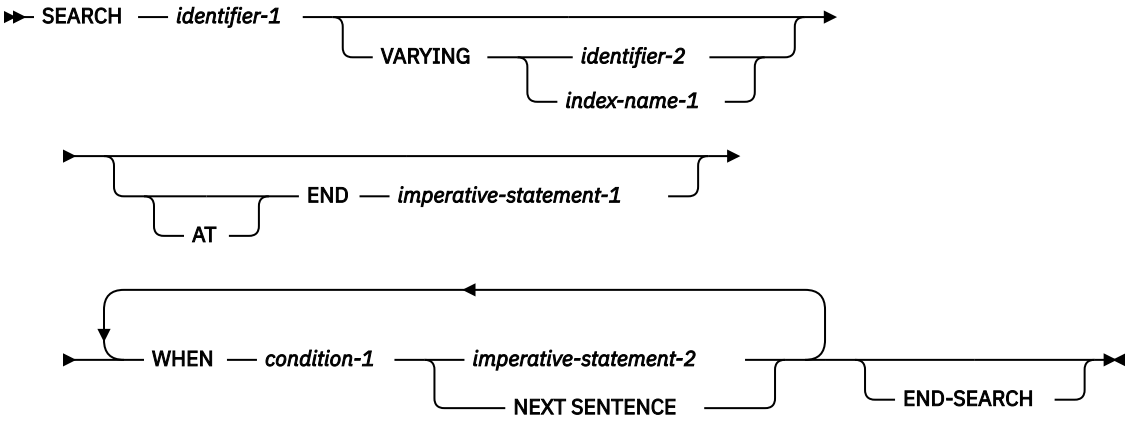
ランダム・アクセス・モードまたは動的アクセス・モードの相対ファイルの場合、INVALID KEY 句または該当する EXCEPTION/ERROR プロシージャーは指定できません。 これらは、両方とも省略することができます。

アクセス・モードがランダムまたは動的である場合、置き換えられるレコードは、 RELATIVE KEY データ項目の中で指定します。 指定したレコードがファイルにない場合は、無効キー条件が生じ、INVALID KEY 命令ステートメントが指定されていれば、それが実行されます。 (『共通の処理機能』で『無効キー条件』を参照してください)。 この場合、更新処理は行われず、レコード名の中のデータは影響を受けません。

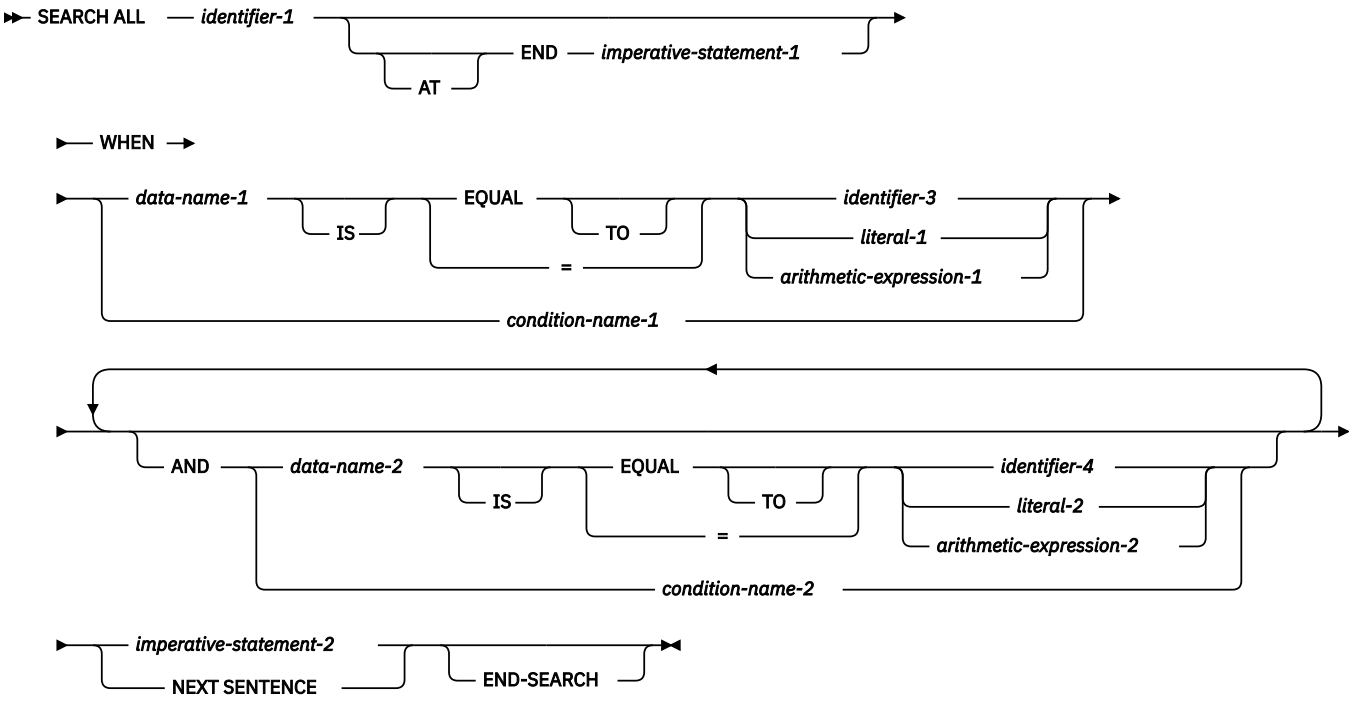
SEARCH ステートメント

SEARCH ステートメントは、指定した条件を満たすエレメントに関してテーブルを 検索します。そして、そのエレメントを指すように関連する指標を調整します。

フォーマット 1: 逐次探索の SEARCH ステートメント



フォーマット 2: 二分探索の SEARCH ステートメント



検索したいテーブルがソートされていない場合は、フォーマット 1 (逐次探索) を使用 してください。 テーブルを逐次探索する場合、あるいは添え字または指標を制御したいときにソート済みのテーブルを検索する場合も、フォーマット 1 を使用します。

テーブル内の全オカレンスを効率的に検索したい場合は、フォーマット 2 (二分探索) を使用してください。テーブルは事前にソートしておく必要があります。また、形式 2 SORT ステートメントを使用してテーブルをソートすることができます。

AT END 句および 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* が実行された場合、これらのス テートメントが GO TO ステートメントで終わっていなければ、SEARCH ステート メントの終わりに制御が渡されます。

AT END 句の機能は、逐次探索および二分探索で同じです。

NEXT SENTENCE

NEXT SENTENCE では、最も近い分離文字ピリオドの後の最初のステートメントに 制御が移されます。

NEXT SENTENCE を END-SEARCH と共に指定すると、END-SEARCH の後のステートメントに制御が 渡されるのではなく、 最も近い後続のピリオドの後のステートメントに制御が渡されます。

フォーマット-2 SEARCH ALL ステートメントの場合、命令ステートメント-*2* および NEXT SENTENCE はいずれも不要です。 それらがなくても、 SEARCH ステートメントは、指標をその条件と一致したテーブルにある値に設定します。

NEXT SENTENCE 句の機能は、逐次探索および二分探索で同じです。

END-SEARCH 句

この明示的範囲終了符号は、SEARCH ステートメントの範囲を区切るために使用されます。 END-SEARCH 句を使用することによって、 条件的な SEARCH ステートメントを他の条件ステートメント内にネストすることができます。

詳しくは、263 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-SEARCH 句の機能は、逐次探索および二分探索で同じです。

逐次探索

このトピックでは、逐次探索のための SEARCH ステートメントの使用について説明します。

***ID-1* (逐次探索)**
ID-1 は検索されるテーブルを識別します。 *ID-1* は、このテーブル内のすべてのオカレンスを参照します。

ID-1 のデータ記述項目には、OCCURS 節を含めなければなりません。

ID-1 のデータ記述項目には、INDEXED BY 句を指定した OCCURS 節を含める必要がありますが、テーブルは、適切に記述された別のテーブルに対して定義された指標を使用して検索することができます。

ID-1 は、OCCURS 節を指定して記述されたデータ項目に従属するデータ項目を参照できます (つまり、*ID-1* は多次元テーブル内の従属テーブルにすることができます)。 この場合、データ記述項目では、テーブルの各次元に対して INDEXED BY 句を指定する必要があります。

ID-1 は、添え字付きまたは参照変更にすることができません。

AT END
関連する WHEN 句で指定された条件を満たすことなく検索操作が終了した場合に存在する条件です。

逐次探索の実行に先立って、*ID-1* に関連付けられた最初の(または唯一の) 指標 (検索指標) の値を、検索対象の最初のオカレンスを示すように設定しておく必要が あります。

多次元テーブルに逐次探索を使用するときには、従属次元ごとに指標の値を設定しておくことも必要です。

SEARCH ステートメントは、検索指標の中の値のみを変更します。VARYING 句が指定されている場合には、指標名-*1* または *ID-2* の値も変更します。 そのため、2 次元から 7 次元のテーブル全体を検索するには、各次元で SEARCH ステートメントを実行する必要があります。 WHEN 句で、すべての次元に指標を指定しなければなりません。 SEARCH ステートメントの実行に先立って、関連する指標を SET ステートメントを使用して初期化しておく必要があります。

SEARCH ステートメントは、現在検索指標が設定されている位置から逐次探索を実行します。

検索が開始されると、*ID-1* に関連付けられた指標の値が、可能な最大の出現数より大きくない場合は、以下の処置が取られます。

- WHEN 句の中にある条件が、それらの記述された順に評価されます。
- 条件が満たされない場合は、*ID-1* の指標が次のテーブル・エレメントに合わせて増加され、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、検索は直ちに終了し、その条件に関連付けられた命令ステートメント-*2* が実行されます。 指標は、条件を満たしたテーブル・エレメントを指しています。NEXT SENTENCE が指定されている場合、制御は最も近いピリオドの後のステートメントに渡されます。
- WHEN 条件が満たされないままテーブルの終わりに達すると (つまり、増分していった指標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)、検索は終了します。

検索が開始されたときに、*ID-1* に関連付けられた指標名の値が、可能な最大の出現数より大きい場合、検索は直ちに終了します。

検索が終了するときに、AT END 句が指定されていると、命令ステートメント-*1* が実行されます。 AT END 句が省略されていると、 制御は SEARCH ステートメントの後にある次のステートメントに渡されます。

例: 多次元逐次探索

以下のコード・フラグメントは、上位テーブル (テーブル R) 内の 3 番目のオカレンスの内部次元 (テーブル C) の検索を示しています。

```
     . . .
     Working-storage section.
     1 G.
       2 R occurs 10 indexed by Rindex.
         3 C occurs 10 ascending key X indexed by Cindex.
           4 X pic 99.
     1 Arg pic 99 value 34.
     Procedure division.
     . . .
* To search within occurrence 3 of table R, set its index to 3
* To search table C beginning at occurrence 1, set its index to 1
     Set Rindex to 3
     Set Cindex to 1
* In the SEARCH statement, specify C without indexes
     Search C
* Specify indexes for both dimensions in the WHEN phrase
       when X(Rindex Cindex) = Arg
       display "Found " X(Rindex Cindex)
     End-search
     . . .
```

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* の指標である場合は、この指標が検索に使用されます。 そうでない場合は、最初の (または唯一の) 指標名が使用されます。
- 指標名-*1* が他のテーブル・エレメントの指標である場合は、*ID-1* 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* によって表される出現数は、検索指標名と同量ずつ、同時に増加されます。

VARYING 指標名-*1* 句が省略された場合は、*ID-1* の最初の (または唯一の) 指標名が検索に使用されます。

INDEXED BY 句の指定のないテーブルを検索するために指標付けが使用される場合、指標付きで定義されたテーブルと指標なしで定義されたテーブルの両方が 同じ長さのテーブル・エレメントを持ち、また同じ数のオカレンスを持つときに限り、 正しい結果が保証されます。

VARYING 句の対象が別のテーブル・エレメントの指標名のときには、1 つの逐次 SEARCH ステートメントは、一度に 2 つのテーブル・エレメントに適用されます。

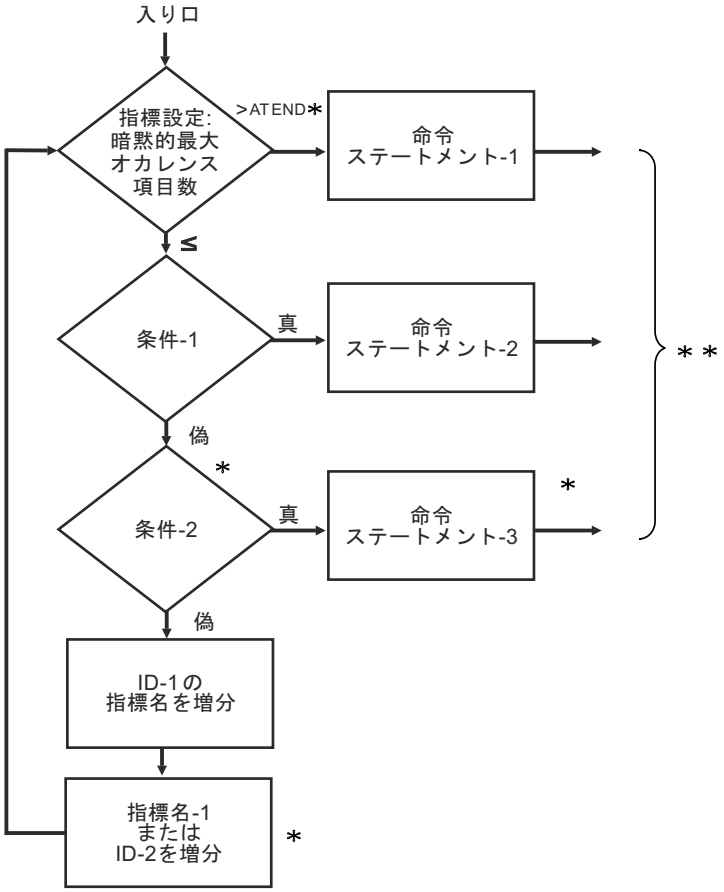
ID-2
指標データ項目または基本整数項目のいずれかでなければなりません。 *ID-2* をウィンドウ表示日付フィールドにすることはできません。 *ID-2* に添え字として、*ID-1* に対して指定した最初の (または唯一の) 指標名を付けることはできません。 検索時には、以下のいずれかの処置が適用されます。

- *ID-2* が指標データ項目である場合には、検索指標が増えるたびに、 指定された指標データ項目が、同時に同じ量だけ増えます。
- *ID-2* が整数データ項目である場合には、検索指標が増えるたびに、 指定されたデータ項目が、同時に 1 だけ増えます。

WHEN 句 (逐次探索)

条件-*1*
238 ページの『条件式』に説明してある条件ならばどの条件でも使用できます。

以下の図は、2 つの WHEN 句を含むフォーマット 1 の SEARCH ステートメントによる処理を示したものです。



* ステートメントにおいて呼び出された場合のみ、これらの操作が行われる。
* * 命令ステートメントがGOTOステートメントで終わっている場合を除き、
制御が次の文に転送される。

二分探索

このトピックでは、二分探索のための SEARCH ステートメントの使用について説明します。

***ID-1* (二分探索)**

ID-1 は検索されるテーブルを識別します。 *ID-1* は、このテーブル内のすべてのオカレンスを参照します。

ID-1 のデータ記述項目には、INDEXED BY 句と KEY IS 句を持つ OCCURS 節が含まれていなければなりません。

ID-1 は、OCCURS 節を含むデータ項目に従属するデータ項目を参照できます (つまり、*ID-1* は多次元テーブル内の従属テーブルにすることができます)。 この場合、データ記述項目では、テーブルの各次元に対して INDEXED BY 句を指定する必要があります。

ID-1 は、添え字付きまたは参照変更にすることができません。

AT END

ここには、WHEN 句で指定された条件を満たせずに検索操作が終了した場合に起こる条件を記述します。

SEARCH ALL ステートメントは、二分探索を使用して実行されます。 *ID-1* に関連付けられた指標 (検索指標) は、SET ステートメントで初期化する必要はありません。 検索指標の値が最初のテーブル・エレメントの値より小さくなったり、最後のテーブル・エレメントの値より大きくなったりすることがないように、検索指標は検索操作中に変更されます。 使用される指標は、必ず OCCURS 節で指定された最初の指標名に関連した指標です。

多次元テーブルに二分探索を使用するときには、事前に SET ステートメントを実行して、従属次元ごとに指標の値を設定しておく必要があります。

SEARCH ステートメントは、検索指標の中の値のみを変更します。 そのため、2 次元から 7 次元のテーブル全体を検索するには、各次元で SEARCH ステートメントを実行する必要があります。 WHEN 句の中で、すべての次元について指標を設定しておかなければなりません。

WHEN 条件を満たさずに検索が終了した場合に AT END 句が指定されていれば、命令ステートメント-*1* が実行されます。 AT END 句が省略されていると、 制御は SEARCH ステートメントの後にある次のステートメントに渡されます。

SEARCH ALL 処理の結果を予測できるのは、次の場合に限ります。

- テーブルの中のデータが、ASCENDING KEY または DESCENDING KEY の順に並んでいる場合。
- WHEN 節の中に指定された ASCENDING KEY または DESCENDING KEY の内容が、 固有のテーブル参照を提供する場合。

WHEN 句 (二分探索)

WHEN 句に比較条件が指定されている場合、その比較の評価はデータ名-*1* が参照 するデータ項目の USAGE に基づきます。 検索引数は、データ名-*1* と同じ USAGE を持つ一時データ項目へ移され、SEARCH に関連する比較演算にはこの一時データ 項目が使用されます。 データ名-1 が数値項目の場合、一時データ項目は、データ名-1 のデータ記述の符号の有無に合わせて、符号付きまたは符号なしとなります。 検索引数が符号付きでデータ名-1 が符号なしの場合、比較を行う前に、検索引数の符号は除去されます。

WHEN 句の条件が、この範囲内にある指標のどの設定値についても満たされない場合、検索は失敗します。この場合、制御は、AT END 句の指定があればその命令ステートメント-*1* に渡されるか、または SEARCH ステートメントの後にある次のステートメントに渡されます。 どちらの場合も、指標の最終的な設定値は予測できません。

WHEN 句の中で条件を満たすことができた場合には、制御は、命令ステートメント-*2* の指定があればそれに渡されるか、または NEXT SENTENCE 句の指定があれ ば、次の実行可能文に渡されます。 この場合の指標には、WHEN 条件を満たすことができた発生項目を指示する値が入っています。

命令ステートメント-*2* が実行された場合、命令ステートメント-*2* が GO TO ステ ートメントで終わっていなければ、SEARCH ステートメントの終わりに制御が渡さ れます。

条件名-*1*、条件名-*2*
指定する条件名はそれぞれ、単一値のみを持ち、このテーブル・エレメントに対して ASCENDING KEY または DESCENDING KEY データ項目と関連付けられている必要があります。

データ名-*1*、 データ名-*2*
ID-1 によって参照されるテーブル・エレメント内で ASCENDING KEY または DESCENDING KEY のデータ項目を指定する必要があり、また *ID-1* に関連する最初の指標名で添え字付けされていなければなりません。 それぞれのデータ名は、修飾することができます。

データ名-*1* は、比較の規則に従って *ID-3*、リテラル-*1*、または算術式-*1* と比較するために有効なオペランドでなければなりません。

データ名-*2* は、比較の規則に従って *ID-4*、リテラル-*2*、または算術式-*2* と比較するために有効なオペランドでなければなりません。

データ名-*1* とデータ名-*2* は、以下を参照することはできません。

- 浮動小数点データ項目
- 可変オカレンス・データ項目を含むグループ項目
- ウィンドウ表示日付フィールド

ID-3*、 *ID-4
ID-1 の ASCENDING KEY または DESCENDING KEY のデータ項目や、*ID-1* の最初の指標名で添え字付けされた項目にはできません。

ID-3 および *ID-4* は、POINTER、FUNCTION-POINTER、または PROCEDURE-POINTER の使用で定義されたデータ項目にはできません。

ID-3 および *ID-4* はウィンドウ表示日付フィールドにはできません。

ID-3 またはリテラル-*1* が国別クラスの場合、データ名-*1* のクラスは国別でなければなりません。

ID-4 またはリテラル-*2* が国別クラスの場合、データ名-*2* のクラスは国別でなければなりません。

リテラル-*1*、 リテラル-*2*
リテラル-*1* またはリテラル-*2* は、データ名-*1* またはデータ名-*2* との比較に有効なオペランドでなければなりません。

算術式
234 ページの『算術式』で定義された式のいずれかにできますが、以下の制限があります。 算術式における ID は、*ID-1* の ASCENDING KEY または DESCENDING KEY のデータ項目や、*ID-1* の最初の指標名で添え字付けされた項目にはできません。

WHEN 句の中で ASCENDING KEY データ項目または DESCENDING KEY データ項目を明示的もしくは暗黙的に指定する場合は、*ID-1* に対するすべての先行する ASCENDING KEY データ名または DESCENDING KEY データ名も指定しなければなりません。

SEARCH ステートメントに関する考慮事項

このトピックでは、SEARCH ステートメントの使用に関する考慮事項について説明します。

指標データ項目は、添え字として使用することはできません。 それらに対する直接参照に制限があるためです。

可変長テーブルに対して SEARCH ステートメントを正しく実行するには、OCCURS DEPENDING ON 節 (データ名-*1*) のオブジェクトに、テーブルの現在の長さを指定する値が含まれている必要があります。

SEARCH ステートメントの範囲は、以下の項目のいずれかによって終了することができます。

- ネスト構造で同じレベルにある END-SEARCH 句。
- 分離文字ピリオド。
- 先行する IF ステートメントに関連する ELSE 句または END-IF 句。

SET ステートメント

このトピックで説明されているように、SET ステートメントは操作を実行するために使用します。

操作は以下のとおりです。

- テーブル・エレメントに関連付けられた値を指標名に関連付けられた指標に入れる。
- オカレンス項目数を増減する。
- 外部スイッチの状況を ON または OFF に設定する。
- 条件を真にするためにデータを条件名に移動する。
- USAGE POINTER データ項目を、あるデータ・アドレスに設定する。
- USAGE PROCEDURE-POINTER データ項目を、ある項目アドレスに設定する。
- USAGE FUNCTION-POINTER データ項目を、ある項目アドレスに設定する。
- 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。

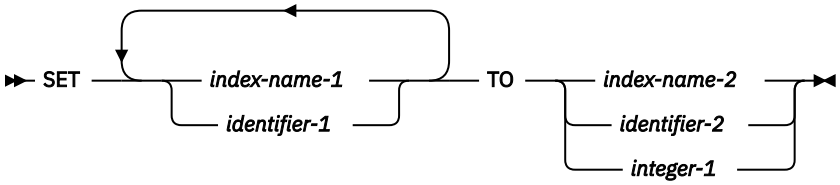
指標名は、OCCURS 節の INDEXED BY 句を通して付与されたテーブルと関連付けられています。 プログラムでさらに定義されることはありません。

SET ステートメントの中で、 送り出しフィールドと受け取りフィールドがそれらのストレージの一部を共用している場合 (つまり、 オペランドのオーバーラップがあると)、SET ステートメントを実行した結果は未定義のままです。

フォーマット 1: 基本的なテーブル処理のための SET

この形式の SET ステートメントを実行すると、受け取りフィールドの現行値が、 送り出しフィールドの値を変換した値で置き換えられます。

フォーマット 1: 基本的なテーブル処理のための SET ステートメント



指標名-*1*
受け取りフィールド。

OCCURS 節の INDEXED BY 句で指定された指標に名前を付ける必要があります。

ID-1
受け取りフィールド。

指標データ項目または基本数字整数項目のいずれかを指定する必要があります。 受け取りフィールドをウィンドウ表示日付フィールドにすることはできません。

指標名-*2*
送り出しフィールド。

OCCURS 節の INDEXED BY 句で指定した指標に名前を付ける必要があります。 SET ステートメントが実行される前の指標値は、関連付けられたテーブルのオカレンス項目数に対応する必要があります。

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を指定する必要があります。 送り出しフィールドをウィンドウ表示日付フィールドにすることはできません。

整数-*1*
送り出しフィールド。

これは、正の整数である必要があります。

以下の表は、フォーマット 1 の SET ステートメントにおける 送り出しフィールドと受け取りフィールドの有効な組み合わせを示しています。

表 *55.* **フォーマット-*1* の *SET* ステートメントの送り出しフィールドと受け取りフィールド**

送り出しフィールド	指標名 受け取りフィールド	指標データ項目受け取りフィールド	整数データ項目受け取りフィールド
指標名*	有効	有効**	有効
指標データ項目*	有効**	有効**	無効
整数データ項目	有効	無効	無効
整数リテラル	有効	無効	無効

*指標名とは、OCCURS 文節の INDEXED BY 句で指定した指標を指します。指標データ項目は、USAGE IS INDEX 節を使用して定義されます。

**変換は何も行われません。

受け取りフィールドは、指定されている順に左から右へ処理されます。 *ID-1* に関連付けられた添え字付けまたは指標付けはいずれも、受け取り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で使用される値は、SET ステートメントの実行開始時の値です。

SEARCH ステートメントまたは PERFORM ステートメント実行後の指標の値は未 定義となることがあります。したがって、他のテーブル処理操作を行う前に、フォ ーマット-1 SET ステートメントでそのような指標を再初期設定してください。

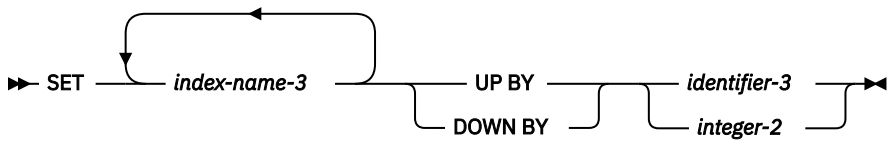
指標名-*2* が、OCCURS DEPENDING ON 節を含む従属項目を持つテーブルに対するものである場合は、未定義の値が *ID-1* に受け取られる可能性があります。

複合 OCCURS DEPENDING ON について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『複合 *OCCURS DEPENDING ON*』を参照してください。

フォーマット 2: 指標調整用の SET

この形式の SET ステートメントを実行すると、受け取り指標値が送り出しフィール ド内の値に対応する値だけ増加 (UP BY) または減少 (DOWN BY) します。

フォーマット 2: 指標調整用の SET ステートメント



受け取りフィールド は指標名-*3* によって指定された指標です。 指標値は、SET ステートメント実行前も実行後も、関連するテーブル内のオカレンス項目数に対応する必要があります。

送り出しフィールド は、*ID-3* または整数-*2* として定義できます。 ID-3 は基本整数データ項目、整数-2 はゼロ以外の整数でなければなりません。 *ID-3* をウィンドウ表示日付フィールドにすることはできません。

フォーマット-2 SET ステートメントが実行されると、受け取りフィールドの内容が、*ID-3* または整数-*2* の値で表される出現数に対応する値だけ増加 (UP BY) または減少 (DOWN BY) します。 受け取りフィールドは、指定されている順に左から右へと処理されます。 SET ステートメントの実行開始時に増加するフィールドまたは減少するフィールドの値が、すべての受け取りフィールドに使用されます。

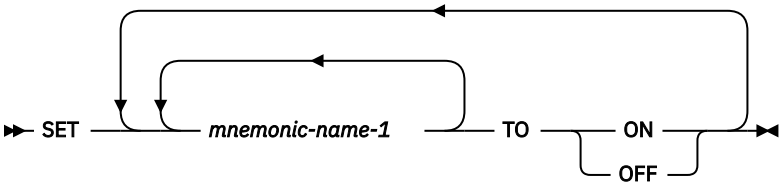
指標名-*3* が OCCURS DEPENDING ON 節を含む従属項目を持つテーブルに対するものである場合、または ODO オブジェクトがフォーマット-2 SET ステートメントの実行前に変更された場合、指標名-*3* には、関連するテーブルの出現数に対応する値が入りません。

複合 OCCURS DEPENDING ON について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『複合 *OCCURS DEPENDING ON*』を参照してください。

フォーマット 3: 外部スイッチ用の SET

この形式の 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外部 スイッチの各状況が、ON または OFF に切り替わります。

フォーマット 3: 外部スイッチ用の SET ステートメント

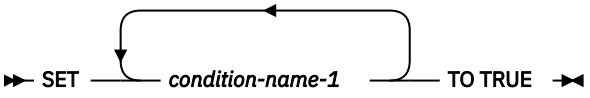


簡略名-*1*
外部スイッチと関連付けられている必要があり、その状況は変更可能です。

フォーマット 4: 条件名用の SET

この形式の SET ステートメントを実行すると、条件名に関連する値が VALUE 節の規則に従って条件変数の中に入れられます。

フォーマット 4: 条件名用の SET ステートメント



条件名-*1*
条件変数と関連付けられている必要があります。

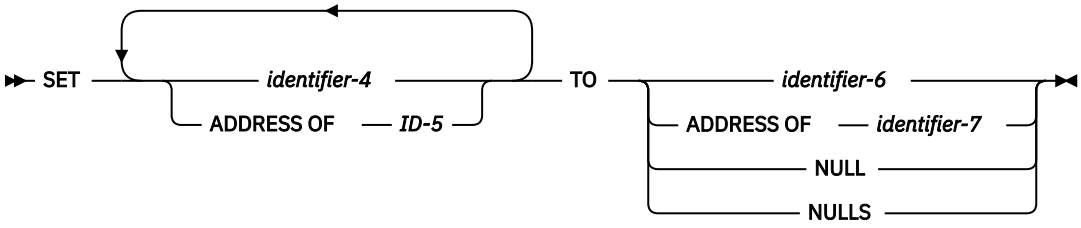
条件名-*1* の VALUE 節内で複数のリテラルが指定されている場合、関連付けられた条件変数は、最初のリテラルに等しく設定されます。

複数の条件名を指定すると、 その実行結果は、それらの条件名が SET ステートメントの中で指定されている順に、 各条件名に関して個別の SET ステートメントを記述したものと同じになります。

フォーマット 5: USAGE IS POINTER データ項目用の SET

この形式の SET ステートメントを実行すると、受け取りフィールドの現行値は、送 り出しフィールドの中にあるアドレス値によって置き換えられます。

フォーマット 5: USAGE IS POINTER データ項目用の SET ステートメント



ID-4
受け入れフィールド。

USAGE IS POINTER として定義されている必要があります。

ADDRESS OF *ID-5*
受け入れフィールド。

ID-5 は、LINKAGE SECTION の中に定義されたレベル 01 またはレベル 77 の項目である必要があります。 これらの項目のアドレスは、TO 句の中に指定されたオペランドの値に設定されます。

ID-5 は、参照変更にすることはできません。

ID-6
送り出しフィールド。

USAGE IS POINTER として定義されている必要があります。

プログラムが所有する WORKING-STORAGE SECTION、FILE SECTION、または LOCAL-STORAGE SECTION 内のアドレスを含めることはできません。

ADDRESS OF*ID-7*
送り出しフィールド。

identifier-7 には、LINKAGE SECTION、WORKING-STORAGE SECTION、または LOCAL-STORAGE SECTION 内の 66 または 88 を除いたレベルの項目を指定する必要があります。 ADDRESS OF *identifier-7* には、ID の内容ではなく、ID のアドレスが入ります。

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。

以下の表は、フォーマット 5 の SET ステートメントにおける 送り出しフィールドと受け取りフィールドの有効な組み合わせを示しています。

表 *56.* **フォーマット *5* の *SET* ステートメントの送り出しフィールドと受け取りフィールド**

送り出しフィールド	USAGE IS POINTER 受け取りフィールド	ADDRESS OF 受け取りフィールド	NULL/NULLS 受け取りフィールド
USAGE IS POINTER	有効	有効	無効
ADDRESS OF	有効	有効	無効
NULL/NULLS	有効	有効	無効

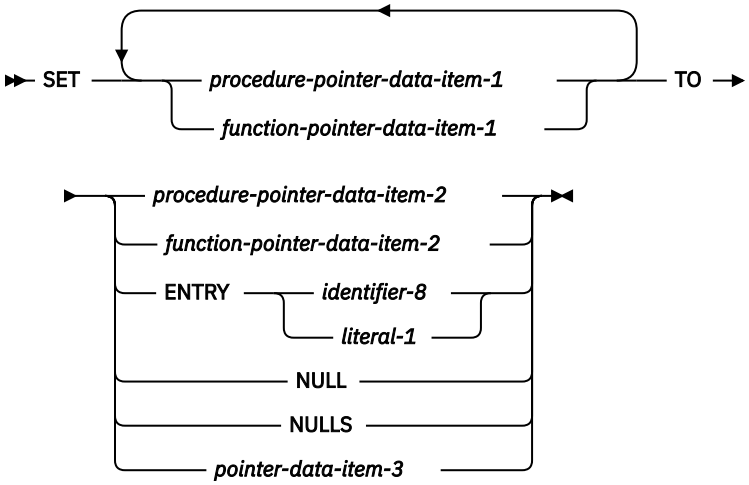
フォーマット 6: プロシージャー・ポインターおよび関数ポインタ ーのデータ項目用の SET

このフォーマットの SET ステートメントを実行すると、受け取りフィールドの現行 値は、送り出しフィールドで指定されたアドレス値によって置き換えられます。

実行時、関数ポインターとプロシージャー・ポインターでは、COBOL プログラムの 1 次入り口点のアドレス、 COBOL プログラムの代替入り口点のアドレス、または COBOL 以外のプログラムの入り口点のアドレスを参照できます。 これらは NULL の場合もあります。

COBOL で C 関数と相互協調処理を行う場合は、プロシージャー・ポインターよりも関数ポインターのほうが簡単に使用できます。

フォーマット 6: プロシージャー・ポインターおよび関数ポインター用の SET ステートメント



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*
USAGE IS PROCEDURE-POINTER として記述され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-*1* は受け取りフィールド、プロシージャー・ポインター・データ項目-*2* は送り出しフィールドです。

関数ポインター・データ項目-*1*、 関数ポインター・データ項目-*2*
USAGE IS FUNCTION-POINTER として記述されている必要があります。 関数ポインター・データ項目-*1* は受け取りフィールド、関数ポインター・データ項目-*2* は送り出しフィールドです。

ID-8
その値をプログラム名にできるように、英字または英数字として定義する必要があります。 詳しくは、84 ページの『PROGRAM-ID 段落』を参照してください。 . COBOL 以外のプログラムの入り口点の場合、 *ID-8* に @、#、および $ の各文字を含めることができます。

リテラル-1
英数字リテラルでなければならず、 またプログラム名形成の規則に適合している必要があります。 形成規則の詳細は、84 ページの『PROGRAM-ID 段落』のプログラム名の説明を参照してください。

ID-8 またはリテラル-*1* は、以下に示すタイプの入り口点の 1 つを参照する必要があります。

- PROGRAM-ID 段落によって定義されている COBOL プログラムの 1 次入り口点。 PROGRAM-ID は、コンパイル単位の一番外側のプログラムを参照する必要があります。 ネストされたプログラムは参照してはなりません。
- COBOL ENTRY ステートメントによって COBOL プログラムのために定義されている代替となる入り口点。
- 非 COBOL プログラムの入り口点。

SET...TO ENTRY ステートメントが参照するプログラム名は、 PGMNAME コンパイラー・オプションの影響を受けることがあります。 詳しくは、*COBOL for Linux on x86* プログラミング・ガイド内の *PGMNAME* を参照してください。

NULL、NULLS
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。

ポインター・データ項目*-3*
USAGE POINTER で定義される必要があります。 ポインター・データ項目*-3* を COBOL 以外のプログラムで設定して、有効なプログラム入り口点を指すようにする必要があります。

COBOL/C インターオペラビリティーの例

以下の例では、関数ポインターをサービスに戻す C 関数への COBOL CALL を説明しており、サービスへの COBOL CALL が続きます。

```
IDENTIFICATION DIVISION.
PROGRAM-ID DEMO.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 FP USAGE FUNCTION-POINTER.
PROCEDURE DIVISION.
    CALL "c-function" RETURNING FP.
    CALL FP.
```

フォーマット 7: USAGE OBJECT REFERENCE データ項目用の SET

このフォーマットは現在サポートされていません。

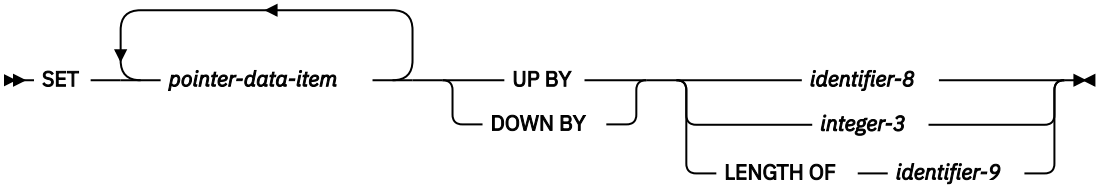
フォーマット 8: 動的長基本項目の長さの SET

このフォーマットは現在サポートされていません。

フォーマット 9: ポインターの調整のための SET

このフォーマットの 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。

フォーマット 9: ポインター調整用の SET ステートメント



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であってはなりません。

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 18 ページの『LENGTH OF』 を参照してください。

フォーマット 10: ロケール・カテゴリーの SET

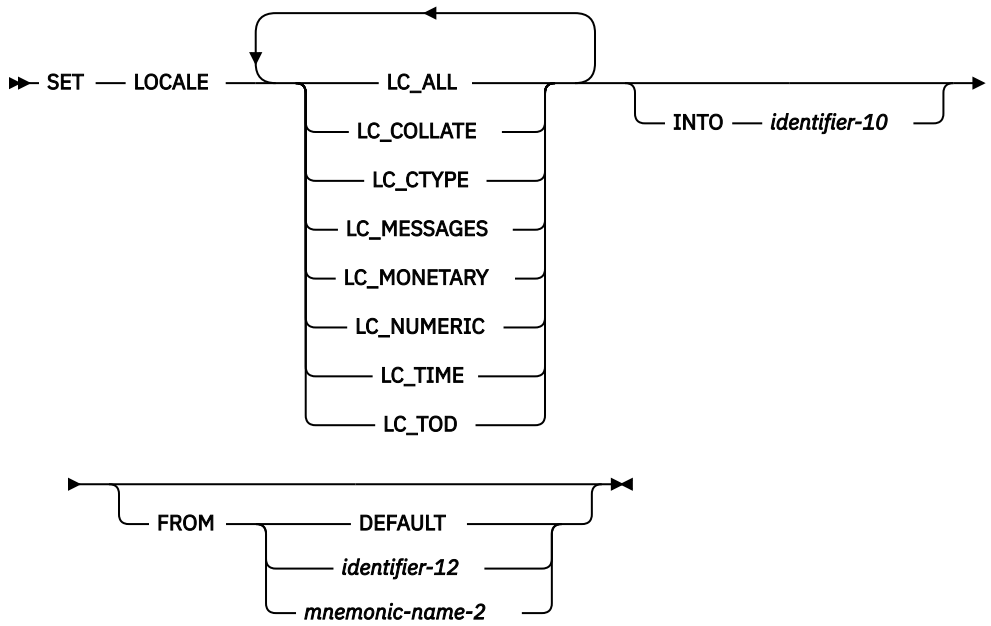
このフォーマットの SET ステートメントによって、現行ロケールのロケール・カテゴリーの設定と照会が可能となります。

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。

フォーマット 10: ロケール・カテゴリー用の SET ステートメント



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。

identifier-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 10 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。

identifier-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。

mnemonic-name-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。

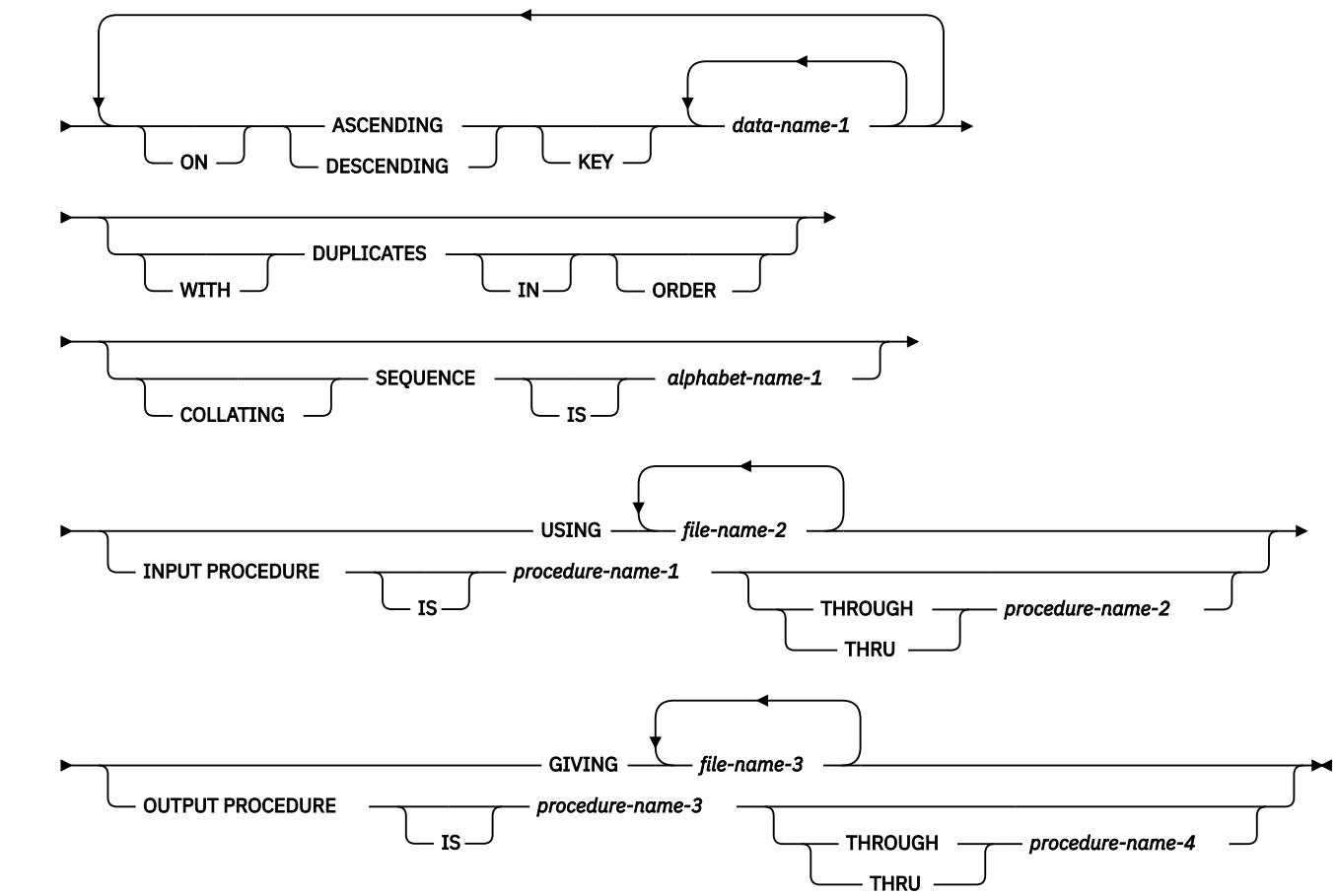
SORT ステートメント

SORT ステートメントにより、一連のレコードまたはテーブル・エレメントがユーザー指定の順序で配置されます。

ファイルをソートする場合、SORT ステートメントは、1 つ以上のファイルからレコードを受け取り、指定されたキーに従ってそれらをソートし、出力プロシージャーを介して、または出力ファイル内で、ソートされたレコードを使用できるようにします。

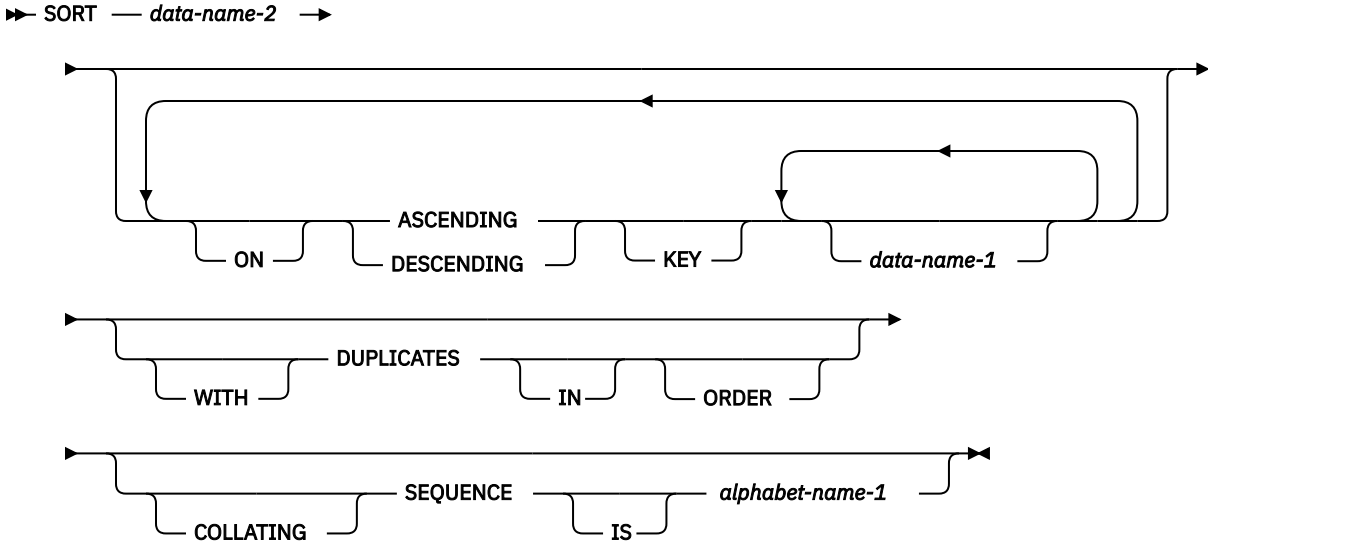
テーブルをソートする場合、SORT ステートメントは、指定されたテーブル・キーに従ってテーブル・エレメントをソートします。

フォーマット



形式 1 SORT ステートメントは、宣言部分内を除き、PROCEDURE DIVISION のどこにでも指定できます。
322 ページの『MERGE ステートメント』も参照してください。

形式 2: テーブル SORT ステートメント



形式 2 SORT ステートメントは、PROCEDURE DIVISION のどこにでも指定できます。

ファイル名-*1*
ソートされるレコードを記述している SD 項目の中で指定されている名前。

SORT ステートメントの中にあるファイル名の対を、同じ SAME SORT AREA 節または SAME SORT-MERGE AREA 節の中で指定することはできません。 GIVING 節に関連したファイル名 (ファイル名-*3* ...) は、SAME AREA 節には指定できません。 ただし、それらを SAME RECORD AREA 節に関連付けることは可能です。

データ名-*2*
以下の規則に従ったテーブル・データ名を指定します。

- データ名-*2* では、データ記述項目内に OCCURS 節がなければなりません。
- データ名-*2* は修飾することができます。
- データ名-*2* には添え字を付けることができます。テーブルの右端または唯一の添え字は省略するか、ワード ALL で置き換える必要があります。

データ名-*2* によって参照されるテーブル・エレメントの出現数は、OCCURS 節内の規則によって決定されます。 ソートされたテーブル・エレメントは、データ名-*2* によって参照されるテーブルと同じテーブルに配置されます。

ASCENDING KEY 句および DESCENDING KEY 句 (形式 1)

この句は、指定されたソート・キーに基づいて、 レコードを昇順または降順 (どちらになるかは指定された句次第) で処理することを指定します。

データ名-*1*
SORT ステートメントがソートの際に基準として使用する KEY データ項目を指定します。 そのようなデータ名はそれぞれ、ファイル名-*1* に関連するレコードの中のデータ項目を識別する必要があります。 KEY という語の後に置かれたデータ名は、 レベルが高いものから順に左から右へと SORT ステートメントの中にリストします。 その際、これらのデータ名が KEY 句の中でどのように分割されるかは関係ありません。 左端のデータ名が最もレベルの高いキーとなり、次のデータ名が 2 番目のレベルのキーとなる、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、各入力ファイルの中で、 物理的に同じ位置になければならず、 また同じデータ・フォーマットを持っていなければなりません。 しかし、同じデータ名を持っている必要はありません。
- ファイル名-*1* が 2 つ以上のレコード記述を持つ場合には、KEY データ項目は、 どちらか一方のレコード記述の中にのみ記述されている必要があります。
- ファイル名-*1* が可変長レコードを含んでいる場合には、KEY データ項目はすべて、 レコードの最初の *n* 個の文字位置内に入っていなければなりません (*n* はファイル名-*1* で指定されている最小レコード・サイズ)。
- KEY データ項目は、OCCURS 節を含んでいたり、OCCURS 節を含む項目に従属していたりすることはできません。
- KEY データ項目には、以下を指定できません。
  - 可変位置項目
  - 可変オカレンス・データ項目を含むグループ項目
  - ウィンドウ表示日付フィールド
  - USAGE NATIONAL で記述された数字カテゴリー (国別 10 進数項目)
  - USAGE NATIONAL で記述された外部浮動小数点カテゴリー (国別浮動小数点項目)
  - DBCS カテゴリー
- KEY データ項目は、 修飾することができます。
- KEY データ項目は、以下のデータ・カテゴリーのいずれかに属することができます。
  - 英字、英数字、英数字編集
  - 数字 (USAGE NATIONAL の数字を除く)

- 数字編集 (USAGE DISPLAY または NATIONAL)
- 内部浮動小数点または display 浮動小数点
- NCOLLSEQ(BINARY) コンパイラー・オプションが有効な場合は、国別または国別編集。 バイナリー照合シーケンスが国別キーに適用されます。

ファイル名-*3* が、基本レコード・キーがレコード・キー名ではない索引付きファイルを参照している場合、データ名-*1* の最初の指定は ASCENDING 句に関連付けられていなければならず、そのデータ名-*1* によって参照されるデータ項目は、そのファイルの基本レコード・キーに関連付けられているデータ項目と同じ文字位置をこのレコード内で占めていなければなりません。

ファイル名-*3* が索引付きファイルを参照しており、その索引付きファイルの基本レコード・キーがレコード・キー名 である場合は、レコード・キー名 の SOURCE 句内の各データ名に対応するデータ名-*1* を指定する必要があります。各データ名-*1* は ASCENDING 句に関連付けられていなければならず、各データ名-*1* は SOURCE 句内の対応するデータ名と同じ文字位置を占めていなければなりません。

レコード・キー名について詳しくは、103 ページの『FILE-CONTROL 段落』を参照してください。

ソート処理の方向は、次に示すように ASCENDING または DESCENDING のどちらのキーワードを指定するかによって異なります。

- ASCENDING を指定すると、最低のキー値から最高のキー値へのシーケンスとなります。
- DESCENDING を指定すると、最高のキー値から最低のキー値へのシーケンスとなります。
- KEY データ項目が英字、英数字、英数字編集、または数字編集の場合は、キー値のシーケンスは使用されている照合シーケンスによって決まります (378 ページの『COLLATING SEQUENCE 句 (両方の形式)』を参照)。
- KEY データ項目が USAGE NATIONAL で記述されている場合、KEY 値のシーケンスは国別文字の 2 進値に基づきます。
- KEY が display 浮動小数点項目の場合は、コンパイラーはデータ項目を、数値データとしてではなく、同じサイズの文字データのキーとして取り扱います。 レコードがソートされるシーケンスは、使用している照合シーケンスによって決まります。
- KEY データ項目が内部浮動小数点項目の場合は、キー値のシーケンスは数値順になります。
- COLLATING SEQUENCE 句が指定されていない場合は、比較条件のオペランド比較規則に従ってキーが比較されます。 243 ページの『一般比較条件』を参照してください。 .
- COLLATING SEQUENCE 句が指定されている場合は、英字、英数字、英数字編集、外部浮動小数点、および数字編集の各カテゴリーのキー・データ項目に対して、指定された照合シーケンスが使用されます。 その他すべてのキー・データ項目に対しては、比較条件でのオペランドの比較規則に従って比較が行われます。

ASCENDING KEY 句および DESCENDING KEY 句 (形式 2)

この句は、指定された句およびソート・キーに基づいて、テーブル・エレメントを昇順または降順に処理することを指定します。

データ名-*1*

以下の規則に従った KEY データ名を指定します。

- キー・データ名で示されるデータ項目は、データ名-*2* によって参照されるデータ項目と同一であるか、そのデータ項目に従属していなければなりません。
- KEY データ項目は、 修飾することができます。
- KEY データ項目は、以下のデータ・カテゴリーのいずれかに属することができます。
  - 英字、英数字、英数字編集
  - 数字 (USAGE NATIONAL の数字を除く)
  - 数字編集 (USAGE DISPLAY または NATIONAL)
  - 内部浮動小数点または display 浮動小数点
  - 国別または国別編集

- KEY データ項目には、以下を指定できません。
  - 可変位置項目
  - 可変オカレンス・データ項目を含むグループ項目
  - USAGE NATIONAL で記述された数字カテゴリー (国別 10 進数項目)
  - USAGE NATIONAL で記述された外部浮動小数点カテゴリー (国別浮動小数点項目)
  - DBCS カテゴリー
  - クラス・ポインター
  - USAGE POINTER、USAGE PROCEDURE-POINTER、または USAGE FUNCTION-POINTER
  - 添え字付き
- KEY データ名で示されるデータ項目がデータ名-*2* に従属している場合は、以下の規則が適用されます。
  - データ項目は OCCURS 節で記述できません。
  - データ項目は、同じくデータ名-*2* に従属し、OCCURS 節を含んでいる項目に従属することはできません。

データ名-*2* によって参照されるテーブルの記述に KEY 句が含まれている場合にのみ、KEY 句を省略できます。

ワード ASCENDING および DESCENDING は、別の ASCENDING または DESCENDING のワードが検出されるまでに出現するすべてのデータ名-*1* にわたって適用されます。

データ名-*1* によって参照されるデータ項目はキー・データ項目であり、これらのデータ項目によって、ソートされるテーブル・エレメントの保管順が決まります。 キーの重要度の順序は、データ項目が SORT ステートメントで指定される順序であり、ASCENDING 句または DESCENDING 句との関連は考慮されません。

SORT ステートメントは、データ名-*2* によって参照されるテーブルをソートし、ソートされたテーブルをデータ名-*2* に示します。 ソート順は、ASCENDING 句および DESCENDING 句 (指定されている場合)、またはデータ名-*2* に関連付けられている KEY 句によって決まります。

ソート操作の方向は、以下のように、キーワード ASCENDING または DESCENDING の指定によって異なります。

- ASCENDING を指定すると、順序は最低キー値から最高キー値になります。
- DESCENDING を指定すると、順序は最高キー値から最低キー値になります。
- KEY データ項目が USAGE NATIONAL で記述されている場合、KEY 値のシーケンスは国別文字の 2 進値に基づきます。
- KEY データ項目が内部浮動小数点である場合、キー値のシーケンスは数値順になります。
- COLLATING SEQUENCE 句が指定されていない場合は、英字、英数字、英数字編集、外部浮動小数点、および数字編集の各カテゴリーのキー・データ項目に対して、EBCDIC シーケンスが使用されます。 その他すべてのキー・データ項目に対しては、比較条件でのオペランドの比較規則に従って比較が行われます。
- COLLATING SEQUENCE 句が指定されている場合は、英字、英数字、英数字編集、外部浮動小数点、および数字編集の各カテゴリーのキー・データ項目に対して、指定された照合シーケンスが使用されます。 その他すべてのキー・データ項目に対しては、比較条件でのオペランドの比較規則に従って比較が行われます。

テーブル・エレメントが保管される相対順位を決定するために、比較条件でのオペランドの比較規則に従って、対応するキー・データ項目の内容が比較されます。 ソートは、以下の規則に従って、最上位キー・データ項目から開始されます。

- 対応するキー・データ項目の内容が同一ではなく、キーが ASCENDING 句に関連付けられている場合、テーブル・エレメントに含まれているキー・データ項目の値が小さいほど、そのテーブル・エレメントのオカレンス番号は小さくなります。
- 対応するキー・データ項目の内容が同一ではなく、キーが DESCENDING 句に関連付けられている場合、テーブル・エレメントに含まれているキー・データ項目の値が大きいほど、そのテーブル・エレメントのオカレンス番号は小さくなります。

- 対応するキー・データ項目の内容が同一である場合は、次の最上位キー・データ項目の内容に基づいて決定が行われます。

KEY 句が指定されていない場合は、データ名-*2* によって参照されるテーブルのデータ記述項目内の KEY 句によって順序が決定されます。

KEY 句が指定されている場合は、その KEY 句が、データ名-*2* によって参照されるテーブルのデータ記述項目に指定されているすべての KEY 句をオーバーライドします。

データ名-*1* が省略されている場合は、データ名-*2* によって参照されるデータ項目がキー・データ項目になります。

DUPLICATES 句 (形式 1)

DUPLICATES 句が指定され、あるレコードに関連するすべてのキー・エレメントの内容が、 1 つまたは複数の他のレコードの中の対応するキー・エレメントに一致している場合は、 これらのレコードは次のような順序で戻されます。

- 関連付けられた入力ファイルの SORT ステートメント中に指定されている通りの順序。 ある 1 個のファイル内では、レコードがそのファイルからアクセスされるときの順序。
- 入力プロシージャーが指定されているとき、 これらのレコードが入力プロシージャーにより解放されるときの順序。

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。

DUPLICATES 句 (形式 2)

以下の両方の条件を満たしている場合、テーブル・エレメントの内容は、ソート操作を行う前の順序と同じ相対順序になります。

- DUPLICATES 句が指定されている。
- あるテーブル・エレメントに関連付けられているすべてのキー・データ項目の内容が、1 つ以上の他のテーブル・エレメントに関連付けられている、対応するキー・データ項目の内容に等しい。

DUPLICATES 句が指定されておらず、2 番目の条件が存在する場合、これらのテーブル・エレメントの内容の相対順序は未定義になります。

COLLATING SEQUENCE 句 (両方の形式)

この句で指定する照合シーケンスは、このソート処理で KEY データ項目に対して行われる英数字比較で使用されます。

COLLATING SEQUENCE 句は、英字または英数字以外のキーには影響を与えません。

COLLATING SEQUENCE 句は、1 バイト ASCII コード・ページが有効な場合にのみ有効です。

英字名-*1*
これは SPECIAL-NAMES 段落の ALPHABET 節で指定されている必要があります。 英字名-*1* は ALPHABET 節の句のいずれか 1 つに関連付けることができ、次のような結果になります。

STANDARD-1
照合シーケンスは、文字の 16 進値の順序に基づきます。

STANDARD-2
照合シーケンスは、文字の 16 進値の順序に基づきます。

NATIVE
ロケールによって指示された照合シーケンスが選択されます。

EBCDIC
EBCDIC 照合シーケンスがすべての英数字比較のために使用されます。 (EBCDIC 照合シーケンスは、535 ページの『付録 D EBCDIC および ASCII の照合シーケンス』 に示されています。)

リテラル
ALPHABET-NAME 節でリテラルを指定したことにより設定された照合シーケンスが、すべての英数字比較のために使用されます。

COLLATING SEQUENCE 句を省略した場合は、 OBJECT-COMPUTER 段落の PROGRAM COLLATING SEQUENCE 節 (指定されている場合) で使用したい照合シーケンスを指定します。 COLLATING SEQUENCE 句と PROGRAM COLLATING SEQUENCE 節を両方とも省略した場合には、 COLLSEQ コンパイラー・オプションによって指示された照合シーケンスが使用されます。

USING 句

ファイル名-*2* , ...
入力ファイル。

USING 句が指定されている場合、ファイル名-*2*、... (つまり、入力ファイル) の中のすべてのレコードは自動的にファイル名-*1* に移動されます。 SORT ステートメントの実行時に、これらのファイルがオープンしないでください。 コンパイラーが自動的にこれらのファイルをオープンし、 読み取り、レコードを使用可能にし、そしてクローズします。 EXCEPTION/ERROR プロシージャーがこれらのファイルに対して指定されていると、 コンパイラーはこれらのプロシージャーへの必要なリンケージを設定します。

すべての入力ファイルは、DATA DIVISION の中の FD 項目に記述されている必要があります。

USING 句が指定され、ファイル名-*1* に可変長レコードが含まれている場合、入力ファイル (ファイル名-*2*、...) に含まれるレコードのサイズは、ファイル名-*1* に記述されている最小レコード以上かつ最大レコード以下でなければなりません。 ファイル名-*1* に固定長レコードが含まれている場合、入力ファイルに含まれるレコードのサイズは、ファイル名-*1* に対して記述されている最大レコード以下でなければなりません。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『ソートまたはマージへの入力の記述』を参照してください。

INPUT PROCEDURE 句

この句によって、ソート処理を開始する前に、 入力レコードを選択したり修正したりするために使う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の中の最初の (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*
入力プロシージャーの最後のセクションまたは段落を識別します。

入力プロシージャーは、ファイル名-*1* によって参照されるファイルに対する RELEASE ステートメントの実行によって 1 つずつ使用可能にされるレコードを選択、 修正、またはコピーするために必要なプロシージャーで構成することができます。 この範囲には、入力プロシージャーの範囲内の CALL、EXIT、GO TO、PERFORM、および XML PARSE の各ステートメントの実行による制御移動の結果として実行されるすべてのステートメントと、 入力プロシージャーの範囲にあるステートメント実行の結果として実行される宣言型プロシージャーの中のすべてのステートメントが含まれます。 入力プロシージャーの範囲内で、MERGE ステートメント、RETURN ステートメント、形式 1 の SORT ステートメントを実行することはできません。

入力プロシージャーが指定されている場合、制御がその入力プロシージャーに渡されない限り、ファイル名-*1* によって参照されたファイルを SORT ステートメントが順序付けすることはできません。 コンパイラーは、入力プロシージャーの中の最後のステートメントの終わりに RETURN 挿入します。 制御が入力プロシージャーの中の最後のステートメントに渡されると、ファイル名-*1* によって参照されるファイルに対して以前に解放されてい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 句が指定されたとき、ファイル名-*1* にあるソートされたレコードはすべて、 自動的に出力ファイル (ファイル名-*3*...) に転送されます。.

すべての出力ファイルは、DATA DIVISION の中の FD 項目に記述されている必要があります。

出力ファイル (ファイル名-*3*、...) に可変長レコードが含まれている場合、ファイル名-*1* に含まれるレコードのサイズは、その出力ファイルについて記述された最小レコード以上かつ最大レコード以下でなければなりません。 出力ファイルに固定長レコードが含まれている場合、ファイル名-*1* に含まれるレコードのサイズは、出力ファイルについて記述された最大レコードを超えてはいけません。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『ソートまたはマージからの出力の記述』を参照してください。

SORT ステートメントの実行時に、出力ファイル (ファイル名-*3*、...) がオープンされないようにします。各出力ファイルに対して、SORT ステートメントが実行されると、 次のことが行われます。

- ファイルの処理が開始されます。 この開始は、 OUTPUT 句指定の OPEN ステートメントが実行された場合と同様にして行われます。
- ソートされた論理レコードが戻され、ファイル上に書き込まれます。 各レコードは、 何もオプショナルの句が指定されていない WRITE ステートメントが実行された場合と同様にして書き込まれます。

  相対ファイルの場合、 戻された最初のレコードの相対キー・データ項目には値 '1' が含まれます。戻された 2 番目のレコードでは、値 '2' が含まれるというようになります。 SORT ステートメントの実行が終わると、相対キー・データ項目は、 ファイルに最後に戻されたレコードを示しています。
- 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。

これらの暗黙の関数は、関連付けられている USE AFTER EXCEPTION/ERROR プロシージャーを実行するように実行されます。 ただし、このような USE プロシージャーの実行によって、ファイル名-*3* が参照するファイルを操作するステートメント、 または関連付けられているレコード域にアクセスする操作を行うステートメントが実行されないように注意してください。 ファイルの外部定義境界を超えて書き込む最初の試みが行われると、 そのファイルに USE AFTER STANDARD EXCEPTION/ERROR プロシージャーが指定されていれば、 それが実行されます。 制御がその USE プロシージャーから戻されるか、 またはこのような USE プロシージャーが指定されていなければ、 ファイルの処理は終了します。

OUTPUT PROCEDURE 句

この句によって、ソート処理から出力レコードを選択したり修正したりするために使う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の中の最初の (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* によって参照されるファイルから RETURN ステートメントの実行によってソート順序に基づいて 1 つずつ使用可能に されるレコードを選択、修正、またはコピーするために必要なプロシージャーで構成することができます。 この範囲には、出力プロシージャーの範囲内で CALL、EXIT、GO TO、PERFORM、および XML PARSE ステートメントによって制御が移動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、この範囲には、出力プロシージャーの範囲内のステートメントが実行されると実行される宣言型プロシージャーの中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内で、MERGE ステートメント、RELEASE ステートメント、形式 1 の SORT ステートメントを実行することはできません。

出力プロシージャーが指定されていると、ファイル名-*1* によって参照されたファイルが SORT ステートメントによって順序付けされてから、制御はこの出力プロシージャーに渡されます。 コンパイラーは、出力プロシージャーの中の最後のステートメントの終わりに RETURN を挿入し、 制御が出力プロシージャーの中の最後のステートメントに移ると、RETURN によってソートが終了し、 制御が SORT ステートメントの後に置かれた次の実行可能ステートメントに移ります。 出力プロシージャーに入る前

に、ソート・プロシージャーは要求されたとき、 次のレコードをソートされた順に選択できる所まで来ています。 出力プロシージャーの中の RETURN ステートメントは、次のレコードを要求することになります。

INPUT PROCEDURE および OUTPUT PROCEDURE は基本的な PERFORM ステートメント用の句と類似しています。 例えば、出力プロシージャーにあるプロシージャーを指定すると、そのプロシージャーは、 それが PERFORM ステートメントの中で指定された場合とまったく同じように、 ソート操作時に実行されます。 PERFORM ステートメントによる場合と同様に、プロシージャーの実行は、 最後のステートメントがその実行を終えると終了します。 入力プロシージャーまたは出力プロシージャーの最後のステートメントとして、 EXIT ステートメントを使用することができます (306 ページの『EXIT ステートメント』を参照)。

SORT 特殊レジスター

特殊レジスターの SORT-CORE-SIZE、SORT-MESSAGE、および SORT-MODE-SIZE は、 ソート制御ファイルの中の制御ステートメントのオプションで使用されるキーワードと同じ働きをします。 ソート制御ファイルの定義は、SORT-CONTROL 特殊レジスターを使用して行います。

使用上の注意: ソート制御ファイルを使用して制御ステートメントを指定する場合は、ソート制御ファイルの中に指定されている値が特殊レジスターにある値に優先します。

SORT-MESSAGE 特殊レジスター
21 ページの『SORT-MESSAGE』を参照してください。 .

SORT-CORE-SIZE 特殊レジスター
21 ページの『SORT-CORE-SIZE』を参照してください。 .

SORT-FILE-SIZE 特殊レジスター
21 ページの『SORT-FILE-SIZE』を参照してください。 .

SORT-MODE-SIZE 特殊レジスター
22 ページの『SORT-MODE-SIZE』を参照してください。 .

SORT-CONTROL 特殊レジスター
21 ページの『SORT-CONTROL』を参照してください。 .

SORT-RETURN 特殊レジスター
22 ページの『SORT-RETURN』を参照してください。 .

セグメント化に関する考慮事項

このトピックでは、SORT ステートメントの使用に関する考慮事項について説明します。

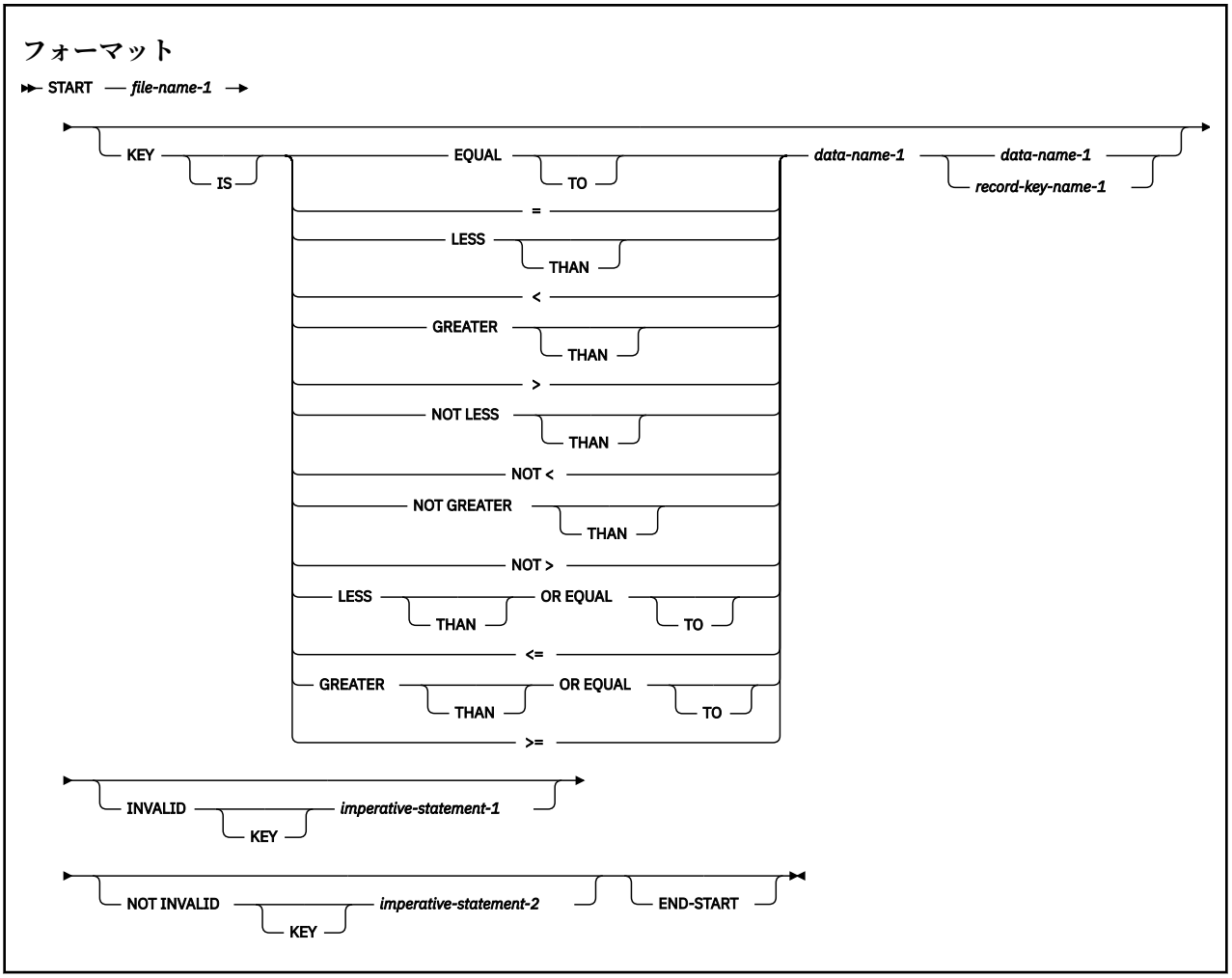
固定セグメントで SORT ステートメントがコード化された場合、この SORT ステートメントが参照するすべての入力または出力プロシージャーは完全に固定セグメントの範囲内にあるか、プロシージャー全体が単一の独立セグメントに含まれている必要があります。

独立セグメントで SORT ステートメントがコード化された場合、この SORT ステートメントが参照するすべての入出力プロシージャーは完全に固定セグメントの範囲内にあるか、プロシージャー全体が SORT ステートメントと同じ独立セグメントに含まれている必要があります。

START ステートメント

START ステートメントは、その後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での位置決めの手段を提供します。

START ステートメントを実行する場合には、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 モードのいずれかでオープンされている必要があります。



制約事項 : レコード・キー名 は、STL ファイル・システムでのみサポートされます。

ファイル名-1
順次アクセス・モードまたは動的アクセス・モードのファイルを指定しなければなりません。 ファイル名-*1* は、DATA DIVISION の FD 項目に定義されている必要があり、また、 ソート・ファイルにすることはできません。

KEY 句

レコード・キー名-*1* を、ファイル名-*1* のファイル制御項目内の RECORD KEY 節または ALTERNATE RECORD KEY 節の SOURCE 句で指定する必要があります。 データ名-*1* または レコード・キー名-*1* は修飾することができます。データ名-*1* には添え字を付けることができません。

KEY 句を指定すると、ファイル位置標識は、 ファイル内で比較の条件を満たすキー・フィールドを持つ論理レコードに位置付けられます。

KEY 句を指定しない場合は、 KEY IS EQUAL (基本レコード・キーに一致する) が暗黙に指定されます。

KEY をデータ項目「より小」または「より小か等しい」に指定した場合、ファイル位置標識は、ファイル内に現在存在する、比較の条件を満たす最後の論理レコードに位置付けられます。

索引付きファイルの場合、比較の条件を満たすキーが重複項目である場合、ファイル位置標識はそれらの最後の項目に位置付けられます。

データ名-1
修飾形でもよいが、添え字付きにはできません。

START ステートメントが実行されると、 キー・データ名の現行値とファイルの指標内の該当キー・フィールドが比較されます。

ファイル制御項目の中に FILE STATUS 節が指定されている場合は、 START ステートメントの実行時に、関連するファイル状況キーが更新されます (269 ページの『ファイル状況キー』を参照)。

INVALID KEY 句

ファイル内のどのレコードでも比較の条件が満たされない場合は、無効キー条件となります。 つまり、ファイル位置標識の位置は未定義となり 、(指定されている場合は) INVALID KEY 命令ステートメントが実行されます (『共通の処理機能』にある 274 ページの『INTO 句および FROM 句』を参照してください)。

INVALID KEY 句および該当する EXCEPTION/ERROR プロシージャーは、両方とも省略することができます。

END-START 句

この明示的範囲終了符号は、START ステートメントの範囲を区切るために使用されます。 END-START を使用すると、 条件付き START ステートメントを他の条件ステートメント内にネストすることができます。 END-START 句は、命令 START ステートメントと共に使用することもできます。

詳しくは、263 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 句を指定したときは、比較のために使用されるキー・データ項目はデータ名-*1* またはレコード・キー名-*1* です。

KEY 句が指定されていない場合、EQUAL TO 比較に使用されるキー・データ項目は基本レコード・キーです。

START ステートメントが正しく実行されると、データ名-*1* またはレコード・キー名-*1* が関連付けられている RECORD KEY または ALTERNATE RECORD KEY は、後続の READ ステートメントで使用される参照キーとなります。

データ名-*1* またはレコード・キー名-*1*
これは以下の項目のいずれかにすることができます。

- 基本 RECORD KEY。
- 任意の ALTERNATE RECORD KEY。
- 左端の文字位置がレコード・キーの左端の文字位置と対応しているファイルにおいて、そのレコード記述内のデータ項目。このデータ項目は修飾できます。 データ項目のサイズは、そのファイルのレコード・キーの長さ以下でなければなりません。

カテゴリーにかかわらず、データ名-*1* または レコード・キー名-*1* は、比較演算を行うために、英数字項目として扱われます。

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を指します。 比較の際にオペランドの長さが同じでない場合は、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行われます。 PROGRAM COLLATING SEQUENCE 節が指定されていても効力を持たないことを除いて、 他の数字および英数字比較の規則がすべて適用されます。

START ステートメントが正しく実行されると、データ名-*1* またはレコード・キー名-*1* が関連付けられている RECORD KEY は、 後続の READ ステートメントで使用される参照キーとなります。

START ステートメントの実行が正常に行われないと、参照キーは定義されません。

相対ファイル

KEY 句を指定する場合は、データ名-*1* に RELATIVE KEY を指定する必要があります。

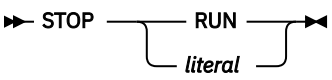
KEY 句の指定の有無にかかわらず、比較の際に使用されるキー・データ項目は、 RELATIVE KEY データ項目です。 数字比較の規則が適用されます。

ファイル位置標識は、指定された比較の条件を満たすキーを持つファイル内の論理レコードを指示します。

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永続的または一時的に停止します。

フォーマット



リテラル
固定小数点数値リテラル (符号あり、または符号なし)、英数字リテラル、またはブール・リテラルを指定できます。 ALL リテラル を除く任意の形象定数を指定できます。

STOP リテラル を指定すると、 そのリテラルをオペレーターに知らせてからオブジェクト・プログラムの実行は中断します。 プログラムの実行は、オペレーターの介入があった場合にのみ再開し、 次の実行可能ステートメントから順に実行が継続されます。

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なる特殊な状況で使用すると便利です。これには、特殊なテープやディスクをマウントする必要がある場合や、特定の日次コードを入力する必要がある場合などがあります。 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、実行が終了し、システムに制御が戻されます。 文の中の一連の命令ステートメントにおいて、STOP RUN が最後のステートメントではない場合、 または唯一のステートメントではない場合、STOP RUN の後にあるステートメントは実行されません。

STOP RUN ステートメントは、実行単位内のプログラムに定義されているすべての ファイルをクローズします。

呼び出すプログラムと呼び出されるプログラムの中で STOP RUN ステートメントを使用する場合は、 以下の表を参照してください。

終了 ステートメント	メインプログラム	サブプログラム
STOP RUN	呼び出し側プログラムへ戻る。 (それがシステムの場合は、アプリケーションを終了する。)	メインプログラムを呼び出したプログラムに直接。 (それがシステムの場合は、アプリケーションを終了する。)

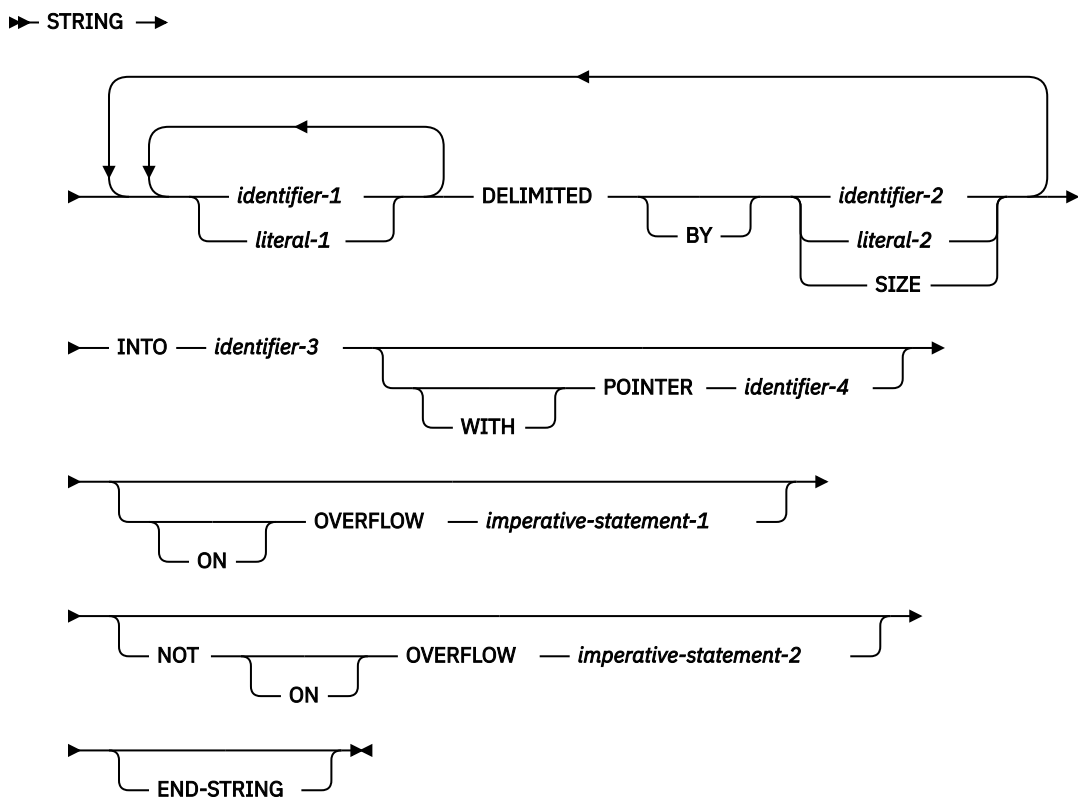
注 : CICS® では、EXEC CICS LINK ステートメントを介して開始されたプログラムから STOP RUN が呼び出された場合、STOP RUN は、GOBACK または EXEC CICS RETURN が発行されたかのように動作します。 STOP RUN は、リンクしているプログラムに制御を戻します。

STRING ステートメント

STRING ステートメント・ストリングは、2 つ以上のデータ項目やリテラルの内容の一部または全部を一緒にして 1 つのデータ項目にまとめます。

MOVE ステートメントをいくつも並べる代わりに 1 つの STRING ステートメントで済ませることができます。

フォーマット



ID-1*、リテラル-*1
これは送り出しフィールド を表します。

DELIMITED BY 句
ストリングの限界を設定します。

ID-2*、リテラル-*2
これらは区切り文字です。つまり、転送するデータを区切る文字です。

SIZE
この指定をすると送り出し領域全体を転送します。

INTO 句
受け取りフィールドを示します。

ID-3
これは受け取りフィールド を表します。

POINTER 句
受け取りフィールド内の文字位置を指します。 ポインター・フィールドは、受け取りフィールドが USAGE DISPLAY、DISPLAY-1、または NATIONAL の場合に、それぞれ相対的な英数字文字位置、DBCS 文字位置、国別文字位置を示します。

ID-4
ポインター・フィールド を表します。 *identifier-4* は、受信フィールドの長さに 1 を加えた値を入れられる十分な大きさにする必要があります。*identifier-4* は、STRING ステートメントの実行が開始される前に、ゼロ以外の値に初期設定する必要があります。

次の規則が適用されます。

- *ID-4* 以外のすべての ID は、USAGE DISPLAY、DISPLAY-1、または NATIONAL として、明示的または暗黙的に記述されているデータ項目を参照しなければなりません。

- リテラル-*1* またはリテラル-*2* は、英数字、DBCS、または国別カテゴリーでなければなりません。また、ALL というワードで始まらない任意の形象定数 (NULL を除く) にすることができます。
- *ID-1* または *ID-2* が数字カテゴリーのデータ項目を参照する場合、それぞれの数字項目は、PICTURE 文字ストリング内に記号「P」を指定しないで整数として記述する必要があります。
- *ID-3* は、数字編集、英数字編集、または国別編集カテゴリーの、USAGE DISPLAY の外部浮動小数点データ項目、または USAGE NATIONAL のデータ項目を参照してはなりません。
- *ID-3* は、JUSTIFIED 節を指定して記述してはなりません。
- *ID-3* が USAGE DISPLAY の場合、*ID-1* および *ID-2* は USAGE DISPLAY で、すべてのリテラルは英数字リテラルでなければなりません。 ALL というワードで始まる形象定数を除き、任意の形象定数を指定できます。 形象定数は、それぞれ 1 文字の英数字リテラルを表します。
- *ID-3* が USAGE DISPLAY-1 の場合、*ID-1* および *ID-2* は USAGE DISPLAY-1 で、すべてのリテラルは DBCS リテラルでなければなりません。 指定できる形象定数は SPACE のみです。これは、1 文字の DBCS リテラルを表します。 *DBCS*-リテラル はすべて指定することはできません。
- *ID-3* が USAGE NATIONAL の場合、*ID-1* および *ID-2* は USAGE NATIONAL でなければならず、すべてのリテラルは国別リテラルでなければなりません。 シンボリック文字 および ALL というワードで始まる形象定数を除き、任意の形象定数を指定できます。 形象定数は、それぞれ 1 文字の国別リテラルを表します。
- *ID-1* または *ID-2* が、数字、数字編集、または英数字編集カテゴリーとして記述されている USAGE DISPLAY の基本データ項目を参照する場合、この項目は英数字カテゴリーとして再定義されたかのように処理されます。
- *ID-1* または *ID-2* が、数字、数字編集、または国別編集カテゴリーの項目として記述されている、USAGE NATIONAL の基本データ項目を参照する場合、この項目は国別カテゴリーとして再定義されたかのように処理されます。
- *ID-4* は、PICTURE 文字ストリングの中に記号 P を指定して記述することはできません。
- STRING ステートメントでは、ID をウィンドウ表示日付フィールドにすることはできません。

添え字、参照変更、可変長、可変位置、および関数 ID の評価は、STRING ステートメントの実行開始時に 1 回のみ行われます。 したがって、*ID-3* または *ID-4* が、STRING ステートメントの中で添え字、参照修飾子、または関数引数として使用されているか、あるいは STRING ステートメントの中のいずれかの ID の長さまたは位置に影響する場合、これらの添え字、参照修飾子、可変長、可変位置、および関数について計算される値は、STRING ステートメントの結果によって影響されません。

ID-3 および *ID-4* が同じストレージ域を占めると、たとえそれらの ID が同じデータ記述項目によって定義されていても、 未定義の結果が生じます。

ID-1 または *ID-2* が、*ID-3* または *ID-4* と同じストレージ域を占めると、 たとえそれらの ID が同じデータ記述項目によって定義されていても、未定義の結果が生じます。

STRING ステートメントの処理について詳しくは、387 ページの『データ・フロー』 を参照してください。

ON OVERFLOW 句

命令ステートメント-*1*

ここにあるステートメントは、 ポインター値が明示的または暗黙のうちに次のような値になると実行されます。

- 1 より小さい値。
- 受け取りフィールドの長さを超える値。

上記の状態のいずれかが生じると、オーバーフロー条件が起こり、 データはそれ以上転送されません。ついで STRING 処理が終了し、NOT ON OVERFLOW 句が指定されている場合には、 それが無視され、制御は STRING ステートメントの終わりに移されるか、または ON OVERFLOW 句が指定されていれば、命令ステートメント-*1* に制御が移されます。

制御が命令ステートメント-*1* に移された場合、命令ステートメント-*1* に指定された各ステートメントの規則に従って、実行が継続されます。 プロシージャー・ブランチまたは明示的な制御の移動を引き

起こす条件ステートメントが実行される場合、 制御はそれを起こすステートメントの規則に従って移されます。 そうでない場合、命令ステートメント-*1* の実行完了時に、 制御は STRING ステートメントの最後に転送されます。

STRING ステートメントの実行時にオーバーフロー条件を生じる状態にならなければ、データ転送の完了後、ON OVERFLOW 句は指定されていても無視されます。そして、制御は STRING ステートメントの終わりか、 または NOT ON OVERFLOW 句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* に移された場合、命令ステートメント-*2* に指定された各ステートメントの規則に従って、実行が継続されます。 明示的な制御の移動を起こす、プロシージャーのブランチ・ステートメントや条件ステートメントが実行された場合は、制御はそのステートメントの規則に従って移されます。それ以外の場合は、命令ステートメント-*2* の実行が完了すると、 制御は STRING ステートメントの終わりに移されます。

END-STRING 句

この明示的範囲終了符号は、STRING ステートメントの範囲を区切るために使用されます。 END-STRING 句を使用することによって、 条件 STRING ステートメントを他の条件ステートメントの中にネストすることができます。 END-STRING 句は、命令 STRING ステートメントと共に使用することもできます。

詳しくは、263 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・フロー

STRING ステートメントの実行時に、文字は送り出しフィールドから受け取りフィールドに転送されます。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から受け取りフィールドに文字が転送される際は、以下の方法が使 用されます。
  - 国別送り出しフィールドの場合は、国別から国別への基本移動に関する MOVE ステートメントの規則を使用してデータが転送されます。 ただし、スペースの埋め込みは行われません。
  - DBCS 送り出しフィールドの場合は、DBCS から DBCS への基本移動に関する MOVE ステートメントの規則を使用して、データが転送されます。ただし、スペースの埋め込みは行われません。
  - それ以外の場合は、英数字間の基本移動に関する MOVE ステートメントの規則を使用して、データが受け取りフィールドに転送されます。ただし、スペースの埋め込みは行われません (327 ページの『MOVE ステートメント』を参照)。
- DELIMITED BY *ID-2* またはリテラル-*2* を指定した場合、各送り出し項目の内容は、左端の文字位置から始めて、1 文字ずつ次のいずれかの時点まで移動されます。
  - その送り出しフィールドの区切り文字に到達したとき (区切り文字自体は移動されない)。
  - その送り出しフィールドの右端の文字が転送されたとき。
- DELIMITED BY SIZE が指定されている場合、各送り出しフィールド全体が受け取りフィールドに移動されます。
- 受け取りフィールドがいっぱいになるか、 またはすべての送り出しフィールドが処理されるとデータ転送操作は終わります。
- POINTER 句を指定すると、COBOL ユーザーは明示的なポインター・フィールドを受け取りフィールドの中の データの配置を制御するために使用できるようになります。 ユーザーは明示的なポインターの初期値を設定しなければなりません。 この初期値は 1 未満であることも、受け取りフィールドの文字位置数を超えることもできません。

  使用上の注意: ポインター・フィールドは、受け取りフィールドの長さに 1 を加えた値を入れられる十分な大きさに定義する必要があります。これは、転送終了時にシステムがポインターを更新する際の算術オーバーフローを防止します。
- POINTER 句を指定しない場合、ユーザーはポインターを使用することはできません。 ただし、システムは、初期値として 1 を持つ概念的な暗黙のポインターを使用します。
- 概念的には、 STRING ステートメントが実行されるときのポインターの初期値 (明示的または暗黙の指定) は、データが移動される受け取りフィールドの最初の文字位置です。 その位置から開始して、データ

は 1 文字ずつ左から右へと位置付けられていきます。 各文字が位置付けされるごとに、明示的または暗黙のポインターが 1 だけ増えます。ポインター・フィールドの値は、この方法によってのみ変更されます。 処理が終了したときのポインター値は、 受け取りフィールドに移動された最後の文字より、常に 1 文字位置だけ先の値を示しています。

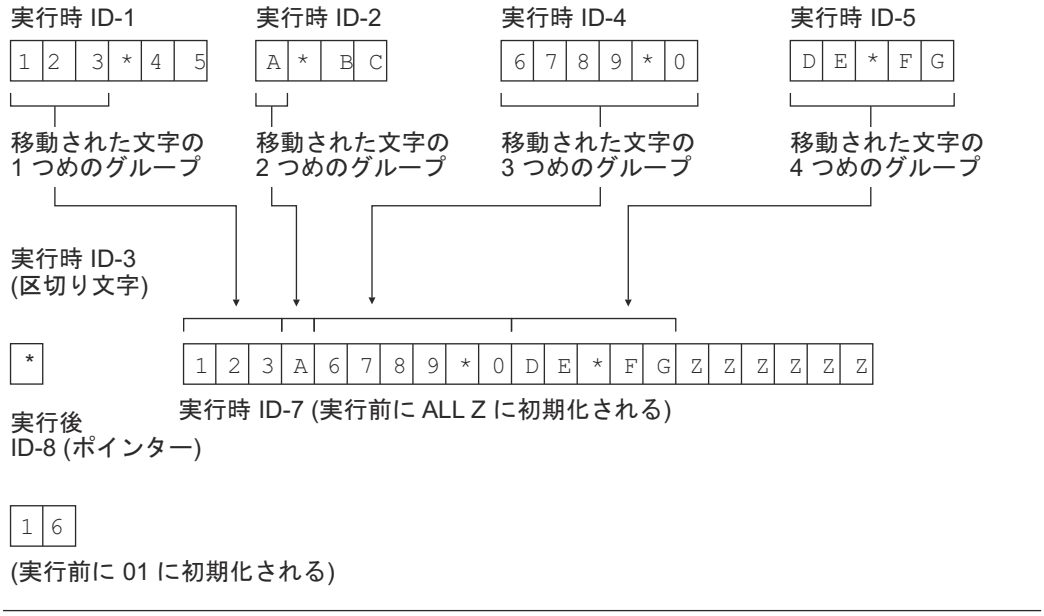
STRING ステートメントの実行が完了すると受け取りフィールドは、 データが移動された部分だけが変更されます。 受け取りフィールドの残りの部分には、 STRING ステートメントの今回の実行以前に存在していたデータが入っています。

STRING ステートメントの例

このトピックでは、STRING ステートメントの例を示します。

次に示す STRING ステートメントを実行すると得られる結果は、ステートメントの後の図に図解したようなものになります。

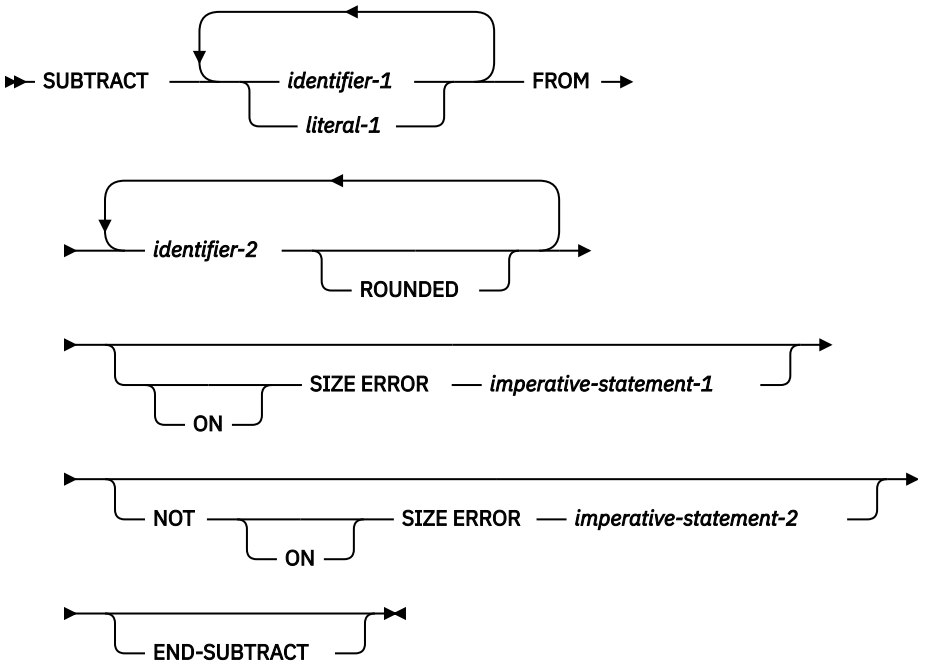
```
STRING ID-1 ID-2 DELIMITED BY ID-3
       ID-4 ID-5 DELIMITED BY SIZE
  INTO ID-7 WITH POINTER ID-8
END-STRING
```



SUBTRACT ステートメント

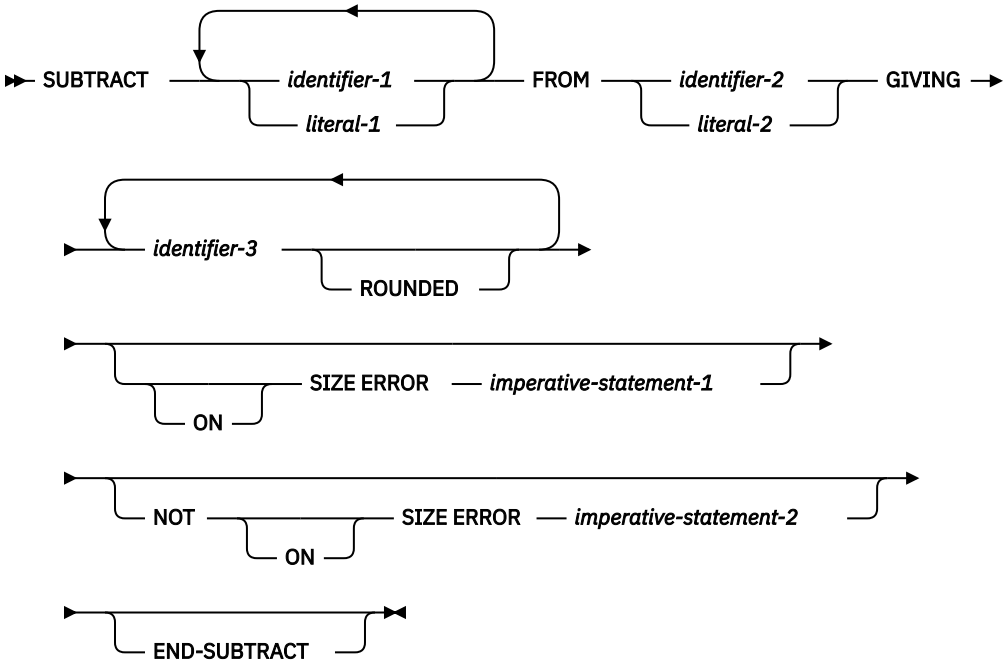
SUBTRACT ステートメントは、1 つまたは複数の数値項目から、 1 つの数値項目または 2 つ以上の数値項目の和を減算して、その結果を保管します。

フォーマット 1: SUBTRACT ステートメント



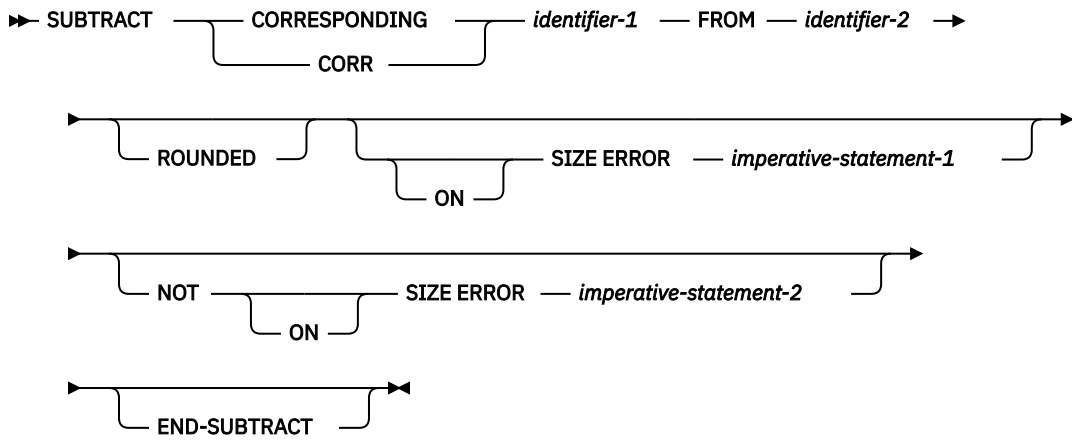
キーワード FROM の前にあるすべての ID またはリテラルは互いに加算され、それらの和が ID-2 から減算され、*ID-2* に直接保管されます。 この処理は、*ID-2* が連続する場合、それぞれの ID-2 ごとに、*ID-2* が指定されている順序で左から右へと繰り返されます。

フォーマット 2: GIVING 句を含む SUBTRACT ステートメント



キーワード FROM の前にあるすべての ID またはリテラルが加算され、 これらの和が *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* によって参照されるデータ項目それぞれの新しい値として保管されます。

フォーマット 3: CORRESPONDING 句を含む SUBTRACT ステートメント



ID-1 内の基本データ項目は *ID-2* の該当する基本データ項目から減算され、 その結果が、その ID-2 内の該当する基本データ項目に保管されます。

ARITH(COMPAT) コンパイラー・オプションが有効な場合は、オペランドの合成が最大 30 桁になります。 ARITH(EXTEND) コンパイラー・オプションが有効な場合は、オペランドの合成が最大 31 桁になります。 算術計算の中間結果について詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『付録 *A.* 中間結果および算術精度』を参照してください。

すべてのフォーマットに関して次のことが言えます。

ID

フォーマット 1 では、基本数字データ項目を指定しなければなりません。

フォーマット 2 では、ID がキーワード GIVING の後にある場合を除き、 基本数字データ項目の名前でなければなりません。 キーワード GIVING の後に置かれた ID はそれぞれ、数字基本項目または数字編集基本データ項目の名前でなければなりません。

フォーマット 3 では、英数字グループ項目または国別グループ項目を指定する必要があります。

以下の制約事項は、日付フィールドに適用されます。

- フォーマット 1 では、*ID-1* は 1 つの日付フィールドしか指定できません。 *ID-1* が日付フィールドを指定する場合は、*ID-2* のすべてのインスタンスでは、*ID-1* で指定された日付フィールドと互換性のある日付フィールドを指定しなければなりません。 *ID-1* が日付フィールドを指定しない場合は、*ID-2* で 1 つまたは複数の日付フィールドを指定することができ、DATE FORMAT 節への制約事項はありません。
- フォーマット 2 では、*ID-1* と *ID-2* がそれぞれ 1 つの日付フィールドでしか指定できません。 *ID-1* が日付フィールドを指定する場合は、FROM *ID-2* は、*ID-1* で指定された日付フィールドと互換性のある日付フィールドでなければなりません。 *ID-3* は 1 つ以上の日付フィールドを指定することができます。*ID-2* が日付フィールドを指定し、*ID-1* が日付フィールドを指定しない場合は、*ID-3* のすべてのインスタンスでは、*ID-2* で指定された日付フィールドと互換性のある日付フィールドを指定しなければなりません。
- フォーマット 3 では、*ID-1* 内の項目が日付フィールドである場合は、*ID-2* 内の該当する項目が互換日付フィールドでなければなりません。
- 年末尾型日付フィールドを SUBTRACT ステートメントに指定できるのは、*ID-1* としてのみ、および減算の結果が非日付データである場合だけです。

1 つ以上の日付フィールドに関連する SUBTRACT ステートメントの結果を判別するには、次の 2 つのステップがあります。

1. 減算: 237 ページの『日付フィールドが関係する減算』で記述されたとおり、減算の結果を判別します。
2. 保管: その結果が受け取りフィールドにどのように保管されるかを判別します。 (フォーマット 1 と 3 では、受け取りフィールドは *ID-2* です。フォーマット 3 では、 受け取りフィールドは GIVING *ID-3* です。) 詳しくは、237 ページの『日付フィールドに関連する算術演算結果の保管』を参照してください。

リテラル
これは、数字リテラルでなければなりません。

数字データ項目とリテラルを指定できる個所に、浮動小数点データ項目およびリテラルを使用することができます。

ROUNDED 句

ROUNDED 句に関する詳細、およびオペランドに関する考慮事項については、265 ページの『ROUNDED 句』を参照してください。

SIZE ERROR 句

SIZE ERROR 句に関する詳細、およびオペランドに関する考慮事項については、266 ページの『SIZE ERROR 句』を参照してください。

CORRESPONDING 句 (フォーマット 3)

264 ページの『CORRESPONDING 句』を参照してください。

END-SUBTRACT 句

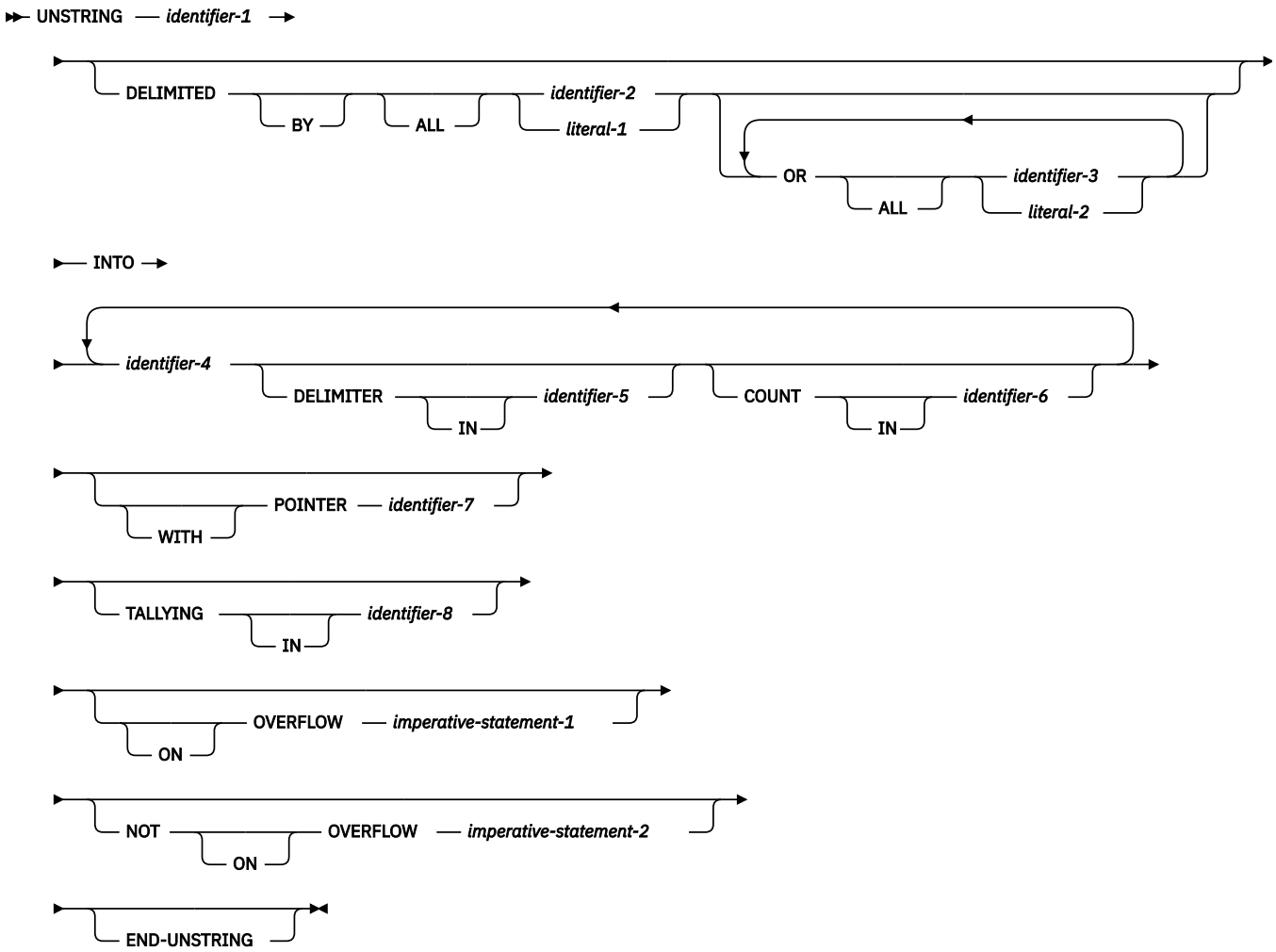
この明示的範囲終了符号は、SUBTRACT ステートメントの範囲を区切るために使用されます。 END-SUBTRACT 句を使用することによって、条件 SUBTRACT ステートメントを他の条件ステートメントの中にネストすることができます。 END-SUBTRACT 句は、命令 SUBTRACT ステートメントと共に使用することもできます。

詳しくは、263 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 ステートメントを使用することによって、送り出しフィールドの中の連続するデータを分割して、 複数の受け取りフィールドに入れることができます。

フォーマット



ID-1
これは送り出しフィールド を表します。 データは、このフィールドからデータ受け取りフィールド (*ID-4*) に転送されます。

ID-1 は、英字、英数字、英数字編集、DBCS、国別、または国別編集カテゴリーのデータ項目を参照しなければなりません。

ID-2*、リテラル-*1*、*ID-3*、 リテラル-*2
1 つ以上の区切り文字を指定します。

ID-2 および *ID-3* は、英字、英数字、英数字編集、DBCS、国別、または国別編集カテゴリーのデータ項目を参照しなければなりません。

リテラル-*1* またはリテラル-*2* は、英数字、DBCS、または国別カテゴリーでなければなりません。また、ALL というワードで始まる形象定数にすることはできません。

ID-4
1 つ以上の受け取りフィールドを指定します。

ID-4 は、英字、英数字、数字、DBCS、または国別カテゴリーのデータ項目を参照しなければなりません。 参照されるデータ項目が数字カテゴリーの場合は、その PICTURE 文字ストリングにはピクチャー記号 P を含めることはできません。また、USAGE DISPLAY または NATIONAL でなければなりません。

ID-5
ID-4 に関連付けられた区切り文字を受け取るフィールドを指定します。

ID-5 は、英字、英数字、DBCS、または国別カテゴリーのデータ項目を参照しなければなりません。

ID-6
ID-4 へ転送される文字数を入れるフィールドを指定します。

ID-6 は、PICTURE 文字ストリングの中で記号 P を使用しないで定義された整数データ項目でなければなりません。

ID-7
UNSTRING 処理中の相対的文字位置を入れるフィールドを指定します。

ID-7 ストリングの記号 P なしで定義された 整数データ項目である必要があります。

ID-7 は、*ID-1* によって参照されるデータ項目内の文字位置数に 1 を加えた値が十分に入るサイズの データ項目として記述する必要があります。

ID-8
処理される区切り文字で区切られているフィールドの数で増分されるフィールドを指定します。

ID-8 は、PICTURE 文字ストリングの中で記号 P を使用しないで定義された整数データ項目でなければなりません。

次の規則が適用されます。

- *ID-4* が USAGE DISPLAY のデータ項目を参照する場合、*ID-1*、*ID-2*、*ID-3*、および *ID-5* も USAGE DISPLAY のデータ項目を参照しなければなりません。また、すべてのリテラルは英数字リテラルでなければなりません。 ALL というワードで始まる形象定数および NULL を除き、任意の形象定数を指定できます。 形象定数は、それぞれ 1 文字の英数字リテラルを表します。
- *ID-4* が USAGE DISPLAY-1 のデータ項目を参照する場合、*ID-1*、*ID-2*、*ID-3*、および *ID-5* も USAGE DISPLAY-1 のデータ項目を参照しなければなりません。また、すべてのリテラルは DBCS リテラルでなければなりません。 形象定数 SPACE が、指定できる唯一の形象定数です。 形象定数は、それぞれ 1 文字の DBCS リテラルを表します。
- *ID-4* が USAGE NATIONAL のデータ項目を参照する場合、*ID-1*、*ID-2*、*ID-3*、および *ID-5* も USAGE NATIONAL のデータ項目を参照しなければなりません。また、すべてのリテラルは国別リテラルでなければなりません。 ALL というワードで始まる形象定数および NULL を除き、任意の形象定数を指定できます。 形象定数は、それぞれ 1 文字の国別リテラルを表します。
- UNSTRING ステートメントのどの ID もウィンドウ表示日付フィールドにはできません。

カウント・フィールド (*ID-6*) およびポインター・フィールド (*ID-7*) は、バイト数ではなく文字位置 (英数字、DBCS、または国別) の数によって増分されます。

UNSTRING ステートメントの実行開始時に、特定のエレメントの 評価または計算が 1 回だけ実行されることを除き、MOVE ステートメントの シリーズを 1 つの UNSTRING ステートメントで置き換えることができます。 詳しくは、397 ページの『UNSTRING ステートメントの実行終了時の値』を参照してください。

移動の規則は、*ID-1* のカテゴリーの基本送り出し項目に対する MOVE ステートメントの規則と 同じであり、該当する *ID-4* が受け取り項目となります (327 ページの『MOVE ステートメント』を参照)。 例えば、*ID-1* が DBCS 項目の場合は、DBCS 項目の移動規則が使用されます。

DELIMITED BY 句

この句は、データ転送を制御するデータ内の区切り文字を指定します。

ID-2、*ID-3*、リテラル-*1*、またはリテラル-*2* は、それぞれ 1 つの区切り文字を表します。

DELIMITED BY 句を指定していない 場合、DELIMITER IN 句および COUNT IN 句を指定してはなりません。

ALL
任意の区切り文字の複数の連続するオカレンスは、唯一のオカレンスのように扱われます。この 1 つのオカレンスは、区切り文字受け取りフィールド (*ID-5*) が指定されていれば、そこに移動されます。送り出しフィールド内の区切り文字は、*ID-1* と同じ USAGE およびカテゴリーの基本項目として扱われます。この区切り文字は、MOVE ステートメントの規則に従って、現在の区切り文字受け取りフィールドに移動されます。

DELIMITED BY ALL が指定されていない 場合には、いずれかの 区切り文字が 2 つ以上連続して出現すると、現在のデータ受け取りフィールド (*ID-4*) は、 データ受け取りフィールドの記述に従って、スペースまたは 0 で埋め込まれます。

2 文字以上の区切り文字

2 文字以上の区切り文字は、区切っている文字が以下の両方である場合にのみ、区切り文字として認識されます。

- 連続している
- 送り出しフィールドで指定したシーケンス内

2 つ以上の区切り文字

2 つ以上の区切り文字を OR 条件で指定すると、重なり合わないいずれかの区切り文字が現れるたびに、送り出しフィールドで区切り文字として、指定した順序で認識されます。

次に例を示します。

```
DELIMITED BY "AB" or "BC"
```

送り出しフィールドに AB または BC が現れると、区切り文字としてみなされます。 ABC が現れると、AB が現れたとみなされます。

INTO 句

この句は、データの移動先フィールドを指定します。

ID-4 はデータ受け取りフィールド を表します。

DELIMITER IN
この句は、区切り文字の移動先フィールドを指定します。

ID-5 は区切り文字受け取りフィールド を表します。

DELIMITED BY 句を指定していない 場合、DELIMITED IN 句を指定してはなりません。

COUNT IN
この句は、検査済み文字位置の数が入れられるフィールドを指定します。

ID-6 は、各データ転送のデータ個数フィールド です。 各フィールドには、この受け取りフィールドへの移動に関して、送り出しフィールド内の 検査済み文字 (区切り文字で終わるか、または送り出しフィールドの終わりで終わる) の 個数が入れられます。 区切り文字はこの個数には含まれません。

DELIMITED BY 句を指定していない 場合、COUNT IN 句を指定してはなりません。

POINTER 句

POINTER 句を指定した場合、ポインター・フィールド *ID-7* の値は、送り出しフィールドの文字位置が検査されるたびに 1 ずつ増分されます。 UNSTRING ステートメントの実行が完了すると、ポインター・フィールドには、 送り出しフィールド内で検査された文字位置数とその初期値を加えた値が入ります。

この句を指定する場合には、UNSTRING ステートメントの実行を開始する前に、ユーザーはポインター・フィールドを初期化する必要があります。

TALLYING IN 句

TALLYING 句が指定されるときに、領域カウント・フィールド *ID-8* には、(UNSTRING ステートメントの実行の終了時に) 受け取り領域が作用したデータ数を初期値に加えた値が入ります。

この句を指定する場合には、UNSTRING ステートメントの実行を開始する前に、ユーザーは領域カウント・フィールドを初期化する必要があります。

ON OVERFLOW 句

次のような場合に、オーバーフロー条件が存在します。

- ポインター値 (明示的または暗黙的に指定) が 1 より小さい場合。
- ポインター値 (明示的または暗黙的に指定) が、送り出しフィールドの長さに 等しい値を超える場合。
- すべてのデータ受け取りフィールドの処理を終えても、送り出しフィールドに 依然として未検査の文字位置がある場合。

オーバーフロー条件が生じる場合

オーバーフロー条件は、次のような操作の結果生じます。

1. データがそれ以上転送されない。
2. UNSTRING 操作が終了する。
3. NOT ON OVERFLOW 句が指定されている場合は、それが無視される。
4. 制御は UNSTRING ステートメントの最後に移されるか、ON OVERFLOW 句が指定されていれば、命令ステートメント-*1* へ移されます。

命令ステートメント-*1*
オーバーフロー条件を処理するステートメント (複数可)。

制御が命令ステートメント-*1* に移された場合、命令ステートメント-*1* に指定された各ステートメントの規則に従って、実行が継続されます。 明示的な制御の移動を起こす、プロシージャーのブランチ・ステートメントや条件ステートメントが実行された場合は、制御はそのステートメントの規則に従って移されます。それ以外の場合は、命令ステートメント-*1* の実行が完了すると、 制御は UNSTRING 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生じない場合

UNSTRING ステートメントの実行中に、オーバーフロー条件を引き起こす 条件が生じないときには、以下のようになります。

1. データの転送が完了する。
2. ON OVERFLOW 句が指定されていれば、無視される。
3. 制御は UNSTRING ステートメントの最後に移されるか、NOT ON OVERFLOW 句が 指定されていれば、命令ステートメント-*2* へ移されます。

命令ステートメント-*2*
生じないオーバーフロー条件を処理するステートメント (複数可)。

制御が命令ステートメント-*2* に移された場合、命令ステートメント-*2* に指定された各ステートメントの規則に従って、実行が継続されます。 明示的な制御の移動を起こす、プロシージャーのブランチ・ステートメントや条件ステートメントが実行された場合は、制御はそのステートメントの規則に従って移されます。 それ以外の場合は、命令ステートメント-*2* の実行が完了すると、制御は UNSTRING ステートメントの終わりに移されます。

END-UNSTRING 句

この明示的範囲終了符号は、UNSTRING ステートメントの範囲を区切るために使用されます。 END-UNSTRING 句を使用することによって、条件 UNSTRING ステートメントを他の条件ステートメント内にネ

ストすることができます。 END-UNSTRING 句は、命令 UNSTRING ステートメントと共に使用することもできます。

詳しくは、263 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・フロー

UNSTRING ステートメントのデータ・フローは、一定の規則に基づいています。

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在のデータ受け取りフィールドに移動されます。

ステージ 1: 検査

1. POINTER 句が指定されている場合には、送り出しフィールドはポインター・フィールドの 値によって指定された相対文字位置から検査が開始されます。

   POINTER 句が指定されていない 場合、送り出しフィールドの文字ストリングは 左端の文字位置から検査が開始されます。

2. DELIMITED BY 句が指定されている場合は、区切り文字が検出されるまで、 文字位置が 1 つずつ、左から右へ検査されます。 区切り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すると、送り出しフィールドの 最後の文字位置で検査が終了します。 受け取りフィールドがさらにある場合には、次のフィールドが選択され、それ以外の場合には、 オーバーフロー条件が起こります。

   DELIMITED BY 句が指定されていない 場合、検査される文字位置の数は、現在のデータ受け取りフィールドのサイズに等しくなります。以下の表の説明を参照してください。受け取りフィールドのカテゴリーの扱いによるサイズについては、320 ページの表 48 を参照してください。

表 *57.* ***DELIMITED BY*** **が指定されていない場合の検査される文字位置**	
受け取りフィールドが以下の場合	**検査される文字位置の数**
英数字または英字	現行受け取りフィールドの英数字文字位置数と等しい
DBCS	現行受け取りフィールドの DBCS 文字位置数と等しい
国別	現行受け取りフィールドの国別文字位置数と等しい
数字	現行受け取りフィールドの整数部の文字位置数と等しい
SIGN IS SEPARATE 節で説明されている	現行受け取りフィールドのサイズより 1 小さい
可変長データ項目として説明されている	UNSTRING 操作の開始時に現行受け取りフィールドのサイズで判別する

ステージ 2: 移動

3. 検査される文字位置 (区切り文字を除く) は、下記の表で示している場合を除き、送り出しフィールドと同じデータ・カテゴリーの基本データ項目として扱われます。

ID-1 **(送り出しフィールド) のカテゴリー**	**基本データ項目のカテゴリー**
英数字編集	英数字
国別編集	国別

基本データ項目は、送り出しフィールドおよび受け取りフィールドのカテゴリーに関する MOVE ステートメントの規則に従って (327 ページの『MOVE ステートメント』を参照)、現在のデータ受け取りフィールドに移動されます。

4. DELIMITER IN 句を指定したときは、送り出しフィールドの中の区切り文字は、 基本英数字項目として扱われ、MOVE ステートメントの規則に従って現在の区切り文字 受け取り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起きた場合は、現在の区切り文字受け取りフィールドにはスペースが入れられます。
5. COUNT IN 句を指定すると、検査された文字位置数に等しい値 (区切り文字を除く) が、 基本移動の規則に従って、データ・カウント・フィールドに移動されます。

   ステージ 3: 連続的な反復
6. DELIMITED BY 句を指定した場合、送り出しフィールドは、区切り文字の右側にある最初の文字位置からさらに検査されます。

   DELIMITED BY 句を指定しない 場合、送り出しフィールドは、検査された最後の文字位置の右側にある最初の文字位置からさらに検査されます。
7. 連続した各データ受け取りフィールドに対して、検査および移動の処理が以下の条件のいずれかが生じるまで繰り返されます。
   - 送り出しフィールドにあるすべての文字が転送される。
   - 満たされていないデータ受け取りフィールドがなくなる。

UNSTRING ステートメントの実行終了時の値

UNSTRING ステートメントの実行開始時に、特定の操作が 1 回のみ行われます。

操作は以下のとおりです。

- 添え字、参照変更、変数の長さ、変数の場所の計算
- 関数の計算

したがって、*ID-4*、*ID-5*、*ID-6*、*ID-7*、または *ID-8* が、 UNSTRING ステートメントの中で添え字、参照修飾子、または関数引数として使用される場合、 または、UNSTRING ステートメント内の ID のいずれかの長さまたは位置に 影響を与える場合、これらの値は UNSTRING ステートメントの実行開始時に決められるので、 UNSTRING ステートメントの実行結果によって影響されることはありません。

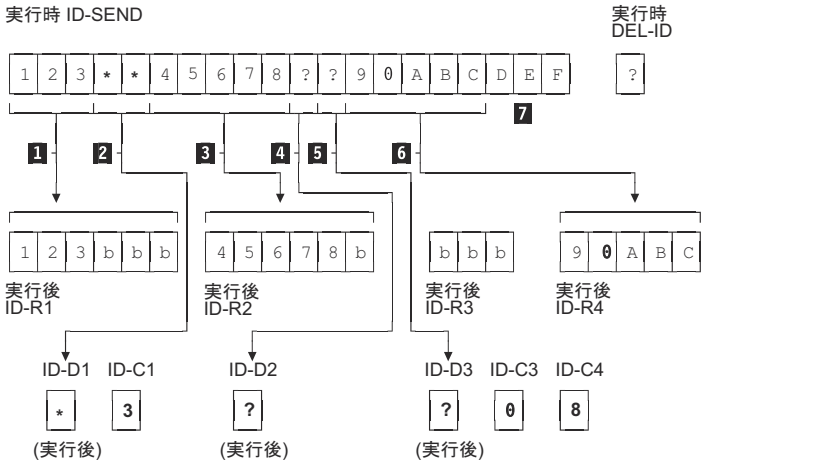
UNSTRING ステートメントの例

このトピックでは、UNSTRING ステートメントの例を示します。

下の図は、UNSTRING ステートメント例の実行結果を示しています。

```
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.
```

(受け取りフィールドのすべてのデータは英数字項目として定義されている)



ID-P (ポインター)

ID-T (累計フィールド)

2 1

0 5

(実行後--- いずれも実行前に 01 に初期化される)

実行の順序は以下のとおり。

1. 3 文字が ID-R1 に格納される。
2. ALL * が指定されているので、連続するすべてのアスタリスクが処理されるが、ID-D1 にはアスタリスクが 1 文字のみ格納される。
3. 5 文字が ID-R2 に格納される。
4. ? が 1 文字 ID-D2 に格納される。現行受け取りフィールドは ID-R3 になる。
5. ? が 1 文字 ID-D3 に格納される。ID-R3 はスペースで埋められる。文字はまったく移送されないので、ID-C3 には 0 が格納される。
6. 区切り文字が検出されないまま 5 文字が ID-R4 に格納される。ID-C4 には 8 が格納される。これは、最後に区切り文字が検出されてから検査した文字の数を表す。
7. ID-P は 21 (送り出しフィールドの全長 + 1) に更新される。ID-T は操作済みのフィールドの数 + 1 の 5 に更新される。ID-SEND には検査されていない文字はないので、OVERFLOW EXIT は取られない。

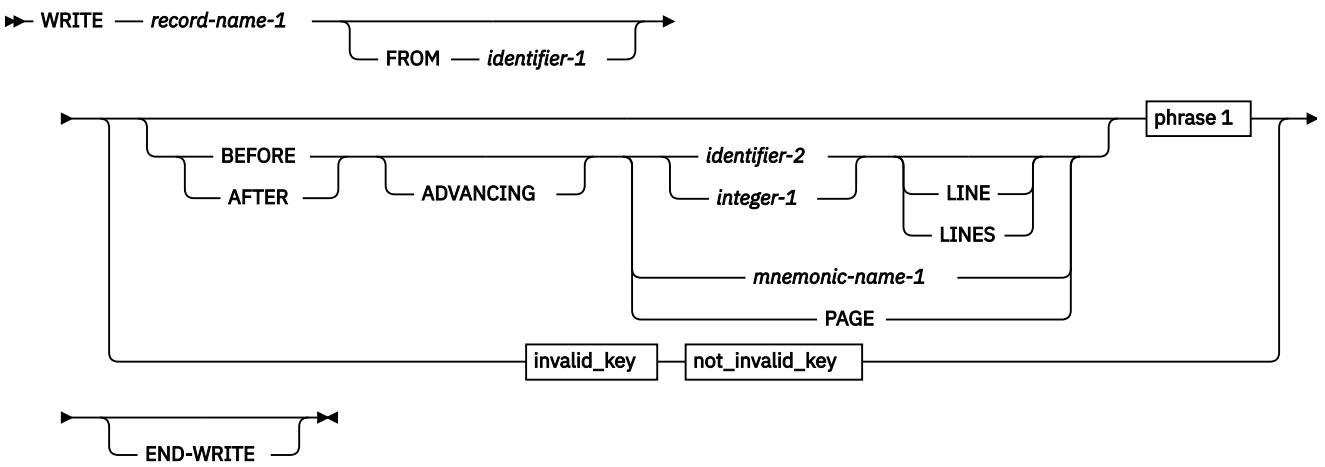
WRITE ステートメント

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に 1 つの論理レコードを解放します。

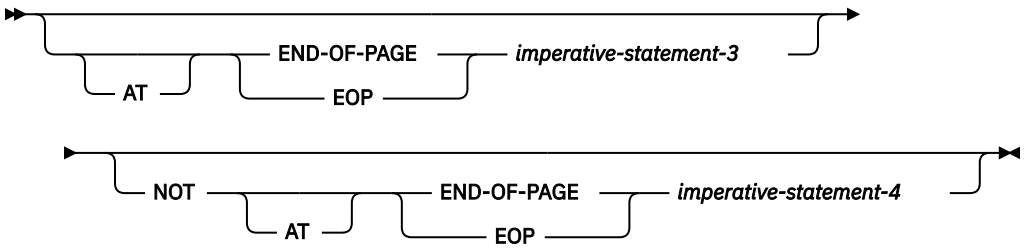
WRITE ステートメントが実行される時には、次のようになっている必要があります。

- 関連付けられている順次ファイルが OUTPUT または EXTEND モードでオープンしている必要があります。
- 関連付けられている指標または相対ファイルが OUTPUT、I-O、または EXTEND モードでオープンしている必要があります。

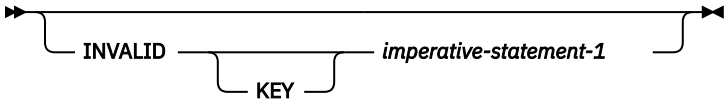
フォーマット 1: 順次ファイルの WRITE ステートメント



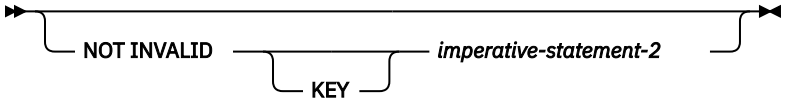
句 1



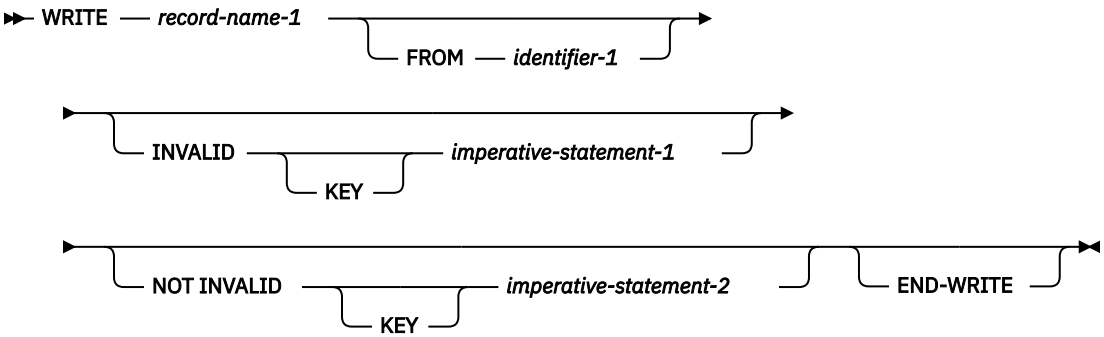
invalid_key



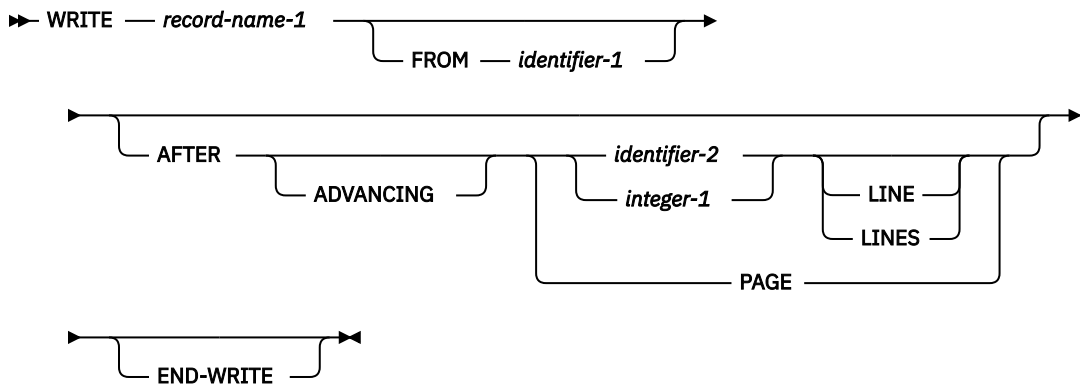
not_invalid_key



フォーマット 2: 索引付きおよび相対ファイルの WRITE ステートメント



フォーマット 3: 行順次ファイルの WRITE ステートメント



レコード名-*1*

DATA DIVISION の FD 項目に定義されている必要があります。 レコード名-*1* は修飾することができます。 ソート・ファイルやマージ・ファイルと関連付けることはできません。

相対ファイルの場合、作成されるレコード内の文字位置数と、 置換されるレコード内の文字位置数は、異なっていても構いません。

FROM 句

FROM *ID-1* 句を指定した WRITE ステートメントの実行結果は、次のステートメントを指定した順序で実行した場合と同じになります。

```
MOVE ID-1 TO レコード名-1.
WRITE record-name-1.
```

MOVE は、CORRESPONDING 句を指定していない MOVE ステートメントの規則に従って行われます。

ID-1

ID-1 は以下の項目のいずれかを参照できます。

- WORKING-STORAGE SECTION、LOCAL-STORAGE SECTION、または LINKAGE SECTION に定義されたデータ項目
- すでにオープンされた別のファイルのレコード記述
- 英数字関数
- 国別関数

ID-1 は、受け取り項目としてレコード名-*1* が指定された MOVE ステートメントに対して、有効な送り出し項目でなければなりません。

ID-1 およびレコード名-*1* は、同じストレージ域を参照することはできません。

WRITE ステートメントの実行後も、*ID-1* の中の情報は使用可能です。(『共通の処理機能』にある 274 ページの『INTO 句および FROM 句』を参照してください)。

ID-2

これは整数データ項目である必要があります。

ADVANCING 句

ADVANCING 句は、ページ上での出力レコードの位置付けを制御します。

WRITE ADVANCING を環境名 C01-C012 または S01-S05 と共に使用する場合、1 行進みます。

ADVANCING 句の規則

ADVANCING 句を指定する場合には、次の規則が適用されます。

1. BEFORE ADVANCING を指定すると、ページが進む前に行が印刷されます。
2. AFTER ADVANCING を指定すると、行が印刷される前にページが進みます。
3. *ID-2* を指定した場合、 そのページは *ID-2* の現行値に等しい行数だけ行送りされます。 *ID-2* には基本整数データ項目を指名する必要があります。 *ID-2* はウィンドウ表示日付フィールドを指名することはできません。
4. 整数を指定すると、ページは、その整数値に等しい行数だけ行送りされます。
5. 整数または *ID-2* の値は 0 とすることができます。
6. PAGE を指定する場合、 使用する句が BEFORE か AFTER かにより装置が次の論理ページに位置付けされる前に (BEFORE)、 または位置付けされた後で (AFTER)、レコードは論理ページ上に印刷されます。 PAGE が使用されている装置で意味を持たない場合、 BEFORE または AFTER (どちらの句が指定されているかに応じて) ADVANCING 1 LINE が想定されます。

   FD 項目に LINAGE 節が含まれている場合には、 その節で指定された次のページの最初の印刷可能行に位置変更されます。 LINAGE 節を省略すると、位置変更は後に続く次のページの第 1 行目になります。

ADVANCING 句を省略すると、AFTER ADVANCING 1 LINE を指定した場合と同様に自動改行が行われます。

LINAGE-COUNTER の規則

ファイルに対して LINAGE 節を指定すると、WRITE ステートメントの実行中、 次の規則に従って、関連する LINAGE-COUNTER 特殊レジスターが変更されます。

1. ADVANCING PAGE が指定されていると、LINAGE-COUNTER は 1 にリセットされます。
2. ADVANCING *ID-2* または整数を指定すると、LINAGE-COUNTER は *ID-2* または整数の値だけ増加されます。
3. ADVANCING 句を省略している場合、LINAGE-COUNTER は 1 だけ増やされます。
4. 装置が新しいページの最初の使用可能行に再配置されると、 LINAGE-COUNTER は 1 にリセットされます。

END-OF-PAGE 句

END-OF-PAGE 句が指定されている場合、WRITE ステートメントの実行時に印刷ページの論理的終わりに達すると、END-OF-PAGE 命令ステートメントが実行されます。 END-OF-PAGE 句を指定する場合、 このファイルの FD 項目には、LINAGE 節がなければなりません。

印刷ページの論理的終わりは、関連する LINAGE 節で指定します。

END-OF-PAGE 条件が起こるのは、 WRITE END-OF-PAGE ステートメントの実行によって、 ページ本体のフッター域内で印刷または行送りが行われるときです。 これが起こるのは、LINAGE-COUNTER 特殊レジスターの値が、 LINAGE 節の WITH FOOTING 句で指定された値に等しくなる、またはそれを超えてしまうような WRITE ステートメントが実行されたときです。 WRITE ステートメントが実行され、次いで END-OF-PAGE 命令ステートメントが実行されます。

ある WRITE ステートメント (END-OF-PAGE 句の指定の有無に関係なく) が現在のページ本体の中で最後まで実行できないとき、自動的なページ・オーバーフロー条件が起こります。 これは、WRITE ステートメントが実行されると、LINAGE-COUNTER の値が LINAGE 節で指定されたページ本体の行数を超えてしまうときに起こります。 この場合は、 装置が次の論理ページの最初の印刷可能な行 (LINAGE 節で指定する) に位置変更される前 (BEFORE)、 または位置変更された後で (AFTER)、 行が印刷されます (前になるか後になるかは BEFORE、AFTER のうちのどちらの句を指定するかによって異なります)。 END-OF-PAGE 句が指定されていれば、次に END-OF-PAGE 命令ステートメントが実行されます。

LINAGE 節の WITH FOOTING 句が指定されていない場合、 ページ終了条件を (ページ・オーバーフロー条件と異なるものとして) 検知することができないために、自動的なページ・オーバーフロー条件が起こります。

WITH FOOTING 句が指定されていても、ある WRITE ステートメントを実行すると、LINAGE-COUNTER が LINAGE 節で指定されたフッター域の値とページ本体の値を両方とも超えてしまう場合には、 ページ終了条件と自動的なページ・オーバーフロー条件が同時に起こります。

キーワード END-OF-PAGE と EOP は同じ意味です。

単一の WRITE ステートメントには、 ADVANCING PAGE 句と END-OF-PAGE 句を両方指定できます。

INVALID KEY 句

無効キー条件は、次の場合に起きます。

- 順次ファイルの場合、外部的に定義されたファイルの境界を超えて書き出そうとした場合。
- 索引付きファイルの場合:
  - 外部的に定義されたファイルの境界を超えて書き出そうとした場合。
  - ACCESS SEQUENTIAL が指定されており、ファイルが OUTPUT モードでオープンされ、 基本レコード・キーの値が前のレコードの基本レコード・キー値よりも大きくない場合。
  - ファイルが OUTPUT モードまたは I-O モードでオープンされ、 基本レコード・キーの値がすでに存在するレコードの基本レコードの値に等しい場合。
- 相対ファイルの場合:
  - 外部的に定義されたファイルの境界を超えて書き出そうとした場合。
  - アクセス・モードがランダムまたは動的である場合に、 RELATIVE KEY データ項目がファイル内の既存レコードを指定している場合。
  - 相対レコード番号の有効数字の数が、 ファイルの相対キー・データ項目のサイズより大きい場合。

無効キー条件が起きると、以下のことが発生します。

- INVALID KEY 句が指定されている場合は、命令ステートメント-*1* が実行されます。 無効キーの処理について詳しくは、『無効キー条件』を参照してください。
- INVALID KEY 句が指定されていない場合、WRITE ステートメントは失敗し、レコード名 の内容は影響を受けません。さらに、以下のことが発生します。
  - 順次ファイルの場合、ファイル状況キーが指定してあれば、更新されて EXCEPTION/ERROR 条件が存在します。

    明示的または暗黙の EXCEPTION/ERROR プロシージャーがファイルに指定されている場合、そのプロシージャーが実行されます。 そのようなプロシージャーが指定されていない場合は、結果は予測できません。
  - 相対ファイルおよび索引付きファイルの場合、プログラム実行は、『共通の処理機能』の『無効キー条件』で説明されている規則に従って進められます。

    OPEN OUTPUT モードの相対ファイルに適用される INVALID KEY 条件は、OPEN EXTEND モードの相対ファイルにも適用されます。
- NOT INVALID KEY 句が指定され、WRITE ステートメントの実行の終わりに有効なキー条件が起きたときには、*ID-4* に制御が移されます。

INVALID KEY 句および該当する EXCEPTION/ERROR プロシージャーは、両方とも省略することができます。

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、条件的な WRITE ステートメントを他の条件ステートメントの中にネストすることができます。 END-WRITE 句は、命令の WRITE ステートメントと共に使用することもできます。

詳しくは、263 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。.

順次ファイル用 WRITE

順次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。

WRITE ステートメントの実行後、以下の場合を除き、論理レコードはレコード名-*1* 内では使用できなくなります。

- 関連するファイルが、SAME RECORD AREA 節内に指定している場合 (この場合、 レコードは SAME RECORD AREA 節で指名された他のファイルのレコードとしても使用可能です)。
- WRITE ステートメントの実行が、境界違反を理由に失敗した場合。

これらの場合には、レコード名-*1* の中の論理レコードは使用可能です。

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を受けません。

ファイル内にレコードを保管するために必要な文字位置の数は、COBOL プログラムの中でレコードの論理記述によって定義された文字位置の数と同じであっても、 同じでなくても構いません (193 ページの『PICTURE 節の編集』および 213 ページの『USAGE 節』を参照してください。)

ファイル制御項目で FILE STATUS 節が指定されている場合は、WRITE ステートメントが実行されると、正常に実行されたかどうかにかかわりなく、関連するファイル状況キーが更新されます。

WRITE ステートメントは、OUTPUT モードでオープンされた順次ファイルに対してのみ実行できます。

索引付きファイル用 WRITE

索引付きファイルに対して WRITE ステートメントを実行する場合、その前に基本レコード・キー (ファイル制御項目で定義した RECORD KEY データ項目) を必要な値に設定しておく必要があります。 RECORD KEY 値は、ファイル内で固有でなければならないことに注意してください。

ファイル制御項目内に ALTERNATE RECORD KEY 節も指定されている場合は、DUPLICATES 句が指定されていない限り、代替レコード・キーはそれぞれ固有でなければなりません。 DUPLICATES 句が指定されている場合、代替レコード・キー値は固有である必要はありません。 この場合、システムは、後でレコードを順次にアクセスする際に、保管時と同じ順序で取り出すことができるようにレコードを保管します。

ファイル制御項目で ACCESS IS SEQUENTIAL が指定されている場合は、RECORD KEY 値の昇順にレコードを解放しなければなりません。

ファイル制御項目で ACCESS IS RANDOM または ACCESS IS DYNAMIC が指定されている場合は、プログラマーが指定した任意の順序でレコードを解放できます。

相対ファイル用 WRITE

相対レコード OUTPUT ファイルの場合、トピックで説明されているように、WRITE ステートメントによって以下の処置が行われます。

- 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。

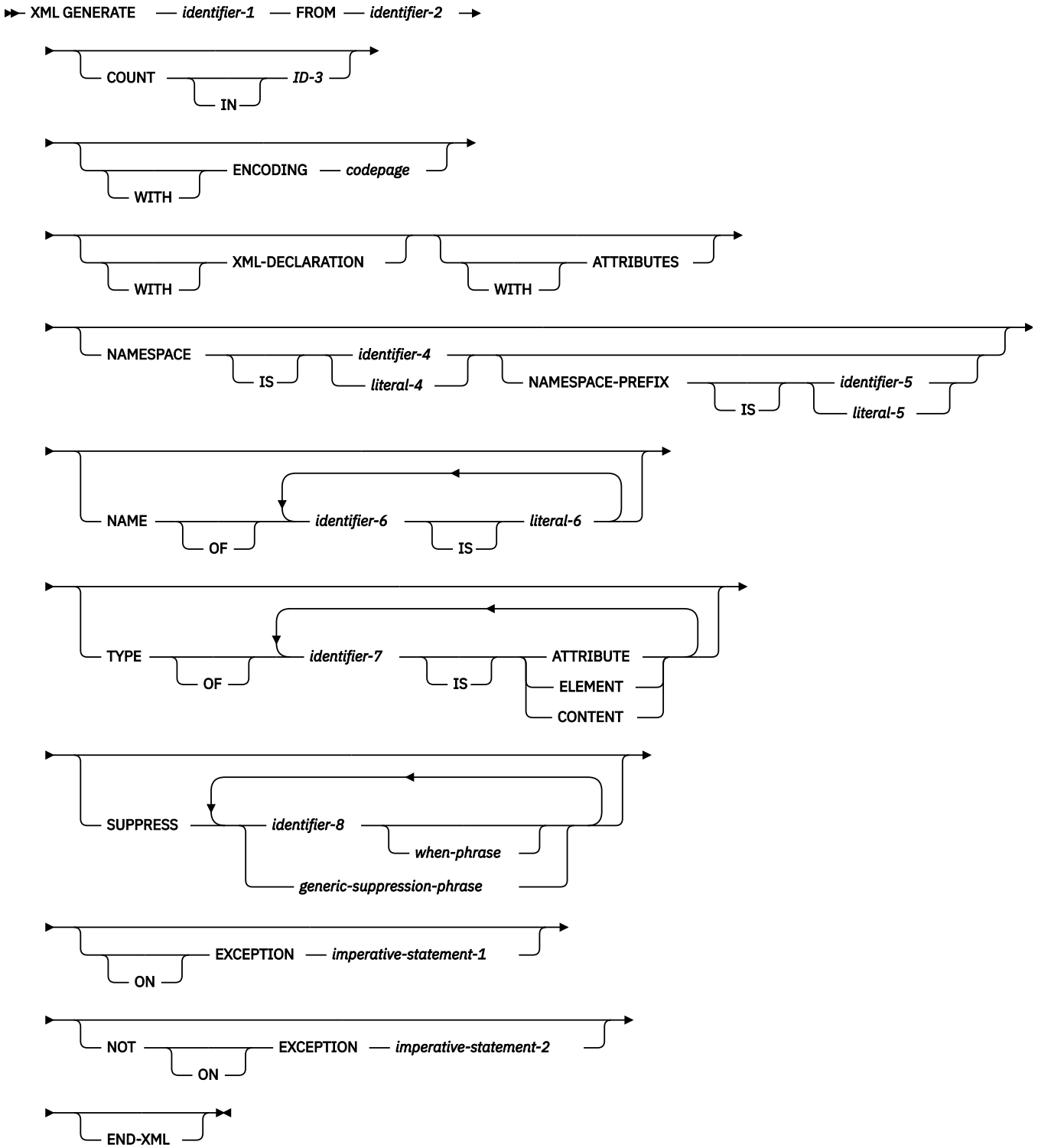
  ファイル制御項目の中で RELATIVE KEY が指定されていれば、 WRITE ステートメントの実行時に、書き込まれたばかりのレコードの相対レコード番号が、 RELATIVE KEY の中に入れられます。
- ACCESS IS RANDOM または ACCESS IS DYNAMIC を指定した場合は、WRITE ステートメントを実行する前に、このレコードの必要な相対レコード番号を RELATIVE KEY に入れておかなければなりません。 WRITE ステートメントが実行されると、このレコードはファイル内の指定された相対レコード番号の位置に入れられます。

I-O ファイルの場合、 ACCESS IS RANDOM または ACCESS IS DYNAMIC のいずれかを指定する必要があります。 WRITE ステートメントは新規レコードをファイルに挿入します。 WRITE ステートメントを実行する前に、このレコードの必要な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。 WRITE ステートメントが実行されると、このレコードは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。

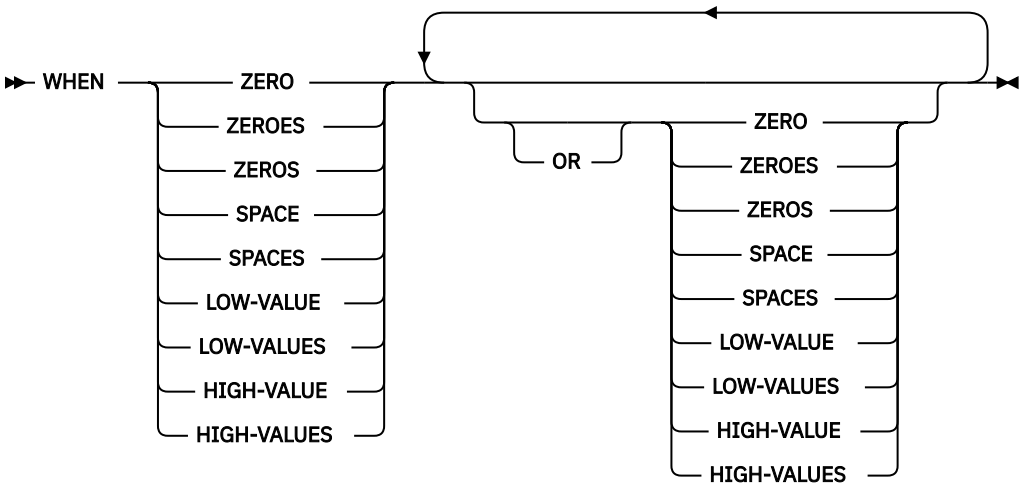
XML GENERATE ステートメント

XML GENERATE ステートメントはデータを XML 形式に変換します。

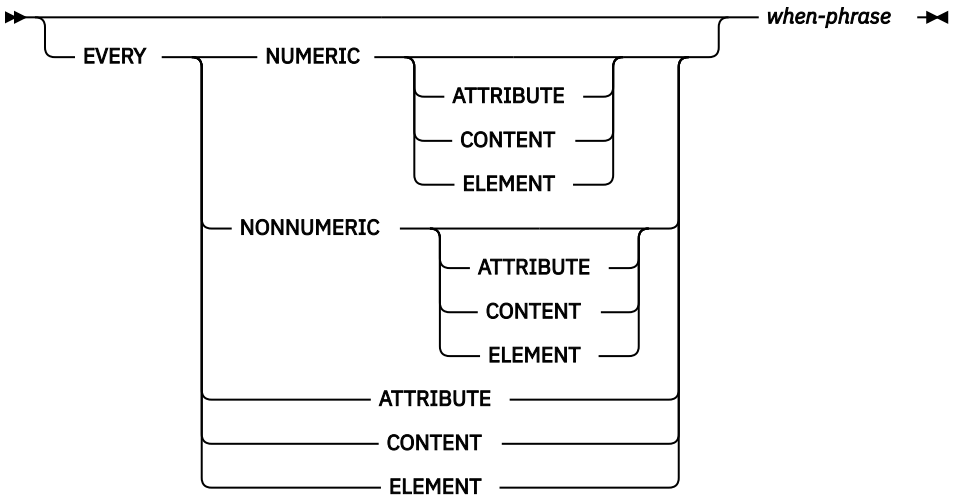
フォーマット



when-phrase 形式



汎用抑止句形式



ID-1

生成された XML 文書の受け取り領域。*ID-1* は、以下の項目のいずれかを参照する必要があります。

- 英数字カテゴリーの基本データ項目
- 英数字グループ項目
- 国別カテゴリーの基本データ項目
- 国別グループ項目

ID-1 が国別グループ項目を参照する場合は、*ID-1* は 国別カテゴリーの基本データ項目として処理されます。 *ID-1* が英数字グループ項目を参照する場合は、*ID-1* は英数字カテゴリーの基本データ項目とし処理されます。

ID-1 は JUSTIFIED 節を使用して記述することはできません。また、関数 ID にすることはできません。*ID-1* は、添え字または参照変更にすることができます。

ID-1 は *ID-2*、*ID-3*、 *codepage* (ID の場合)、 *ID-4*、または *ID-5* とオーバーラップすることはできません。

生成された XML 出力は、ENCODING 句の説明に従ってエンコードされます。

ID-1 は、国別カテゴリーのデータ項目を参照する必要があります。 あるいは、*ID-1* が英数字カテゴリーのときに次の条件が該当する場合、 文書エンコードは Unicode UTF-8 でなければなりません。

- *ID-4* または *ID-5* が国別カテゴリーのデータ項目を参照する。
- *literal-4*、*literal-5*、または *literal-6* が国別カテゴリーである。
- *codepage* は国別リテラルであるか、または国別カテゴリーのデータ項目を参照する。
- 生成された XML に次のものを指定する *ID-2* からのデータが含まれる。
  - 国別クラスまたは DBCS クラスの任意のデータ項目
  - マルチバイト名を持つ任意のデータ項目 (つまり、名前が マルチバイト文字からなるデータ項目)
  - マルチバイト文字を含む英数字クラスの任意のデータ項目

ID-1 には生成された XML 文書を入れるだけの大きさが必要です。通常は、*ID-2* のサイズの 5 倍から 10 倍の大きさでなければなりません (*ID-2* 内の 1 つ以上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は、XML GENERATE ステートメントの終わりにエラー条件が存在します。

ID-2

XML 形式に変換されるグループ・データ項目または基本データ項目。

ID-2 が国別グループ項目を参照する場合は、ID-2 はグループ項目として処理されます。ID-2 が従属国別グループ項目を含んでいるときには、その従属項目はグループ項目として処理されます。

ID-2 は関数 ID にすることや参照修飾することはできませんが、添え字を付けることはできます。

ID-2 は、*ID-1* または *ID-3* とオーバーラップすることはできません。

ID-2 のデータ記述項目に RENAMES 節が含まれていてはなりません。

ID-2 によって指定された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。

- 任意の従属名前なし基本データ項目または基本 FILLER データ項目
- SYNCHRONIZED 項目に挿入された任意の遊びバイト
- REDEFINES 節を指定して記述されているか、またはそのような再定義項目に従属する *ID-2* に従属する任意のデータ項目
- RENAMES 節を指定して記述された *ID-2* に従属する任意のデータ項目
- すべての従属データ項目が無視される任意のグループ・データ項目

前の規則に従って無視されない、*ID-2* によって指定された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。

- それぞれの基本データ項目は、英字、英数字、数字、または国別クラスを持つか、または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、USAGE FUNCTION-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。)
- 上記のような基本データ項目が最低 1 つ必要です。
- FILLER 以外のそれぞれのデータ名は、直接上位にあるグループ・データ項目内で固有である必要があります。
- マルチバイト データ名は、Unicode に変換する場合、XML specification バージョン 1.0 で規定された正しい名前でなければなりません。XML の指定の詳細については、*XML* の指定 を参照してください。
- データ項目は DATE FORMAT 節を指定することはできません。また、DATEPROC コンパイラー・オプションが有効であってはなりません。

例えば、次のようなデータ宣言があると考えてください。

```
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-5 PIC S9(9).
```

```
    03 ASFNPTR REDEFINES COUNTER USAGE FUNCTION-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-5 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).
```

前の例の以下のデータ項目は *ID-2* として指定できます。

- STRUCT。その従属データ項目 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。その従属データ項目 FLAGS、COUNTER、および UNREFERENCED は変換されます。(データ記述項目で 03 PIC X(3) を指定する項目は、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。)
- STRUCT に従属する任意の基本データ項目。ただし、以下を除きます。
  - ASFNPTR または PTR (使用禁止)
  - UNREFERENCED OF NG-AREA2 (非固有なデータ項目名。固有であれば有効)
  - 任意の FILLER データ項目

以下のデータ項目は *ID-2* として指定することはできません。

- NG-AREA1。これは、従属データ項目 PTR が USAGE POINTER を指定するが、REDEFINES 節を指定しないためです。(PTR で REDEFINES 節を指定した場合は無視されます。)
- NG-AREA2。これは、従属基本データ項目に非固有名 UNREFERENCED が指定されているためです。

COUNT IN 句

COUNT IN 句を指定すると、*ID-3* には (XML GENERATE ステートメントの実行後に) 生成された XML 文字エンコード・ユニット数が含まれます。 *ID-1* (受け取り側) が国別カテゴリーの場合、個数は UTF-16 文字エンコード・ユニット数になります。 UTF-8 を含めそれ以外のエンコード方式の場合はすべて、個数はバイト単位です。

ID-3

データ個数フィールド。 PICTURE ストリングの中で記号 P を使用しないで定義された整数データ項目でなければなりません。

ID-3 は *ID-1*、*ID-2*、 *codepage* (ID の場合)、 *ID-4*、または *ID-5* とオーバーラップすることはできません。

ENCODING 句

指定した ENCODING 句は、生成された XML 文書のエンコード方式を決定します。

codepage

符号なし整数データ項目、符号なし整数リテラル、英数字リテラル、国別リテラルのいずれかになっているか、または英数字カテゴリーや国別カテゴリーのデータ項目を参照していなければなりません。 *codepage* は、リテラルの場合は形象定数であってはなりません。

codepage は、英数字クラスまたは国別クラスの 場合は、ICU 変換ライブラリーでサポートされている基本コード・ページ名またはコード・ページ別名を 識別しなければなりません (「*International Components for Unicode: Converter Explorer*」 を参照)。 *codepage* が整数の場合、その整数は有効な CCSID 番号でなければなりません。

ID-1 が国別カテゴリーのデータ項目を参照する場合、*codepage* はリトル・エンディアン・フォーマットの UTF-16 を識別する必要があります。

ID-1 が英数字カテゴリーのデータ項目を参照する場合、*codepage* は次のように UTF-8、1 バイト ASCII、または 1 バイト EBCDIC コード・ページを識別する必要があります。

- CHAR(EBCDIC) コンパイラー・オプションが無効になっているか、または *ID-1* のデータ記述項目に NATIVE 句が含まれている場合、*codepage* は UTF-8 または 1 バイト ASCII コード・ページを識別する必要があります。
- CHAR(EBCDIC) が有効になっていて、*ID-1* のデータ記述項目に NATIVE 句が含まれていない場合、*codepage* は 1 バイト EBCDIC コード・ページを識別する必要があります。

codepage は、ID の場合は *ID-1* や *ID-3* とオーバーラップすることはできません。

ENCODING 句が省略されて、*ID-1* が国別カテゴリーである場合、 文書エンコードは Unicode UTF-16 (リトル・エンディアン・フォーマット) です。

ENCODING 句が省略されて、*ID-1* が英数字カテゴリーの場合、次のようになります。

- CHAR(EBCDIC) が無効になっているか、または *ID-1* の データ記述項目に NATIVE 句が含まれている場合、XML 文書は、ランタイム・ロケールのコード・ページを使用してエンコードされます。
- CHAR(EBCDIC) オプションが有効になっていて、*ID-1* の データ記述項目に NATIVE 句が含まれていない場合、XML 文書は、EBCDIC_CODEPAGE 環境変数のコード・ページを使用してエンコードされます。 EBCDIC_CODEPAGE が設定されていない場合、エンコードは、ランタイム・ロケールに関連したデフォルト EBCDIC コード・ページです。

XML-DECLARATION 句

XML-DECLARATION 句を指定した場合、生成された XML 文書は、XML バージョン情報およびエンコード宣言を含む XML 宣言で始まります。

ID-1 が国別カテゴリーである場合、エンコード宣言の値は UTF-16 (encoding="UTF-16") となります。

ID-1 が英数字カテゴリーの場合、エンコード宣言は、ENCODING 句から得られる (ENCODING 句が指定されている場合) か、または ランタイム・ロケールか EBCDIC_CODEPAGE 環境変数から得られます (ENCODING 句が指定されていない場合)。 詳細については、ENCODING 句の説明を参照してください。

XML-DECLARATION 句をコーディングした場合の効果の例については、「*COBOL for Linux on x86* プログラミング・ガイド」内の『*XML* 出力の生成』を参照してください。

XML-DECLARATION 句を省略した場合、生成された XML 文書には XML 宣言が含まれません。

ATTRIBUTES 句

ATTRIBUTES 句を指定した場合、生成された XML 文書に含まれる適格な各項目は、XML エレメントの子エレメントとしてではなく、適格なその項目の直接上位にあるデータ項目に対応する XML エレメントの属性として表現されます。 適格となるには、データ項目は、基本項目であり、FILLER 以外の名前を持つ必要がありますが、そのデータ記述項目に OCCURS 節が指定されていてはなりません。

TYPE 句が特定の ID に指定されている場合、その TYPE 句は、それらの ID において WITH ATTRIBUTES 句よりも優先されます。

ATTRIBUTES 句の効果の例については、*COBOL for Linux on x86* プログラミング・ガイド内の *XML* 出力の生成を参照してください。

NAMESPACE および NAMESPACE-PREFIX 句

NAMESPACE 句を使用すると、生成された XML 文書のネーム・スペースを識別できます。 NAMESPACE 句を指定しなかった場合、または *ID-4* が長さゼロであるか、全桁スペースである場合、XML GENERATE ステートメントによって作成された XML 文書のエレメント名はどのネーム・スペースにもありません。

NAMESPACE-PREFIX 句を使用すると、生成された XML 文書で各エレメントの開始タグと終了タグを接頭部で限定できます。

NAMESPACE-PREFIX 句を指定しなかった場合、または *ID-5* が長さゼロであるか、全桁スペースを含む場合、NAMESPACE 句によって指定されたネーム・スペースは、文書にデフォルトのネーム・スペースを指定します。 この場合、ルート・エレメントで宣言されたネーム・スペースが、そのルート・エレメントも含め、文書内の各エレメント名にデフォルトで適用されます。 (デフォルトのネーム・スペース宣言は、直接には属性名に適用されません。)

NAMESPACE-PREFIX 句を指定し、*ID-5* が長さゼロでなく、全桁スペースを含まない場合、生成された文書で各エレメントの開始タグと終了タグが指定された接頭部で限定されます。 したがって、この接頭部はできれば短いものにしてください。 XML GENERATE ステートメントを実行するときには、接頭

部は有効な XML 名でなければなりませんが、コロン (:) は使用しないでください。これについては、「*Namespaces in XML 1.0*」で定義されています。 接頭部の末尾にスペースを含めることができますが、使用前に除去されます。

***ID-4*、リテラル-4; *ID-5*、リテラル-5**

ID-4、リテラル-*4*: ネーム・スペース ID。有効な *URI* でなければなりません。これについては、「*Uniform Resource Identifier (URI): Generic Syntax*」で定義されています。

ID-5、リテラル-*5*: ネーム・スペース接頭部。ネーム・スペース ID の別名として働きます。

ID-4 および *ID-5* は、英数字または国別のカテゴリーのデータ項目を参照する必要があります。

ID-4 および *ID-5* は、*ID-1* または *ID-3* とオーバーラップすることはできません。

リテラル-*4* および リテラル-*5* は、英数字または国別のカテゴリーでなければなりませんが、形象定数とすることはできません。

ネームスペースの詳細については、「*Namespaces in XML 1.0*」を参照してください。

NAMESPACE および NAMESPACE-PREFIX 句の使用例については、*COBOL for Linux on x86* プログラミング・ガイド内の *XML* 出力の生成を参照してください。

NAME 句

エレメントおよび属性名を指定できます。

ID-6 は、*ID-2* またはその従属データ項目の 1 つを参照する必要があります。 この ID は関数 ID にはできず、参照変更または添え字付けを行うことはできません。 XML GENERATE ステートメントで無視されるデータ項目を指定してはいけません。 *ID-2* の詳細情報については、*ID-2* の説明を参照してください。 *ID-6* を NAME 句で複数回指定すると、最後の指定が使用されます。

リテラル-6 は、*ID-6* に対応する XML 文書で生成される属性またはエレメント名を含む英数字または国別リテラルでなければなりません。 また、有効な XML ローカル名でなければなりません。 リテラル-*6* が国別リテラルである場合は、*ID-1* で国別カテゴリーのデータ項目を参照するか、ENCODING 句で UTF-16 を指定する必要があります。

TYPE 句

属性およびエレメント生成を制御できるようにします。

ID-7 は、*ID-2* に従属する基本データ項目を参照する必要があります。 この ID は関数 ID にはできず、参照変更または添え字付けを行うことはできません。 XML GENERATE ステートメントで無視されるデータ項目を指定してはいけません。 *ID-2* の詳細情報については、*ID-2* の説明を参照してください。 *ID-7* を TYPE 句で複数回指定すると、最後の指定が使用されます。

- XML GENERATE ステートメントに WITH ATTRIBUTES 句も組み込まれている場合、*ID-7* において TYPE 句が優先されます。
- ATTRIBUTE を指定する場合、*ID-7* は XML 属性として適格でなければなりません。 *ID-7* は、生成される XML で、子エレメントではなく、*ID-7* のすぐ上位にある XML エレメントの属性として表されます。
- ELEMENT を指定すると、*ID-7* は、生成される XML でエレメントとして表されます。 XML エレメント名は *ID-7* から派生し、エレメント文字内容は、411 ページの『XML GENERATE の操作』で説明されているように、*ID-7* の変換された内容から派生します。
- CONTENT を指定すると、*ID-7* は、生成される XML で、*ID-7* のすぐ上位にあるデータ項目に対応する XML エレメントのエレメント文字内容として表されます。 411 ページの『XML GENERATE の操作』で説明されているように、エレメント文字内容の値は、*ID-7* の変換された内容から派生します。同じ上位 ID にすべてが対応する、複数の ID に CONTENT を指定すると、エレメント文字内容への複数のコントリビューションが連結されます。

SUPPRESS 句

ID-2 に従属する項目を特定して無条件に抑止し、XML GENERATE ステートメントの出力を選択的に生成することを可能にします。 SUPPRESS 句を指定する場合、*ID-1* は、抑止前に生成された XML 文書を入れるために十分な大きさでなければなりません。

generic-suppression-phrase を使用すると、通常は XML GENERATE 操作では無視されない、*identifier-2* に従属する基本項目は、一般的に抑止の可能性があると見なされます。 数字クラスの項目 (NUMERIC キーワードが指定されている場合)、数字クラスでない項目 (NONNUMERIC キーワードが指定されてい

る場合)、またはその両方が抑止される可能性があります。 ATTRIBUTE キーワードを指定すると、生成される XML 文書で XML 属性として表される項目のみが、抑止の可能性がある項目として特定されます。 ELEMENT キーワードを指定すると、生成される XML 文書で XML エレメントとして表わされる項目のみが、抑止の可能性がある項目として特定されます。 CONTENT キーワードを指定すると、生成される XML 文書で、CONTENT データ項目の上位にあるデータ項目に対応する XML エレメントのエレメント文字内容として表わされる項目のみが、抑止の可能性がある項目として特定されます。

複数の *generic-suppression-phrase* を指定した場合、効果は累積されます。

ID-8 は、抑止の可能性がある項目を明示的に特定します。 WHEN 句を指定する場合、*ID-8* は、*ID-2* に従属している、ほかの場合には XML GENERATE 操作によって無視されることのない、基本データ項目を参照していなければなりません。 *ID-8* は関数 ID にはできず、参照変更も添え字付けも行えません。 WHEN 句を省略すると、*ID-8* は、基本データ項目だけではなく、グループ・データ項目も参照できます。 当該グループ・データ項目、およびそのグループ項目に従属するすべてのデータ項目が抑止されます。 *ID-8* を SUPPRESS 句で複数回指定すると、最後の指定が使用されます。 *ID-8* が一般的に識別される ID の 1 つでもある場合、*ID-8* の明示的な抑止指定は、*generic-suppression-phrase* で暗黙指定された抑止指定をオーバーライドします。

ID-8 を指定する場合は、以下の規則が適用されます。

- WHEN 句で ZERO、ZEROES、または ZEROS を指定する場合、*ID-8* は USAGE DISPLAY-1 であってはなりません。
- WHEN 句で SPACE または SPACES を指定する場合、*ID-8* は、USAGE DISPLAY、DISPLAY-1、または NATIONAL でなければなりません。 *ID-8* は、ゾーン 10 進数項目または国別 10 進数項目の場合は整数でなければなりません。
- WHEN 句で LOW-VALUE、LOW-VALUES、HIGH-VALUE、または HIGH-VALUES を指定する場合、*ID-8* は USAGE DISPLAY または NATIONAL でなければなりません。 *ID-8* は、ゾーン 10 進数項目または国別 10 進数項目の場合は整数でなければなりません。

generic-suppression-phrase を指定する場合、以下の規則に従って、抑止の可能性があるデータ項目が選択されます。

- WHEN 句で ZERO、ZEROES、または ZEROS を指定する場合は、USAGE DISPLAY-1 で定義されたデータ項目を除くすべてのデータ項目が選択されます。
- WHEN 句で SPACE または SPACES を指定する場合は、USAGE DISPLAY、DISPLAY-1、または NATIONAL のデータ項目が選択されます。 ゾーン 10 進数項目または国別 10 進数項目の場合は、整数のみが選択されます。
- WHEN 句で LOW-VALUE、LOW-VALUES、HIGH-VALUE、または HIGH-VALUES を指定する場合は、USAGE DISPLAY または NATIONAL のデータ項目が選択されます。 ゾーン 10 進数項目または国別 10 進数項目の場合は、整数のみが選択されます。

項目を抑止するかどうかを決定する比較演算は、「形象定数を含む比較」の表に示されている比較条件です。 すなわち、指定した値が ZERO、ZEROS、または ZEROES で、項目が数字クラスである場合、比較は数値比較です。 その他の場合はすべて、比較演算は、項目の用途が DISPLAY、DISPLAY-1、または NATIONAL のいずれであるかに応じて、それぞれ英数字、DBCS、または国別の比較です。

SUPPRESS 句が指定されると、*identifier-2* に従属するグループ項目は、そのグループ項目に従属する適格な項目がすべて抑止される場合、または何らかの従属項目が抑止された後でそのグループ項目に対応する XML が属性なしの空のエレメントになる場合に、生成された XML 文書において抑止されます。 *ID-2* に従属するすべての項目が抑止される場合でも、ルート・エレメントは常に生成されます。

ON EXCEPTION 句

XML 文書の生成中にエラーが発生した場合、例えば、*ID-1* に生成された XML 文書を含めるだけの大きさがない場合は、例外条件が存在します。 この場合、XML 生成は停止し、受け取り側である *ID-1* の内容は未定義となります。 COUNT IN 句を指定すると、*ID-3* には生成された文字位置の数 (0 から *ID-1* の長さまで) が含まれます。

ON EXCEPTION 句が指定されている場合は、制御は 命令ステートメント*-1* に移ります。 ON EXCEPTION 句が指定されていない場合は、NOT ON EXCEPTION 句が指定されていても無視されます。この場合、制御は、XML GENERATE ステートメントの終わりに移ります。 特殊レジスター XML-CODE には例外コードが含まれます。詳細については、「*COBOL for Linux on x86* プログラミング・ガイド」内の『*XML GENERATE* 例外の処理』を参照してください。

NOT ON EXCEPTION 句
XML 文書の生成中に例外条件が発生しない場合、制御は命令ステートメント-*2* (指定されている場合) に渡されます。指定されていない場合は、XML GENERATE ステートメントの終わりに渡されます。 ON EXCEPTION 句は、指定されていても無視されます。 XML GENERATE ステートメントを実行すると、特殊レジスター XML-CODE にゼロが格納されます。

END-XML 句
この明示範囲終了符号は、XML GENERATE ステートメントまたは XML PARSE ステートメントの有効範囲を設定します。 END-XML は条件 XML GENERATE ステートメントまたは XML PARSE ステートメント (つまり、ON EXCEPTION 句または NOT ON EXCEPTION 句を指定する XML GENERATE ステートメントまたは XML PARSE 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を許可します。

条件 XML GENERATE または XML PARSE ステートメントの有効範囲は、次のいずれかによって終了します。

- ネスト構造で同じレベルにある END-XML 句。
- 分離文字ピリオド。

END-XML は、ON EXCEPTION 句または NOT ON EXCEPTION 句を指定しない XML GENERATE ステートメントまたは XML PARSE ステートメントと共に使用することもできます。

明示的範囲終了符号の詳細については、263 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。

ネストされた XML GENERATE ステートメントまたは XML PARSE ステートメント

XML GENERATE ステートメントまたは 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* として出現する場合、 あるいは別の XML GENERATE ステートメントまたは 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* の一部として出現する場合、その XML GENERATE ステートメントまたは XML PARSE ステートメントはネストされた XML GENERATE ステートメントまたは XML PARSE ステートメントになります。

ネストされた XML GENERATE ステートメントまたは XML PARSE ステートメントは、一致した XML GENERATE と END-XML の組み合わせ、または XML PARSE と END-XML の組み合わせとみなされ、左から右に処理されます。 したがって、検出される END-XML 句はすべて、暗黙的または明示的に終了されていない、 先行の最も近い場所にある XML GENERATE ステートメントまたは XML PARSE ステートメントと一致します。

XML GENERATE の操作

SUPPRESS 句に従って XML 生成が抑止されていない、*ID-2* 内の適格な各基本データ項目の内容は、文字フォーマットに変換されます。

それぞれの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。 また、RENAMES 節によって有効に定義されたデータ項目も含まれません。

基本データのフォーマット変換については、412 ページの『基本データのフォーマット変換』および 413 ページの『生成された XML データのトリミング』を参照してください。

TYPE OF 句が指定されると、変換された内容はその句の仕様に従って、エレメントの文字内容または属性値として処理されます。 TYPE OF 句が指定されないと、変換された内容はデフォルトで、生成された XML 文書にエレメントの文字内容として、あるいは WITH ATTRIBUTES 句が指定され、データ項目が属性として表現されるのに適格な場合は、属性の値として挿入されることになります。

XML エレメント名および属性名は、NAME 句 (指定されている場合) から取得されます。 そうではない場合はデフォルトで、413 ページの『XML エレメント名および属性名の形成』で説明されているように *ID-2* 内のデータ名から派生します。 選択済み基本項目を含むグループ項目の名前は、親エレメントとして保持されます。NAMESPACE-PREFIX 句を指定した場合、末尾スペースを取り去った接頭部値は、各エレメントの開始タグと終了タグを限定するのに使用されます。

生成された XML をより読みやすくするために追加の空白文字 (改行、字下げなど) が挿入されることはありません。 XML 宣言は、XML-DECLARATION 句を指定した場合に生成されます。

ID-1 によって指定された受け取り領域の長さが生成された XML 文書を格納するのに十分でない場合は、エラー条件が存在します。 詳細については、上記の ON EXCEPTION 句の説明を参照してください。

ID-1 の長さが生成された XML 文書よりも長い場合は、*ID-1* 内の XML が生成された部分のみが変更されます。 *ID-1* の残りの部分には、XML GENERATE ステートメントの今回の実行以前に存在していたデータが入っています。 そのデータを参照しないようにするには、 *ID-1* を初期化して XML GENERATE ステートメントの前にスペースを入れるか、または COUNT IN 句を指定します。

COUNT IN 句を指定すると、*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 (UTF-16 エンコード・ユニットまたはバイト) が含まれます。 *ID-3* を参照変更の長さフィールドとして使用して、生成された XML 文書を含む *ID-1* の一部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 には正常に完了したことを示すゼロ、またはゼロ以外の例外コードが含まれます。 詳細については、「*COBOL for Linux on x86* プログラミング・ガイド」内の『*XML GENERATE* 例外の処理』を参照してください。

また、XML PARSE ステートメントも特殊レジスター XML-CODE を使用します。 したがって、XML PARSE ステートメントの処理プロシージャーで XML GENERATE ステートメントをコーディングするときは、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ておいた値を復元してください。

Unicode エンコードの XML 文書の場合、バイト・オーダー・マークは生成されません。

基本データのフォーマット変換

ID-2 内の基本データ項目は、以下のように、一連のステップに従って変換されます。 一部のステップはオプションです。

文字フォーマットへの変換

基本データ項目は、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。

- カテゴリーが英字、英数字、英数字編集、DBCS、外部浮動小数点、国別、国別編集、および数字編集のデータ項目は変換されません。
- COMPUTATIONAL-5 (COMP-5) 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、または TRUNC(BIN) コンパイラー・オプションを使用してコンパイルされたバイナリー・データ項目は、以下を持つ数字編集項目に移動されたものとして変換されます。
  - 数値項目と同じだけの整数桁。最低でも 1 つの整数桁
  - 数値項目に最低 1 つの小数点位がある場合は、明示小数点
  - 数値項目と同じ小数点位数
  - データ項目が符号付き (PICTURE 節に S がある場合) の場合は、先行する '-' ピクチャー記号
- COMPUTATIONAL-5 (COMP-5) バイナリー・データ項目、または TRUNC(BIN) コンパイラー・オプションを使用してコンパイルされたバイナリー・データ項目は、整数桁数以外は他の固定小数点数値項目と同様に変換されます。 整数桁の数は、ピクチャー文字ストリング内の '9' 記号の数に応じて以下のように計算されます。
  - ピクチャー記号 '9' がデータ項目に 1 個から 4 個ある場合は、5 から小数点以下の桁数を減算して得られた値
  - ピクチャー記号 '9' がデータ項目に 5 個から 9 個ある場合は、10 から小数点以下の桁数を減算して得られた値
  - ピクチャー記号 '9' がデータ項目に 10 個から 18 個ある場合は、20 から小数点以下の桁数を減算して得られた値
- 内部浮動小数点データ項目は、以下のようにデータ項目に移動されたものとして変換されます。
  - 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 (桁位置の数が原因で不正となります)
- 指標データ項目は、USAGE COMP-5 PICTURE S9(9) と宣言されたものとして変換されます。

トリミング

文字フォーマットへの変換後は、先頭や末尾のスペースおよび先行ゼロは除去されます。詳細については、413 ページの『生成された XML データのトリミング』で解説しています。

文書エンコードへの変換

すべての値は必要に応じて、以下のように文書のエンコードに変換されます。 *ID-1* の内容によって、次のようになります。

- 英数字カテゴリーである場合。NATIVE 句がデータ記述に指定されているか、CHAR(NATIVE) が有効であれば、 値は UTF-8 または選択された ASCII コード・ページに変換されます。
- 英数字カテゴリーである場合。NATIVE 句がデータ記述に指定されていないときに CHAR(EBCDIC) が有効であれば、 値は選択された EBCDIC コード・ページに変換されます。
- 国別カテゴリーである場合。非国別値が国別フォーマットに変換されます。

XML 参照への特殊文字の変換

5 つの文字 & (アンパーサンド)、' (アポストロフィ)、> (より大符号)、< (より小符号)、および " (引用符) の残りのインスタンスは、同等の XML 参照である '&'、'''、'>'、'<'、および '"' にそれぞれ変換されます。

範囲外の Unicode 文字の置換

x'FFFF' より大きい Unicode スカラー値を持つ残りの Unicode 文字は、XML 文字参照で置換されます。 例えば、UTF-16LE の文書に Unicode スカラー値 x'10813' の文字が含まれる場合、その値はサロゲート・ペア (NX'02D8', NX'13DC') で表されます。これは参照 '𐠓' で置換されます。文書エンコードが UTF-8 の場合、文字参照 '𐠓' と同等のバイト・シーケンスは X'F090A093' です。

生成された XML データのトリミング

トリミングは、データ値を文字フォーマットに変換した後にそのデータ値に対して実行されます。

変換について詳しくは、412 ページの『基本データのフォーマット変換』を参照してください。

符号付き数値から変換された値の場合、値が正であれば先行スペースが除去されます。

数値項目から変換された値の場合、実際または暗黙の小数点の直前の桁まで (直前の桁は含まない) の先行ゼロ (冒頭の負符号の後) が除去されます。 小数点の後ろの後続ゼロは保持されます。 次に例を示します。

- -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、英数字、DBCS、および国別クラスのデータ項目の文字値は、対応するデータ項目に左方 (デフォルト) または右方への位置調整があるかどうかに応じて、それぞれ後続スペースまたは先行スペースが除去されます。 つまり、値に対応するデータ項目で JUSTIFIED 節が指定されていない場合、値から後続スペースが除去されます。 値に対応するデータ項目で JUSTIFIED 節が指定されている場合、値から先行スペースが除去されます。 文字値がスペースのみで構成される場合は、トリミングの完了後に 1 つのスペースが値として残ります。

XML エレメント名および属性名の形成

ID-2 から生成された XML 文書において、XML エレメント名および属性名は、 NAME 句が指定されている場合、名前はその句から取得されます。 この句が指定されていない場合は、*ID-2* で指定されたデータ項目の名前から、および *ID-2* に従属する適格なデータ名から得られます。

XML エレメント名および属性名の形成は以下のとおりです。

- データ記述項目で指定されたデータ名の英大/小文字混合の正確なスペルは維持されます。 データ項目への任意の参照からのスペル (例えば、OCCURS DEPENDING ON 節で指定されているもの) は使用されません。

- 数字で始まるデータ名には接頭部として下線が付けられます。 例えば、データ名「3D」は XML タグまたは属性名「_3D」になります。
- 大文字と小文字を任意に組み合わせた文字 'xml' で始まるデータ名には、接頭部として下線が付けられます。 例えば、データ名「Xml」は XML タグまたは属性名「_Xml」になります。

マルチバイト データ名は、Unicode に変換する場合、XML specification のバージョン 1.0 で規定された正しい名前でなければなりません。 XML の指定の詳細については、*XML の指定*を参照してください。

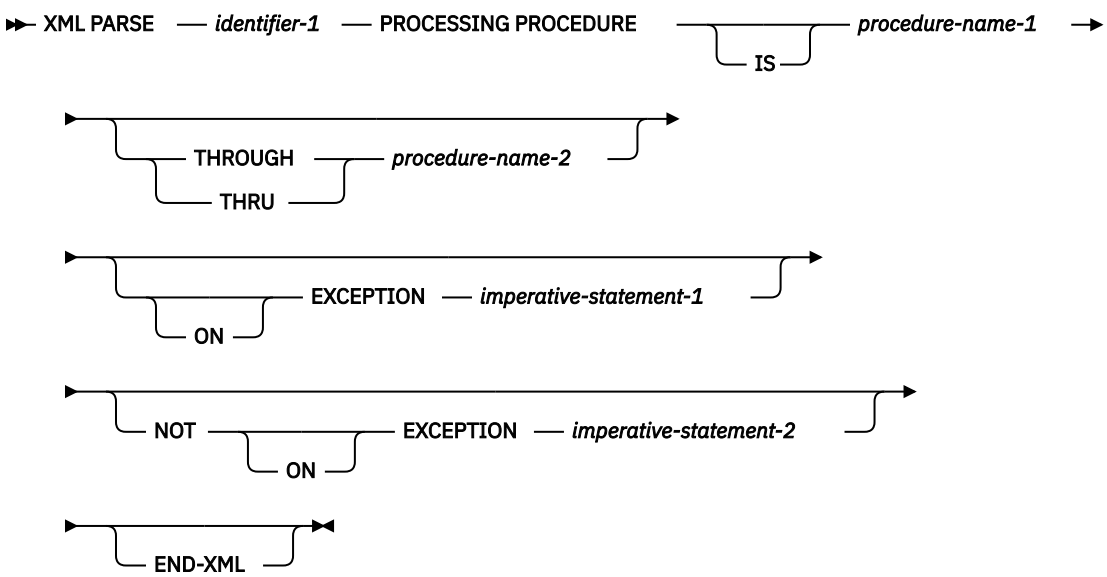
XML PARSE ステートメント

XML PARSE ステートメントは、COBOL ランタイムの一部となっている高速 XML パーサーへの COBOL 言語インターフェースです。

XML PARSE ステートメントは、XML 文書を解析し、それを個々の部分に分割します。それぞれの部分は、一度に 1 つずつ、ユーザーが作成した処理プロシージャーに渡されます。

XML PARSE ステートメントは、宣言型プロシージャーには指定できません。

フォーマット



ID-1

ID-1 は国別カテゴリーの基本データ項目、国別グループ、 英数字カテゴリーの基本データ項目、または英数字グループ項目でなければなりません。 *ID-1* を関数 ID にすることはできません。 *ID-1* には XML 文書の文字ストリームが含まれます。

ID-1 が国別グループ項目の場合、*ID-1* は国別カテゴリーの基本データ項目として処理されます。

ID-1 が国別カテゴリーの場合、 その内容は、Unicode UTF-16LE (CCSID 1200) を使用してエンコードする必要があります。 *ID-1* には複数のエンコード・ユニットを使用して表現される文字エンティティーを含めることはできません。文字参照を使用してそのような文字を表します。例:

- "񧘃" または
- "𐠓"

文字 x は小文字でなければなりません。

CHAR(NATIVE) および英数字 ID-1

ID-1 が英数字で CHAR(EBCDIC) コンパイラー・オプションが無効な場合、*ID-1* の内容は、ICU 変換ライブラリーでサポートされている UTF-8 Unicode または 1 バイト ASCII コード・ページを使用してエンコードする必要があります (「*International Components for Unicode: Converter Explorer*」 を参照)。

このようなデータ項目の文書が ASCII ではなく UTF-8 で構文解析されるようにするには、以下の条件があります。

- ランタイム・ロケールのコード・ページは UTF-8 ロケールでなければなりません。または、
- 文書に UTF-8 を指定する XML エンコード宣言が含まれている必要があります。または、
- 文書の先頭に UTF-8 バイト・オーダー・マークが必要です。

UTF-8 文書には、x'FFFF' より大きな Unicode スカラー値を持つ文字を含めることはできません。 そのような文字には、文字参照を使用してください。

そのようなデータ項目の XML 文書でエンコード宣言を指定していないときに、XML 文書の先頭に UTF-8 バイト・オーダー・マークがない場合は、現行のランタイム・ロケールが示すコード・ページで構文解析されます。

CHAR(EBCDIC) および英数字 ID-1

ID-1 が英数字で CHAR(EBCDIC) コンパイラー・オプションが有効な場合、*ID-1* の内容は、ICU 変換ライブラリーでサポートされている 1 バイト EBCDIC コード・ページを使用してエンコードする必要があります (「*International Components for Unicode: Converter Explorer*」 を参照)。 *ID-1* が基本項目である場合、NATIVE キーワードをそのデータ記述項目には指定できません。

そのようなデータ項目の XML 文書でエンコード宣言を指定していない場合、XML 文書は、EBCDIC_CODEPAGE 環境変数で指定されたコード・ページで構文解析されます。 あるいは、EBCDIC_CODEPAGE 環境変数が設定されていない場合は、現在のランタイムのロケールに選択されているデフォルトの EBCDIC コード・ページを使用して解析されます。 詳しくは、『サポートされるロケールおよびコード・ページ』を参照してください。

ランタイム・ロケールおよびコード・ページの設定および使用法

ランタイム・ロケールおよびコード・ページの設定および使用の詳細については、 「サポートされるロケールおよびコード・ページ」を参照してください。1 バイト ASCII および EBCDIC コード・ページとは、表「サポートされるロケールおよびコード・ページ」のラベルが「言語グループ」の列 (右端の列) が「表意文字言語」となっていないもののことです。

PROCESSING PROCEDURE 句

XML パーサーにより生成された各種のイベントを処理する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、プロシージャー名-2
PROCEDURE DIVISION の中のセクションまたは段落を指名し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* には、宣言セクションにあるプロシージャー名を使用してはいけません。

プロシージャー名-*1*
処理プロシージャーの中の最初の (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*
処理プロシージャー内にある最後のセクションまたは段落を指定します。

XML イベントごとに、パーサーはプロシージャー名-*1* というプロシージャーの最初のステートメントに制御を移します。 制御は常に処理プロシージャーから XML パーサーに戻されます。 制御がどの地点で戻されるかは、次のようにして決定されます。

- 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、プロシージャー名-*1* の段落の最後のステートメントの実行後に制御の戻りが行われます。
- 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、プロシージャー名-*1* のセクションにある最後の段落の最後のステートメントの実行後に制御の戻りが行われます。
- プロシージャー名-*2* が指定されており、それが段落名である場合、プロシージャー名-*2* の段落の最後のステートメントの実行後に制御の戻りが行われます。
- プロシージャー名-*2* が指定されており、それがセクション名である場合、プロシージャー名-*2* のセクションにある最後の段落の最後のステートメントの実行後に制御の戻りが行われます。

プロシージャー名-*1* とプロシージャー名-*2* との間で保持しなければならない唯一の関係は、連続する一連の処理を、プロシージャー名-*1* によって指名されるプロシージャーから始まり、プロシージャー

名-*2* によって指名されるプロシージャーの実行によって終了するように定義する、ということだけです。

戻る地点への論理パスが 2 つ以上ある場合、 EXIT ステートメントだけからなる段落の名前をプロシージャー名-*2* として指定することができます。 その場合、戻り点へのすべてのパスは、この段落に導かれます。

処理プロシージャーは、XML イベントを処理するすべてのステートメントから構成されます。処理プロシージャーの範囲の中には、プロシージャーの範囲内の CALL、EXIT、GO TO、GOBACK、MERGE、PERFORM、および SORT ステートメントにより 実行されるすべてのステートメントと、処理プロシージャーの範囲にあるステートメント実行の結果として実行される宣言型プロシージャーの中のすべてのステートメントが含まれます。

処理プロシージャーの範囲内では、GOBACK ステートメントまたは EXIT PROGRAM ステートメントを実行させることはできません。ただし、処理プロシージャーの範囲内で実行された CALL ステートメントによってそれぞれ制御が渡されたメソッドまたはプログラムから制御が戻る場合を除きます。

処理プロシージャーの範囲内では、 XML PARSE ステートメントを実行させることはできません。ただし、処理プロシージャーの範囲内で実行された CALL ステートメントによって制御が渡されたメソッドまたは最外部プログラムで XML PARSE ステートメントが実行される場合を除きます。

複数のスレッド上でプログラムが実行されている場合は、同じ XML ステートメント、 または別の XML ステートメントを同時に実行できます。

処理プロシージャーでは、STOP RUN ステートメントを指定して、実行単位を終了できます。

処理プロシージャーの詳細については、417 ページの『制御フロー』を参照してください。

ON EXCEPTION

ON EXCEPTION 句では、XML PARSE ステートメントで例外条件が発生したときに実行する命令ステートメントを指定します。

XML 文書の処理中に XML パーサーでエラーが検出されると、例外条件が発生します。 最初にパーサーは、特殊レジスター XML-EVENT に 'EXCEPTION' が設定された処理プロシージャーに制御を渡して、XML 例外をシグナル通知します。 またパーサーは、特殊レジスター XML-CODE に数字のエラー・コードを提供します。詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の 『*XML PARSE* の例外処理』を参照してください。

パーサーに通常の XML イベントが戻される前に、処理プロシージャーが XML-CODE を -1 に設定した場合、例外条件が存在します。 この場合、パーサーは EXCEPTION XML イベントをシグナル通知せず、構文解析処理は終了します。

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* に移します。 ON EXCEPTION 句が指定されていない場合は、NOT ON EXCEPTION 句が指定されていても無視され、 制御は、XML PARSE ステートメントの終わりに移ります。

XML PARSE ステートメントの実行後、特殊レジスター XML-CODE には XML 例外を示す数字のエラー・コードまたは -1 が格納されます。

パーサーに制御が戻る前に、処理プロシージャーで XML 例外イベントを処理し、 XML-CODE にゼロを設定すると、例外条件は発生しません。 パーサーの終了前に、その他の処理対象外の例外が発生しない場合は、NOT ON EXCEPTION 句の命令ステートメント-*2* に制御が移ります (NOT ON EXCEPTION 句が指定されている場合)。

NOT ON EXCEPTION

XML PARSE 処理の終了時に例外条件が存在しない場合もありますが、NOT ON EXCEPTION 句では、そうした場合に実行する命令ステートメントを指定します。

XML PARSE 処理の終了時に例外条件が存在しない場合は、NOT ON EXCEPTION 句の命令ステートメント-*2* に制御が移ります (NOT ON EXCEPTION 句が指定されている場合)。 NOT ON EXCEPTION 句が指定されていない場合は、XML PARSE ステートメントの終わりに制御が移ります。 ON EXCEPTION 句は、指定されていても無視されます。

XML PARSE ステートメントを実行すると、特殊レジスター XML-CODE にゼロが格納されます。

END-XML 句

この明示範囲終了符号は、XML GENERATE ステートメントまたは XML PARSE ステートメントの有効範囲を設定します。 END-XML は条件 XML GENERATE ステートメントまたは XML PARSE ステートメント (つまり、ON EXCEPTION 句または NOT ON EXCEPTION 句を指定する XML GENERATE ステートメントまたは XML PARSE 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を許可します。

条件 XML GENERATE または XML PARSE ステートメントの有効範囲は、次のいずれかによって終了します。

- ネスト構造で同じレベルにある END-XML 句。
- 分離文字ピリオド。

END-XML は、ON EXCEPTION 句または NOT ON EXCEPTION 句を指定しない XML GENERATE ステートメントまたは XML PARSE ステートメントと共に使用することもできます。

明示的範囲終了符号の詳細については、263 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。

ネストされた XML GENERATE ステートメントまたは XML PARSE ステートメント

XML GENERATE ステートメントまたは 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* として出現する場合、 あるいは別の XML GENERATE ステートメントまたは 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* の一部として出現する場合、その XML GENERATE ステートメントまたは XML PARSE ステートメントはネストされた XML GENERATE ステートメントまたは XML PARSE ステートメントになります。

ネストされた XML GENERATE ステートメントまたは XML PARSE ステートメントは、左から右に処理される、一致した XML GENERATE と END-XML、または XML PARSE と 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 END-XML 句はすべて、暗黙的または明示的に終了されていない、 先行の最も近い場所にある XML GENERATE ステートメントまたは XML PARSE ステートメントと一致します。

制御フロー

XML パーサーは、XML PARSE ステートメントから制御を受け取ると、XML 文書を分析し、プロセスの特定の時点で制御を転送します。

そのような時点は以下のとおりです。

- プロセス解析を開始するとき。
- 文書のフラグメントを検出したとき。
- XML 文書の解析時にパーサーがエラーを検出したとき。
- XML 文書の処理を終了するとき。

処理プロシージャーが終了すると、XML パーサーに制御が戻ります。

以下の状態になるまでは、パーサーと処理プロシージャーとの間で制御のやり取りが行われます。

- XML 文書全体が解析され、END-OF-DOCUMENT イベントで終了した場合。
- パーサーに制御が戻る前に、処理プロシージャーが XML-CODE に -1 を設定して、解析を強制終了した場合。
- パーサーは例外 (エンコード矛盾以外の例外) を検出し、処理プロシージャーはパーサーに制御を戻す前に特殊レジスター XML-CODE をゼロにリセットしません。
- パーサーはエンコード矛盾の例外を検出し、処理プロシージャーは特殊レジスター XML-CODE をゼロまたは文書エンコードの CCSID にリセットしません。

いずれの場合にも、処理プロシージャーは制御をパーサーに戻します。 その後パーサーが終了し、制御が XML PARSE ステートメントに戻ると、XML-CODE 特殊レジスターには、パーサーによって設定された最新の値または -1 (パーサーまたは処理プロシージャーによって設定された可能性がある) が格納されます。

XML イベントが処理プロシージャーに渡されるたびに、 XML-CODE、 および XML-EVENT 特殊レジスターには、特定のイベントに関する情報が格納されます。 特殊レジスター XML-EVENT には、'START-OF-DOCUMENT' などのイベント名が設定されます。 ほとんどのイベントでは、XML-TEXT または XML-NTEXT 特殊レジスターに文書テキストが格納されます。 詳しくは、24 ページの『XML-EVENT』を参照してください。

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。 処理プロシージャーの範囲の外では、 その他すべての XML 特殊レジスターの内容が未定義となります。

通常の XML イベントの場合は、制御が処理プロシージャーに移ると、特殊レジスター XML-CODE にゼロが格納されます。 XML 例外イベントの場合、XML-CODE には XML 例外コードが含まれています。 これについては、「*COBOL for Linux on x86* プログラミング・ガイド」の『継続を許可する *XML PARSE* 例外』および『継続を許可しない *XML PARSE* 例外』を参照してください。

XML 特殊レジスターの詳細については、以下を参照してください。

- 23 ページの『XML-CODE』
- 24 ページの『XML-EVENT』
- 26 ページの『XML-NTEXT』
- 26 ページの『XML-TEXT』

特殊レジスターの概要については、15 ページの『特殊レジスター』を参照してください

EXCEPTION イベントおよび例外処理の詳細については、「*COBOL for Linux on x86* プログラミング・ガイド」の 『*XML PARSE* の例外処理』を参照してください。

第 7 部 組み込み関数

第 20 章 組み込み関数

組み込み関数 は、算術演算、文字演算、または論理演算を行うための関数です。 組み込み関数を使用して、実行中に値が自動的に得られるデータ項目を参照することができます。

データ処理に関わる問題では、オブジェクト・プログラムに関連付けられたデータ・ストレージの中で直接アクセスすることができず、他のデータ操作を行うことによって取り込む値を使用する必要があります。 組み込み関数 は、算術演算、文字演算、論理演算を行うための関数で、 これらを使用して、オブジェクト・プログラムの実行中に値が自動的に得られるデータ項目を参照することができます。

組み込み関数は、実行されるサービスのタイプに応じて以下の 6 種類に分類できます。

- 算術
- 統計
- 日時
- 財務
- 文字処理
- その他

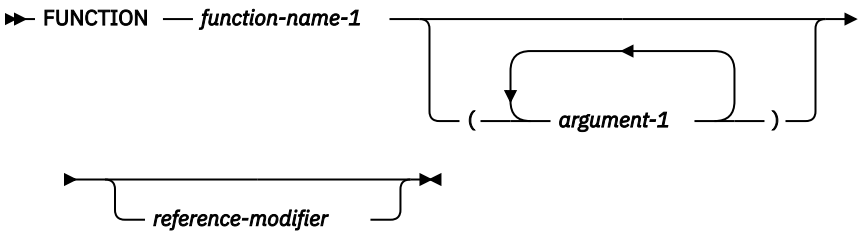
関数は、その名前を必要な引数と共に指定することによって、PROCEDURE DIVISION の中のステートメントで参照できます。

関数は基本データ項目であり、英数字、国別文字、数値、または整数値、整数、ブール値、または日時値を返します。 関数は、受信オペランドとして使用することはできません。

関数の指定

このトピックでは、関数 ID の一般的なフォーマットについて説明します。

フォーマット: 関数 ID



関数名-*1*
関数名-*1* は、組み込み関数の名前のうちの 1 つでなければなりません。

引数-*1*
引数-*1* は、指定された関数の引数要件を満たす、ID、リテラル (形象定数以外)、または算術式でなければなりません。

引数-*1* は、UNDATE 組み込み関数の場合を除いて、ウィンドウ表示日付フィールドであってはなりません。

参照修飾子
タイプが英数字または国別の関数についてのみ指定できます。

関数 ID は、その関数タイプのデータ項目を使用できるところであればどこでも指定できます。 関数の引数は、同じ関数についての別の評価を含め、結果がその引数の要件を満たす、任意の関数または関数を含む式にすることができます。

PROCEDURE DIVISION のステートメント内では、それぞれの関数 ID は、それと同じ位置にも ID に関連付けられた参照変更や添え字付けがあれば、それらが評価される時に同時に評価されます。

関数の定義と評価

ある関数のクラスと特性、およびそれが必要とする引数の個数とタイプは、 その関数定義によって決定されます。

関数の有する特性には、次のようなものがあります。

- 英数字および国別タイプの関数の場合は、戻り値のサイズ。
- 数字および整数タイプの関数の場合は、戻り値の符号、およびその関数が整数かどうか。
- 関数によって戻される実際の値。
- 日時関数の戻り値の長さ。

いくつかの関数の場合、そのクラスと特性は、 その関数への引数によって決定されます。

組み込み関数の評価はコンテキストには影響されません。 つまり、関数評価は関数以外の操作やオペランドには影響されません。 ただし、関数の評価は、引数の属性によって影響を受ける場合があります。

PROCEDURE DIVISION のステートメント内では、それぞれの関数 ID は、それと同じ位置にも ID に関連付けられた参照変更や添え字付けがあれば、それらが評価される時に同時に評価されます。

関数のタイプ

このトピックでは、COBOL での関数のタイプについて説明します。

COBOL の関数には以下のタイプがあります。

- 英数字
- ブール
- 日時
- DBCS
- 国別
- 数字
- 整数

英数字 関数は、英数字のクラスとカテゴリーに属します。 戻り値は、暗黙のうちに USAGE になります。 DISPLAY (NATIVE 句なし) になります。 戻り値の文字位置の数は、関数定義によって決定されます。

ブール 関数はブールのクラスとカテゴリーに属します。 戻り値では DISPLAY が暗黙に使用されます。 この戻り値はブール値の真 (B"1") またはブール値の偽 (B"0") のいずれかです。

日時 関数は、日時クラスと、日付、時刻、またはタイム・スタンプのカテゴリーに属します。戻り値は、暗黙のうちに USAGE DISPLAY になります。戻り値の文字位置の数は、関数定義によって決定されます。

DBCS 関数のクラスおよびカテゴリーは DBCS です。 戻り値では DISPLAY-1 が暗黙的に使用され、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 関数は、国別のクラスとカテゴリーに属します。 戻り値は、暗黙のうちに USAGE NATIONAL を持ち、国別文字 (UTF-16) で表現されます。 戻り値の文字位置の数は、関数定義によって決定されます。

数字 関数は、数字のクラスとカテゴリーに属します。戻り値は、常に演算符号を持つとみなされ、数字の中間結果になります。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」内の 『数字組み込み関数の使用』 を 参照してください。

整数 関数は、数字のクラスとカテゴリーに属します。 戻り値は、常に演算符号を持つとみなされ、整数の中間結果になります。 戻り値の桁数は、関数定義によって決定されます。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」内の 『数字組み込み関数の使用』 を 参照してください。

使用の規則

このトピックでは、さまざまなタイプの関数の使用規則について説明します。

英数字関数

英数字関数は、クラスおよびカテゴリーが英数字のデータ項目を使用できる、一般フォーマットに関連する規則が関数への参照を特別に禁止していない一般フォーマットの中であれば、どこでも指定することができます。ただし、以下の例外があります。

英数字引数は使用できる任意の関数の引数として、英数字関数を使用することができます。

英数字関数の参照変更は許可されています。 ある関数に対して参照変更が指定されている場合、その参照変更の評価は、その関数の評価の直後に行われます。 つまり、関数の戻り値は参照変更されます。

次の場合、英数字関数は使用できません。

- 任意のステートメントの受け入れ側のオペランドとする場合。
- 一般フォーマットに関連する規則が、参照されるデータ項目にある一定の特性 (クラスとカテゴリー、USAGE、サイズ、および暗黙的値など) を要求する場合で、 しかもその定義に従った関数の評価と指定された特定の引数がこれらの特性を持たない場合。

ブール関数

一般フォーマットにおいてブール ID が許可されていて、一般フォーマットに関連付けられている規則において関数への参照が特に禁じられていなければ、いつでもブール関数を指定できます。 ただし、以下の場合、ブール関数は指定できません。

- 任意のステートメントの受け入れ側のオペランドとする場合。
- 一般フォーマットに関連する規則が、参照されるデータ項目にある一定の特性 (クラスとカテゴリー、USAGE、サイズ、および暗黙的値など) を要求する場合で、しかもその定義に従った関数の評価と指定された特定の引数がこれらの特性を持たない場合。

ブール関数は比較条件の一部として使用できます。 ブール関数が比較条件において指定されると、ブール関数が評価された直後に比較条件が評価されます。

ブール関数は、ブール引数を許可する関数の引数として参照できます。

日時関数

一般フォーマットにおいて日時 ID が許可されていて、一般フォーマットに関連付けられている規則において関数への参照が特に禁じられていなければ、いつでも日時関数を指定できます。 ただし、以下の場合、ブール関数は指定できません。

- 任意のステートメントの受け入れ側のオペランドとする場合。
- 一般フォーマットに関連する規則が、参照されるデータ項目にある一定の特性 (クラスとカテゴリー、USAGE、サイズ、および暗黙的値など) を要求する場合で、しかもその定義に従った関数の評価と指定された特定の引数がこれらの特性を持たない場合。

日時関数は比較条件の一部として使用できます。 日時関数が比較条件において指定されると、日時関数が評価された直後に比較条件が評価されます。

日時関数は、日時引数を許可する関数の引数として参照できます。

国別関数

国別関数は、クラスおよびカテゴリーが国別のデータ項目を使用できる、一般フォーマットに関連する規則が関数への参照を特別に禁止していない一般フォーマットの中であれば、どこでも指定することができます。ただし、以下の例外があります。

国別引数は使用できる任意の関数の引数として、国別関数を使用することができます。

国別関数の参照変更は許可されています。 ある関数に対して参照変更が指定されている場合、その参照変更の評価は、その関数の評価の直後に行われます。 つまり、関数の戻り値は参照変更されます。

国別関数は使用できません。

- 任意のステートメントの受け入れ側のオペランドとする場合。

- 一般フォーマットに関連する規則が、参照されるデータ項目にある一定の特性 (クラスとカテゴリー、USAGE、サイズ、 および暗黙的値など) を要求する場合で、 しかもその定義に従った関数の評価と指定された特定の引数がこれらの特性を持たない場合。

数字関数
数字関数は、算術式を指定できる個所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。

たとえ個々の参照が整数値をもたらす場合でも、 数字関数は、整数オペランドが必要とされるところでは使用することができません。関数 INTEGER または関数 INTEGER-PART を使用すると、 引数のタイプを数字から整数に強制的に変えることができます。

整数関数
整数関数は、算術式を指定できる個所でのみ使用することができます。

整数関数は、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。

使用上の注意:

- 関数 ID を CALL ステートメントの BY REFERENCE 句で使用することはできません (つまり、CALL ステートメントの *identifier-2* を関数 ID にしないでください)。
- COPY ステートメントでは、REPLACING 句であらゆる種類の関数 ID を受け入れます。
- MOVE ステートメントは、送り出しフィールド (*identifier-1*) の英数字、数字、整数、および国別関数 ID を受け入れます。

引数

関数によって戻り値は、その関数が評価されるときに、関数 ID の中で指定された引数によって決定される場合があります。 関数の中には引数が不要なものもあれば、引数の固定数が必要なもの、さらには引数の可変数を受け入れるものがあります。

引数は、以下の項目のいずれかのうちの 1 つでなければなりません。

- データ項目 ID
- 算術式
- 関数 ID
- 形象定数以外のリテラル
- 特殊レジスター
- 簡略名
- キーワード
- タイプ名

関数に固有の引数指定については、428 ページの『関数の定義』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、または英文字だけを含む英数字リテラルのクラスに属する基本データ項目。 引数の内容は、関数の値を決定するために使用されます。 引数の長さは、関数の値を決定するために使用される場合があります。

英数字
英字クラス、英数字または英数字リテラルのクラスに属するデータ項目。 引数の内容は、関数の値を決定するために使用されます。 引数の長さは、関数の値を決定するために使用される場合があります。

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。

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ために使用されます。 引数の長さは、関数の値を決定するために使用される場合があります。

DBCS
DBCS クラスまたは DBCS リテラルのクラスに属する基本データ項目。 引数の内容は、関数の値を決定するために使用されます。 引数の長さは、関数の値を決定するために使用される場合があります。 (DBCS データ項目または DBCS リテラルは、引数としては NATIONAL-OF 関数に対してのみ使用できます。)

整数
結果が常に整数値になる算術式。 符号も含めてこの式の値は、関数の値を決定するために使用されます。

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。

国別
国別クラスに属するデータ項目 (カテゴリーは国別、国別編集、または数字編集)。 引数の内容は、関数の値を決定するために使用されます。 引数の長さは、関数の値を決定するために使用される場合があります。

数字
算術式。 式には、数値リテラルと、カテゴリーが数字、内部浮動小数点、および外部浮動小数点のデータ項目を含めることができます。数字データ項目は、データ項目のカテゴリーに対して許可されている任意の USAGE (NATIONAL を含む) を持つことができます。 符号も含めてこの式の値は、関数の値を決定するために使用されます。

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。

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。

関数によっては、その引数に対して制約 (受け入れ可能な値の範囲など) を設けているものがあります。 関数に対して引数として割り当てられる値が、 指定の制限に適合しない場合には、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る場合、その引数の評価は、 その関数より外側の関数が持つ引数によって影響を受けることはありません。

同一の関数のレベルにある引数だけが、相互作用します。 この相互作用は、次の 2 つの領域で発生します。

- 関数の引数として現れる算術式の計算は、その関数の他の引数によって影響を受けます。
- 関数の評価は、それが持つすべての引数の属性を考慮に入れて行われます。

関数の評価においては、その引数は、引数リストに指定された順番に左から右へと個別的に評価されます。評価される引数は、関数 ID、または関数 ID を含む式であることができます。

算術式を引数として指定し、その式の最初の演算子が単項演算子の加算記号または減算記号である場合、式はその直前に左括弧を必要とします。

浮動小数点リテラルは、数字引数が使用できる個所ならどこでも使用できます。また、数字引数が許される関数の中で使用する算術式中でも使用できます。

内部浮動小数点項目および外部浮動小数点項目 (display 浮動小数点および国別浮動小数点の両方) は、数字引数が許される個所であればどこでも使用できます。また、数字引数が許される関数への引数として算術式の中でも使用できます。

浮動小数点項目および浮動小数点リテラルは、整数引数が必要とされる個所や、英数字または国別クラスの引数が必要とされる個所 (LOWER-CASE、REVERSE、 UPPER-CASE、NUMVAL、および NUMVAL-C 関数など) では使用できません。

例

さまざまなタイプの組み込み関数の使用例を調べてください。

以下のステートメントは、英数字引数の中の各小文字を対応する大文字に置き換える、組み込み関数 UPPER-CASE の使用例です。

```
MOVE FUNCTION UPPER-CASE('hello') TO DATA-NAME.
```

このステートメントは、HELLO を DATA-NAME へ移動します。

以下のステートメントは、国別引数の中の各小文字に対応する大文字に置き換える組み込み関数 LOWER-CASE の使用例です。

```
MOVE FUNCTION LOWER-CASE(N'HELLO') TO N-DATA-NAME.
```

このステートメントは、国別文字 hello を N-DATA-NAME へ移動します。

以下のステートメントは、数字組み込み関数の使用例です。

```
COMPUTE NUM-ITEM = FUNCTION SUM(A B C)
```

このステートメントは数字関数 SUM を使用して、A、B、および C の値を加算し、その結果を NUM-ITEM へ格納します。

ALL 添え字

関数が 1 つの引数を任意の回数繰り返して使用できる場合は、テーブルを識別するデータ名と修飾子を指定することにより、そのテーブルを参照することができます。 これはその直後に、 1 つ以上の添え字として ALL の添え字付けを行うことができます。

ヒント: ALL 添え字の評価は、少なくとも 1 つの引数の中に結果としてもたらされる必要があります。 さもなければ、その関数によって戻り値は、未定義になってしまいます。 ただし、SSRANGE コンパイラー・オプションと CHECK ランタイム・オプションを指定すれば、実行時に状況の診断ができます。

添え字として ALL を指定するのは、その添え字の位置にあらゆる有効な添え字を使用して、すべての可能なテーブル・エレメントを指定することと同じです。

テーブル名 (ALL) としてテーブル引数を指定する場合、 1 つの引数として左から右への順序で各テーブル・エレメントの暗黙指定をします。この場合、最初 (左端) の引数が、テーブル名 (1) であり、ALL が 1 によって置き換えられます。 次の引数はテーブル名 (2) です。ここで、添え字の値は 1 だけ増加されています。 このプロセスは、暗黙の引数を 1 つずつ生成するために添え字の値を 1 ずつ増加しながら続行され、 添え字の値が ALL の値の範囲の最大値に達するまで行われます。

例えば、

```
FUNCTION MAX(Table(ALL))
```

は次のものと同じになります。

```
FUNCTION MAX(Table(1) Table(2) Table(3) ... Table(n))
```

ここで *n* は、Table の中のエレメント数です。

テーブル名 (ALL, ALL, ALL) のように ALL 添え字が複数ある場合、最初の暗黙の引数はテーブル名 (1, 1, 1) となります。ここで、各 ALL 添え字は 1 によってそれぞれ置き換えられています。次の引数はテーブル名 (1, 1, 2) となります。ここでは、右端の添え字が 1 だけ増加されてます。右端の ALL によって表現される添え字は、その値の範囲の限度まで増加され、それぞれの値に対する暗黙の引数を作り出します。

いったん ALL として指定された添え字が、その値の範囲の限度まで増加されると、 その左にある ALL として指定されている次の添え字が 1 だけ増加されます。 新たに増加される添え字の右にある ALL として指定されている各添え字は、1 に設定されて暗黙の引数が作り出されます。 ここで再び、右端の ALL によって表現されている添え字がその値の範囲の限度まで増加され、 その値に対し暗黙の引数を作り出します。この処理は、ALL として指定されている各添え字がその値の範囲の限度に増加されるまで繰り返されます。

例えば、

```
FUNCTION MAX(Table(ALL, ALL))
```

は次のものと同じになります。

```
FUNCTION MAX(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 Table の列の次元の中のエレメント数であり、*m* は Table の行の次元の中のエレメント数です。

ALL 添え字は、リテラル、データ名、 または指標名の各添え字と結合して多次元テーブルを参照することができます。

例えば、

```
FUNCTION MAX(Table(ALL, 2))
```

は次のものと同じになります。

```
FUNCTION MAX(Table(1, 2)
             Table(2, 2)
             Table(3, 2)
             ...
             Table(m, 2))
```

ここで、*m* は Table の行の次元の中のエレメント数です。

ALL 添え字が、ある引数に対して指定され、その引数が参照変更を行う場合、 その参照修飾子は、テーブルのエレメントとして暗黙に指定されたそれぞれのエレメントに適用されます。

ALL 添え字が参照変更を行うオペランドに対して指定されている場合、 その参照修飾子は、 テーブルのエレメントとして暗黙に指定されたそれぞれのエレメントに適用されます。

ALL 添え字が OCCURS DEPENDING ON 節に関連付けられている場合、 その値の範囲は、OCCURS DEPENDING ON 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レコードの定義があります。

```
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することによって、 その年のその日までの就業時間の合計、週当たりの最大勤務時間、 および最大勤務時間があった週を識別することができます。

```
COMPUTE YTD-HOURS = FUNCTION SUM (PAYROLL-HOURS(ALL))
COMPUTE MAX-HOURS = FUNCTION MAX (PAYROLL-HOURS(ALL))
COMPUTE MAX-WEEK  = FUNCTION ORD-MAX (PAYROLL-HOURS(ALL))
```

これらの関数呼び出しにおいて、ALL 添え字を使用することによって PAYROLL-HOURS 配列の (PAYROLL-WEEK フィールドの実行時間値に応じて異なる) すべてのエレメントを参照できます。

関数の定義

このセクションでは、各組み込み関数の引数タイプ、関数タイプ、および返される値の概要を示します。

組み込み関数について詳しくは、429 ページの表 58 を参照してください。

引数のタイプと関数のタイプは、次のような省略形を使用しています。

省略形	意味
A	英字
B	ブール
D	DBCS
DA	日時
I	整数
IX	索引
K	キーワード
M	簡略名
N	数字
O	関数で定義されるその他のタイプ (ポインター、関数ポインター、またはプロシージャー・ポインター)
P	ポインター
S	特殊レジスター
T	タイプ名
U	国別
X	英数字

"DP" のマークの付いた関数の動作は、DATEPROC または NODATEPROC コンパイラー・オプションが有効であるかによって異なります。

DATEPROC コンパイラー・オプションが有効な場合、以下の組み込み関数は日付フィールドを戻します。

関数	暗黙の DATE FORMAT を持つ戻り値
DATE-OF-INTEGER	YYYYXXXX
DATE-TO-YYYYMMDD	YYYYXXXX
DAY-OF-INTEGER	YYYYXXX
DAY-TO-YYYYDDD	YYYYXXX
YEAR-TO-YYYY	YYYY
DATEVAL	DATEVAL で指定されたフォーマットによって異なる
YEARWINDOW	YYYY

NODATEPROC コンパイラー・オプションが有効な場合

- 以下の組み込み関数は、DATEPROC が有効な場合と同じ値を戻しますが、戻り値は非日付データです。

  - DAY-OF-INTEGER
  - DATE-TO-YYYYMMDD
  - DAY-TO-YYYYDDD
  - YEAR-TO-YYYY
- DATEVAL および UNDATE 組み込み関数は効力を持たず、単にその (最初の) 引数を未変更のまま戻します。
- YEARWINDOW 組み込み関数は、無条件に 0 を戻します。

各組み込み関数については、以下の表の後のトピックで詳しく説明します。

表 *58.* **組み込み関数表**

関数名	引数	関数のタイプ	戻される値
ACOS	N1	N	N1 の逆余弦
ADD-DURATION	DA1、K2、I3	DA	期間が加算された日時項目
ANNUITY	N1、I2	N	N1 の金利で I2 期にわたり支払われる年金の初期投資額に対する割合
ASIN	N1	N	N1 の逆正弦
ATAN	N1	N	N1 の逆正接
CHAR	I1	X	プログラムを有する照合シーケンスの I1 の位置にある文字
COS	N1	N	N1 の余弦
CURRENT-DATE	なし	X	現在の日時とグリニッジ標準時からの時間差
CONVERT-DATE-TIME	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
DATE-OF-INTEGER[DP]	I1	I	整数で表された日付に相当する標準フォーマットの日付 (YYYYMMDD)
DATE-TO-YYYYMMDD[DP]	I1、I2	I	I1 (ウィンドウ表示西暦年 YYMMDD) に相当する年を、I2 と実行時の年との和 が終了年である 100 年間隔に従って変更した、標準フォーマットの日付 (YYYYMMDD)。
DATEVAL[DP]	I1	I	I1 に相当する日付フィールド
	X1	X	X1 に相当する日付フィールド
DAY-OF-INTEGER[DP]	I1	I	整数で表された日付に相当する年間通算日フォーマットの日付 (YYYYDDD)
DAY-TO-YYYYDDD[DP]	I1、I2	I	I1 (ウィンドウ表示西暦年の年間通算日フォーマット YYDDD) に相当する年を、 I2 と実行時の年との和が終了年である 100 年間隔に従って変更した、 年間通算日フォーマットの日付 (YYYYDDD)。

表 58. **組み込み関数表** (続き)

関数名	引数	関数のタイプ	戻される値
DISPLAY-OF	U1 または U1、I2、または U1、X2、または D2	X	I2 が指定された場合は I2 によって示されるコード・ページを使用して、I2 が指定されていない場合はランタイム・ロケールによって指定されたコード・ページを使用して、対応する文字表現に変換された U1 の各文字。
EXTRACT-DATE-TIME	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	DA1、DA2、または K3	I	2 つの日時項目間の整数の期間
INTEGER	N1	I	N1 を超えない最大の整数値
INTEGER-OF-DATE	I1	I	標準フォーマットの日付 (YYYYMMDD) に相当する整数で表された日付
INTEGER-OF-DAY	I1	I	年間通算日 (YYYYDDD) に相当する整数で表された日付
INTEGER-PART	N1	I	N1 の整数部分
LENGTH	A1、B1、D1、DA1、IX1、N1、O1、P1、T1、X1、または U1	I	引数のタイプによって異なる、国別文字位置、英数字位置またはバイト数のいずれかの引数の長さ
LOG	N1	N	N1 の自然対数
LOG10	N1	N	N1 の常用対数
LOWER-CASE	A1 または X1	X	引数内のすべての文字が小文字に設定される
	U1	U	引数内のすべての文字が小文字に設定される
MAX	A1...	X	最大引数の値。 関数のタイプは引数によって決定されることに注意
	I1...	I	最大引数の値。 関数のタイプは引数によって決定されることに注意
	N1...	N	最大引数の値。 関数のタイプは引数によって決定されることに注意
	X1...	X	最大引数の値。 関数のタイプは引数によって決定されることに注意
	U1...	U	最大引数の値。 関数のタイプは引数によって決定されることに注意
MEAN	N1...	N	引数の算術平均
MEDIAN	N1...	N	引数の中央値
MIDRANGE	N1...	N	引数の最小値と最大値の平均

表 58. **組み込み関数表** (続き)

関数名	引数	関数のタイプ	戻される値
MIN	A1...	X	最小引数の値。 関数のタイプは引数によって決まることに注意
	I1...	I	最小引数の値。 関数のタイプは引数によって決まることに注意
	N1...	N	最小引数の値。 関数のタイプは引数によって決まることに注意
	X1...	X	最小引数の値。 関数のタイプは引数によって決まることに注意
	U1...	U	最小引数の値。 関数のタイプは引数によって決まることに注意
MOD	I1、I2	I	I1 モジュロ I2
NATIONAL-OF	A1、X1、または D1	U	I2 が指定された場合は I2 によって示されるコード・ページを使用して、I2 が指定されていない場合はランタイム・ロケールによって指定されたコード・ページを使用して、国別文字に変換された引数の文字。
	A1、X1、または D1 。I2	U	I2 が指定された場合は I2 によって示されるコード・ページを使用して、I2 が指定されていない場合はランタイム・ロケールによって指定されたコード・ページを使用して、国別文字に変換された引数の文字。
	A1 または X1 または D1、U2	U	I2 が指定された場合は I2 によって示されるコード・ページを使用して、I2 が指定されていない場合はランタイム・ロケールによって指定されたコード・ページを使用して、国別文字に変換された引数の文字。
NUMVAL	X1 または U1	N	単純数字ストリングの数値
NUMVAL-C	X1 または U1。 X1、X2。 U1、U2	N	オプショナル・コンマや通貨符号の付いた数字ストリングの数値
ORD	A1 または X1	I	照合シーケンスにおける引数の順序位置
ORD-MAX	A1...、N1...、X1...、または U1...	I	最大引数の順序位置
ORD-MIN	A1...、N1...、X1...、または U1...	I	最小引数の順序位置
PRESENT-VALUE	N1、N2...	N	割引率が N1 で将来的な期間満了時の総額が N2 である一連の数字の現価
RANDOM	I1、なし	N	乱数

表 58. **組み込み関数表** (続き)

関数名	引数	関数のタイプ	戻される値
RANGE	I1...	I	最大引数の値から最小引数の値を減算したもの。 関数のタイプは引数によって決定されることに注意
	N1...	N	最大引数の値から最小引数の値を減算したもの。 関数のタイプは引数によって決定されることに注意
REM	N1、N2	N	N1/N2 の剰余
REVERSE	A1 または X1	X	引数の文字の逆配列
	U1	U	引数の文字の逆配列
SIN	N1	N	N1 の正弦
SQRT	N1	N	N1 の平方根
STANDARD-DEVIATION	N1...	N	引数の標準偏差
SUBTRACT-DURATION	DA1、K2、I3	DA	期間が減算された日時項目
SUM	I1...	I	引数の和。関数のタイプは引数によって決定されることに注意
	N1...	N	引数の和。関数のタイプは引数によって決定されることに注意
TAN	N1	N	N1 の正接
TEST-DATE-TIME	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	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		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
	D2 または D1、D2	D	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
	NL1 または NL1、NL2	NL	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
TRIML	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		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
	D1 または D1、D2	D	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
	NL1 または NL1、NL2	NL	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

表 58. **組み込み関数表** (続き)

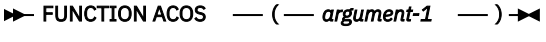
関数名	引数	関数のタイプ	戻される値
TRIMR	A1 または X1	X	右ブランクを持つ、または指定した文字を切り取ったストリング
	A1、A2 または X1、X2	X	右ブランクを持つ、または指定した文字を切り取ったストリング
	D1 または D1、D2	D	右ブランクを持つ、または指定した文字を切り取ったストリング
	NL1 または NL1、NL2	NL	右ブランクを持つ、または指定した文字を切り取ったストリング
UNDATE[DP]	I1	I	日付フィールド I1 または X1 に相当する非日付データ
	X1	X	日付フィールド I1 または X1 に相当する非日付データ
UPPER-CASE	A1 または X1	X	引数内のすべての文字が大文字に設定される
	U1	U	引数内のすべての文字が大文字に設定される
UTF8STRING	A1、X1、D1、または NL1	A	可変長 UTF-8 ストリング
VARIANCE	N1...	N	引数の分散
WHEN-COMPILED	なし	X	プログラムがコンパイルされた日時
YEAR-TO-YYYY[DP]	I1、I2	I	I1 (ウィンドウ表示西暦年 YY) に相当する年を、 実行時の年から I2 年後を終了年とする 100 年ウィンドウを使用して拡張した年 (YYYY)。
YEARWINDOW[DP]	なし	I	DATEPROC コンパイラー・オプションが有効な場合は、 YEARWINDOW コンパイラー・オプションによって指定された世紀ウィンドウの開始年 (YYYY フォーマット) が戻される。 NODATEPROC が有効な場合は、 0 が戻される。

ACOS

ACOS 関数は、指定された引数の逆余弦に近似する数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*

この引数のクラスは数字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。

戻り値は引数の逆余弦の近似値で、 0 以上 Pi 以下です。

ADD-DURATION

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。

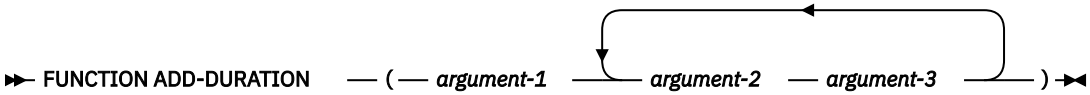
戻り値の長さは、引数-*1* に指定する日付項目、時刻項目、またはタイム・スタンプ項目の長さによって決まります。戻り値は、引数-*1* の長さまで切り捨てられます。

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 DATETIME ・コンパイラー・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.
```

関連参照
460 ページの『SUBTRACT-DURATION』

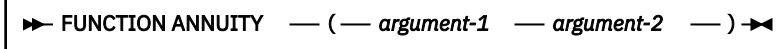
ANNUITY

ANNUITY 関数は、所定の期間数につき所定の金利で、各期の終わりに支払われる年金の初期値に対する比率を近似値で表す数字を戻します。

期間の数は引数-*2* によって指定します。金利は引数-*1* によって指定します。 例えば、引数-*1* が 0、引数-*2* が 4 であるとすると、 戻り値は比率 1/4 の近似値となります。

関数タイプは数字です。

フォーマット



引数-*1*
この引数のクラスは数字でなければなりません。 引数-*1* の値は 0 より大きいか等しくなければなりません。

引数-*2*
これは、正の整数である必要があります。

引数-*1* が 0 である場合、関数によって戻り値は次のような近似値になります。

1 / 引数-*2*

引数-*1* の値が 0 ではない場合、関数の値は次のような近似値になります。

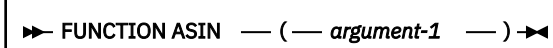
引数-*1* / (1 - (1 + 引数-*1*) ** (- 引数-*2*))

ASIN

ASIN 関数は、指定された引数の逆正弦に近似する数字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*
この引数のクラスは数字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。

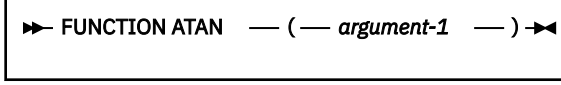
戻り値は、引数-*1* の逆正弦の近似値で、-Pi/2 以上 +Pi/2 以下です。

ATAN

ATAN 関数は、指定された引数の逆正接に近似する数字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*
この引数のクラスは数字でなければなりません。

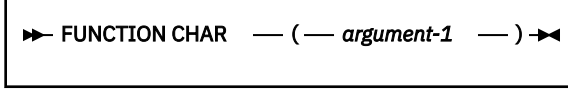
戻り値は、引数-*1* の逆正接の近似値で、-pi/2 以上 +pi/2 以下になります。

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。

この関数の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。この値は、0 以上で英数字データ項目 (最大 256) に関連した照合シーケンス内にある位置の数以下でなければなりません。

複数の文字が、プログラム照合シーケンス内で同じ位置を持っている場合、 関数の値として戻される文字は、 ALPHABET 節の中でその文字位置に対して指定された最初のリテラルの文字になります。

現在のプログラム照合シーケンスが ALPHABET 節によって指定されていない場合は、COLLSEQ コンパイラー・オプションで使用される照合シーケンスが示されます。 例えば、COLLSEQ(EBCDIC) を指定し、PROGRAM COLLATING SEQUENCE を指定しなかった (または NATIVE である) 場合、EBCDIC 照合シーケンスが適用されます。 (238 ページの『条件式』を参照。)

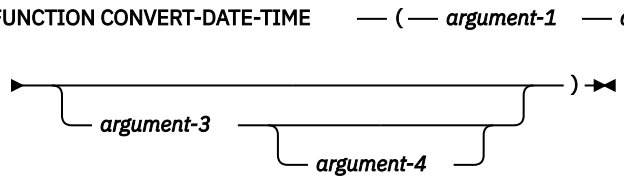
CONVERT-DATE-TIME

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 98 ページの『FORMAT 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。

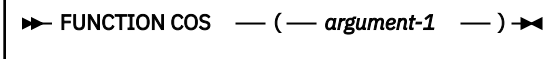
```
MOVE FUNCTION CONVERT-DATE-TIME ('95/05/30' DATE)
     TO  date-1.
MOVE FUNCTION CONVERT-DATE-TIME
     ('95/05/30' DATE '%y/%m/%d')
     TO  date-1.
MOVE FUNCTION CONVERT-DATE-TIME
     ('95/05/30' DATE '%y/%m/%d' my-locale)
     TO  date-1.
MOVE FUNCTION CONVERT-DATE-TIME
     ('95/05/30' DATE LOCALE my-locale)
     TO  date-1.
```

COS

COS 関数は、引数によって指定された角度または円弧の余弦に近似する数字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*
この引数のクラスは数字でなければなりません。

戻り値は引数の余弦の近似値で、-1 以上 +1 以下の値です。

CURRENT-DATE

CURRENT-DATE 関数は、カレンダー上の日付、時刻、およびこの関数が評価されるシステムで提供されているグリニッジ標準時との時差を表す、21 文字の英数字値を戻します。

この関数のタイプは英数字です。

フォーマット

以下の戻り値の 21 文字は、左から右への順序で以下のように解釈します。

文字位置	内容
1 から 4	グレゴリオ暦におけるその年を表す 4 桁の数字。
5 から 6	月を表す 2 桁の数字で、01 から 12 の範囲にある値。
7 から 8	日を表す 2 桁の数字で、01 から 31 の範囲にある値。
9 から 10	深夜午前 0 時からの時間を表す 2 桁の数字で、00 から 23 の範囲にある値。
11 から 12	分を表す 2 桁の数字で、00 から 59 の範囲にある値。
13 から 14	秒を表す 2 桁の数字で、00 から 59 の範囲にある値。
15 から 16	100 分の 1 秒を表す 2 桁の数字で、00 から 99 の範囲の値。 関数が評価されるシステムが、 秒よりも細かい単位の時間を供給する機能を備えていない場合は、 値 00 が戻されます。
17	'-' または '+' のいずれかの文字。 先行する文字位置に示されている地方時がグリニッジ標準時より遅れている場合には、文字 '-' が戻されます。 地方時がグリニッジ標準時より進んでいるかまたは等しい場合には、文字 '+' が戻されます。 この関数を評価するシステムが現地時間の時差を計算して渡す機能を備えていない場合は、 文字 '0' が戻されます。
18 から 19	17 の文字位置が '-' の場合は、時間を表す 00 から 12 の範囲の 2 桁の数字が戻されます。ここに表示されるのは、グリニッジ標準時より遅れている時間数です。 文字位置 17 が「+」の場合は、報告された時刻がグリニッジ標準時より進んでいる時間数を示す 2 桁の数字が 00 から 12 までの範囲で返されます。 17 の文字位置が '0' の場合は、値 00 が戻されます。
20 から 21	前記の時間差に加えるべき分単位の時間を表す 00 から 59 の範囲の 2 桁の数字が戻されます。17 の文字位置が '+' か '-' かに応じて、それぞれここ表示される分の数だけグリニッジ標準時より進んでいるか、または遅れています。 17 の文字位置が '0' の場合は、値 00 が戻されます。

詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『例: 数字組み込み関数』を参照してください。

DATE-OF-INTEGER

DATE-OF-INTEGER 関数は、グレゴリオ暦の日付を整数で表された日付形式から標準形式の日付 (YYYYMMDD) に変換します。

この関数のタイプは整数です。

この関数のもたらす結果は、8 桁の整数です。

DATEPROC コンパイラー・オプションが有効な場合、戻り値は暗黙 DATE FORMAT YYYYXXXX を持つ拡張日付フィールドです。

フォーマット

引数-*1*

正の整数で、グレゴリオ暦の 1601 年 1 月 1 日以降の日数を表します。 有効な範囲は 1 から 3,067,671 で、 これは 1601 年 1 月 1 日から 9999 年 12 月 31 日の範囲に対応します。

戻り値は、引数-*1* として指定された整数に相当する国際標準化機構 (ISO) の標準フォーマットの日付を表します。

戻り値は、YYYYMMDD 形式の整数で、 YYYY はグレゴリオ暦の年を、 MM はその年の月を、 DD はその月の日を表します。

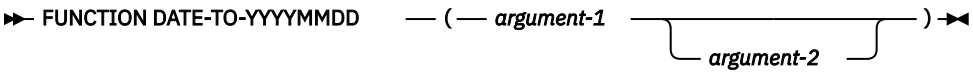
DATE-TO-YYYYMMDD

DATE-TO-YYYYMMDD 関数は、引数-*1* の値を、2 桁の年の日付 (YYnnnn) から 4 桁の年の日付 (YYYYnnnn) に変換します。 実行時に引数-*2* が年に追加されると、100 年間隔の終了年を定義するか、引数-*1* の年を入れるスライディング世紀ウィンドウを定義します。

この関数のタイプは整数です。

DATEPROC コンパイラー・オプションが有効な場合、戻り値は暗黙 DATE FORMAT YYYYXXXX を持つ拡張日付フィールドです。

フォーマット



引数-*1*
0 または 991,232 未満の正の整数でなければなりません。

注: COBOL 実行時には、値が有効日付であるかどうかの検証は行われません。

引数-*2*
整数でなければなりません。引数-*2* を省略すると、関数は値 50 が指定されたものと想定して評価されます。

実行時における年と引数-*2* の値の合計は、10,000 よりも小さく、1,699 よりも大きくなければなりません。

DATE-TO-YYYYMMDD 関数からの戻り値について以下の例を確認してください。

現在の年	引数-*1* の値	引数-*2* の値	戻り値
2002	851003	120	20851003
2002	851003	-20	18851003
2002	851003	10	19851003
1994	981002	-10	18981002

DATEVAL

DATEVAL 関数は、非日付データを日付フィールドに変換して、日付フィールドで使用してもあいまいにならないようにします。

DATEPROC コンパイラー・オプションが有効な場合、戻り値は引数-*1* の値が未変更のままの日付フィールドです。 結果としての日付フィールドの使用方法については、以下を参照してください。

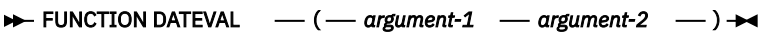
- 算術式の場合、236 ページの『日付フィールドを使用する算術計算』を参照してください。
- 条件式の場合、250 ページの『日付フィールドの比較』を参照してください。

NODATEPROC コンパイラー・オプションが有効であると、DATEVAL 関数は効力を持たず、引数-1 の値が未変更のまま戻されます。

関数のタイプは、次に示すように引数-*1* のタイプに応じて異なります。

引数のタイプ	関数のタイプ
英数字	英数字
整数	整数

フォーマット



引数-*1*
以下のいずれかにする必要があります。

- 引数-*2* で指定された日付フォーマットと同じ文字数を持つクラス英数字項目。
- 整数。 これを使用して負の値を含め、引数-*2* で指定された範囲外の値を指定することができます。

引数-*1* の値は、引数-*2* で指定された形式の日付を表します。

引数-*2*
162 ページの『DATE FORMAT 節』に定義されているとおり、日付パターンを指定する英数字リテラルでなければなりません。 日付パターンは YY または YYYY (それぞれウィンドウ表示西暦年と拡張西暦年を表す) で構成されており、 場合によっては以下に示すように 1 つまたは複数の X (月日など日付の他の部分) がその前後に付くことがあります。 値の大/小文字は区別されません。引数-*2* の文字 X と Y は、大文字と小文字の混在が可能です。

日付パターン・ストリング	指定する引数-*1* の内容
YY	ウィンドウ表示 (2 桁) 年。
YYYY	拡張 (4 桁) 年。
X	1 文字。例えば、1 学期または四半期 (1 から 4) などを表す数字。
XX	2 文字。例えば、月を表す数字 (01 から 12)。
XXX	3 文字。例えば、1 年のうちの何日目かを表す数字 (001 から 366)。
XXXX	4 文字。例えば、月を表す 2 桁 (01 から 12) と月の何日かを表す 2 桁 (01 から 31)。

DAY-OF-INTEGER

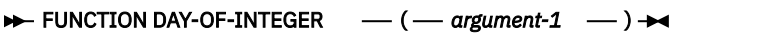
DAY-OF-INTEGER は、グレゴリオ暦の日付を整数で表された日付から年間通算日形式 (YYYYDDD) に変換します。

この関数のタイプは整数です。

この関数のもたらす結果は、7 桁の整数です。

DATEPROC コンパイラー・オプションが有効な場合、戻り値は暗黙 DATE FORMAT YYYYXXX を持つ拡張日付フィールドです。

フォーマット



引数-*1*
正の整数で、グレゴリオ暦の 1601 年 1 月 1 日以降の日数を表します。 有効な範囲は 1 から 3,067,671 で、 これは 1601 年 1 月 1 日から 9999 年 12 月 31 日の範囲に対応します。

戻り値は、引数-*1* として指定された整数に相当する年間通算日を表します。 戻り値は、YYYYDDD 形式の整数で、 YYYY はグレゴリオ暦の年を、 DDD はその年の日を表します。

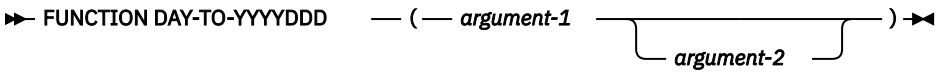
DAY-TO-YYYYDDD

DATE-TO-YYYYDDD 関数は、引数-*1* の値を、 2 桁の年の日付 (YYnnn) から 4 桁の年の日付 (YYYYnnn) に変換します。 実行時に引数-*2* が年に追加されると、100 年間隔の終了年を定義するか、引数-*1* の年を入れるスライディング世紀ウィンドウを定義します。

この関数のタイプは整数です。

DATEPROC コンパイラー・オプションが有効な場合、戻り値は暗黙 DATE FORMAT YYYYXXX を持つ拡張日付フィールドです。

フォーマット



引数-*1*

0 または 99,367 未満の正の整数でなければなりません。

COBOL 実行時には、値が有効日付であるかどうかの検証は行われません。

引数-*2*

整数でなければなりません。引数-*2* を省略すると、関数は値 50 が指定されたものと想定して評価されます。

実行時における年と引数-*2* の値の合計は、10,000 よりも小さく、1,699 よりも大きくなければなりません。

DAY-TO-YYYYDDD 関数から戻り値の例を以下に示しています。

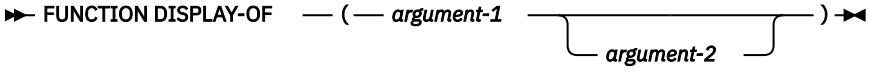
現在の年	引数-*1* の値	引数-*2* の値	戻り値
2002	10004	-20	1910004
2002	10004	-120	1810004
2002	10004	20	2010004
2013	95005	-10	1995005

DISPLAY-OF

DISPLAY-OF 関数は、特定のコード・ページ表現に変換された *argument-1* の内容から構成される英数字ストリングを戻します。

この関数のタイプは英数字です。

フォーマット



引数-*1*

国別クラス (USAGE NATIONAL を指定して記述されている、国別、国別編集、数字編集カテゴリー) でなければなりません。 引数-*1* は変換に使用するソース・ストリングを示します。

引数-*2*

整数でなければなりません。これは整数または英数字クラスで なければなりません。 引数-*2* は変換に使用する出力コード・ページを示します。

引数-*2* は、英数字クラスの 場合は、ICU 変換ライブラリーでサポートされている基本コード・ページ名またはコード・ページ別名を 識別しなければなりません (「*International Components for Unicode: Converter Explorer*」 を参照)。

引数-*2* が整数の場合、その整数は有効な CCSID 番号でなければなりません。

引数-*2* を省略すると、出力コード・ページは、ランタイム・ロケールから決定されます。

戻り値は、出力コード・ページの表現に変換された引数-*1* の文字で構成される英数字ストリングです。 ソース文字を出力コード・ページの文字に変換できないときは、ソース文字が置換文字によって置き換えられます。 一般的に使用されているいくつかのコード・ページの置換文字を以下の表に示します。

出力コード・ページ	**置換文字**
SBCS ASCII PC Windows SBCS	X'7F'
ISO SBCS	X'1A'
EBCDIC SBCS	X'3F'
ASCII DBCS	X'FCFC'
EBCDIC DBCS (タイ語以外)	X'FEFE'
EBCDIC DBCS (タイ語)	X'41B8'
PC DBCS (日本語または中国語)	X'FCFC'
PC DBCS (韓国語)	X'BFFC'
EUC (韓国語)	X'AFFE'
EUC (日本語)	X'747E'
UTF-8	SBCS から変換する場合: X'1A' MBCS から変換する場合: X'EFBFBD'
UTF-16	SBCS から変換する場合: X'1A00' MBCS から変換する場合: X'FDFF'

例外条件は発生しません。

戻り値の長さは、引数-*1* の内容および出力コード・ページの特性によって異なります。

使用上の注意

- コード・ページ名を使用すると、他の Linux ソフトウェアと整合性が出ますが、ソース・コードは Enterprise COBOL for z/OS に移植できません。
- UTF-8 を表す CCSID は 1208 です。
- 出力コード・ページに DBCS 文字が含まれる場合は、戻り値に SBCS ストリングと DBCS ストリングが混合する場合があります。
- DISPLAY-OF 関数は、引数-*2* とともに指定すると、ASCII または EBCDIC データに有効なコード・ページとは異なるコード・ページで表現された文字データを生成するために使用できます。 その後のそのデータに対する COBOL 操作として、データが有効な ASCII または EBCDIC コード・ページで表現されていることを前提とした暗黙の変換を実行できます。 複数のコード・ページを使用して表されたデータを 1 つのプログラム内で処理する例およびプログラミング手法については、「*COBOL for Linux on x86* プログラミング・ガイド」の『国別 *(*ユニコード*)* 表記との間の変換』を参照してください。

例外: 変換が失敗した場合、重大なランタイム・エラーが発生します。

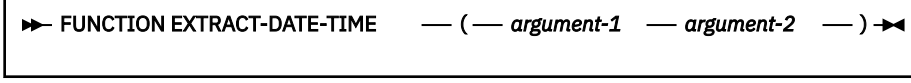
EXTRACT-DATE-TIME

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。

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')

その他の有効な変換指定のリストについては、SPECIAL-NAMES 段落の FORMAT 節の説明にある表 1 を参照してください。

引数-*2* で使用され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 (この場合、%p は @p になり、@p はロケールに基づく変換指定子の %p に相当します)。
4. PICOSECONDS は、タイム・スタンプ項目からのみ抽出できます。

例

```
COMPUTE integer-1 = FUNCTION EXTRACT-DATE-TIME (date-3 MONTHS).
```

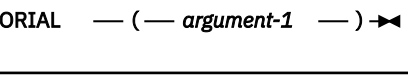
```
COMPUTE integer-1 = FUNCTION EXTRACT-DATE-TIME (date-3 '%m').
MOVE FUNCTION EXTRACT-DATE-TIME (date-2 '%m/%d') to alphanum-1.
```

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。

この関数のタイプは整数です。

フォーマット



引数-*1*
ARITH(COMPAT) コンパイラー・オプションが有効である場合は、引数-*1* は、0 以上 28 以下の整数でなければなりません。 ARITH(EXTEND) コンパイラー・オプションが有効である場合は、引数-*1* は、0 以上 29 以下の整数でなければなりません。

引数-*1* の値が 0 の場合、値 1 が戻されます。0 以外の場合は、引数-*1* の階乗が戻されます。

FIND-DURATION

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。

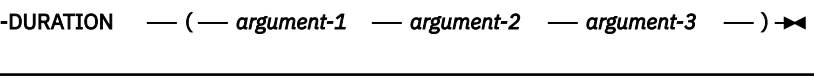
FIND-DURATION 関数は、以下の間の期間を計算する場合に使用します。

- 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
- 2 つのタイム・スタンプ

この関数のタイプは整数です。

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。 たとえば 2020 年 3 月 17 日と 2020 年 5 月 2 日との間の期間は 1.5 か月です。FIND-DURATION は整数のみを戻すので、.5 は切り捨てられ、戻される実際の値は 1 になります。
4. PICOSECONDS 期間は、引数-*1* と 引数-*2* がタイム・スタンプ項目である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。

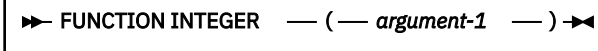
```
COMPUTE integer-1 = FUNCTION FIND-DURATION (date-3 date-4 MONTHS).
COMPUTE integer-1 = FUNCTION FIND-DURATION (timestamp-1 date-5 MONTHS).
```

INTEGER

INTEGER 関数は、指定された引数より小さいか等しい最大の整数値を戻します。

この関数のタイプは整数です。

フォーマット



引数-*1*
この引数のクラスは数字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) は値として 2 を戻し、FUNCTION INTEGER (-2.5) は値として -3 を戻します。

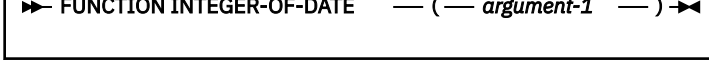
INTEGER-OF-DATE

INTEGER-OF-DATE 関数は、グレゴリオ暦の日付を標準形式の日付 (YYYYMMDD) から整数で表された日付形式に変換します。

この関数のタイプは整数です。

この関数のもたらす結果は、1 から 3,067,671 の範囲の 7 桁の整数です。

フォーマット



引数-*1*
YYYYMMDD 形式の整数である必要があります。その値は、(YYYY * 10,000) + (MM * 100) + DD という計算から得られます。この場合、以下が適用されます。

- YYYY は、グレゴリオ暦の年を表します。 この値は、1,600 より大きく、9,999 以下の整数でなければなりません。
- MM は月を表し、13 より小さい正の整数でなければなりません。
- DD は日を表し、32 より小さい正の整数でなければなりません。 ただし、指定の年と月に対して有効な日付にしてください。

戻り値は整数で、引数-*1* によって表された日付をグレゴリオ暦で 1601 年 1 月 1 日以降の日数に換算したものです。

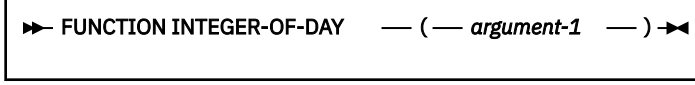
INTEGER-OF-DAY

INTEGER-OF-DAY 関数は、グレゴリオ暦の日付を年間通算日形式 (YYYYDDD) から整数で表された日付形式に変換します。

この関数のタイプは整数です。

この関数のもたらす結果は、7 桁の整数です。

フォーマット



引数-*1*

YYYYDDD 形式の整数である必要があります。その値は、 (YYYY * 1000) + DDD という計算から得られます。この場合、以下が適用されます。

- YYYY は、グレゴリオ暦の年を表します。 この値は、1,600 より大きく、9,999 以下の整数でなければなりません。
- DDD は年間通算日を表します。 この値は、指定の年に対して有効な 367 より小さい正の整数でなければなりません。

戻り値は整数で、引数-*1* によって表された日付をグレゴリオ暦で 1601 年 1 月 1 日以降の日数に換算したものです。

INTEGER-PART

INTEGER-PART 関数は、指定された引数の整数部分である整数を戻します。

この関数のタイプは整数です。

フォーマット

引数-*1*

この引数のクラスは数字でなければなりません。

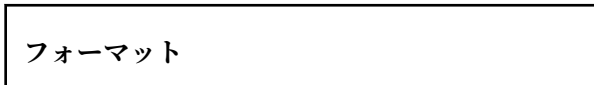
引数-*1* の値がゼロの場合、戻り値は 0 です。 引数-*1* の値が正の場合、戻り値は引数-*1* の値以下である最大の整数となります。 引数-*1* の値が負の場合、戻り値は引数-*1* の値以上である最小の整数となります。

LENGTH

LENGTH 関数は、引数の長さ (USAGE NATIONAL の引数については国別文字位置数、その他すべての引数については英数字文字位置数またはバイト数) に等しい整数を戻します。 英数字位置とバイトは等価です。

この関数のタイプは整数です。

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。

- 英数字リテラル、または国別リテラル

- DBCS 以外の任意のクラスのデータ項目
- USAGE POINTER、PROCEDURE-POINTER、または FUNCTION-POINTER として記述されるデータ項目
- ADDRESS OF 特殊レジスター
- LENGTH OF 特殊レジスター
- XML-NTEXT 特殊レジスター
- XML-TEXT 特殊レジスター

戻り値は、9 桁の整数 で、以下のように決定されます。

- 引数-*1* が英数字リテラル、あるいは英字または英数字クラスの基本データ項目である場合、戻り値は、引数の英数字文字位置の数と等しくなります。

  引数-*1* がヌル終了英数字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と等しくなります。

  英数字データ項目の長さまたは 1 バイト文字と 2 バイト文字が混在するリテラルの長さは、 各バイトが 1 バイト文字であるものとして数えられます。
- 引数-*1* が英数字グループ項目である場合、戻り値は、グループの内容に関係なく、引数-*1* の英数字文字位置分の長さに等しくなります。 引数-*1* に従属するいずれかのデータ項目が OCCURS 節の DEPENDING 句を使用して記述されている場合、引数-*1* の長さは DEPENDING 句の中で指定されたデータ項目の内容によって決定されます。 この評価は、送り出しデータ項目に関する OCCURS 節における規則に従って実施されます。 詳細については、173 ページの『OCCURS 節』 および 213 ページの『USAGE 節』 の説明を参照してください。

  戻り値は、暗黙の FILLER 位置がある場合はその位置を含みます。
- 引数-*1* が国別リテラル、または USAGE NATIONAL を指定して記述された基本データ項目である場合、戻り値は、引数-*1* の国別文字位置分の長さに等しくなります。

  例えば、引数-*1* が USAGE NATIONAL で PIC 9(3) と定義されている場合、引数のストレージ・サイズは 6 バイトですが、戻り値は 3 になります。
- 引数-*1* が国別グループ項目である場合、戻り値は、引数-*1* の国別文字位置分の長さに等しくなります。 引数-*1* に従属するいずれかのデータ項目が OCCURS 節の DEPENDING 句を使用して記述されている場合、引数-*1* の長さは DEPENDING 句の中で指定されたデータ項目の内容によって決定されます。 この評価は、送り出しデータ項目に関する OCCURS 節における規則に従って実施されます。 詳細については、173 ページの『OCCURS 節』 および 213 ページの『USAGE 節』 の説明を参照してください。

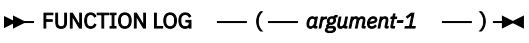
  戻り値は、暗黙の FILLER 位置がある場合はその位置を含みます。
- それ以外の場合は、戻り値は引数-*1* が占めるストレージのバイト数になります。

LOG

LOG 関数は、指定された引数の e (natural log) を底とする対数に近似する数値を戻します。

関数タイプは数字です。

フォーマット



引数-*1*
この引数のクラスは数字でなければなりません。 引数-*1* の値は 0 よりも大きくなければなりません。

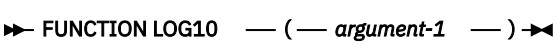
戻り値は e を底とする引数-*1* の対数の近似値です。

LOG10

LOG10 関数は、指定された引数の 10 を底とする対数に近似する数値を戻します。

関数タイプは数字です。

フォーマット



引数-1
この引数のクラスは数字でなければなりません。 引数-*1* の値は 0 よりも大きくなければなりません。

戻り値は 10 を底とする引数-*1* の対数の近似値です。

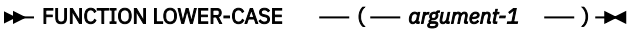
LOWER-CASE

LOWER-CASE 関数は、引数内の文字を含む文字ストリングを、大文字を対応する小文字にそれぞれ置き換えて返します。

関数のタイプは、次に示すように引数のタイプに応じて異なります。

引数のタイプ	関数のタイプ
英字	英数字
英数字	英数字
国別	国別

フォーマット



引数-1
英字、英数字、または国別のクラスでなければならず、少なくとも 1 文字位置の長さでなければなりません。

各大文字がそれに対応する小文字に置き換えられるという点を除いては、引数-*1* と同じ文字ストリングが返されます。

大文字から小文字への文字の変換は、適用できるランタイム・ロケールの文字属性の仕様に基づいています。

ロケールによっては、大文字から小文字への文字の変換によって、文字ストリングの長さが引数-*1* の長さと異なる場合があります。 これは、使用できるすべてのタイプの引数で起こる可能性があります。 大文字の英字 (A から Z)、小文字の英字 (a から Z)、および数字 (0 から 9) のみからなる英字および英数字引数の場合、戻される文字ストリングの長さは引数-*1* と同じになります。

MAX

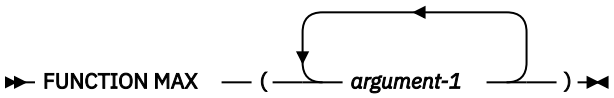
MAX 関数は、最大値を含む引数の内容を戻します。

関数のタイプは、次に示すように引数のタイプに応じて異なります。

引数のタイプ	関数のタイプ
英字	英数字
英数字	英数字

引数のタイプ	関数のタイプ
国別	国別
すべての引数が整数 (使用法が NATIONAL の整数引数を含む)	整数
数字 (一部の引数が整数) (USAGE NATIONAL の数字引数を含む)	数字

フォーマット



引数-1
クラスの英字、英数字、国別、または数字でなければなりません。

すべての引数が同じクラスに属している必要があります。ただし、例外として英字と英数字の引数の組み合わせが許可されています。

戻り値は、最大値を持っている引数-*1* の内容です。 最大値を決定するために使用される比較は、単純条件に関する規則に従って行われます。 詳しくは、238 ページの『条件式』を参照してください。.

複数の引数-*1* が同一の最大値を持つ場合、その値を持つもののうち左端の引数-*1* が戻されます。

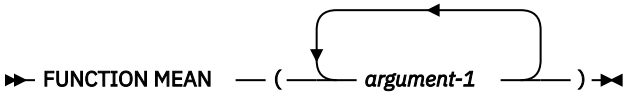
関数のタイプが英数字または国別である場合、戻り値のサイズは選択した引数-*1* のサイズと同じです。

MEAN

MEAN 関数は、その引数の算術平均に近似する数値を戻します。

関数タイプは数字です。

フォーマット



引数-1
この引数のクラスは数字でなければなりません。

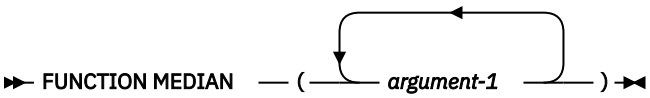
戻り値は一連の引数-*1* の算術平均値です。 戻り値は一連の引数-*1* の合計を 引数-*1* によって参照した出現数で除算したものとして定義されます。

MEDIAN

MEDIAN 関数は、複数の引数をソート順に並べ変えて作成したリスト中で中央値を値として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*
この引数のクラスは数字でなければなりません。

戻り値は、すべての引数-*1* の値をソート順に並べ変えて作成したリスト中で中央値を持つ引数-*1* の内容です。

引数-*1* によって参照されるオカレンス項目数が奇数である場合、戻り値は 次のようなものになります。つまり、引数-*1* として参照されるオカレンスの少なくとも半分が持つ値は、 戻り値よりも大きいかまたはそれに等しく、もう一方の半分が持つ値は、戻り値よりも小さいかまたはそれに等しくなります。 引数-*1* によって参照されるオカレンス項目数が偶数である場合、戻り値は中央にある 2 つのオカレンス項目によって指示される 2 つの値の算術平均になります。

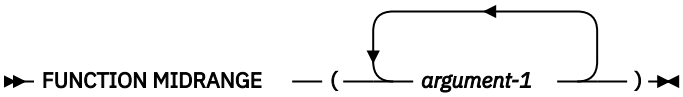
ソート順で引数値を並べ変えるために使用される比較は、 単純条件に関する規則に従って行われます。 詳しくは、238 ページの『条件式』を参照してください。.

MIDRANGE

MIDRANGE 関数は、最小の引数の値と最大の引数の値の算術平均に近似した数値を戻します。

関数タイプは数字です。

フォーマット



引数-*1*
この引数のクラスは数字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* の値の算術平均です。 最大値と最小値を決定するために使用される比較は、 単純条件に関する規則に従って行われます。 詳しくは、238 ページの『条件式』を参照してください。

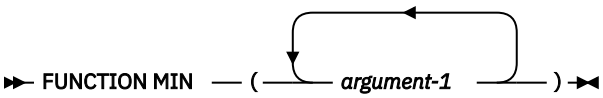
MIN

MIN 関数は、最小値を含む引数の内容を戻します。

関数のタイプは、次に示すように引数のタイプに応じて異なります。

引数のタイプ	関数のタイプ
英字	英数字
英数字	英数字
国別	国別
すべての引数が整数 (使用法が NATIONAL の整数引数を含む)	整数
数字 (一部の引数が整数) (USAGE NATIONAL の数字引数を含む)	数字

フォーマット



引数-1
クラスの英字、英数字、国別、または数字でなければなりません。

すべての引数が同じクラスに属している必要があります。ただし、例外として英字と英数字の引数の組み合わせが許可されています。

戻り値は、最小値を持っている引数*-1* の内容です。 最小値を決定するために使用される比較は、 単純条件に関する規則に従って行われます。 詳しくは、238 ページの『条件式』を参照してください。.

複数の引数*-1* が同一の最小値を持つ場合、その値を持つもののうち左端の引数*-1* が戻されます。

関数のタイプが英数字または国別である場合、戻り値のサイズは選択した引数*-1* のサイズと同じです。

MOD

MOD 関数は、引数*-2* をモジュロとする引数*-1* である整数値を戻します。

この関数のタイプは整数です。

関数のもたらす結果は、引数*-1* および引数*-2* のうち桁数の少ない方と同じ桁数の整数になります。

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。0 にすることはできません。

戻り値は、引数*-2* をモジュロとする引数*-1* 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*))

引数*-1* および引数*-2* のいくつかの値について、予期される結果を下の表に示します。

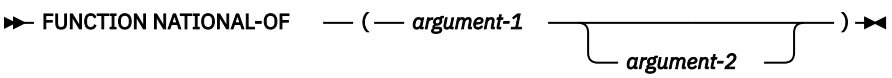
引数-1	***引数-2***	**戻り値**
11	5	1
-11	5	4
11	-5	-4
-11	-5	-1

NATIONAL-OF

NATIONAL-OF 関数は、引数*-1* の文字の国別文字表現で構成される国別文字ストリングを戻します。

この関数のタイプは国別です。

フォーマット



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・ストリングを指定します。

引数-*2*
整数でなければなりません。これは整数または英数字クラスで なければなりません。 引数-*2* は変換のソース・コード・ページを指定します。

引数-*2* は、英数字クラスの 場合は、ICU 変換ライブラリーでサポートされている基本コード・ページ名またはコード・ページ別名を 識別しなければなりません (「*International Components for Unicode: Converter Explorer*」 を参照)。

引数-*2* が整数の場合、その整数は有効な CCSID 番号でなければなりません。

引数-*2* を省略すると、ソース・コード・ページは、次のように決定されます。

- 引数-*1* が固有項目 (ASCII または ASCII DBCS、EUC、または UTF-8 データを含む USAGE DISPLAY または USAGE DISPLAY-1) である場合、ソース・コード・ページはランタイム・ロケールから決定されます。
- 引数-*1* が EBCDIC または EBCDIC DBCS データを含む USAGE DISPLAY または USAGE DISPLAY-1 項目である場合、ソース・コード・ページは、EBCDIC_CODEPAGE 環境変数が設定されていればそれから決定されます。 EBCDIC_CODEPAGE 環境変数が設定されていない場合、ソース・コード・ページは、「*COBOL for Linux on x86* プログラミング・ガイド」の『サポートされるロケールおよびコード・ページ』で指定されているデフォルトのコード・ページです。

戻り値は、国別文字の表現に変換された引数-*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ときは、ソース文字は置換文字に変換されます。 置換文字は以下のとおりです。

- 1 バイト文字に変換する場合は X'1A00'
- マルチバイト文字に変換する場合は X'FDFF'

例外条件は発生しません。

戻り値の長さは、引数-*1* の内容およびソース・コード・ページの特性によって異なります。

使用上の注意:

- コード・ページ名を使用すると、他の Linux ソフトウェアと整合性が出ますが、ソース・コードは Enterprise COBOL for z/OS に移植できません。
- UTF-8 を表す CCSID は 1208 です。
- UTF-16LE を表す CCSID は 1200 です。

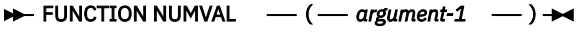
例外: 変換が失敗した場合、重大なランタイム・エラーが発生します。

NUMVAL

NUMVAL 関数は、引数として指定されている英数字文字ストリングまたは国別文字ストリングによって表される数値を戻します。 この関数は、ストリング内に先行スペースまたは後続スペースがある場合にはそれを除去し、数値を生成します。

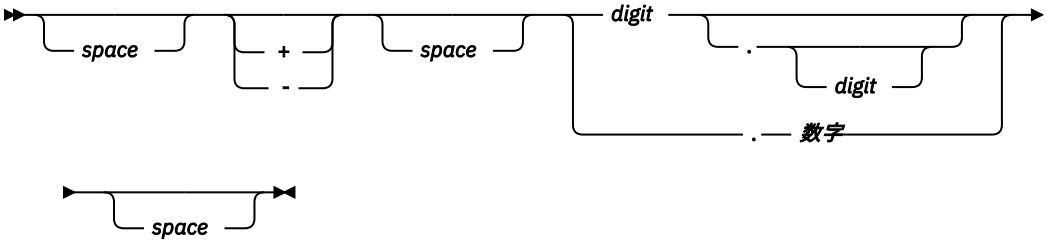
関数タイプは数字です。

フォーマット

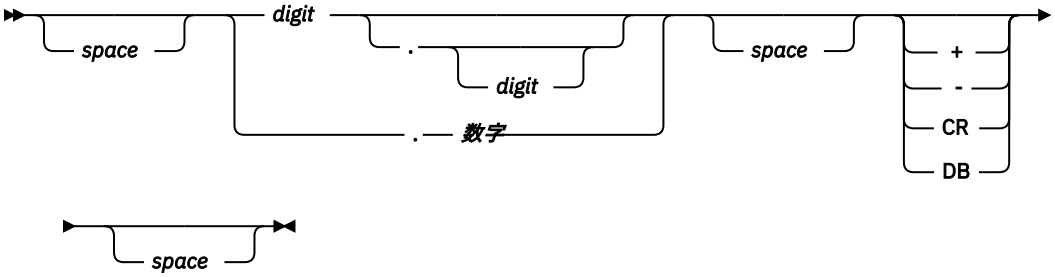


引数-*1*
英数字リテラル、国別リテラル、あるいは以下のいずれかのフォーマットの文字ストリングを含む国別クラスまたは英数字クラスのデータ項目でなければなりません。

フォーマット 1: 引数-1



フォーマット 2: 引数-1, 通貨フォーマット



スペース
1 つ以上のスペースで構成されるストリング。

数字
1 つ以上の数字で構成されるストリング。 ARITH(COMPAT) コンパイラー・オプションが有効な場合は、桁数の合計数は 18 を超えてはなりません。 ARITH(EXTEND) コンパイラー・オプションが有効な場合は、桁数の合計数は 31 を超えてはなりません。

DECIMAL-POINT IS COMMA 節が、SPECIAL-NAMES 段落の中に指定されている場合には、引数-*1* の中には小数点ではなくコンマを使用しなければなりません。

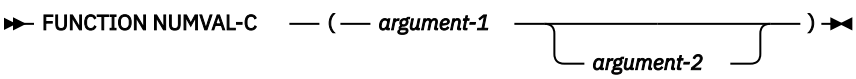
戻り値は、引数-*1* によって表される数値の浮動小数点近似値です。 戻り値の精度は、ARITH コンパイラー・オプションの設定値によって異なります。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『数値への変換 *(NUMVAL、NUMVAL-C、NUMVAL-F)*』を参照してください。

NUMVAL-C

NUMVAL-C 関数は、引数-*1* として指定されている英数字文字ストリングまたは国別文字ストリングによって表される数値を戻します。 通貨ストリング、およびグループ化された分離文字 (複数のコンマまたは複数のピリオド) がある場合には除去され、数値が生成されます。

関数タイプは数字です。

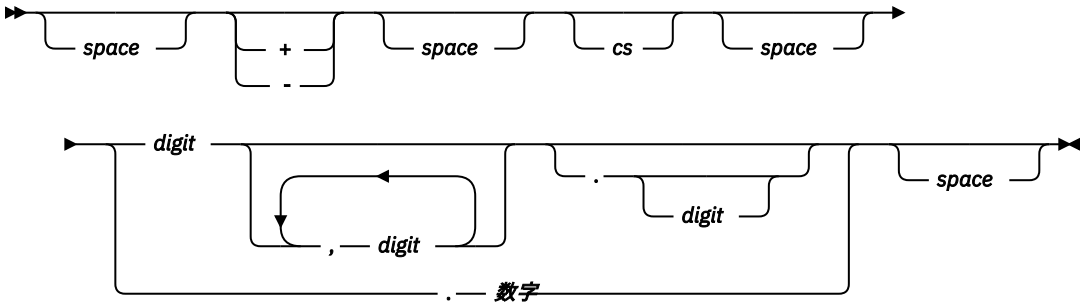
フォーマット



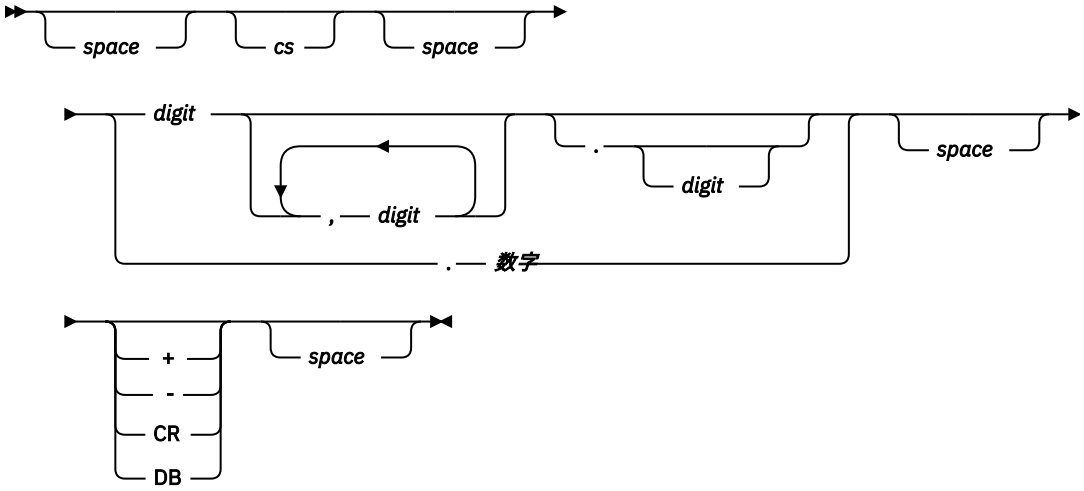
引数-*1*

英数字リテラル、国別リテラル、あるいは以下のいずれかのフォーマットの文字ストリングを含む英数字クラスまたは国別クラスのデータ項目でなければなりません。

フォーマット 1: 引数-1



フォーマット 2: 引数-1, 通貨フォーマット



スペース

1 つ以上のスペースで構成されるストリング。

cs

通貨記号を形成する、1 つ以上の文字のストリング。 *cs* によって指定された文字は、1 回に限り引数-*1* の中で行うことができます。

数字

1 つ以上の数字で構成されるストリング。 ARITH(COMPAT) コンパイラー・オプションが有効な場合は、桁数の合計数は 18 を超えてはなりません。 ARITH(EXTEND) コンパイラー・オプションが有効な場合は、桁数の合計数は 31 を超えてはなりません。

DECIMAL-POINT IS COMMA 節が、SPECIAL-NAMES 段落の中で指定されている場合には、引数-*1* の中のコンマと小数点は、その機能を交換します。

引数-*2*

通貨ストリング値を指定します。

次の規則が適用されます。

- プログラムに複数の CURRENCY SIGN 節が含まれる場合は、引数-*2* を指定する必要があります。
- 引数-*2* が指定されている場合、これは引数-*1* と同じクラスでなければなりません。
- 引数-*2* は、0 から 9 の数字、先頭や末尾のスペース、または特殊文字 '+'、'-'、'.'、または ','のいずれも含むことはできません。

- 引数-*2* は、0 を含め、引数-*2* のクラスの基本データ項目またはグループ・データ項目で有効な任意の長さにできます。
- 引数-*2* の指定は、大文字と小文字を区別します。 例えば、引数-*2* に 'CHF' と指定した場合、'ChF'、'chf' または 'chF' のいずれも一致しません。

引数-*2* を指定しないと、*cs* として使用される文字は、 プログラムで指定されている通貨記号になります。

戻り値は、引数-*1* によって表される数値の浮動小数点近似値です。 戻り値の精度は、ARITH コンパイラー・オプションの設定値によって異なります。 詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『数値への変換 *(NUMVAL、NUMVAL-C、NUMVAL-F)*』を参照してください。

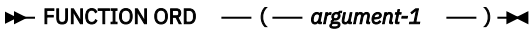
ORD

ORD 関数は、プログラム照合シーケンスの中で、この引数が持つ順序位置に等しい整数値を戻します。 最も小さな順序位置値は、1 です。

この関数のタイプは整数です。

この関数のもたらす結果は、3 桁の整数です。

フォーマット



引数-*1*
この引数は長さが 1 文字で、英字または英数字のクラスに属さなければなりません。

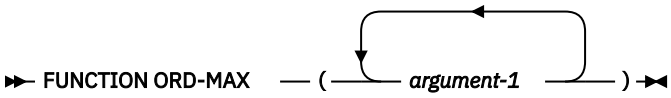
戻り値は、プログラムの照合シーケンス内の引数-*1* の順序位置です。 照合シーケンスに応じて、1 から 256 の値になります。

ORD-MAX

ORD-MAX 関数は、最大値を持つ引数の引数リスト内での順序位置に等しい値を戻します。

この関数のタイプは整数です。

フォーマット



引数-*1*
クラスの英字、英数字、国別、または数字でなければなりません。

すべての引数が同じクラスに属している必要があります。ただし、例外として英字と英数字の引数の組み合わせが許可されています。

戻り値は、一連の引数-*1* の中で最大値を持つ引数-*1* の位置に対応する序数です。

最大値を持つ引数-*1* を決定するために使用される比較は、単純条件に関する規則に従って行われます。 詳しくは、238 ページの『条件式』を参照してください。.

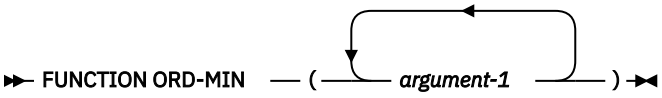
複数の引数-*1* が同一の最大値を持つ場合、その値を持つ引数-*1* のうち左端の位置に対応した引数-1 の序数が戻されます。

ORD-MIN

ORD-MIN 関数は、最小値を持つ引数の引数リスト内での順序位置に等しい値を戻します。

この関数のタイプは整数です。

フォーマット



引数-*1*
クラスの英字、英数字、国別、または数字でなければなりません。

すべての引数が同じクラスに属している必要があります。ただし、例外として英字と英数字の引数の組み合わせが許可されています。

戻り値は、一連の引数-*1* の中で最小値を持つ引数-*1* の位置に対応する序数です。

最小値を持つ引数-*1* を決定するために使用される比較は、単純条件に関する規則に従って行われます。 詳しくは、238 ページの『条件式』を参照してください。.

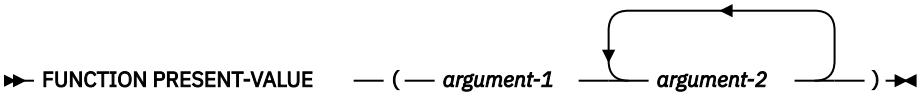
複数の引数-*1* が同一の最小値を持つ場合、その値を持つ引数-*1* のうち左端の位置に対応した引数-1 の序数が戻されます。

PRESENT-VALUE

PRESENT-VALUE 関数は、引数-*2* によって指定された将来的な期間満了時の一連の量の現在額に近似する値を、引数-*1* によって指定された割引率で戻します。

関数タイプは数字です。

フォーマット



引数-*1*
この引数のクラスは数字でなければなりません。 -1 より大きくなければなりません。

引数-*2*
この引数のクラスは数字でなければなりません。

戻り値は、次に示す形式で各期ごとに行われた一連の計算の総計の近似値です。

引数-*2* / (1 + 引数-*1*) ** *n*

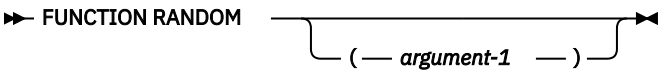
引数-*2* のオカレンスごとに 1 つの期間があります。 指数 *n* は、引数-2 の一連の指定において各期ごとに 1 から 1 ずつ増加されます。

RANDOM

RANDOM 関数は、長方形分布から疑似乱数である数値を戻します。

関数タイプは数字です。

フォーマット



引数-*1*
引数-*1* を指定する場合には、0 または正の整数でなければなりません。 ただし、ゼロから 2,147,483,645 の範囲内 (2,147,483,645 を含む) にある値のみが、明確な疑似乱数のシーケンスを発生させます。

2 回目以降の参照で引数-*1* が指定してあれば、新たな疑似乱数のシーケンスの作成が開始されます。

実行単位内においてこの関数への最初の参照が引数-*1* を指定していない場合、シード値は 0 です。

引数-*1* の指定がない以降の参照では、参照が行われるたびに、 現在の数字列の生成において次に生成される数字を戻します。

戻り値は必ず 0 と 1 の間の数字です。

ある指定されたシード値に関しては、疑似乱数のシーケンスは常に同じです。

RANDOM 関数は、スレッド化プログラムで使用できます。 初期シードの場合、RANDOM の起動時に実行しているスレッドに関係なく、疑似乱数の単一シーケンスが戻されます。

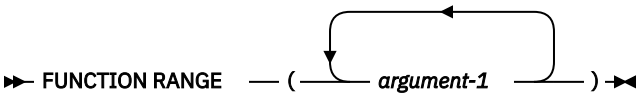
RANGE

RANGE 関数は、最大引数の値から最小引数の値を減算して得られた値を戻します。

関数のタイプは、次に示すように引数のタイプに応じて異なります。

引数のタイプ	関数のタイプ
すべての引数が整数	整数
数字 (一部の引数が整数)	数字

フォーマット



引数-*1*
この引数のクラスは数字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* を減算して得られた値に等しくなります。 最大値と最小値を決定するために使用される比較は、 単純条件に関する規則に従って行われます。 詳しくは、238 ページの『条件式』を参照してください。.

REM

REM 関数は、引数-*1* を引数-*2* で除算したその剰余に等しい値を戻します。

関数タイプは数字です。

フォーマット

引数-*1*
この引数のクラスは数字でなければなりません。

引数-*2*
この引数のクラスは数字でなければなりません。 0 にすることはできません。

戻り値は、引数-*1* を引数-*2* によって除算したその剰余の値です。 これは次の式によって定義されます。

引数-*1* - (引数-*2* * FUNCTION INTEGER-PART (引数-*1*/引数-*2*))

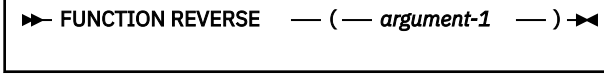
REVERSE

REVERSE 関数は、引数内に指定された文字と同一の文字を使用し、文字の順序を逆にして、引数と同一の長さの文字ストリングとして戻します。国別タイプの引数の場合は、文字位置が逆になります。

関数のタイプは、次に示すように引数のタイプに応じて異なります。

引数のタイプ	関数のタイプ
英字	英数字
英数字	英数字
国別	国別

フォーマット



引数-*1*
クラスの英字、英数字、または国別でなければならず、少なくとも 1 文字の長さを持たなければなりません。

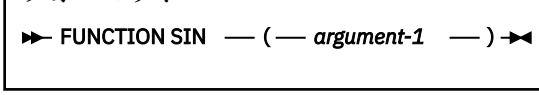
戻り値は、引数-*1* と同じ長さの文字ストリングであり、引数-*1* の文字の逆順になります。 例えば、引数-*1* に ABC が含まれている場合、戻り値は CBA になります。

SIN

SIN 関数は、引数によって指定された角度または円弧の正弦に近似する数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*
この引数のクラスは数字でなければなりません。

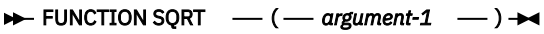
戻り値は引数-*1* の正弦の近似値で、-1 以上 +1 以下の値です。

SQRT

SQRT 関数は、指定された引数の平方根に近似する数値を戻します。

関数タイプは数字です。

フォーマット



引数-*1*

この引数のクラスは数字でなければなりません。 引数-*1* の値は、0 または正の数でなければなりません。

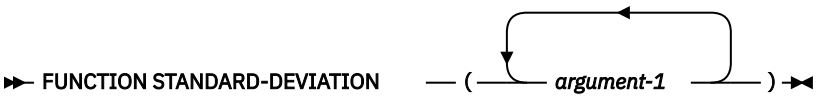
戻り値は、引数-*1* の平方根の近似値の絶対値です。

STANDARD-DEVIATION

STANDARD-DEVIATION 関数は、引数の標準偏差に近似する数値を戻します。

関数タイプは数字です。

フォーマット



引数-*1*

この引数のクラスは数字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、以下のようにして計算されます。

1. それぞれの引数-*1* と一連の引数-*1* の算術平均との差を計算し、これを二乗します。
2. 得られた値を次にすべて加算します。 この値を一連の引数-*1* の値の数で除算します。
3. 得られた商の平方根を次に計算します。 戻り値は、この平方根の絶対値です。

一連の引数-*1* が 1 つの値のみで構成される場合、または一連の引数-*1* がすべての可変オカレンス・データ項目から構成され、それらデータ項目のオカレンスの総数が 1 である場合、戻り値は 0 です。

SUBTRACT-DURATION

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。

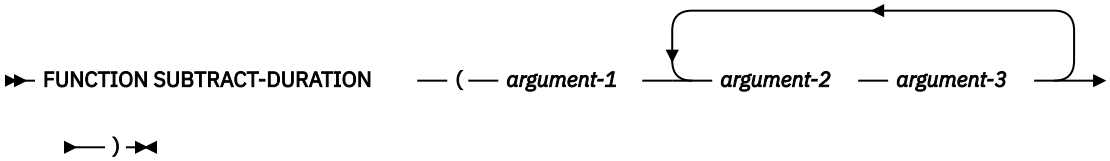
戻り値の長さは、引数-*1* に指定する日付項目、時刻項目、またはタイム・スタンプ項目の長さによって決まります。戻り値は、引数-*1* の長さまで切り捨てられます。

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 DATETIME ・コンパイラー・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、2020 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 2020 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 2020 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.
```

関連参照
434 ページの『ADD-DURATION』

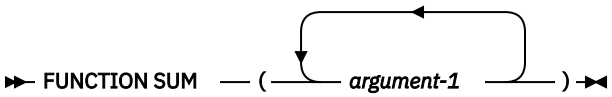
SUM

SUM 関数は、引数の合計に等しい値を戻します。

関数のタイプは、次に示すように引数のタイプに応じて異なります。

引数のタイプ	関数のタイプ
すべての引数が整数	整数
数字 (一部の引数が整数)	数字

フォーマット



引数-*1*
この引数のクラスは数字でなければなりません。

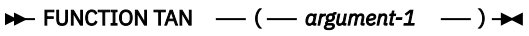
戻り値は、引数の合計です。 一連の引数-*1* がすべて整数である場合には、戻り値は整数です。 一連の引数-*1* がすべて整数とは限らない場合には数字が戻されます。

TAN

TAN 関数は、引数によって指定された角度または円弧の正接に近似する数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*
この引数のクラスは数字でなければなりません。

戻り値は、引数-*1* の正接の近似値です。

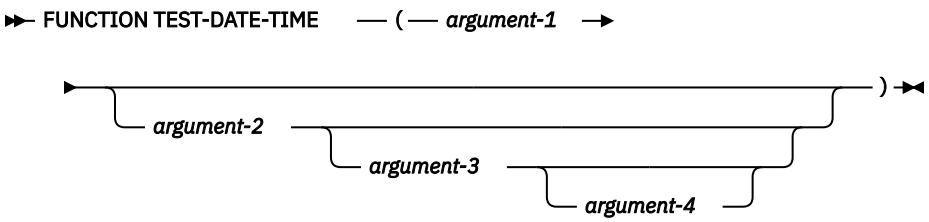
TEST-DATE-TIME

TEST-DATE-TIME 関数は、日付、時刻、タイム・スタンプ、英数字、数値パックまたは数字ゾーンの項目をとります。この関数は、それが有効な日付、時刻、またはタイム・スタンプ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*

有効な引数-*1* は以下のとおりです。

- 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

有効な引数-*2* は以下のとおりです。

- 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

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 を含んでいて、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 を含んでいて、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。

例えば、次のように指定します。

以下の例で TEST-DATE-TIME 組み込み関数の使用法を示します。

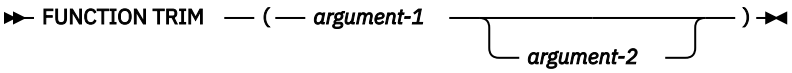
```
WORKING-STORAGE SECTION.
01 mydate1 PIC X(8) VALUE '07312013'.

PROCEDURE DIVISION.

IF FUNCTION TEST-DATE-TIME (mydate1 DATE '%d%m%Y') = B'1'
   DISPLAY 'Valid Date'
END-IF.
```

TRIM

TRIM 関数は、与えられたストリングから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から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。

フォーマット



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 (' ' or X'20')、引数-*1* がクラス DBCS である場合、2 バイト・スペース (X'2020')、または 引数-*1* がクラス国別の場合、国別スペース (X'2000' または X'0030') 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。

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しません。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。

例えば、次のように指定します。

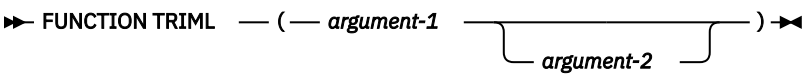
```
FUNCTION TRIM("xxxABxCxxx", "x")             // returns 'ABxC'
FUNCTION TRIMR(">>>>ABC<<<<<", "<>")         // returns '>>>>ABC'
MOVE "xyz" TO tc.
FUNCTION TRIML("xxyyzzyyzzABCxyzyxzxy", tc)  // returns 'ABCxyzyxzxy'
```

TRIML

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。

フォーマット



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、464 ページの『TRIM』を参照してください。

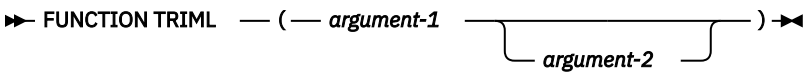
関連参照
464 ページの『TRIM』

TRIMR

TRIMR 関数は、与えられたストリングから末尾ブランクを除去して、または、与えられたストリングから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。

フォーマット



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、464 ページの『TRIM』を参照してください。

関連参照
464 ページの『TRIM』

UNDATE

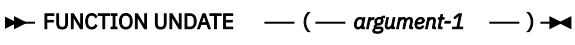
UNDATE 関数は、日付フィールドを非日付データに変換して、非日付データで使用してもあいまいとならないようにします。

NODATEPROC コンパイラー・オプションが有効であると、 UNDATE 関数は効力を持ちません。

関数のタイプは、次に示すように引数-*1* のタイプに応じて異なります。

引数のタイプ	関数のタイプ
英数字	英数字
整数	整数

フォーマット



引数-*1*
日付フィールド。

戻り値は、引数-*1* の値が未変更のまま入れられた非日付データになります。

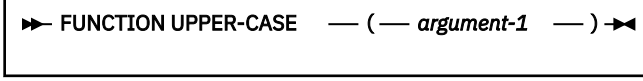
UPPER-CASE

UPPER-CASE 関数は、引数のそれぞれの小文字をそれに対応する大文字に置き換えて、引数の文字が含まれた文字ストリングを戻します。

関数のタイプは、次に示すように引数のタイプに応じて異なります。

引数のタイプ	関数のタイプ
英字	英数字
英数字	英数字
国別	国別

フォーマット



引数-*1*

英字、英数字、または国別のクラスでなければならず、少なくとも 1 文字位置の長さでなければなりません。

各小文字がそれに対応する大文字に置き換えられるだけで、引数-*1* と同じ長さの文字ストリングが戻されます。

小文字から大文字への文字の変換は、適用できるランタイム・ロケールの文字属性の仕様に基づいています。

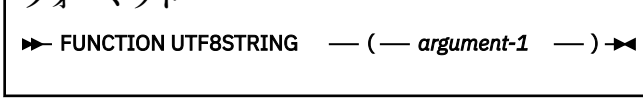
ロケールによっては、小文字から大文字への文字の変換によって、文字ストリングの長さが引数-*1* の長さと異なる場合があります。 これは、使用できるすべてのタイプの引数で起こる可能性があります。 大文字の英字 (A から Z)、小文字の英字 (a から Z)、および数字 (0 から 9) のみからなる英字および英数字引数の場合、戻される文字ストリングの長さは引数-*1* と同じになります。

UTF8STRING

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。

この関数のタイプは英数字です。

フォーマット



引数-*1*

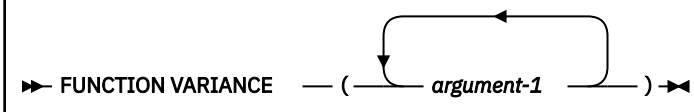
英字、英数字、DBCS、または国別文字であり、 長さは 1 文字以上でなければなりません。

VARIANCE

VARIANCE 関数は、引数の分散に近似する数値を戻します。

関数タイプは数字です。

フォーマット



引数-*1*

この引数のクラスは数字でなければなりません。

戻り値は、一連の引数-*1* の分散の近似値です。

戻り値は、一連の引数-*1* の標準偏差の二乗として定義されます。 この値は次のようにして計算されます。

1. それぞれの引数-*1* の値と一連の引数-*1* の算術平均との差を計算し、これを二乗します。
2. 得られた値を次にすべて加算します。 この値を一連の引数の数で除算します。

一連の引数-*1* が 1 つの値のみで構成される場合、または一連の引数-*1* がすべての可変オカレンス・データ項目から構成され、それらデータ項目のオカレンスの総数が 1 である場合、戻り値は 0 です。

WHEN-COMPILED

WHEN-COMPILED 関数は、プログラムがコンパイルされたシステムにより提供されるプログラムのコンパイル日時を戻します。

この関数のタイプは英数字です。

フォーマット

以下の戻り値の 21 文字は、左から右への順序で以下のように解釈します。

文字 位置	内容
1 から 4	グレゴリオ暦におけるその年を表す 4 桁の数字。
5 から 6	月を表す 2 桁の数字で、01 から 12 の範囲にある値。
7 から 8	日を表す 2 桁の数字で、01 から 31 の範囲にある値。
9 から 10	深夜午前 0 時からの時間を表す 2 桁の数字で、00 から 23 の範囲にある値。
11 から 12	分を表す 2 桁の数字で、00 から 59 の範囲にある値。
13 から 14	秒を表す 2 桁の数字で、00 から 59 の範囲にある値。
15 から 16	100 分の 1 秒を表す 2 桁の数字で、00 から 99 の範囲の値。 関数を評価するシステムが、 秒よりも細かい単位の時間を供給する機能を備えていない場合は、 値 00 が戻されます。
17	'-' または '+' のいずれかの文字。 先行する文字位置に示されている地方時がグリニッジ標準時より遅れている場合には、文字 '-' が戻されます。 地方時がグリニッジ標準時より進んでいるかまたは等しい場合には、文字 '+' が戻されます。 この関数を評価するシステムが現地時間の時差を計算して渡す機能を備えていない場合は、 文字 '0' が戻されます。
18 から 19	17 の文字位置が '-' の場合は、時間を表す 00 から 12 の範囲の 2 桁の数字が戻されます。ここに表示されるのは、グリニッジ標準時より遅れている時間数です。 17 の文字位置が '+' の場合は、 グリニッジ標準時より進んでいる時間数を示す 00 から 13 の範囲の 2 桁の数字が戻されます。 17 の文字位置が '0' の場合は、値 00 が戻されます。
20 から 21	前記の時間差に加えるべき分単位の時間を表す 00 から 59 の範囲の 2 桁の数字が戻されます。17 の文字位置が '+' か '-' かに応じて、それぞれここ表示される分の数だけグリニッジ標準時より進んでいるか、または遅れています。 17 の文字位置が '0' の場合は、値 00 が戻されます。

戻り値は、この関数を含むソース単位のコンパイル日時です。 プログラムが他のプログラムに含まれているプログラムである場合は、戻り値はこのプログラムを含むプログラムに関連付けられたコンパイル日時です。

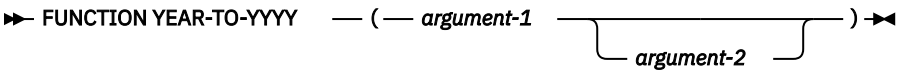
YEAR-TO-YYYY

YEAR-TO-YYYY 関数は、引数-*1* (2 桁の年) を 4 桁の年に変換します。 実行時に引数-*2* が年に追加されると、100 年間隔の終了年を定義するか、引数-*1* の年を入れるスライディング世紀ウィンドウを定義します。

この関数のタイプは整数です。

DATEPROC コンパイラー・オプションが有効な場合、戻り値は暗黙 DATE FORMAT YYYY を持つ拡張日付フィールドです。

フォーマット



引数-*1*
100 未満の負ではない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* を省略すると、関数は値 50 が指定されたものと想定して評価されます。

実行時における年と引数-*2* の値の合計は、10,000 よりも小さく、1,699 よりも大きくなければなりません。

YEAR-TO-YYYY 関数からの戻り値の例を、以下に示します。

現在の年	引数-*1* の値	引数-*2* の値	戻り値
1995	4	23	2004
1995	4	-15	1904
2008	98	23	1998
2008	98	-15	1898

YEARWINDOW

DATEPROC コンパイラー・オプションが有効な場合、YEARWINDOW 関数は、YEARWINDOW コンパイラー・オプションによって指定された世紀ウィンドウの開始年を戻します。

戻り値は拡張日付フィールドであり、暗黙の DATE FORMAT は YYYY です。

NODATEPROC コンパイラー・オプションが有効であると、YEARWINDOW 関数は 0 を戻します。

この関数のタイプは整数です。

フォーマット

第 8 部 コンパイラー指示ステートメントおよびコンパイラー指示

第 21 章 コンパイラー指示ステートメント

コンパイラー指示ステートメント とは、コンパイル時にコンパイラーに特定の処置を行わせるステートメントです。

コンパイラー指示ステートメントは、以下の目的に使用できます。

- 拡張ソース・ライブラリーの制御 (BASIS、DELETE、および INSERT ステートメント)
- ソース・テキストの操作 (COPY および REPLACE ステートメント)
- 例外処理 (USE ステートメント)
- コンパイラー・リストの制御 (*CONTROL、 *CBL、 EJECT、 TITLE、 SKIP1、 SKIP2、 および SKIP3 ステートメント)
- コンパイラー・オプションの指定 (CBL および PROCESS ステートメント)
- COBOL 例外処理プロシージャーの指定 (USE ステートメント)

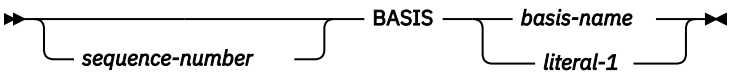
コンパイラー指示ステートメントの ENTER、READY または RESET TRACE、SERVICE LABEL、および SERVICE RELOAD は、影響を与えません。

BASIS ステートメント

BASIS ステートメントは、拡張ソース・テキスト・ライブラリー・ステートメントです。 これはコンパイル時のソースとして完全な COBOL プログラムを用意します。

完全なプログラムを、ユーザー定義のライブラリーの中に 1 つの項目として含めておき、これをコンパイルのソースとして使用できます。 コンパイラー入力は、BASIS ステートメントとオプションでそれに続けていくつかの INSERT ステートメントおよび DELETE ステート メントからなります。

フォーマット



シーケンス番号
これはオプションであり、 使用する場合には第 1 から 6 桁に記入し、後にスペースを 1 つ付けます。 このフィールドの内容は、無視されます。

BASIS
1 から 72 桁のどこにでも (固定ソース形式のとき)、または 1 から 252 桁のどこにでも (拡張ソース形式 (Extended Source Format) のとき) 置くことができます。後に基本名を続けます。このステートメントの中に他のテキストを入れないでください。

基本名、リテラル-1
システム環境は、この名前によってライブラリーの項目を認識します。

形成規則と処理規則については、476 ページの『COPY ステートメント』のリテラル-1 およびテキスト名 の説明を参照してください。

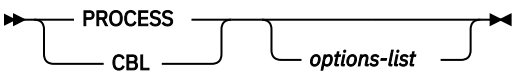
ソース・ファイルは、BASIS ステートメントの実行後も変更されません。

使用上の注意: INSERT ステートメントや DELETE ステートメントが、BASIS ステートメントによって提供される COBOL ソース・テキストを修正するために使用される場合、 COBOL ソース・テキストのシーケンス・フィールドには、シーケンス番号が昇順で入っていなければなりません。

PROCESS (CBL) ステートメント

PROCESS(CBL) ステートメントを使用すると、プログラムのコンパイル時に使用するコンパイラー・オプションを指定することができます。 PROCESS(CBL) ステートメントは、最外部のプログラムの IDENTIFICATION DIVISION ヘッダーの前に置かなければなりません。

フォーマット



オプション・リスト

一連の 1 つ以上のコンパイラー・オプション、それぞれのコンパイラー・オプションは、コンマかスペースで区切られています。

コンパイラー・オプションについて詳しくは、「*COBOL for Linux on x86* プログラミング・ガイド」の『コンパイラー・オプション』を参照してください。

PROCESS(CBL) ステートメントは、1 から 6 桁目にシーケンス番号を前に付けることができ ます。シーケンス番号の最初の文字は数字でなければなりません。 PROCESS や CBL は 8 桁目以降から始めます。シーケンス番号を指定しない場合、PROCESS や CBL は 1 桁目から始めることができます。

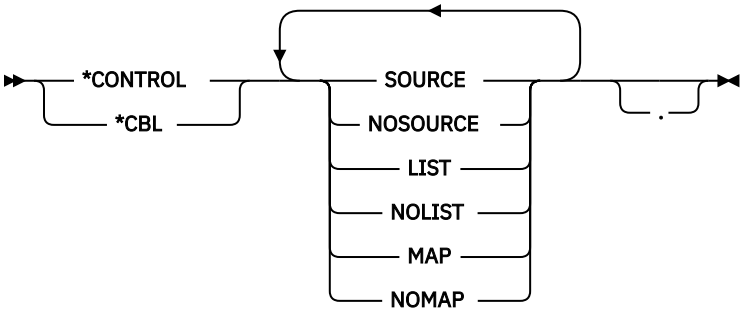
PROCESS(CBL) ステートメントは 72 桁以前に (固定ソース形式のとき)、または 252 桁以前に (拡張ソース形式 (Extended Source Format) のとき)、終わらなければなりません。また、複数の PROCESS(CBL) ステートメントに渡ってオプションを続けることはできません。 ただし、複数の PROCESS(CBL) ステートメントを使用することはできます。 複数の PROCESS(CBL) は、ステートメント間に他のタイプのステートメントを入れずに続けなければなりません。

PROCESS(CBL) ステートメントは、 コメント行や他のコンパイラー指示ステートメントがある場合には、それより前に置かなければなりません。

*CONTROL (*CBL) ステートメント

*CONTROL (または *CBL) ステートメントを使用すると、ソース・テキスト全体でソース・コードおよびストレージ・マップのリストを選択して表示または抑制することができます。

フォーマット



このオプションを指定することによって得られる出力の詳しい説明については、「*COBOL for Linux on x86* プログラミング・ガイド」の『リストの入手』を参照してください。

リストが UTF-8 でエンコードされるか、それともコンパイル時ロケールで指定されたコード・ページでエンコードされるかを指定する詳細については、*COBOL for Linux on x86* プログラミング・ガイドの *LSTFILE* 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じ意味です。 *CONTROL が受け入れられるところはどこでも、*CBL は受け入れられます。

*CONTROL または *CBL という文字は、7 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を挟んで、1 つ以上のオプションのキーワードを続けます。 オプションのキーワードは、 1 つまたは複数のスペースまたはコンマで区切らなければなりません。 このステートメントは、その行にある唯一のステートメントである必要があり、継続させることはできません。 この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。

*CONTROL ステートメントと *CBL ステートメントは、 プログラムのソース・コードの中に組み込まなければなりません。 例えば、バッチ・アプリケーションの場合は、 *CONTROL ステートメントおよび *CBL ステートメントは PROCESS (CBL) ステートメントと プログラムの終わりの間 (END PROGRAM マーカーが指定されていればその前) に置かれなければなりません。

*CONTROL (*CBL) ステートメントを含むソース・コード行は、 ソース・プログラムのリストには現れません。

固定オプションとしてインストール時に定義されているオプションがあると、その固定オプションは以下のすべてのパラメーターおよびステートメントに優先します。

- PARM (使用可能である場合)
- CBL ステートメント
- *CONTROL (*CBL) ステートメント

要求されたオプションは、次のように取り扱われます。

1. あるオプションまたはそのオプションの否定が *CONTROL ステートメントの中に複数回現れる場合、そのオプションのうち最後に指定されたものが使用されます。
2. コンパイラーに対するパラメーターとして CORRESPONDING オプションが要求されていた場合、 否定のオプショナル・ワードを指定した *CONTROL ステートメントは、 リスト出力が禁止されるソース・テキストの部分より前になければなりません。 肯定のオプショナル・ワードを指定した *CONTROL ステートメントが現れると、 リスト出力は再開されます。
3. コンパイラーに対するパラメーターとして CORRESPONDING オプションの否定が要求されていた場合、 リストは常に 禁止されます。
4. *CONTROL ステートメントは、 それが記述されているソース・プログラム内 (それに含まれるプログラムも含めて) でのみ有効です。 このステートメントは、 複数の COBOL ソース・プログラムのバッチ・コンパイルにわたって有効になることはありません。

ソース・コード・リスト

このトピックでは、入力ソース・テキスト行のリスト作成を制御するステートメントについて説明します。

以下のいずれかのステートメントを使用できます。

```
*CONTROL SOURCE        [*CBL SOURCE]
*CONTROL NOSOURCE      [*CBL NOSOURCE]
```

*CONTROL NOSOURCE ステートメントで、 かつ SOURCE がコンパイル・オプションとして要求されていた場合、 これ以降はソース・リストの印刷は抑止されます。 「ソースの印刷は抑止されました」という通知 (I-レベル) メッセージが表示されます。

オブジェクト・コード・リスト

コンパイラーは、常にオブジェクト・コード・リストを作成します。 *CONTROL (または *CBL) ステートメントの LIST オプションと NOLIST オプションは構文チェックされますが、オブジェクト・コードには何も影響しません。

ストレージ・マップ・リスト

このトピックでは、DATA DIVISION に現れるストレージ・マップ項目のリスト作成を制御するステートメントについて説明します。

以下のいずれかのステートメントを使用できます。

```
*CONTROL MAP           [*CBL MAP]
*CONTROL NOMAP         [*CBL NOMAP]
```

*CONTROL NOMAP ステートメントで、 かつ MAP がコンパイル・オプションとして要求されていた場合、ここ以降はストレージ・マップの項目のリストは抑止されます。

例えば、次のどちらかの組のステートメントを使用すると、A および B が現れないストレージ・マップのリストを作成します。

```
*CONTROL NOMAP         *CBL NOMAP
    01  A                  01  A
    02  B                  02  B
*CONTROL MAP           *CBL MAP
```

COPY ステートメント

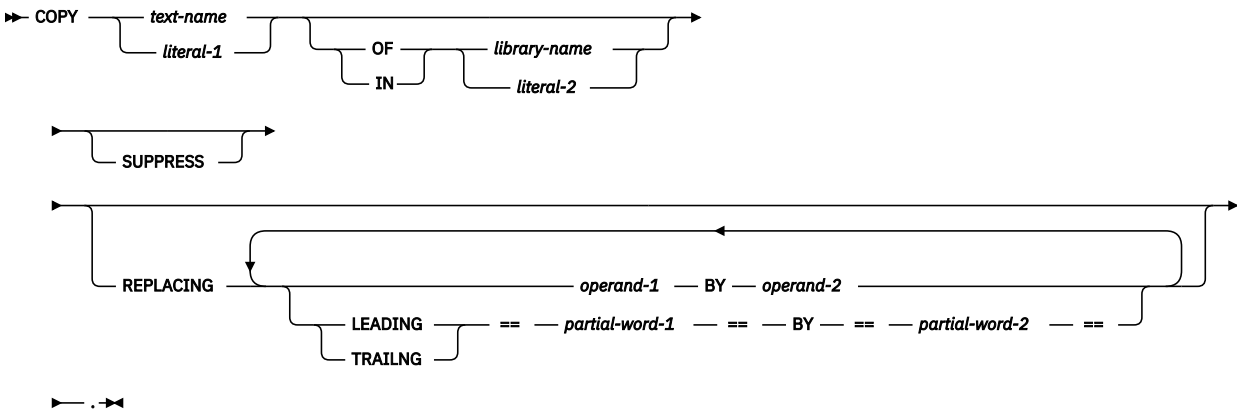
COPY ステートメントは、事前に書かれたテキストを COBOL コンパイル単位に入れるライブラリー・ステートメントです。

事前に書かれたソース・コード項目を、コンパイル時にコンパイル単位の中に含めることができます。 したがって、インストール先は、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度コーディングすることなく使用することができます。 その後、これらの項目とプロシージャーは、ユーザー作成のライブラリーに保管できます。 また、COPY ステートメントによってプログラムおよびクラス定義に組み込むこともできます。

COPY ステートメントを含むソース・コードをコンパイルするのは、すべての COPY ステートメントを処理してから、 その結果として得られるソース・テキストの処理と同じになります。

COPY ステートメントが処理されると、COPY ワードで始まりピリオドで終わる COPY ステートメント全体が論理的に 置き換えられ、テキスト名 に関連するライブラリー・テキストがコンパイル単位にコピーされます。 REPLACING 句を指定していない場合、 ライブラリー・テキストは変更されずにコピーされます。

フォーマット



テキスト名、ライブラリー名

テキスト名 はコピー・テキストを識別します。 ライブラリー名 はコピー・テキストが存在する場所を識別します。

- 1 から 30 文字の長さにできます。
- 英大文字 A から Z、英小文字 a から z、数字 0 から 9、ハイフン、および下線を使用できます。

テキスト名 もライブラリー名 もプログラム内で固有である必要はありません。 これらは、プログラム内の他のユーザー定義語と同じであっても構いません。

テキスト名 を修飾する必要はありません。 テキスト名 を修飾しない場合には、ライブラリー名は SYSLIB とみなされます。

リテラル-*1*、 リテラル-*2*
英数字リテラルでなければなりません。 リテラル-*1* はコピー・テキストを識別します。 リテラル-*2* はコピー・テキストが存在する場所を識別します。

リテラルは 1 から 160 文字の長さにできます。

テキスト名 およびライブラリー名 の固有性は、システム依存名の形成規則と変換規則が適用された後で判定されます。

テキスト名、ライブラリー名、およびリテラル内の文字のマッピングの詳細については、 *COBOL for Linux on x86* プログラミング・ガイド 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。

オペランド-*1*、オペランド-*2*
疑似テキスト、ID、関数 ID、リテラル、または COBOL ワード (COPY ワードを除く) のいずれかを指定できます。 詳しくは、478 ページの『REPLACING 句』を参照してください。

ライブラリー・テキストには、ソース・テキストに記述可能な任意のワード、ID、またはリテラルのいずれでも使用できます。これには、マルチバイト・ユーザー定義語、マルチバイト・リテラル、および国別リテラルが含まれます。

部分語-*1*、部分語-*2*
部分語を指定できます。詳しくは、478 ページの『REPLACING 句』を参照してください。

各 COPY ステートメントは、 前にスペースが 1 つなければならず、 分離文字ピリオドで終わらなければなりません。

文字ストリングまたは区切り文字が使用できるところならばどこでも、ソース・テキストの中で COPY ステートメントを使用することができます。

COPY ステートメントはネストできます。 また、ネストされた COPY ステートメントのチェーン内のどの COPY ステートメントにも、REPLACING 句を指定できます。 これは、チェーン内にそのような COPY ステートメントが 1 つだけ存在する場合に限られます。 ネストされた COPY ステートメントのチェーン内に現れる COPY ステートメントに REPLACING 句が指定されると、その REPLACING 句は、REPLACING 句のある COPY ステートメントの下にネストされた COPY ステートメントに含まれている、すべてのライブラリー・テキストに適用されます。

ネストされた COPY ステートメントが再帰することはできません。 すなわち、ある COPY メンバーに関してファイルの終わりに達するまでに、 その COPY メンバーは、 1 組のネストされた COPY ステートメントの中で一度しか指定できません。 例えば、ソース・テキストに `COPY X.` というステートメントがあり、ライブラリー・テキスト X に `COPY Y.` というステートメントがあるとします。

この場合、Y に含まれるライブラリー・テキストには、`COPY X` ステートメントも `COPY Y` ステートメントも含まれていてはなりません。

ライブラリー・テキストおよび疑似テキストには、デバッグ行を入れることができます。 デバッグ行内のテキスト・ワードは、「D」が標識域にない場合と同様に、突き合わせ規則の適用対象になります。 デバッグ行は、ソース・テキスト内の、開始疑似テキスト区切り文字の後、対応する終了疑似テキスト区切り文字の前で始まる場合に、疑似テキスト内で指定されます。

COPY ステートメントの結果としてさらに行がソース・テキストに挿入される場合、COPY ステートメントがデバッグ行上で開始されているか、挿入されるテキスト・ワードがライブラリー・テキスト内のデバッグ行にあると、挿入される各テキスト・ワードはデバッグ行上に置かれます。 BY 句の中で指定されたテキスト・ワードが挿入されるときには、 そのテキスト・ワードは、 置き換えられるその最初のライブラリー・テキスト・ワードがデバッグ行上で指定されている場合には、 デバッグ行上に置かれます。

COPY ステートメントがデバッグ行上に指定されているときには、コピーされるテキストは、 そのテキスト中のコメント行がその COPY ステートメントの実行結果として得られた ソース・テキストの中でコメント行として現れる場合を除き、デバッグ行上に入れられたものとして扱われます。

COPY ワードがコメント項目に置かれているか、 またはコメント項目を指定できるところにある場合には、その COPY ワードはコメント項目の一部とみなされます。

SOURCE-COMPUTER 段落に WITH DEBUGGING MODE 節が指定されていない場合、すべての COPY ステートメントおよび REPLACE ステートメントの処理後に、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、インライン・コメント、またはブランク行は、ライブラリー・テキスト中に置かれることがあります。 ライブラリー・テキストに置かれているコメント行、インライン・コメント、またはブランク行は、結果ソース・テキストにそのままコピーされます。ただし、ライブラリー・テキストに置かれているコメント行、インライン・コメント、またはブランク行が、オペランド-*1* に一致するテキスト・ワードの並びの中に置かれている場合、そのコメント行、インライン・コメント、またはブランク行はコピーされません (480 ページの『比較および置換の規則』 を参照)。).

*CONTROL (*CBL)、EJECT、SKIP1、SKIP2、SKIP3、または TITLE の各ステートメントを含む行を、 ライブラリー・テキストの中に記述することができます。 それらの行は、COPY ステートメントの処理中はコメント行として扱われます。

ライブラリー・テキストが構文上正しいかどうかは、別々に判定することはできないので、すべての COPY ステートメントおよび REPLACE ステートメントの処理が完全に終了するまで、 COBOL ソース・テキスト全体が構文的に正しいかどうかは判定できません。

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、 結果として得られるプログラムの中で、 そのライブラリー・テキストがライブラリーの中にあったときと同じ領域に入れられます。 ライブラリー・テキストは、85 COBOL 標準 フォーマットの規則に適合している必要があります。

注: COBOL ワードおよびセパレーター用に定義されるそれらの外側の文字は、コメント 行、インライン・コメント、コメント項目、英数字リテラル、DBCS リテラル、または国別リテラルである場合 を除いて、ライブラリー・テキストまたは疑似テキストで表示してはいけません。

SUPPRESS 句

SUPPRESS 句は、ライブラリー・テキストをソース・リストに印刷させないように指定します。

REPLACING 句

REPLACING 句を指定すると、ライブラリー・テキストがコピーされ、ライブラリー・テキスト内にあるオペランド-*1* または部分語-*1* は、それが完全に一致するたびに関連するオペランド-*2* または部分語-*2* によって置き換えられます。

以下の説明で、REPLACING 句の LEADING キーワードまたは TRAILING キーワードを指定する場合、REPLACING 句の各オペランドは部分語でなければなりません。 そうでない場合、各オペランド は、以下のいずれかの項目によって構成できます。

- 疑似テキスト
- ID
- リテラル
- COBOL ワード (COPY ワードを除く)
- 関数 ID
- 部分語

pseudo-text

疑似テキスト区切り文字 (==) によって区切られた一連の文字ストリングや 区切り文字 (疑似テキスト区切り文字は含まれません)。各疑似テキスト区切り文字の両方の文字が 1 つの行になければなりません。 ただし、疑似テキスト内の文字ストリングは継続することができます。

疑似テキスト内の個々の文字ストリングは、いずれも最大 322 文字までが可能です。 これらの文字ストリングは、ソース・コード形式の通常の継続規則に従って継続させることができます。

文字ストリングは区切り文字によって区切らなければならないことに注意してください。 詳しくは、3 ページの『第 1 章 文字』を参照してください。

疑似テキスト-*1* は、オペランド-*1* として使用されるときの疑似テキストを指し、疑似テキスト-*2* は、オペランド-*2* として使用されるときの疑似テキストを指しています。

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のみで構成することができます。 疑似テキスト-*2* は、ヌルであり、スペース文字、コメント行、またはインライン・コメントだけで構成することができます。

疑似テキストに COPY ワードを含めることはできません。

プログラムの中にコピーされる疑似テキスト-*2* の中の各テキスト・ワードは、 結果として得られるプログラムの中で、それが疑似テキスト-*2* の中にあったときと同じ領域に入れられます。

疑似テキストには、ソース・テキストに記述可能な任意のワード (COPY ワードを除く)、ID、またはリテラルのいずれでも使用できます。 これは、マルチバイト・ユーザー定義語、DBCS リテラル、および国別リテラルを含みます。

マルチバイト ユーザー定義語は、完全形式でなければなりません。 つまり、マルチバイト ワードを部分語で置き換えることはできません。

マルチバイト 文字を含むワードまたはリテラルは、行をまたいで継続することはできません。

PICTURE 文字ストリングを置き換える場合は、疑似テキストを使用します。 あいまいさを回避するため、キーワード PICTURE や PIC などを含め、PICTURE 節全体を 疑似テキスト-*1* で指定する必要があります。

ID
DATA DIVISION の任意のセクションで定義することができます。

リテラル
数字リテラル、英数字リテラル、DBCS リテラル、または国別リテラルにすることができます。

ワード
マルチバイト・ユーザー定義語を含む、任意の 1 つの COBOL ワード (COPY を除く) にすることができます。 マルチバイト・ユーザー定義語は、完全形式でなければなりません。マルチバイト ワードを部分語で置き換えることはできません。

分離文字以外の COBOL 文字 (例えば + * / $ < > =) は、REPLACING オペランドとして使用する場合は COBOL ワードの一部として組み込むことができます。 さらに、ハイフンまたは下線を語の先頭に使用することも、ハイフンを語の末尾に使用することもできます。

関数 *ID*
関数の評価からの結果であるデータ項目を一意的に参照する、文字ストリングおよび分離文字のシーケンス。 詳しくは、64 ページの『関数 ID』を参照してください。

部分語
疑似テキスト区切り文字 (==) によって区切られた単一のテキスト・ワード (疑似テキスト区切り文字を含みません)。 各疑似テキスト区切り文字の両方の文字が 1 つの行になければなりません。 ただし、部分語内のテキスト・ワードは継続することができます。

以下の規則が部分語-*1* および部分語-*2* に適用されます。

- 部分語-*1* は 1 つのテキスト・ワードから構成されます。
- 部分語-*2* はゼロまたは 1 つのテキスト・ワードから構成されます。
- 部分語-*1* および 部分語-*2* には、英数字リテラル、国別リテラル、DBCS リテラル、マルチバイト ワードのいずれも使用できません。

突き合わせのために、各 ID、リテラル、ワード、または関数 ID はそれぞれ、その ID、リテラル、ワード、または関数 ID のみを含む疑似テキストとして扱われます。

比較および置換の規則

このトピックでは、比較および置換の規則について詳しく説明します。

- 算術演算子と論理演算子は、テキスト・ワードとみなされ、疑似テキスト・オペランドによってのみ置き換えることができます。
- 先頭と末尾のブランクは、テキストの比較処理ではその対象となりません。 テキストの間に埋め込まれるブランクは、 テキストの比較処理では複数のテキスト・ワードを分離するものとして使用されます。
- オペランド-*1* が形象定数である場合、オペランド-*1* はまったく同じ形象定数とのみ一致します。 例えば、ALL "AB" がオペランド-*1* に指定された場合、ライブラリー・テキスト内の "ABAB" は一致とみなされません。 ALL "AB" のみが一致とみなされます。
- ライブラリー・テキスト内の左端のワードの前にある分離文字のコンマ、セミコロン、またはスペースはいずれも、ソース・テキストにコピーされます。 左端のライブラリー・テキスト・ワードおよび REPLACING 句で指定された最初のオペランド-*1* または部分語-*1* から始めて、キーワード BY の前にある REPLACING オペランド全体が、それと等しい個数の連続するライブラリー・テキスト・ワードと比較されます。
- オペランド-*1* を形成する順序付けられたテキスト・ワードのシーケンスが、順序付けられたライブラリー・ワードのシーケンスの 1 文字 1 文字とすべて等しい場合にのみ、オペランド-*1* はライブラリー・テキストに一致します。 国別文字の場合、国別文字のシーケンスは、順序付けられた一連のライブラリー・ワードと 1 文字 1 文字がすべて 等しくなければなりません。
- LEADING 句を指定する場合、部分語-*1* を形成する、連続する文字のシーケンスが、ライブラリー・テキスト・ワードの左端の文字位置から始まる同数の連続する文字の 1 文字 1 文字とすべて等しい場合にのみ、部分語-*1* はライブラリー・テキストに一致します。

  TRAILING 句を指定する場合、部分語-*1* を形成する、連続する文字のシーケンスが、ライブラリー・テキスト・ワードの右端の文字位置で終わる同数の連続する文字の 1 文字 1 文字とすべて等しい場合にのみ、部分語-*1* はライブラリー・テキストに一致します。
- 突き合わせでは、分離文字コンマ、分離文字セミコロン、または 1 つ以上の分離文字スペース・シーケンスの各オカレンスは、シングル・スペースと見なされます。

  ただし、オペランド-*1* または部分語-*1* が分離文字コンマまたは分離文字セミコロンのみで構成されている場合、突き合わせでは、オペランド-*1* または 部分語-*1* はテキスト・ワードとして扱われます。 この場合、コンマ分離文字またはセミコロン分離文字に続くスペースは省略可能です。

  ライブラリー・テキストに終了引用符が含まれており、その直後に分離文字スペース、分離文字コンマ、分離文字セミコロン、または分離文字ピリオドがない場合、その終了引用符は分離文字引用符と見なされます。
- 一致しない場合は、一致するものが見つかるまで、またはこれ以上 REPLACING オペランドがなくなるまで、後続のオペランド-*1* または部分語-*1* (指定されている場合) のそれぞれについて比較が繰り返されます。
- オペランド-*1* とライブラリー・テキストが一致するたびに、対応するオペランド-*2* がソース・テキストにコピーされます。 部分語-*1* とライブラリー・テキスト・ワードが一致するたびに、そのライブラリー・テキスト・ワードの一致した文字は、部分語-*2* に置き換えられるか、部分語-*2* がゼロ個のテキスト・ワードで構成されている場合は削除されます。
- すべてのオペランドが比較され、一致するものがない場合は、左端のライブラリー・テキスト・ワードがソース・テキストにコピーされます。
- ライブラリー・テキスト・ワードの処理が終了し、変更されずにソース・テキストにコピーされるか、一致したために置換されると、次に続くまだコピーされていないライブラリー・テキスト・ワードが、左端のテキスト・ワードと見なされ、オペランド-*1* または部分語-*1* が最初に現れる位置から比較サイクルが再開されます。 右端のライブラリー・テキスト・ワードが比較されるまで、処理は続行されます。
- ライブラリー・テキスト、疑似テキスト-*1*、および 部分語-*1* 内のテキスト・ワードのシーケンスは、参照形式に関する規則によって決定されます。 詳しくは、41 ページの『第 6 章 参照形式』を参照してください。
- テキスト・ワードがソース・テキスト内に配置される場合、既にスペース (ソース行間の想定スペースを含む) が存在するテキスト・ワードの間にのみ、追加のスペースが挿入されます。

- 以下の規則が、コメント行、行内コメント、およびブランク行に適用されます。
  - ライブラリー・テキスト、疑似テキスト-*1*、または部分語-*1* 内のコメント行、行内コメント、またはブランク行は、突き合わせでは無視されます。
  - ライブラリー・テキスト内のコメント行、行内コメント、またはブランク行は、結果として得られるソース・テキストに未変更のままコピーされます。 ただし、ライブラリー・テキスト内のコメント行、行内コメント、またはブランク行が、疑似テキスト-*1* または部分語-*1* に一致するテキスト・ワードのシーケンス内に現れる場合、そのコメント行、行内コメント、またはブランク行はコピーされません。
  - 疑似テキスト-*2* または部分語-*2* 内のコメント行、行内コメント、またはブランク行は、テキスト置換の結果として疑似テキスト-*2* または部分語-*2* がソース・テキスト内に配置されるときは必ず、結果として得られるプログラムに未変更のままコピーされます。
  - COPY ワードがコメント項目に置かれているか、 またはコメント項目を指定できるところにある場合には、 その COPY ワードはコメント項目の一部とみなされます。
- COPY REPLACING は、コンパイラー・ディレクティブ・ステートメント EJECT、SKIP1、SKIP2、SKIP3、または TITLE に作用しません。
- *CONTROL (*CBL)、EJECT、SKIP1、SKIP2、SKIP3、または TITLE の各ステートメントを含む行を、ライブラリー・テキスト内に記述できます。 これらの行は、結果のソース・テキストに未変更のままコピーされます。
- 以下の規則がデバッグ行に適用されます。
  - ライブラリー・テキストおよび疑似テキストには、デバッグ行を入れることができます。 デバッグ行内のテキスト・ワードは、文字「D」が標識域にない場合と同様に、突き合わせ規則の適用対象になります。 デバッグ行は、ソース・テキスト内の、開始疑似テキスト区切り文字の後、対応する終了疑似テキスト区切り文字の前で始まる場合に、疑似テキスト内で指定されます。
  - COPY ステートメントの結果としてさらに行がソース・テキストに挿入される場合、COPY ステートメントがデバッグ行上で開始されているか、挿入されるテキスト・ワードがライブラリー・テキスト内のデバッグ行にあると、挿入される各テキスト・ワードはデバッグ行上に置かれます。 BY 句で指定されたテキスト・ワードが挿入される場合、置き換えられる最初のライブラリー・テキスト・ワードがデバッグ行上で指定されていると、そのテキスト・ワードはデバッグ行上に置かれます。
  - COPY ステートメントがデバッグ行上に指定されているときには、コピーされるテキストは、 そのテキスト中のコメント行がその COPY ステートメントの実行結果として得られた ソース・テキストの中でコメント行として現れる場合を除き、デバッグ行上に入れられたものとして扱われます。
  - SOURCE-COMPUTER 段落に WITH DEBUGGING MODE 節が指定されていない場合、すべての COPY ステートメントおよび REPLACE ステートメントの処理後に、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。
- ライブラリー・テキストが構文上正しいかどうかは、別々に判定することはできないので、 すべての COPY ステートメントおよび REPLACE ステートメントの処理が完全に終了するまで、 COBOL ソース・テキスト全体が構文的に正しいかどうかは判定できません。
- (この規則は疑似テキストにのみ適用されます。) ソース・テキストで、コロンで区切られたダミー・オペランド **:**TAG**:** がプログラム・テキストにあると、コンパイラーはそのダミー・オペランドを必要なテキストに置き換えます。 例 3 に、ダミー・オペランド **:**TAG**:** の使用方法を示します。 コロンは区切り文字の役割を果たすため、TAG はスタンドアロンのオペランドになります。
- REPLACING 句を指定した COPY ステートメントを使用して、語の一部を置き換えることができます。 これは、LEADING|TRAILING *partial-word-1* BY *partial-word-2* 句、または疑似テキスト **:**TAG**:** 方式を使用して行います。
- 置換後、テキスト・ワードは、85 COBOL 標準 フォーマット規則に従ってソース・テキスト内に入れられます。

比較および置換の例

このトピックでは、比較および置換の例を示します。

複数のプログラムに共通するコードのシーケンス (ファイル記述およびデータ記述、エラー、例外ルーチンなど) は、ライブラリーに保管して、COPY ステートメントで使用することができます。 そのような共通コ

ードに関して命名規約が確立されていれば、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は、REPLACING 句を使用して、そのプログラムに意味のある名前を指定できます。

例 1

この例では、ライブラリー・テキスト PAYLIB は、次のような DATA DIVISION の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.
```

プログラムの DATA DIVISION 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.
```

このプログラム内に、ライブラリー・テキストがコピーされます。 結果のテキストは、以下のように書かれたものとして扱われます。

```
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.
```

例 2

ライブラリー・テキスト内の一部またはすべての名前を変更するには、REPLACING 句を使用することができます。

```
COPY PAYLIB REPLACING A BY PAYROLL
                      B BY PAY-CODE
                      C BY GROSS-PAY
                      D BY HOURS.
```

このプログラム内に、ライブラリー・テキストがコピーされます。 結果のテキストは、以下のように書かれたものとして扱われます。

```
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.
```

ここに示された変更は、このプログラムに対してのみ行われます。 ライブラリー内にあるテキストは未変更のままです。

例 3

ライブラリー・テキスト内で以下のような規約に従っている場合は、名前の一部 (例えばデータ名の接頭部部分) を REPLACING 句を使って変更することができます。

この例では、ライブラリー・テキスト PAYLIB は、次のような DATA DIVISION の項目から構成されています。

```
01  :TAG:.
  02  :TAG:-WEEK          PIC S99.
  02  :TAG:-GROSS-PAY     PIC S9(5)V99.
  02  :TAG:-HOURS         PIC S999  OCCURS 1 TO 52 TIMES
      DEPENDING ON :TAG:-WEEK OF :TAG:.
```

プログラムの DATA DIVISION 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 REPLACING ==:TAG:== BY ==Payroll==.
```

使用上の注意: この例では、コロンまたは括弧を区切り文字としてライブラリー・テキスト内で使用する必要があります。 括弧は、テーブル・エレメントの参照など、添え字に使用できるため、明確に区別するためにコロンの使用をお勧めします。

このプログラム内に、ライブラリー・テキストがコピーされます。 結果のテキストは、以下のように書かれたものとして扱われます。

```
01  PAYROLL.
  02  PAYROLL-WEEK        PIC S99.
  02  PAYROLL-GROSS-PAY   PIC S9(5)V99.
  02  PAYROLL-HOURS       PIC S999  OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
```

ここに示された変更は、このプログラムに対してのみ行われます。 ライブラリー内にあるテキストは未変更のままです。

例 4

この例は、PICTURE 節内の数値を置き換えることなく、レベル番号を選択的に置き換える方法を示しています。

```
COPY xxx REPLACING ==(01)== BY ==(01)==
                   == 01 == BY == 05 ==.
```

例 5

この例は、COPY ステートメント内の REPLACING 句の LEADING キーワードの使用方法を示しています。 ライブラリー・テキスト PAYLIB は、以下の DATA DIVISION 項目から構成されています。

```
01  DEPT.
  02  DEPT-WEEK       PIC S99.
  02  DEPT-GROSS-PAY  PIC S9(5)V99.
  02  DEPT-HOURS      PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON DEPT-WEEK OF DEPT.
```

プログラムの DATA DIVISION 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 REPLACING LEADING == DEPT == BY == PAYROLL ==.
```

このプログラム内に、ライブラリー・テキストがコピーされます。 結果のテキストは、以下のように書かれたものとして扱われます。

```
01  PAYROLL.
  02  PAYROLL-WEEK       PIC S99.
  02  PAYROLL-GROSS-PAY  PIC S9(5)V99.
  02  PAYROLL-HOURS      PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
```

ここに示された変更は、このプログラムに対してのみ行われます。 ライブラリー内にあるテキストは未変更のままです。

例 6

この例は、COPY ステートメント内の REPLACING 句の TRAILING キーワードの使用方法を示しています。 ライブラリー・テキスト PAYLIB は、以下の DATA DIVISION 項目から構成されています。

```
01  PAYROLL.
  02  PAYROLL-WEEK       PIC S99.
  02  PAYROLL-GROSS-PAY  PIC S9(5)V99.
```

```
  02  PAYROLL-HOURS      PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
```

プログラムの DATA DIVISION 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 REPLACING TRAILING == GROSS-PAY == BY == NET-PAY ==.
```

このプログラム内に、ライブラリー・テキストがコピーされます。 結果のテキストは、以下のように書かれたものとして扱われます。

```
01  PAYROLL.
  02  PAYROLL-WEEK     PIC S99.
  02  PAYROLL-NET-PAY  PIC S9(5)V99.
  02  PAYROLL-HOURS    PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
```

ここに示された変更は、このプログラムに対してのみ行われます。 ライブラリー内にあるテキストは未変更のままです。

例 7

この例では、単一の REPLACING 句に 2 つのタイプの部分語置換が指定されているシナリオについて説明します。 ライブラリー・テキスト PAYLIB は、以下の DATA DIVISION 項目から構成されています。

```
01  PAYROLL.
  02  PAYROLL-WEEK     PIC S99.
  02  :TAG:-GROSS-PAY  PIC S9(5)V99.
  02  PAYROLL-HOURS    PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
```

プログラムの DATA DIVISION 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 REPLACING == :TAG: == BY == PAYROLL ==
            TRAILING == GROSS-PAY == BY == NET-PAY ==.
```

このプログラム内に、ライブラリー・テキストがコピーされます。 結果のテキストは、以下のように書かれたものとして扱われます。

```
01  PAYROLL.
  02  PAYROLL-WEEK       PIC S99.
  02  PAYROLL-GROSS-PAY  PIC S9(5)V99.
  02  PAYROLL-HOURS      PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
```

2 つのタイプの部分語置換が同一の REPLACING 操作に指定されていますが、通常どおり、1 つのライブラリー・テキスト・ワードに対して 1 つの置換が行われます。 そのため、:TAG:-GROSS-PAY が、両方の置換操作の第 1 オペランドとの一致と見なされた場合でも、最初の一致および置換の実行後に、そのワードに対してさらに置換が行われることはありません。

ここに示された変更は、このプログラムに対してのみ行われます。 ライブラリー内にあるテキストは未変更のままです。

例 8

この例では、ネストされた COPY ステートメントのチェーン内の REPLACING 句のサポートについて説明します。 この例では、REPLACING 句が指定されているのは最外部の COPY ステートメントですが、チェーン内のネストされたどの COPY ステートメントに対しても REPLACING 句を指定できることに注意してください (1 つのステートメントにのみ指定する場合に限られます)。 ライブラリー・テキスト PAYLIB は、以下の DATA DIVISION 項目から構成されています。

```
01  PAYROLL.
  02  PAYROLL-WEEK     PIC S99.
  02  :TAG:-GROSS-PAY  PIC S9(5)V99.
```

```
  02  PAYROLL-HOURS    PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.
      COPY PAYLIB2
```

ライブラリー・テキスト PAYLIB2 は、以下の DATA DIVISION 項目から構成されています。

```
01  PAYROLL2.
  02  PAYROLL2-WEEK     PIC S99.
  02  :TAG:2-GROSS-PAY  PIC S9(5)V99.
  02  PAYROLL2-HOURS    PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL2-WEEK OF PAYROLL2.
```

プログラムの DATA DIVISION 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 REPLACING == :TAG: == BY == PAYROLL ==
            TRAILING == GROSS-PAY == BY == NET-PAY ==.
```

このプログラム内に、ライブラリー・テキストがコピーされます。 結果のテキストは、以下のように書かれたものとして扱われます。

```
01  PAYROLL.
  02  PAYROLL-WEEK       PIC S99.
  02  PAYROLL-NET-PAY    PIC S9(5)V99.
  02  PAYROLL-HOURS      PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL-WEEK OF PAYROLL.

01  PAYROLL2.
  02  PAYROLL2-WEEK      PIC S99.
  02  PAYROLL2-NET-PAY   PIC S9(5)V99.
  02  PAYROLL2-HOURS     PIC S999 OCCURS 1 TO 52 TIMES
      DEPENDING ON PAYROLL2-WEEK OF PAYROLL2.
```

最外部の COPY ステートメントにある REPLACING 句は、PAYLIB のライブラリー・テキストだけでなく、PAYLIB2 のテキストにも適用されます。

ここに示された変更は、このプログラムに対してのみ行われます。 ライブラリー内にあるテキストは未変更のままです。

DELETE ステートメント

DELETE ステートメントは拡張ソース・ライブラリー・ステートメントです。 このステートメントは、BASIS ステートメントによって挿入されたソース・プログラムから COBOL ステートメントを取り除きます。

フォーマット

シーケンス番号
これはオプションであり、 使用する場合には第 1 から 6 桁に記入し、後にスペースを 1 つ付けます。 このフィールドの内容は、無視されます。

DELETE
1 から 72 桁のどこにでも (固定ソース形式のとき)、または 1 から 252 桁のどこにでも (拡張ソース形式 (Extended Source Format) のとき) 置くことができます。 キーワード DELETE の後には、スペースとシーケンス番号フィールドを続ける必要があります。 このステートメントの中に他のテキストを入れないでください。

シーケンス番号フィールド
それぞれの番号は、BASIS ソース・プログラムの中にあるシーケンス番号 と一致している必要があります。 このシーケンス番号 は、プログラマーが COBOL コーディング用紙の 1 から 6 桁に記入した 6

桁の番号です。 INSERT ステートメントまたは DELETE ステートメントのシーケンス番号フィールドの中で参照する番号は、数字の昇順で指定しなければなりません。

シーケンス番号フィールド は、次のオプションのうちのいずれか 1 つでなければなりません。

- 1 つの番号
- 複数の番号
- 1 つの番号範囲 (範囲の両端の番号をハイフンによって分けて示したもの)
- 複数の番号の範囲
- 番号 (1 つ以上) と番号の範囲 (1 つ以上) の組み合わせ

シーケンス番号フィールド の中の各項目は、 コンマとその後につけるスペースによって、前の項目と区切る必要があります。 次に例を示します。

```
000250 DELETE  000010-000050, 000400, 000450
```

DELETE ステートメントの後にソース・プログラム・ステートメントを続けることができます。 それらのソース・プログラム・ステートメントは、 BASIS ソース・プログラムの中の、 削除されるステートメントの最後のものの後にあるステートメントの前に挿入されます (すなわち、上記の例では削除されるステートメント 000450 の次にあるステートメントの前に挿入されます)。

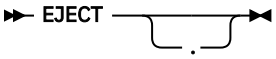
DELETE ステートメントが 12 桁目またはそれ以降の桁から指定されており、有効なシーケンス番号フィールド がキーワード DELETE の後になければ、 コンパイラーはこの DELETE ステートメントを COBOL の DELETE ステートメントと見なします。

使用上の注意: INSERT ステートメントや DELETE ステートメントが、 BASIS ステートメントによって提供される COBOL ソース・プログラムを修正するために使用される場合、 COBOL ソース・プログラムのシーケンス・フィールドには、シーケンス番号が昇順で入っていなければなりません。 ただし、ソース・ファイルは変更されないままです。 これらのシーケンス番号を参照する INSERT ステートメントや DELETE ステートメントは、 シーケンス番号の数字が昇順で順番になっていなければなりません。

EJECT ステートメント

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最上部に印刷するように指定します。

フォーマット



EJECT ステートメントは、その行にある唯一のステートメントでなければなりません。 このステートメントは、領域 A または領域 B のいずれに記述してもよく、 また分離文字ピリオドで終わることもできます。

EJECT ステートメントは、 プログラムのソース・コードの中に埋め込まなければなりません。 例えば、バッチ・アプリケーションの場合には、EJECT ステートメントは CBL (PROCESS) ステートメントとプログラムの終わり の間 (END PROGRAM マーカーが指定されていればその前) に置かなければなりません。

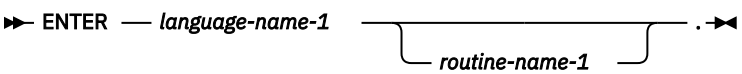
EJECT ステートメントは、ソース単位自体のコンパイルには影響しません。

ENTER ステートメント

ENTER ステートメントは、同じソース・プログラムの中で複数のソース言語の使用を容易にするように設計されています。 ただし、COBOL のみがソース・プログラムで許可されます。

ENTER ステートメントは、構文チェックを受けていますが、プログラムの実行には影響しません。

フォーマット



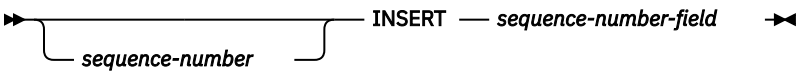
言語名-*1*
定義された意味を持たないシステム名。 形式の正しいユーザー定義語を指定するか、「COBOL」という語を指定します。 少なくとも 1 文字は英字でなければなりません。

ルーチン名-*1*
この名前は、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。

INSERT ステートメント

INSERT ステートメントは、BASIS ステートメントによって組み込まれたソース・プログラムに COBOL ステートメントを付け加えるライブラリー・ステートメントです。

フォーマット



シーケンス番号
これはオプションであり、 使用する場合には第 1 から 6 桁に記入し、後にスペースを 1 つ付けます。 このフィールドの内容は、無視されます。

INSERT
1 から 72 桁のどこにでも (固定ソース形式のとき)、または 1 から 252 桁のどこにでも (拡張ソース形式 (Extended Source Format) のとき) 置くことができます。後にスペースとシーケンス番号フィールドを続けます。 このステートメントの中に他のテキストを入れないでください。

シーケンス番号フィールド
番号は、BASIS ソース・プログラムの中にあるシーケンス番号 と一致している必要があります。 このシーケンス番号 は、プログラマーが COBOL ソース行の 1 から 6 桁に記入した 6 桁の番号です。

INSERT ステートメントまたは DELETE ステートメントのシーケンス番号フィールド の中で参照する番号は、数字の昇順で指定しなければなりません。

シーケンス番号フィールド は、例えば、000130 というような 1 つの番号でなければなりません。 シーケンス番号フィールド で指定するステートメント番号の後に挿入するために、 少なくとも 1 つの新しいソース・プログラム・ステートメントが INSERT ステートメントの後になければなりません。

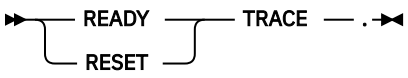
INSERT ステートメントに続く新しいソース・プログラム・ステートメントには、任意の COBOL 構文を含めることができます。

使用上の注意: INSERT ステートメントや DELETE ステートメントが、BASIS ステートメントによって提供される COBOL ソース・プログラムを修正するために使用される場合、 COBOL ソース・プログラムのシーケンス・フィールドには、シーケンス番号が昇順で入っていなければなりません。 ただし、ソース・ファイルは変更されないままです。 これらのシーケンス番号を参照する INSERT ステートメントや DELETE ステートメントは、 シーケンス番号の数字が昇順で順番になっていなければなりません。

READY TRACE ステートメントおよび RESET TRACE ステートメント

READY または RESET TRACE ステートメントは、プロシージャーの実行をトレースするように設計されています。 READY または RESET TRACE ステートメントは、PROCEDURE DIVISION のみに記述できますが、プログラムには影響しません。

フォーマット



プロシージャーの実行は、USE FOR DEBUGGING 宣言を使用してトレースすることができます。 これについては、 「*COBOL for Linux on x86* プログラミング・ガイド」の『例*: USE FOR DEBUGGING*』に説明があります。

REPLACE ステートメント

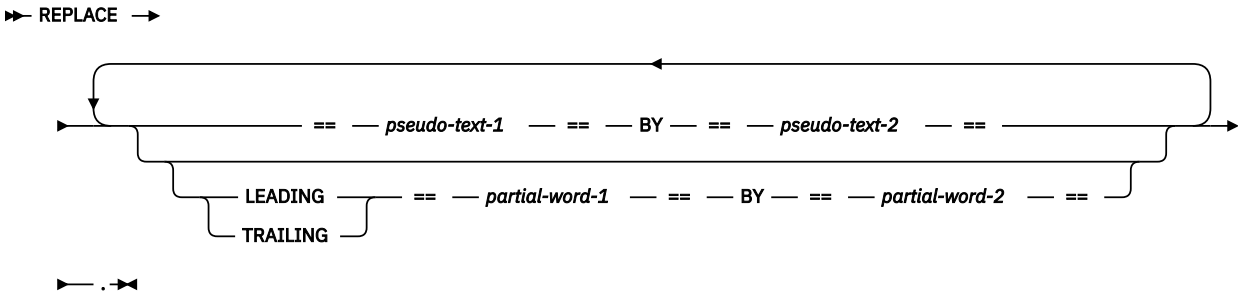
REPLACE ステートメントは、ソース・テキストのテキストを置き換えるために使用されます。

REPLACE ステートメントは、ソース・テキストの中で文字ストリングが使えるところならばどこにでも記述できます。これが別々にコンパイルされたプログラムの中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置かなければなりません。 後に分離文字ピリオドを付けなければなりません。

REPLACE ステートメントは、1 つの COBOL コンパイル・グループ全体を変更するために使用することも、またはコンパイル・グループの一部を変更するために使用することもできます。 この際、変更を必要とする場所を手作業で探して修正する必要がありません。 単純なストリングの置換が簡単な方法で行えます。これは、COPY ライブラリーにあるテキストだけを対象にするのではなく、 ソース・テキスト全体を対象に動作するという点を除けば、 COPY ステートメントに REPLACING 句を指定した場合と同じです。

ワード REPLACE がコメント項目内にある場合、またはコメント項目を指定できる場所にある場合、そのワードはコメント項目の一部と見なされます。

フォーマット 1



ソース・テキストの中で疑似テキスト*-1* に一致したテキストが現れるたびに、対応する疑似テキスト*-2* によって置き換えられます。

フォーマット 2

現在有効なテキストの置換は、フォーマット 2 の REPLACE 形式を使用すると、中断されます。 形式 2 を指定しない場合、ある 1 つの REPLACE ステートメントが作用するのは、指定された時点から、次の REPLACE ステートメントが現れるまで、または別々にコンパイルされたプログラムの終わりまでです。

疑似テキスト-*1*

これは 1 つ以上のテキスト・ワードを含まなければなりません。 文字ストリングは標準ソース・コード規則に従って継続させることができます。

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のみで構成することができます。

疑似テキスト-*2*

これはテキスト・ワードを 1 つも含まないことも、 1 つだけ含むことも、複数含むこともできます。 文字ストリングは標準ソース・コード規則に従って継続させることができます。

疑似テキスト-*1* および疑似テキスト-*2* には、ソース・テキストに記述可能な任意のテキスト・ワード (COPY ワードを除く) を使用できます。これは、国別リテラル、DBCS リテラル、および マルチバイト ユーザー定義語を含みます。

COBOL ワードおよびセパレーターに許可されるそれらの外側の文字は、コメント行、コメント項目、インライン・コメント、英数字リテラル、DBCS リテラル、または国別リテラルである場合を除いて、ライブラリー・テキストまたは疑似テキストの中にあることはできません。

マルチバイト ユーザー定義語は、完全形式でなければなりません。 つまり、マルチバイト ワードを部分語で置き換えることはできません。

疑似テキスト-*1* および疑似テキスト-*2* には、1 バイト文字および マルチバイト文字をコメント行またはコメント項目に含めることができます。

疑似テキスト内の個々の文字ストリングは、いずれも最大 322 文字長までが可能です。ただし、マルチバイト文字を含むストリングを継続することはできません。

部分語-*1*、部分語-*2*

疑似テキスト区切り文字 (==) によって区切られた単一のテキスト・ワード (疑似テキスト区切り文字を含みません)。 各疑似テキスト区切り文字の両方の文字が 1 つの行になければなりません。 ただし、部分語内のテキスト・ワードは継続することができます。

以下の規則が部分語-*1* および部分語-*2* に適用されます。

- 部分語-*1* は 1 つのテキスト・ワードから構成されます。
- 部分語-*2* はゼロまたは 1 つのテキスト・ワードから構成されます。
- 部分語-*1* および 部分語-*2* には、英数字リテラル、国別リテラル、DBCS リテラル、マルチバイト ワードのいずれも使用できません。

コンパイラーは、COPY ステートメントの指定があればそれをすべて処理してから、 ソース・テキストの REPLACE ステートメントを処理します。完全なソース・テキストを組み立てるために COPY ステートメントがまず処理されなければなりません。その後で REPLACE ステートメントを使用して単純なストリングの置換を実施し、そのソース・テキストを修正できます。REPLACE ステートメントは、それ自体に COPY ステートメントを含むことができません。

REPLACE ステートメントの処理を行った結果として得られるテキストは、REPLACE ステートメントを含むことはできません。

疑似テキストの継続規則

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、領域 A または領域 B のいずれかで始めることができます。 ただし、行の標識域にハイフンがあり、そのハイフンが疑似テキストの開始区切り文字の後に続く場合、その行の領域 A はブランクでなければなりません。 行の継続に関する通常の規則が、テキスト・ワードの形成に適用されます。 44 ページの『継続行』 を参照してください。

比較規則

テキスト置換を決定する比較演算は、REPLACE ステートメントに続く左端のソース・テキスト・ワードと、疑似テキスト-*1* または部分語-*1* の最初のワードから開始されます。

- 疑似テキスト-*1* は、 連続しているソース・プログラム・テキスト・ワードのうち同じ順序位置にあるソース・テキスト・ワードと比較されます。疑似テキスト-*1* を形成する順序付けられた一連のテキスト・

ワードが、順序付けられたソース・テキスト・ワードのシーケンスの 1 文字 1 文字とすべて等しい場合にのみ、疑似テキスト-*1* はソース・テキストに一致します。 国別文字の場合、国別文字のシーケンスは、順序付けられた一連のライブラリー・ワードと 1 文字 1 文字がすべて 等しくなければなりません。

- LEADING 句を指定する場合、部分語-*1* を形成する、連続する文字のシーケンスが、ソース・テキスト・ワードの左端の文字位置から始まる同数の連続する文字の 1 文字 1 文字とすべて等しい場合にのみ、部分語-*1* はソース・テキストに一致します。

  TRAILING 句を指定する場合、部分語-*1* を形成する、連続する文字のシーケンスが、ソース・テキスト・ワードの右端の文字位置で終わる同数の連続する文字の 1 文字 1 文字とすべて等しい場合にのみ、部分語-*1* はソース・テキストに一致します。
- 突き合わせでは、分離文字コンマ、分離文字セミコロン、または 1 つ以上の分離文字スペース・シーケンスの各オカレンスは、シングル・スペースと見なされます。

  ただし、疑似テキスト-*1* または部分語-*1* が分離文字コンマまたは分離文字セミコロンのみで構成されている場合、突き合わせでは、これらのコンマまたはセミコロンはテキスト・ワードとして扱われます。この場合、コンマ分離文字またはセミコロン分離文字に続くスペースは省略可能です。

  ソース・テキストに終了引用符が含まれており、その直後に分離文字スペース、分離文字コンマ、分離文字セミコロン、または分離文字ピリオドがない場合、その終了引用符は分離文字引用符と見なされます。
- 一致しない場合は、一致するものが見つかるまで、またはこれ以上 REPLACING オペランドがなくなるまで、後続の疑似テキスト-*1* または部分語-*1* (指定されている場合) のオカレンスのそれぞれについて比較が繰り返されます。
- すべての疑似テキスト-*1* または部分語-*1* が比較され、一致するものがない場合は、次に続くソース・テキスト・ワードが左端のソース・テキスト・ワードと見なされ、疑似テキスト-*1* または部分語-*1* が最初に現れる位置から比較サイクルが再開されます。
- 疑似テキスト-*1* とソース・テキストが一致するたびに、対応する疑似テキスト-*2* が、ソース・テキスト内の一致したテキストと置き変わります。 部分語-*1* とソース・テキスト・ワードが一致するたびに、そのソース・テキスト・ワードの一致した文字は、部分語-*2* に置き換えられるか、部分語-*2* がゼロ個のテキスト・ワードで構成されている場合は削除されます。 続いて、突き合わせが行われた右端のテキスト・ワードの直後に続くソース・テキスト・ワードが、左端のソース・テキスト・ワードと見なされます。疑似テキスト-*1* または部分語-*1* が最初に現れる位置から比較サイクルが再開されます。
- 比較演算は、REPLACE ステートメントの有効範囲内にあるソース・テキスト内の右端のテキスト・ワードが、マッチングの対象となるか、左端のソース・テキスト・ワードとみなされて完全な比較サイクルの対象となるまで続けられます。

置換規則

このトピックでは、置換の規則について詳しく説明します。

- ソース・テキスト、疑似テキスト-*1*、および 部分語-*1* 内のテキスト・ワードのシーケンスは、参照形式に関する規則によって決定されます。 詳しくは、41 ページの『第 6 章 参照形式』を参照してください。
- REPLACE ステートメントの処理結果としてソース・テキストに挿入されたテキスト・ワードは、参照形式に関する規則に従ってソース・テキストに入れられます。疑似テキスト-*2* または部分語-*2* のテキスト・ワードがソース・テキスト内に挿入される場合、スペース (ソース行間の想定スペースを含む) が既に存在するテキスト・ワードの間にのみ追加のスペースが挿入されます。
- REPLACE ステートメントの処理の結果として追加の行がソース・テキストに挿入される場合、挿入された行の標識域には、置き換えられるテキストが開始する行にある文字と同じものが入ります。ただし、その行がハイフンを持つ場合には、挿入される行はスペースを持つことになります。
- 疑似テキスト-*2* または 部分語-*2* 内のリテラルが長すぎて、結果として得られるプログラム内の別の行に継続しないと 1 行に収まらず、リテラルがデバッグ行に入れられない場合は、そのリテラルの余った部分を入れる追加の継続行が挿入されます。デバッグ行において次の行に継続しなければならないリテラルを置換する要求が生じると、 プログラムにエラーが発生します。
- 結果として得られるプログラムの中に入れられるはずの疑似テキスト-*2* または部分語-*2* の各ワードは、結果として得られるプログラムの中でも、それが疑似テキスト-*2* または部分語-*2* の中にあるときと同じ領域から開始されます。
- 以下の規則が、コメント行、行内コメント、およびブランク行に適用されます。

  - ソース・テキスト、疑似テキスト-*1*、または部分語-*1* 内のコメント行、行内コメント、またはブランク行は、突き合わせでは無視されます。
  - ソース・テキスト内のコメント行、行内コメント、またはブランク行は、結果として得られるソース・テキストに未変更のままコピーされます。 ただし、ソース・テキスト内のコメント行、行内コメント、またはブランク行が、疑似テキスト-*1* または部分語-*1* に一致するテキスト・ワードのシーケンス内に現れる場合、そのコメント行、行内コメント、またはブランク行はコピーされません。
  - 疑似テキスト-*2* または部分語-*2* 内のコメント行、行内コメント、またはブランク行は、テキスト置換の結果として疑似テキスト-*2* または部分語-*2* がソース・テキスト内に配置されるときは必ず、結果として得られるプログラムに未変更のままコピーされます。
- *CONTROL (*CBL)、EJECT、SKIP1、SKIP2、SKIP3、または TITLE の各ステートメントを含む行を、ソース・テキスト内に記述できます。 これらの行は、結果のソース・テキストに未変更のままコピーされます。
- 以下の規則がデバッグ行に適用されます。
  - 疑似テキストまたは部分語には、デバッグ行を入れることができます。デバッグ行内のテキスト・ワードは、文字「D」が標識域にない場合と同様に、突き合わせ規則の適用対象になります。
  - デバッグ行に REPLACE ステートメントが指定されている場合、このステートメントは、標識域に文字「D」がないものとして扱われます。
  - SOURCE-COMPUTER 段落に WITH DEBUGGING MODE 節が指定されていない場合、すべての COPY ステートメントおよび REPLACE ステートメントの処理後に、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。
- COPY ステートメントと REPLACE ステートメントを除き、ソース・テキストの構文が正しいかどうかは、すべての COPY ステートメントおよび REPLACE ステートメントが完全に処理されるまで判別できません。
- (この規則は疑似テキストにのみ適用されます。) ソース・テキストで、コロンで区切られたダミー・オペランド :TAG: がプログラム・テキストにあると、コンパイラーはそのダミー・オペランドを必要なテキストに置き換えます。 コロンは区切り文字の役割を果たすため、TAG はスタンドアロンのオペランドになります。

  例えば、プログラムの DATA DIVISION で、次のように REPLACE ステートメントを使用することができます。

```
REPLACE ==:TAG:== BY ==Payroll==.

01  :TAG:.
  02  :TAG:-WEEK          PIC S99.
  02  :TAG:-GROSS-PAY     PIC S9(5)V99.
  02  :TAG:-HOURS         PIC S999  OCCURS 1 TO 52 TIMES
      DEPENDING ON :TAG:-WEEK OF :TAG:.
```

- REPLACE ステートメントを使用して、語の一部を置き換えることができます。 これは、LEADING|TRAILING *partial-word-1* BY *partial-word-2* 句、または疑似テキスト :TAG: 方式を使用して行います。

SERVICE LABEL ステートメント

SERVICE LABEL ステートメントは構文チェックされますが、プログラムの実行には何も影響しません。です。

フォーマット

SERVICE LABEL ステートメントは、宣言セクションではなく PROCEDURE DIVISION にのみ記述できます。

SERVICE RELOAD ステートメント

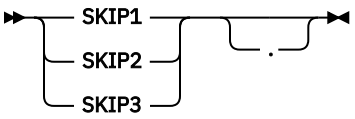
SERVICE RELOAD ステートメントは構文チェックされますが、プログラムの実行には何も影響しません。

フォーマット

SKIP ステートメント

SKIP1、SKIP2、および SKIP3 ステートメントは、ソース・リストの印刷時にコンパイラーが追加するブランク行数を指定します。 SKIP ステートメントは、ソース・テキストのコンパイル自体には影響しません。

フォーマット



SKIP1
これによってソース・プログラムのリストに挿入すべきブランク行が、 1 行であることを指定します。

SKIP2
これによってソース・プログラムのリストに挿入すべきブランク行が、 2 行であることを指定します。

SKIP3
これによってソース・プログラムのリストに挿入すべきブランク行が、 3 行であることを指定します。

SKIP1、SKIP2、または SKIP3 は、 領域 A または領域 B のどこにでも書き込むことができ、 分離文字ピリオドを付けて終了することができます。 このステートメントは、その行にある唯一のステートメントでなければなりません。

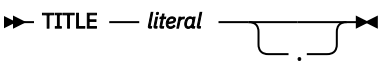
SKIP ステートメントは、プログラムのソース・コードの中に埋め込まなければなりません。 例えば、バッチ・アプリケーションの場合には、SKIP1、SKIP2、または SKIP3 ステートメントは PROCESS (CBL) ステートメントと プログラムの終了の間 (END PROGRAM マーカーが指定されていればその前) に置かなければなりません。

TITLE ステートメント

TITLE ステートメントは、コンパイル時に作成されるソース・プログラムのリストで各ページの上部に印刷されるタイトルを指定します。

TITLE ステートメントが指定されていなければ、 コンパイラーと現行リリース・レベルを識別するタイトルが生成されます。 タイトルは、タイトル行に左寄せで印刷されます。

フォーマット



リテラル
これは英数字リテラル、DBCS リテラル、または国別リテラルでなければなりません。 分離文字ピリオドを後に付けることができます。

形象定数にすることはできません。

デフォルトまたは指定したタイトルに加え、タイトル行の右側には以下の項目が入ります。

- プログラムの場合、最外部 PROGRAM-ID 段落から得られるプログラム名 (この部分は、最外部プログラムの PROGRAM-ID 段落以前のページでは、ブランクになります)。
- クラスの場合、CLASS-ID 段落から得られるクラス名
- 現行ページ番号。
- コンパイルの日付と時間

TITLE ステートメントは、次のようなことを行います。

- SOURCE コンパイラー・オプションが有効になっている場合には、 直ちに改ページされます。
- ステートメント自体はソース・リストに印刷されません。
- コンパイルに対して他の影響を及ぼしません。
- プログラムの実行に影響しません。
- 別の行に継続できません。
- どの部のどんな場所にも記述できます。

LIST オプションによって作成されるリストの各ページに対して、タイトル行を作成します。 このタイトル行は、ソース・ステートメントの中にある最後の TITLE ステートメントまたはデフォルト値を使用します。

TITLE ワードは、領域 A または領域 B のどちらからでも開始できます。

TITLE ステートメントは、クラスまたはプログラム・ソースに埋め込まなければなりません。 例えば、バッチ・アプリケーションの場合には、 TITLE ステートメントは PROCESS (CBL) ステートメントとプログラムの終了の間 (END PROGRAM マーカーが指定されていればその前) に置かなければなりません。

TITLE ステートメントと同じ行に他のステートメントを記述することはできません。

USE ステートメント

USE ステートメントは、ステートメントに続くプロシージャーを実行する条件を定義します。

USE ステートメントのフォーマットには、次のものがあります。

- EXCEPTION/ERROR 宣言
- DEBUGGING 宣言

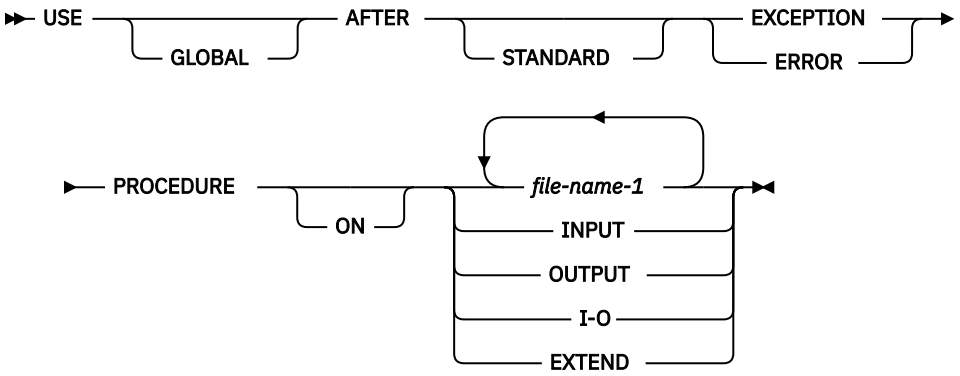
宣言の全般的な情報については、232 ページの『宣言部分』を参照してください。

EXCEPTION/ERROR 宣言

EXCEPTION/ERROR 宣言は、標準のシステム・プロシージャーの他に実行される、 入出力例外処理やエラー処理のためのプロシージャーを指定します。

EXCEPTION ワードと ERROR ワードは、 同義語でそれぞれ入れ替えて使用することができます。

フォーマット 1: EXCEPTION/ERROR 宣言の USE ステートメント



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のみプロシージャーが実行されます。 ファイル名は、ソート・ファイルやマージ・ファイルを参照することはできません。 どのファイルについても、指定できるのは、 1 つの EXCEPTION/ERROR プロシージャーだけです。 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を同時に要求することはできません。

ファイルの名前を指定した USE AFTER EXCEPTION/ERROR 宣言ステートメントは、ファイルのオープン・モードを指定する宣言ステートメントを優先します。

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た場合、 または INPUT モードでオープンされる途中でエラーを起こした場合は、 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た場合、 または OUTPUT モードでオープンされる途中でエラーを起こした場合は、 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た場合、 または I-O モードでオープンされる途中でエラーを起こした場合は、 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。

EXTEND
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、EXTEND モードでオープンされた場合、 または EXTEND モードでオープンされる途中でエラーを起こした場合は、 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 プロシージャーは、次のいずれかの場合に実行されます。

- システム定義の入出力エラー・ルーチンの実行の完了後。
- 入出力ステートメントに INVALID KEY 句または AT END 句の指定がなく、 INVALID KEY 条件または AT END 条件が検出された場合。
- ファイル状況キー 1 を 9 に設定する IBM 定義の条件が認識されたとき。(269 ページの『ファイル状況キー』を参照してください。)

EXCEPTION/ERROR プロシージャーの実行後、制御は、 入出力制御システム内の呼び出しルーチンに戻されます。 入出力状況値が重大な入出力エラーを示していない場合には、入出力制御システムは、例外条件を引き起こした実行の入出力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を戻します。

READ、WRITE、REWRITE、START、OPEN、CLOSE、 または DELETE の各ステートメントの実行中に入出力エラーが生じたときに、該当する EXCEPTION/ERROR プロシージャーがアクティブになります。 どのような条件がエラーであるかを判別するには、269 ページの『共通の処理機能』を参照してください。

宣言型プロシージャーで非宣言型プロシージャーを参照しないでください。

非宣言型プロシージャーの PERFORM ステートメントでは、宣言型プロシージャーを参照できます。それ以外は、宣言型プロシージャーを非宣言型プロシージャーから参照することはできません。

すでに呼び出されていて、まだ制御権を持っている 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も構いません。 ただし、無限ループを起こさないように、 下部に最終的な出口が確実にあるように注意してください。

EXCEPTION/ERROR プロシージャーは、入出力エラーが発生したときに、ファイル状況キーの値を調べるために使用できます。

ネストされたプログラムの優先規則

プログラムが他のプログラム内に含まれている場合は、特殊な優先順位規則に従います。

これらの規則の適用においては、最初の修飾宣言のみが実行のために選択されます。 宣言を選択するときの優先順位は、次のとおりです。

1. 修飾条件を引き起こしたステートメントを含んでいるプログラム内のファイル特定の宣言 (つまり、USE AFTER ERROR ON ファイル名-*1* 形式の宣言)。
2. 修飾条件を引き起こしたステートメントを含んでいるプログラム内のモード特定の宣言 (つまり、 USE AFTER ERROR ON INPUT 形式の宣言)。
3. GLOBAL 句を指定しており、かつ修飾条件に関して最後に調べられたプログラムを直接的に含んでいるプログラム内にあるファイル特定の宣言。
4. GLOBAL 句を指定しており、かつ修飾条件に関して最後に調べられたプログラムを直接的に含んでいるプログラム内にあるモード特定の宣言。

最後に最外部のプログラムが検査されるまで、あるいは修飾する宣言が検索できるまで、ステップ 3 およびステップ 4 を繰り返します。

DEBUGGING 宣言

デバッグ・セクションは、最外部のプログラムでのみ可能です。 ネストされているプログラム内では無効になります。 デバッグ・セクションは、 ネストされたプログラムに含まれるプロシージャーによって起動されることはありません。

デバッグ・セクションは、以下では許可されていません。

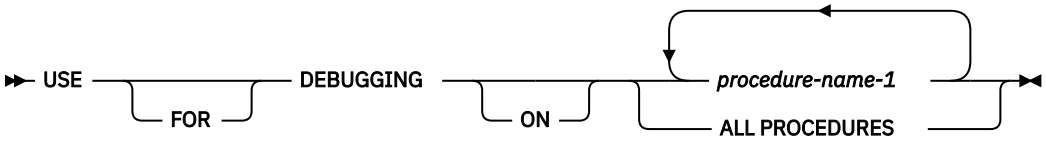
- RECURSIVE 属性で定義されたプログラム

SOURCE-COMPUTER 段落の WITH DEBUGGING MODE 節は、 コンパイルされてオブジェクト・コードに含まれているすべてのデバッグ・セクションとデバッグ行をアクティブにします。 詳しくは、543 ページの『付録 E ソース言語のデバッグ』を参照してください。

WITH DEBUGGING MODE 節を指定せずにデバッグ・モードを抑止したときは、 すべての USE FOR DEBUGGING 宣言型プロシージャーおよびすべてのデバッグ行は動作を禁止されます。

デバッグ・セクションの中にあるステートメントによって、 デバッグ・セクションの実行が自動的に引き起こされることはありません。

フォーマット 2: DEBUGGING 宣言の USE ステートメント



USE FOR DEBUGGING
すべてのデバッグ用ステートメントを 1 つのセクションにまとめて、 DECLARATIVES ヘッダーのすぐ後に書き込まなければなりません。

USE FOR DEBUGGING 文そのものを除き、デバッグ・プロシージャー内では非宣言型プロシージャーを参照することはできません。

プロシージャー名-*1*
デバッグ・セッションの中で定義することはできません。

496 ページの表 59 は、有効な各オプションについて、プログラム実行のどの時点で USE FOR DEBUGGING プロシージャーが実行されるかを示します。

いかなるプロシージャー名も、1 つの USE FOR DEBUGGING 文の中にしか現れてはならず、 その文の中で一度しか使用できません。 すべてのプロシージャーは、最外部のプログラムの中に記述しなければなりません。

ALL PROCEDURES
プロシージャー名-*1* は、USE FOR DEBUGGING 文の中で指定することはできません。 ALL PROCEDURES 句は、プログラムの中で一度だけ指定できます。 最外部プログラムに置かれたプロシージャーだけが、 デバッグ・セクションの実行を起動できます。

表 *59.* デバッグ宣言の実行

USE FOR DEBUGGING オペランド	**USE FOR DEBUGGING プロシージャーが直ちに実行される**
プロシージャー名-*1*	指定されたプロシージャーの実行前。 指定したプロシージャーを参照している ALTER ステートメントの実行後。
ALL PROCEDURES	最外部プログラム内の各非デバッグ・プロシージャーのそれぞれの実行前。 最外部プログラム内の各 ALTER ステートメント (宣言型プロシージャーの中の ALTER ステートメントは除く) の実行後。

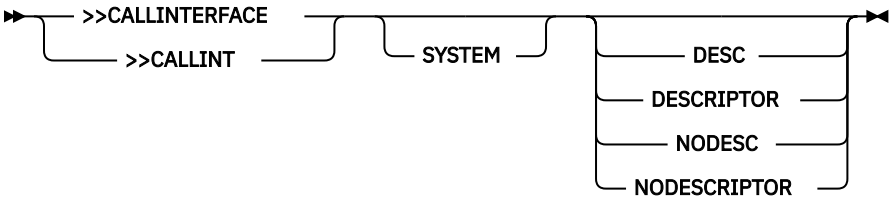
第 22 章 コンパイラー指示

コンパイラー指示とは、コンパイル時にコンパイラーに特定の処置を行わせるステートメントです。

CALLINTERFACE

CALLINTERFACE ディレクティブは、CALL のインターフェース規約を指定します。指定した規約は、ソース内で別の CALLINTERFACE ディレクティブが検出されるまで有効のままになります。

フォーマット



SYSTEM
呼び出しインターフェース規約がプラットフォームの標準システム・リンケージ規約であることを指定します。

DESC、DESCRIPTOR
引数記述子が CALL ステートメント上の各引数に渡されることを示します。

NODESC、NODESCRIPTOR
引数記述子が CALL ステートメント上のどの引数に対しても渡されないことを示します。 NODESC/NODESCRIPTOR がデフォルトです。

>>CALLINTERFACE は手続き部にのみ指定します。

COPY および REPLACE ステートメントがあればその処理の後で、CALL ステートメントの CALLINTERFACE ディレクティブに対する相対位置が認識されます。 例えば、COPY テキスト内の CALL ステートメントおよび CALLINTERFACE ディレクティブは、その CALLINTERFACE ディレクティブで指定された規則によって処理されます。

サポートされていない構文を CALLINTERFACE ディレクティブ内で指定した場合は、CALLINTERFACE ディレクティブ全体が無視されます。

構文と一般規則

- >>CALLINTERFACE は領域 B 内に単独で 1 行に指定する必要があります。
- 次の場合、>>CALLINTERFACE は指定できません。
  - COPY または REPLACE ステートメント内
  - 続いている文字ストリングの行間
  - COBOL ステートメントの中間
- >>CALLINTERFACE の指定は、現行コンパイル単位に限定されます。
- REPLACE ステートメント、および COPY ステートメントの REPLACING 句は、 CALLINTERFACE ディレクティブには影響しません。

条件付きコンパイル

条件付きコンパイルは、DEFINE ディレクティブによって指定されるリテラルの値に基づき、選択されたソース・コードの行を含めたり省略したりする方法を提供します。この方法によって、同じプログラムの複数のバリアントを作成できます。別々のソース・ストリームを維持する必要はありません。

条件付きコンパイルに使用されるコンパイラー・ディレクティブは、DEFINE ディレクティブ、EVALUATE ディレクティブ、および IF ディレクティブです。DEFINE ディレクティブは、コンパイル・グループに含められたり省略されたりするソース・コードの行を選択するために EVALUATE ディレクティブおよび IF ディレクティブで参照されるコンパイル変数を定義するために使用されます。

条件付きコンパイル・ディレクティブは以下の規則に従って処理されます。

- ソース・ファイル内で、条件付きコンパイル・ディレクティブが COPY ステートメントまたは REPLACE ステートメントの前に現れた場合、条件付きコンパイル・ディレクティブは COPY ステートメントまたは REPLACE ステートメントが処理される前に処理されます。つまり、条件付きコンパイル・ディレクティブは COPY ステートメントおよび REPLACE ステートメントをプログラムから除外するために使用できます。
- 条件付きコンパイル・ディレクティブは REPLACE ステートメントまたは COPY ステートメントの REPLACING 句の結果として実行される置き換えによる影響を受けません。
- 条件付きコンパイル・ディレクティブはコピーブックで使用できます。

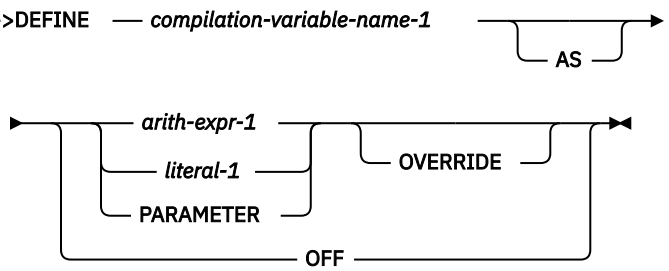
注 : 条件付きコンパイル・ディレクティブは、BASIS ステートメントが含まれるファイルに組み込むことができますが、そのようなファイルでは、そのようなファイルにソースを組み込んだりそのようなファイルからソースを除外したりすることを条件付きコンパイル・ディレクティブは制御しません。 代わりに、条件付きコンパイル・ディレクティブは、BASIS ファイルにおいて INSERT ステートメントでも DELETE ステートメントでもない他のすべてのソース行と同様に処理されます。 また、条件付きコンパイル・ディレクティブは、後からライブラリー・フェーズで処理するためにアセンブルされるソースに渡されます。

関連参照
498 ページの『DEFINE』
500 ページの『EVALUATE』
502 ページの『IF』
例: 条件付きコンパイル出力 (*COBOL for Linux on x86 プログラミング・ガイド*)

DEFINE

DEFINE ディレクティブはコンパイル変数を定義または定義解除します。 コンパイル変数はいずれかの条件付きコンパイル・ディレクティブ (DEFINE、EVALUATE、および IF) の内部で使用できます。コンパイル変数は、現在示しているリテラル値に対するシンボル参照として扱われます。

フォーマット



>>DEFINE
領域 A または B で新しい行として開始する必要があり、その行で全体を指定する必要があります。

compilation-variable-name-1
条件付きコンパイラー・ディレクティブ・キーワードと同じであってはならず、いずれかの事前定義されたコンパイル変数名であってはなりません。

DEFINE ディレクティブが OFF 句と OVERRIDE 句のいずれも指定しない場合、以下のいずれかの条件が true でなければなりません。

- 同じコンパイル・グループ内で *compilation-variable-name-1* が以前宣言されていない。
- *compilation-variable-name-1* を参照する、以前の DEFINE ディレクティブが OFF 句で指定されている必要がある。
- *compilation-variable-name-1* を参照する、以前の DEFINE ディレクティブが同じ値を指定している必要がある。

literal-1
以下の項目のいずれかにする必要があります。

- 通常の英数字リテラル ('abcd') または 16 進数リテラル (x'F1F2F3') として指定可能な英数字リテラル。 国別リテラル、DBCS リテラル、ヌル終了英数字リテラル (Z リテラル) はサポートされていません。
- 整数リテラル。
- ブール・リテラル (B'0' および B'1' のみサポート)

arith-expr-1
505 ページの『コンパイル時の算術式』に記載されている算術式規則に一致した書式にする必要があります。

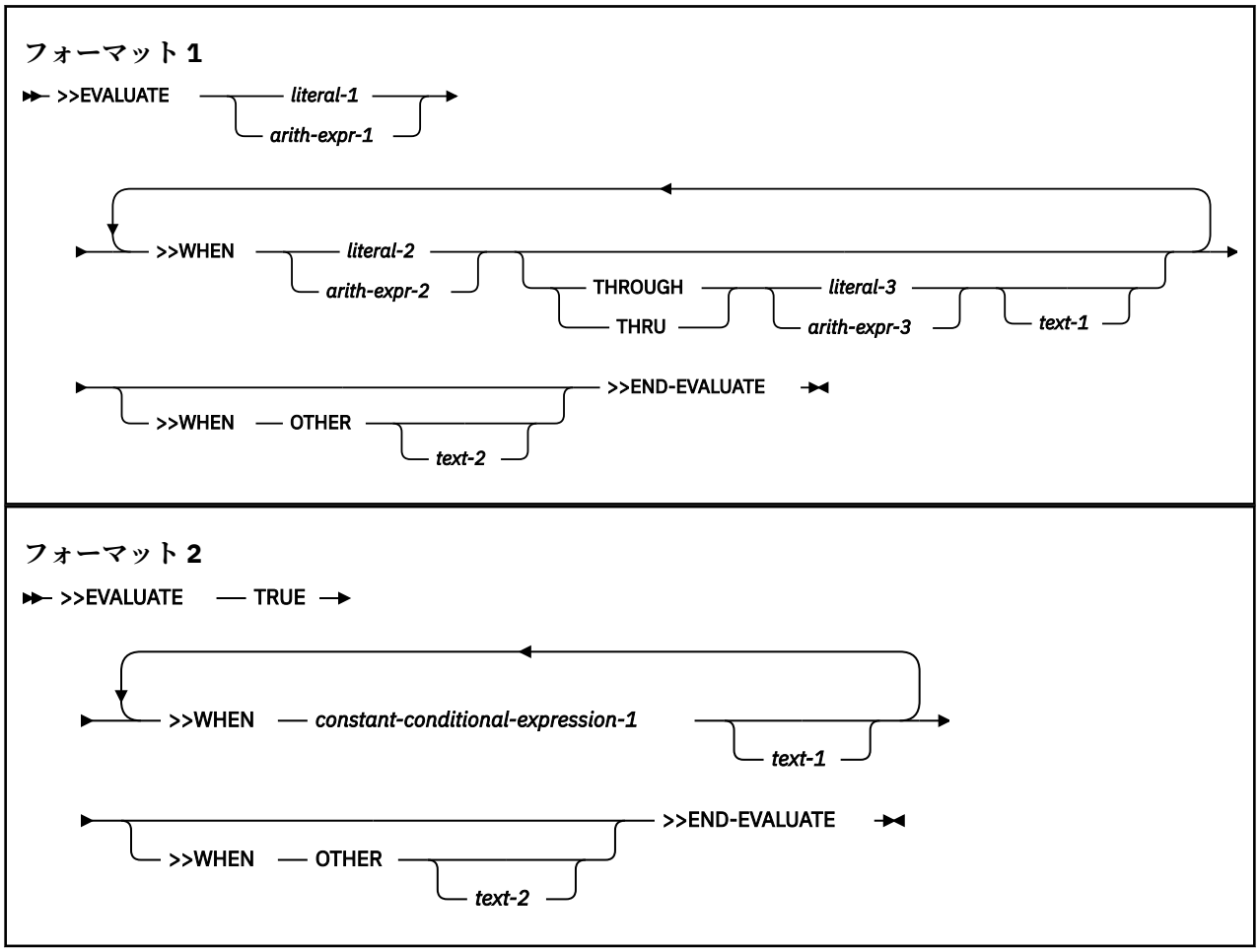
一般規則

- 他の条件付きコンパイル・ディレクティブの結果として省略されるコード内に現れる DEFINE ディレクティブは処理されません。
- *compilation-variable-name-1* を定義し、OFF 句が組み込まれていない DEFINE ディレクティブに続くテキストでは、*compilation-variable-name-1* は、定義済み条件およびブール条件に組み込まれている、名前に関連付けられたカテゴリーのリテラルが許可される場所で使用できます。
- OFF 句を使用して *compilation-variable-name-1* を指定する DEFINE ディレクティブに続くテキスト内では、後続の DEFINE 句で OFF 句を使用せずに *compilation-variable-name-1* が再定義されない限り、*compilation-variable-name-1* を定義済み条件でのみ使用できます。
- OVERRIDE 句が指定された場合、*compilation-variable-name-1* は指定されたオペランドの値を参照するように無条件に設定されます。
- AS PARAMETER 句が指定された場合、*compilation-variable-name-1* によって参照される値は、*compilation-variable-name-1* の DEFINE オプションから取得されます (そのようなオプションが存在する場合)。*compilation-variable-name-1* の DEFINE オプションがない場合、*compilation-variable-name-1* は定義されません。
- *arith-expr-1* が指定された場合、*arith-expr-1* は 505 ページの『コンパイル時の算術式』 に従って評価され、*compilation-variable-name-1* は結果の値を参照します。
- *literal-1* が指定された場合、*compilation-variable-name-1* は *literal-1* を参照します。

関連参照
504 ページの『定義済み条件』
506 ページの『事前定義されたコンパイル変数』
DEFINE (*COBOL for Linux on x86* プログラミング・ガイド)

EVALUATE

EVALUATE ディレクティブは、コンパイル・グループに組み込むソース行を選択する分岐方式を提供します。



このトピックでは説明のために以下のようにします。

- *operand-1* はフォーマット 1 では *literal-1* または *arith-expr-1* を参照し、フォーマット 2 では TRUE を参照します。
- *operand-2* はフォーマット 1 では *literal-2* または *arith-expr-2* を参照し、フォーマット 2 では *constant-conditional-expression-1* を参照します。
- *operand-3* はフォーマット 1 で *literal-3* または *arith-expr-3* を参照します。

すべてのフォーマット:

>>EVALUATE、>>WHEN、>>WHEN OTHER、>>END-EVALUATE
領域 A または B で新しい行として開始する必要があり、その行で全体を指定する必要があります。

text-1*、*text-2
これは、新しい行で開始されなければなりません。これは複数行になってもかまいません。

これは、コンパイラー・ディレクティブなど、どのような種類のソース行であってもかまいません。*text-1* または *text-2* には COPY ステートメントを含めることができます。

特定の EVALUATE ディレクティブの句は、すべて同じライブラリー・テキストまたはソース・テキストで指定する必要があります。この規則のため、*text-1* および *text-2* はその EVALUATE ディレクティブの句とはみなされません。 *text-1* または *text-2* で指定されるネストされた EVALUATE ディレクティブは新しい EVALUATE ディレクティブとみなされます。

フォーマット 1:

>>EVALUATE
1 つの EVALUATE ディレクティブのオペランドはすべて、同じカテゴリーでなければなりません。 この規則のため、算術式は数値カテゴリーです。

literal-1*、*arith-expr-1
選択サブジェクト。

literal-2*、*literal-3*、*arith-expr-2*、*arith-expr-3
選択オブジェクト。

THROUGH、THRU
ワード THROUGH および THRU は同じ意味です。THROUGH 句が指定された場合、すべての選択サブジェクトおよび選択オブジェクトは数値カテゴリーでなければなりません。

arith-expr-1*、*arith-expr-2*、*arith-expr-3
505 ページの『コンパイル時の算術式』に記載されている算術式規則に一致した書式にする必要があります。

フォーマット 2:

constant-conditional-expression-1
504 ページの『定数条件式』に記載されている定数条件式規則に一致した書式にする必要があります。

一般規則

すべてのフォーマット:

- *text-1* および *text-2* は EVALUATE コンパイラー・ディレクティブ行の一部ではありません。EVALUATE ステートメントの最初の true 分岐にある *text-1* および *text-2* は、COPY ステートメントおよび REPLACE ステートメントの突き合わせ規則および置換規則に従います。
- TRUE に評価された WHEN 句がなく、WHEN OTHER 句も見つからないまま END-EVALUATE 句に到達した場合、すべての WHEN 句に関連付けられたすべての *text-1* の行は結果のテキストから省略されます。

フォーマット 1:

- 選択サブジェクトは以下のように、各 WHEN 句に指定された値と順々に比較されます。
  - THROUGH 句が指定されない場合、選択サブジェクトが *operand-2* と等しければ TRUE の結果が返されます。
  - THROUGH 句が指定された場合、選択サブジェクトが *operand-2* 以上で、かつ *operand-3* 以下であれば TRUE の結果が返されます。
- WHEN 句が TRUE に評価された場合、その WHEN 句に関連付けられた *text-1* のすべての行が結果のテキストに含まれます。その EVALUATE ディレクティブの他の WHEN 句に関連付けられた *text-1* のすべての行と、WHEN OTHER 句に関連付けられた *text-2* のすべての行は、結果のテキストから省略されます。
- TRUE に評価される WHEN 句がない場合、WHEN OTHER 句に関連付けられた *text-2* のすべての行は、結果のテキストに含められます (WHEN OTHER 句が指定された場合)。その他の WHEN 句に関連付けられた *text-1* のすべての行は、結果のテキストから省略されます。
- *literal-1* が英数字リテラルの場合、各文字のエンコーディングのバイナリー値に基づいた、文字単位での等価かどうかの比較が使用されます。リテラルの長さが異なっていれば、等しくありません。

フォーマット 2:

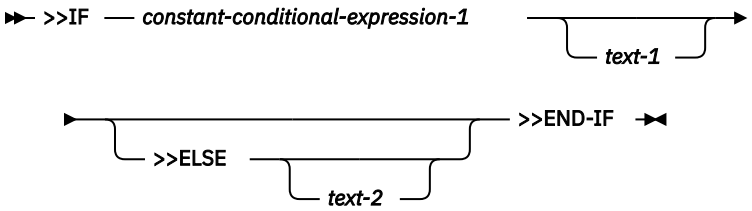
- 各 WHEN 句について順々に、504 ページの『定数条件式』の規則に従って、*constant-conditional-expression-1* が評価されます。
- WHEN 句が TRUE に評価された場合、その WHEN 句に関連付けられた *text-1* のすべての行が結果のテキストに含まれます。その EVALUATE ディレクティブの他の WHEN 句に関連付けられた *text-1* のすべての行と、WHEN OTHER 句に関連付けられた *text-2* のすべての行は、結果のテキストから省略されます。
- TRUE に評価される WHEN 句がない場合、WHEN OTHER 句に関連付けられた *text-2* のすべての行は、結果のテキストに含められます (WHEN OTHER 句が指定された場合)。その他の WHEN 句に関連付けられた *text-1* のすべての行は、結果のテキストから省略されます。

関連参照
476 ページの『COPY ステートメント』
488 ページの『REPLACE ステートメント』

IF

IF ディレクティブは、片方向または両方向条件付きコンパイルを提供します。

フォーマット



>>IF、>>ELSE、>>END-IF
領域 A または B で新しい行として開始する必要があり、その行で全体を指定する必要があります。

constant-conditional-expression-1
504 ページの『定数条件式』に記載されている定数条件式規則に一致した書式にする必要があります。

text-1*、*text-2

これは、新しい行で開始されなければなりません。これは複数行になってもかまいません。

これは、コンパイラー・ディレクティブなど、どのような種類のソース行であってもかまいません。*text-1* または *text-2* には COPY ステートメントを含めることができます。

特定の IF ディレクティブの句は、すべて同じライブラリー・テキストまたはソース・テキストで指定する必要があります。この規則のため、*text-1* および *text-2* は IF ディレクティブの句とみなされません。*text-1* または *text-2* で指定されるネストされた IF ディレクティブは新しい IF ディレクティブとみなされます。

一般規則

- *text-1* および *text-2* は IF コンパイラー・ディレクティブ行の一部ではありません。IF ディレクティブの true 分岐にあるテキスト (*text-1* または *text-2*) は COPY ステートメントおよび REPLACE ステートメントの突き合わせ規則および置換規則に従います。
- *constant-conditional-expression-1* が TRUE に評価された場合、*text-1* のすべての行が結果のテキストに含められ、*text-2* のすべての行が結果のテキストから省略されます。
- *constant-conditional-expression-1* が FALSE に評価された場合、*text-2* のすべての行が結果のテキストに含められ、*text-1* のすべての行が結果のテキストから省略されます。

関連参照
476 ページの『COPY ステートメント』
488 ページの『REPLACE ステートメント』

条件付きコンパイルの例

例 1: ブール・コンパイル変数をソースの外部からインポートしてテストする

DEFINE(DEBUG) が有効であるとします。 この場合、DEBUG はパラメーター値 B'1' のカテゴリー・ブールのコンパイル変数を指します。

```
>>DEFINE DEBUG AS PARAMETER
. . .
>>IF DEBUG IS DEFINED
```

```
     display “DEBUG: debugging mode is on”
>>END-IF
```

例 2: 数値変数値をソースの外部からインポートしてテストする

DEFINE(VAR1=10) が有効である場合:

```
>>DEFINE VAR1 AS PARAMETER
. . .
>>DEFINE VAR2 AS VAR1 + 2
. . .
>>IF VAR2 < 12
     compute x = x + 1 *> this code should NOT be included
>>ELSE
     compute x = x – 1 *> this code should be included
>>END-IF
```

例 3: フォーマット 1 の EVALUATE ディレクティブを数値コンパイル変数とともに使用する

```
>>DEFINE VAR1 AS 6
>>DEFINE VAR2 AS 1
. . .
>>EVALUATE VAR1
>>WHEN VAR2 + 2
       compute x = x + 1 *> this code should NOT be included
>>WHEN 4 THRU 8
       compute x = x – 1 *> this code should be included
>>WHEN OTHER
       compute x = x * 2 *> this code should NOT be included
>>END-EVALUATE
```

例 4: フォーマット 2 の EVALUATE ディレクティブを英数字コンパイル変数とともに使用する

```
>>DEFINE VAR1 AS 'MOO'
. . .
>>EVALUATE TRUE
>>WHEN VAR2 IS DEFINED
       compute x = x + 1 *> this code should NOT be included
>>WHEN VAR1 IS EQUAL TO 'GOO' OR VAR1 IS EQUAL TO 'MOO'
       compute x = x – 1 *> this code should be included
>>END-EVALUATE
```

例 5: DEFINE ディレクティブで OVERRIDE および OFF を使用する

```
>>DEFINE VAR AS 12
. . .
>>DEFINE VAR OFF
. . .
>>IF VAR IS DEFINED
     compute x = x + 1 *> this code should NOT be included
>>ELSE
     compute x = x - 1 *> this code should be included
>>END-IF
. . .
>>DEFINE VAR AS 16
. . .
>>DEFINE VAR AS VAR - 2 OVERRIDE
. . .
>>IF VAR IS EQUAL TO 16
     compute x = x + 1 *> this code should NOT be included
>>ELSE
     compute x = x - 1 *> this code should be included
>>END-IF
```

例 6: ブール・リテラルとコンパイル変数を汎用的に使用する

```
>>DEFINE B1 B'1' *> B1 is category boolean
>>DEFINE B2 B'0' *> B2 is category boolean
. . .
>>IF B1 AND B2
     display “Both B1 and B2 are true” *> not included
>>ELSE
  >>IF B1
       display “Only B1 is true” *> included
  >>ELSE
    >>IF B2
         display “Only B2 is true” *> not included
    >>ELSE
         display “Neither B1 nor B2 is true” *> not included
    >>END-IF
  >>END-IF
>>END-IF
```

定数条件式

定数条件式は一種の式です。この式は、結果プログラムに組み込まれるテキストを決定するために条件付きコンパイル・ディレクティブで指定されて、そのディレクティブの処理時に評価されます。

注 : このトピックでは、「リテラル」にもコンパイル変数が含まれています。 つまり、定数条件式でコンパイル変数を使用できます。

定数条件式は、以下の項目のいずれかでなければなりません。

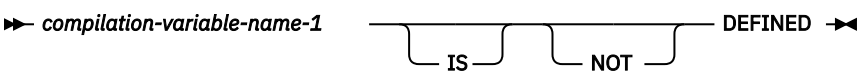
- 両方のオペランドがリテラルであるか、リテラル項のみを含む算術式である比較条件。 条件は比較条件の規則に従う必要があり、以下の追加事項があります。
  - オペランドは同じカテゴリーでなければなりません。算術式は数値カテゴリーです。
  - リテラルが指定され、それらが数値リテラルではない場合、関係演算子は “IS EQUAL TO”、“IS NOT EQUAL TO”、“IS =”、“IS NOT =”、または “IS <>” でなければなりません。
- 定義済み条件。
- ブール条件。
- AND、OR、および NOT を使用して、前述の単純条件の形式を結合することによって形成される複合条件。 簡略複合比較条件は指定してはなりません。

関連参照
242 ページの『比較条件』
504 ページの『定義済み条件』
257 ページの『簡略複合比較条件』

定義済み条件

定義済み条件式は、コンパイル変数が定義されているかどうかをテストします。

フォーマット



compilation-variable-name-1
これは条件付きコンパイラー・ディレクティブ・キーワードと同じであってはなりません。

IS DEFINED
IS DEFINED 構文を使用する定義済み条件は、*compilation-variable-name-1* が定義された場合に TRUE に評価されます。

DEFINE コンパイラー・オプションで定義されたコンパイル変数が定義済み条件で参照されるにもかかわらず、プログラム内の定義済み条件の前に、AS PARAMETER 句を持つ相応の DEFINE ディレクティ

ブも、コンパイル変数に対して OFF 句を持たない DEFINE ディレクティブもない場合、コンパイル変数の定義済み条件は FALSE に評価されます。

IS NOT DEFINED
IS NOT DEFINED 構文を使用する定義済み条件は、*compilation-variable-name-1* が定義されない場合に TRUE に評価されます。

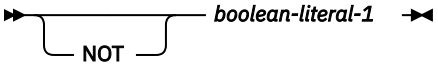
最新の定義で DEFINE ディレクティブと OFF 句を使用したコンパイル変数は、未定義とみなされます。

関連参照
506 ページの『事前定義されたコンパイル変数』
DEFINE (*COBOL for Linux on x86* プログラミング・ガイド)

ブール条件

ブール条件は、ブール・リテラルが true であるか false であるかを決定する。

フォーマット



boolean-literal-1
これは、B'1' の場合に true に評価され、B'0' の場合に false に評価されます。

条件 NOT *boolean-literal-1* は *boolean-literal-1* の真理値の逆に評価されます。

関連参照
DEFINE (*COBOL for Linux on x86* プログラミング・ガイド)

コンパイル時の算術式

コンパイル時の算術式は、DEFINE ディレクティブおよび EVALUATE ディレクティブで指定したり、定数条件式 (IF ディレクティブにあるものや、EVALUATE ディレクティブの WHEN 句にあるものなど) の一部として指定したりできます。

注 : このトピックでは、「リテラル」にもコンパイル変数が含まれています。 つまり、コンパイル時の算術式でコンパイル変数を使用できます。

コンパイル時の算術式は通常の算術式規則に従いますが、以下の例外があります。

- 指数演算子を指定することはできません。
- オペランドはすべて、整数リテラルか、すべてのオペランドが整数リテラルである演算式でなければなりません。
- 式はゼロによる除算が発生しないように指定する必要があります。
- 中間結果は「*COBOL for Linux on x86* プログラミング・ガイド」 の『固定小数点データと中間結果』 に記載されている規則に従って計算されます。 このため、コンパイル時の算術式での整数オペランドは、小数桁数が 0 の固定小数点数とみなされます。中間結果用に維持される精度の桁数を決定するときに ARITH(COMPAT|EXTEND) オプション設定が考慮されます。

関連参照
234 ページの『算術式』
ARITH (*COBOL for Linux on x86* プログラミング・ガイド)

事前定義されたコンパイル変数

コンパイラーによって自動的に定義されるコンパイル変数があります。このトピックにリストされているこれらのコンパイル変数は、コンパイル変数が許可される条件付きコンパイル・ディレクティブで参照できます。

表 *60.* 事前定義されたコンパイル変数

事前定義されたコンパイル変数名	説明	値
IGY-BINARY	バイナリー・データ項目のエンディアンネス・モードを示します (USAGE BINARY、USAGE COMP、および USAGE COMP-4)。	プログラムをコンパイルするために使用された BINARY オプションの値: • 0 は BINARY(NATIVE) を表します • 1 は BINARY(BE) を表します • 2 は BINARY(LE) を表します
IGY-CICS	これは、組み込み CICS ステートメントが受け入れられるかどうかを示します。	B'1' (CICS コンパイラー・オプションが有効になっている場合) または B'0' (それ以外の場合)。
IGY-COMPILER-VRM	コンパイラーのバージョンを示します。	VVRRMM 形式の整数で、詳しくは以下のとおりです。 • VV はバージョン番号を表します。 • RR はリリース番号を表します。 • MM はモディフィケーション番号を表します。 例えば、コンパイラー・バージョン 1.1.0 では、IGY-COMPILER-VRM 値は 010100 です。
IGY-DYNAM	CALL リテラル・ステートメントを通じて呼び出されたプログラムがランタイム時に動的にロードまたは削除されるかどうかを示します。	B'1' (DYNAM コンパイラー・オプションが有効になっている場合) または B'0' (それ以外の場合)。
IGY-FLOAT	浮動小数点データ項目の表現を示します (USAGE COMP-1 および USAGE COMP-2)。	プログラムをコンパイルするために使用された FLOAT オプションの値: • 0 は FLOAT(NATIVE) を表します • 1 は FLOAT(BE) を表します • 2 は FLOAT(LE) を表します
IGY-OPTIMIZE	最適化レベルを示します。	プログラムをコンパイルするときに使用された最適化レベル: 0、1、または 2。 0、1、または 2 は NOOPTIMIZE、OPTIMIZE(STD)、および OPTIMIZE(FULL) をそれぞれ表します。

表 *60.* 事前定義されたコンパイル変数 *(続き)*		
事前定義されたコンパイル変数名	**説明**	**値**
IGY-SQL	組み込み SQL ステートメントの処理が有効であるかどうかを示します。	B'1' (SQL コンパイラー・オプションが有効になっている場合) または B'0' (それ以外の場合)。
IGY-UTF16	国別データ項目のエンディアンネス・モードを示します (USAGE NATIONAL)。	プログラムをコンパイルするために使用された UTF16 オプションの値: • 0 は UTF16(NATIVE) を表します • 1 は UTF16(BE) を表します • 2 は UTF16(LE) を表します

関連参照
498 ページの『DEFINE』
DEFINE (*COBOL for Linux on x86* プログラミング・ガイド)

付録 A IBM 拡張

IBM 拡張とは、COBOL 標準ではなく、IBM によって定義されたフィーチャー、構文規則、または振る舞いを指します。

COBOL 標準は、565 ページの『付録 I 業界仕様』にリストされています。

509 ページの表 61 には、IBM 拡張が要旨と共に記載されています。標準の振る舞いが分かりにくい場合は、それを大括弧 [ ] で囲んで示しています。 拡張については、この文書の全編にわたって詳しく説明していますが、拡張として詳細に示されてはいません。

IBM 拡張の多くは、その構文によって標準言語と区別されています。それ以外については、コンパイラー・オプションを使用して 標準または拡張の振る舞いのいずれかを選択します。 通常、関連するコンパイラー・オプションが詳細規則に記載されています。 コンパイラー・オプションの詳細については、「*COBOL for Linux on x86* プログラミング・ガイド」の 「*COBOL for Linux on x86* プログラミング・ガイド」の『コンパイラー・オプション』 を参照してください。

項目が拡張としてリストされている場合は、すべての関連規則もまた拡張になります。 例えば、DBCS 文字用の USAGE DISPLAY-1 は拡張としてリストされています。 したがって、これをステートメントおよび節で多数使用する場合もまた拡張になります。 ただし、個別にはリストされていません。

表 *61.* ***IBM*** **拡張言語エレメント**

言語領域	**拡張エレメント**
COBOL ワード	マルチバイト文字で書かれたユーザー定義語 マルチバイト文字で書かれたシステム名 下線をユーザー定義語に含めることはできますが、先頭文字には使用できません。
国別文字サポート (Unicode サポート)	USAGE NATIONAL を使用した UTF-16 に対するサポート USAGE DISPLAY を使用した UTF-8 に対する許可 データ・カテゴリーが国別、国別編集、数字、数字編集、外部 10 進数、および外部浮動小数点の場合の USAGE NATIONAL NATIONAL 句を使用した GROUP-USAGE 節 国別リテラル (基本および 16 進数) 形象定数 SPACE、ZERO、QUOTE、HIGH-VALUES、LOW-VALUES、ALL リテラルの国別文字値 データ変換のための組み込み関数 • DISPLAY-OF • NATIONAL-OF UPPER-CASE 関数と LOWER-CASE 関数による大文字小文字の拡張マッピング

表 *61.* ***IBM* 拡張言語エレメント** *(続き)*	
言語領域	**拡張エレメント**
暗黙の項目	特殊レジスター • ADDRESS OF • LENGTH OF • RETURN-CODE • SHIFT-IN • SHIFT-OUT • SORT-CONTROL • SORT-CORE-SIZE • SORT-FILE-SIZE • SORT-MESSAGE • SORT-MODE-SIZE • SORT-RETURN • TALLY • WHEN-COMPILED • XML-CODE • XML-EVENT • XML-NTEXT • XML-TEXT
形象定数	形象定数 QUOTE の値としてアポストロフィ (') を選択 ポインター用およびオブジェクト・リファレンス用の NULL および NULLS

表 61. ***IBM* 拡張言語エレメント** *(続き)*	
言語領域	**拡張エレメント**
リテラル	区切り文字を開始および終了するときの引用符 (") の代わりとしてアポストロフィ (') を使用。 英数字リテラルにおける 1 バイト文字とマルチバイト文字の混在 (混合リテラル)。 開始区切り文字 X" および X' によって定義された、英数字リテラルに対する 16 進数表記。 開始区切り文字 Z" および Z' によって定義されたヌル終了英数字リテラル。 開始区切り文字 N"、N'、G"、および G' によって定義された DBCS リテラル。NSYMBOL(DBCS) コンパイラー・オプションが有効な場合、N" と N' は DBCS として定義されます。 連続した英数字リテラル (最初のリテラルを継続される行の列 72 で終了し、次のリテラルを継続行内で引用符を使用して開始することにより、2 つの連続した英数字リテラルをコーディングします)。 リテラルの内容を国別文字として保管するための国別リテラル N"、N'、NX"、NX'。 NSYMBOL(NATIONAL) コンパイラー・オプションが有効な場合、N" と N' は国別として定義されます。 19 (から 31) 桁の固定小数点数字リテラル。 [85 COBOL 標準 では、最大 18 桁を指定します。] 浮動小数点数字リテラル。
コメント	IDENTIFICATION DIVISION ヘッダーの前のコメント行。 マルチバイト文字を含むコメント行およびコメント項目 インライン・コメント
索引付けと添え字付け	テーブルと別のテーブルに定義された索引名との照会。 相対添え字付けにおける演算子 + または - の後の正符号付き整数リテラルの指定。
2000 年言語拡張および日付フィールド	DATE FORMAT 節 ウィンドウ表示日付フィールド、拡張日付フィールド、年末尾型日付フィールド、互換日付フィールド、日付形式、および世紀ウィンドウ。 以下の組み込み関数。 • DATEVAL • UNDATE • YEARWINDOW • DATE-TO-YYYYMMDD • DAY-TO-YYYYDDD • YEAR-TO-YYYY
参照形式	拡張ソース形式

表 *61.* ***IBM*** **拡張言語エレメント** *(続き)*	
言語領域	**拡張エレメント**
プログラムの IDENTIFICATION DIVISION	IDENTIFICATION の省略 ID RECURSIVE 節 PROGRAM-ID、AUTHOR、INSTALLATION、DATE-WRITTEN、および SECURITY 段落ヘッダーの後のオプショナル分離文字ピリオド。 [85 COBOL 標準 では、それぞれの段落ヘッダーの後にピリオドが必要です。] PROGRAM-ID 段落内のプログラム名の後のオプショナル分離文字ピリオド。 [85 COBOL 標準 では、プログラム名の後にピリオドが必要です。] PROGRAM-ID 段落内のプログラム名を示す英数字リテラル。下線を先頭文字にすることが可能。プログラム名の長さは最大 160 文字。 [85 COBOL 標準 では、プログラム名をユーザー定義語として指定する必要があります。]
終了マーカー	リテラルのプログラム名。 [85 COBOL 標準 では、プログラム名をユーザー定義語として指定する必要があります。]
SPECIAL-NAMES 段落	節のオプションの順序。 [85 COBOL 標準 では、節を構文図に示されている順にコーディングする必要があります。] 節がコーディングされていない場合に、最後の節の後にピリオドを加えるかどうかのオプション。 [85 COBOL 標準 では、節がコーディングされていない場合にもピリオドが必要です。] 複数の CURRENCY SIGN 節。 [85 COBOL 標準 では、単一の CURRENCY SIGN 節を許可します。] CURRENCY SIGN 節の WITH PICTURE SYMBOL 句 CURRENCY SIGN 節での複数文字および大/小文字混合の通貨符号 (WITH PICTURE SYMBOL 句が指定されている場合)。 [85 COBOL 標準 では 1 つの文字のみが許可され、これが通貨符号および通貨ピクチャー・シンボルになります。標準通貨符号は、以下のものであってはなりません。 • 標準ピクチャー・シンボルのいずれかと同じ文字 • 0 から 9 桁 • いずれかの特殊文字 * + - , ; ( ) " = / • スペース] 通貨符号としての小文字の英字の使用。 [85 COBOL 標準 では、大文字のみを許可します。] ロケール設定でマルチバイト・コード・ページが示されている場合に SYMBOLIC CHARACTERS 節の使用を許可しない。

表 *61.* ***IBM*** **拡張言語エレメント** *(続き)*	
言語領域	**拡張エレメント**
INPUT-OUTPUT SECTION、FILE-CONTROL 段落	「FILE-CONTROL.」のオプション (INPUT-OUTPUT SECTION が指定されていて、ファイル制御段落が指定されていない場合、およびコンパイル単位にファイルが定義されていない場合)。 [「INPUT-OUTPUT SECTION.」をコーディングした場合、85 COBOL 標準では「FILE-CONTROL.」をコーディングする必要があります。] 「FILE CONTROL.」構文が指定されていて、コンパイル単位にファイルが定義されていない場合の、ファイル制御段落のオプション。 [「INPUT-OUTPUT SECTION.」をコーディングした場合、85 COBOL 標準では FILE-CONTROL 段落をコーディングする必要があります。] ASSIGN 節の USING 句 PASSWORD 節 FILE STATUS 節の 2 番目のデータ名 ALTERNATE RECORD KEY 節の RECORD のオプション。[85 COBOL 標準 では、ワード RECORD が必要です。] 実行での RESERVE 節の欠落による影響 数字、数字編集、英数字編集、英字、内部浮動小数点、外部浮動小数点、国別、国別編集、または DBCS の基本項目あるいは代替レコード・キー・データ項目。 [85 COBOL 標準 では、キーが英数字でなければなりません。] 可変長レコードを含む索引ファイル用の最小レコード・サイズの範囲外で定義された基本または代替レコード・キー。 [85 COBOL 標準 では、基本および代替レコード・キーを最小レコード・サイズ内に収めることが必要です。] 降順でのレコードへの順次アクセス。 [85 COBOL 標準 では昇順アクセスのみが可能です。] A numeric data item of usage DISPLAY or NATIONAL in the FILE STATUS clause. [85 COBOL 標準 では、英数字ファイル状況データ項目が必要です。] ORGANIZATION IS LINE SEQUENTIAL 節およびおよび行順次ファイル制御形式 PADDING CHARACTER 節の国別リテラル
INPUT-OUTPUT SECTION、I-O-CONTROL 段落	APPLY WRITE-ONLY 節 順次、索引付き、およびソート・マージ形式の I-O-control 項目の SAME 節に 1 つのファイル名のみを指定。[85 COBOL 標準 では、最低でも 2 つのファイル名が必要です。] RERUN 節のキーワード ON のオプション。 [85 COBOL 標準 は、ON をコーディングする必要があります。] 行順次形式の I-O-control 項目 ソート・マージ I-O-control 項目の RERUN 節

表 61. ***IBM* 拡張言語エレメント** *(続き)*	
言語領域	**拡張エレメント**
DATA DIVISION	LOCAL-STORAGE SECTION LINKAGE SECTION の GLOBAL 節 データ記述項目の同じ階層レベルにある他のレベル番号よりも低いレベル番号の指定。 [85 COBOL 標準 では、階層の同じレベルにあるすべての基本項目またはグループ項目には同一のレベル番号を割り当てる必要があります。] 内部浮動小数点、外部浮動小数点、DBCS、国別、および国別編集のデータ・カテゴリー USAGE NATIONAL の数字データ・カテゴリー USAGE NATIONAL の数字編集データ・カテゴリー
FILE SECTION	LABEL RECORDS 節のデータ名 (ユーザー・ラベルの指定用) RECORDING MODE 節 行順次形式ファイル記述項目
ソート/マージ・ファイル記述項目	以下の節: • BLOCK CONTAINS • LABEL RECORDS • VALUE OF • LINAGE • CODE-SET • WITH FOOTING • LINES AT
VALUE OF 節	VALUE 節がないと、SD の元で指定された場合の実行に影響します。
DATA RECORDS 節	指定したデータ名 に対する 01 レコード記述項目のオプション。 [85 COBOL 標準 では、同じデータ名 を持つ 01 レコードを指定する必要があります。]
LINAGE 節	EXTEND モードでオープンされたファイルに対する LINAGE の指定
データ記述項目	DATE-FORMAT 節
BLANK WHEN ZERO 節	ZERO に対する代替スペル ZEROS および ZEROES
GLOBAL 節	LINKAGE SECTION での GLOBAL の指定
INDEXED BY 句	固有でない非参照索引名

表 *61.* ***IBM*** **拡張言語エレメント** *(続き)*	
言語領域	**拡張エレメント**
OCCURS 節	可変長テーブルに対する "*integer-1* TO" の省略 複合 OCCURS DEPENDING ON。 [85 COBOL 標準 では、OCCURS DEPENDING ON を含む項目の後ろには従属項目のみが続き、OCCURS DEPENDING ON を含む項目が OCCURS DEPENDING ON を含む項目に従属しないようにする必要があります。] キー名が固有でない場合の、修飾子なしで指定されたキーの暗黙の修飾 INDEXED BY 句の指定がないテーブルの索引付けによる参照 ASCENDING/DESCENDING KEY 句の、 USAGE COMPUTATIONAL-1、 COMPUTATIONAL-2、 COMPUTATIONAL-3、 COMPUTATIONAL-4、 および COMPUTATIONAL-5 のキー 参照されない固有でない索引名の受け入れ
PICTURE 節	31 から 50 までの文字を含む PICTURE 文字ストリング。 [85 COBOL 標準 では、文字の長さは 30 文字までです。] ピクチャー・シンボル G および N ピクチャー・シンボル E および外部浮動小数点ピクチャー・フォーマット PICTURE 節がデータ記述項目の最後の節ではない場合の、 直後に分離文字コンマまたは分離文字セミコロンが付く末尾コンマ挿入文字または末尾ピリオド挿入文字のコーディング [85 COBOL 標準 では、コンマまたはピリオドで終了するピクチャーを含む PICTURE 節は項目内の最後の節でなければならず、その直後に分離文字ピリオドを付ける必要があります。] CURRENCY コンパイラー・オプションを使用した通貨符号および通貨記号の選択 大文字小文字が区別される通貨記号 USAGE DISPLAY および PACKED-DECIMAL の数値項目および USAGE DISPLAY の数字編集項目に対する最大 31 桁の指定 USAGE を BINARY、COMPUTATIONAL、または COMPUTATIONAL-4 で記述されたデータ項目の値に対する TRUNC コンパイラー・オプションの影響
REDEFINES 節	再定義データ項目の REDEFINES の指定 従属レベルでの、再定義データ項目よりサイズが大きいデータ項目の再定義の指定
SYNCHRONIZED 節	レベル 01 項目に対する SYNCHRONIZED の指定

表 61. ***IBM* 拡張言語エレメント** *(続き)*	
言語領域	**拡張エレメント**
USAGE 節	以下の句: • NATIVE • COMP-1 および COMPUTATIONAL-1 • COMP-2 および COMPUTATIONAL-2 • COMP-3 および COMPUTATIONAL-3 • COMP-4 および COMPUTATIONAL-4 • COMP-5 および COMPUTATIONAL-5 • DISPLAY-1 • NATIONAL • POINTER • PROCEDURE-POINTER • FUNCTION-POINTER USAGE INDEX の項目に対する SYNCHRONIZED 節の使用
条件名項目の VALUE 節	ファイルおよび LINKAGE SECTION における、条件名項目以外の VALUE 節 DISPLAY 以外の USAGE を持つグループでの条件名項目に対する VALUE 節 ロケール定義照合シーケンスを使用して THROUGH 句に値の範囲を指定 VALUE IS NULL および VALUE IS NULLS
手続き部	セクション名の省略 セクション名が省略されている場合の段落名の省略 PROCEDURE DIVISION のヘッダーで USING 句を使用しない LINKAGE SECTION のデータ項目の参照 (それらのデータ名 が ADDRESS OF 句または ADDRESS OF 特殊レジスターのオペランドである場合) 以下のステートメント: • ENTRY • GOBACK • XML PARSE • XML GENERATE
PROCEDURE DIVISION ヘッダー	BY VALUE 句 RETURNING 句 データ項目の記述に REDEFINES 節がある場合の USING 句でのデータ項目の指定 USING 句内での所定のデータ項目の 複数インスタンスの指定
宣言型プロシージャー	アクティブ宣言の実行

表 61. ***IBM* 拡張言語エレメント** *(続き)*	
言語領域	**拡張エレメント**
プロシージャー	優先順位番号 を正の符号付き数字リテラルとして指定。 [85 COBOL 標準 では、符号なし整数でなければなりません。] 宣言の後、または宣言がない場合のセクション・ヘッダーの省略。 [85 COBOL 標準では、「DECLARATIVES.」構文および「END DECLARATIVES.」構文の後にセクション・ヘッダーが必要です。] 宣言がない場合の初期段落名 の省略。 [85 COBOL 標準 では、以下の場合に段落名が必要です。 • 宣言型プロシージャーにステートメントがある場合は、USE ステートメントの後 • 宣言型プロシージャーの外部のセクション・ヘッダーの後ろ • 宣言がない場合は、プロシージャー・ステートメント (ある場合) の前 また、85 COBOL 標準 ではプロシージャー・ステートメントを段落内に含めることが必要です。] セクション内に含まれていない段落の指定 (セクション内に含まれている段落がある場合でも指定できる)。 [85 COBOL 標準 では、宣言がない場合を除いて、段落はセクション内にある必要があります。85 COBOL 標準 では、すべての段落をセクション内に含めるか、またはまったく含めない必要があります。]
条件式	DBCS および KANJI クラス条件 NUMERIC クラス・テストにおける USAGE COMPUTATIONAL-3 または USAGE PACKED-DECIMAL データ項目の指定
比較条件	英数字、DBCS、または国別リテラルを括弧で囲む データ・ポインター・フォーマット、プロシージャー・ポインター、および関数ポインター・フォーマット 索引名の演算式との比較 簡略複合比較条件内の括弧の使用
CORRESPONDING 句	充てん項目に従属する ID の指定
INVALID KEY 句	INVALID KEY 句および適当な EXCEPTION/ERROR プロシージャーの省略。 [85 COBOL 標準 では、最低でも上記の 1 つが必要です。]
ACCEPT ステートメント	FROM 句の *environment-name* オペランド DATE YYYYMMDD 句 DAY YYYYDDD 句
ADD ステートメント	18 桁より大きいオペランドの合成

表 61. ***IBM* 拡張言語エレメント** *(続き)*	
言語領域	**拡張エレメント**
CALL ステートメント	呼び出されるプログラムを識別するためのプロシージャー・ポインターおよび関数ポインターのオペランド 以下の句およびパラメーター: • ADDRESS OF • LENGTH OF • OMITTED • BY VALUE • RETURNING 英字またはゾーン 10 進数データ項目での呼び出されるプログラム名の指定 従属グループ項目として定義された引数の指定。 [85 COBOL 標準 では、引数はレベル 01 で定義された基本データ項目またはグループ項目である必要があります。]
CANCEL ステートメント	英字またはゾーン 10 進数データ項目での取り消すプログラム名の指定 取り消すプログラムの名前に対する PGMNAME コンパイラー・オプションの影響
CLOSE ステートメント	WITH NO REWIND 句 行順次形式
COMPUTE ステートメント	等号 (=) に代わるワード EQUAL の使用
DISPLAY ステートメント	UPON 句の *environment-name* オペランド 符号付き数字リテラルおよび非整数数字リテラルの表示
DIVIDE ステートメント	18 桁より大きいオペランドの合成
EXIT ステートメント	EXIT ステートメントの前または後にステートメントを持つ文の中、または別の文を持つ段落の中での EXIT ステートメントの指定。 [85 COBOL 標準 では、EXIT ステートメントは、段落内の唯一の文である文の中に自身によって指定する必要があります。]
EXIT PROGRAM ステートメント	命令ステートメントのシーケンス内の最後のステートメントに先立つ EXIT PROGRAM の指定。 [85 COBOL 標準 では、EXIT PROGRAM ステートメントを命令ステートメントのシーケンス内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。]
GO TO ステートメント	命令ステートメントのシーケンス内の最後のステートメントに先立つ無条件フォーマットのコーディング。 [85 COBOL 標準 では、以下のように無条件 GO TO をコーディングする必要があります。 • プロシージャー名が指定されていない場合は、単一ステートメント段落内でのみコーディングする • それ以外の場合は、文の最後のステートメントとしてコーディングする。]
IF ステートメント	END-IF と NEXT SENTENCE 句との併用。 [85 COBOL 標準 では、END-IF を NEXT SENTENCE と併用することはできません。]

表 61. ***IBM* 拡張言語エレメント** (続き)	
言語領域	**拡張エレメント**
INITIALIZE ステートメント	REPLACING 句の中の DBCS、EGCS、NATIONAL、および NATIONAL-EDITED OCCURS 節の DEPENDING 句を含むデータ項目の初期化
MERGE ステートメント	SAME 節でのファイル名の指定
MULTIPLY ステートメント	18 桁より大きいオペランドの合成
OPEN ステートメント	行順次形式 LINAGE 節を持つファイルに対する EXTEND 句の指定
PERFORM ステートメント	空の行内 PERFORM ステートメント 複数のアクティブな PERFORMS に対する共通出口
READ ステートメント	PREVIOUS 句 AT END 句および適当な宣言型プロシージャーの省略 INVALID KEY 句および適当な宣言型プロシージャーの省略 グループ項目ではなく基本英数字項目でもない項目の読み取り
RETURN ステートメント	英数字グループ項目ではなく基本英数字項目でもない項目へのリターン
REWRITE ステートメント	INVALID KEY 句および適当な宣言型プロシージャーの省略 再書き込みされるレコードの中の文字位置の数以外の文字位置の数によるレコードの再書き込み
SEARCH ステートメント	END SEARCH を NEXT SENTENCE と共に指定 二分探索フォーマットにおける NEXT SENTENCE 句と命令ステートメントの省略
SET ステートメント	データ・ポインター・フォーマット プロシージャー・ポインターおよび関数ポインター・フォーマット
SORT ステートメント	SAME 節での GIVING ファイル名の指定
START ステートメント	INVALID KEY 句および適当な例外プロシージャーの省略 英数字以外のカテゴリーのキーの使用 以下の関係演算子: • LESS THAN • < • NOT GREATER THAN • NOT > • LESS THAN OR EQUAL TO • <=
STOP ステートメント	非整数固定小数点リテラルまたは符号付き数値整数または非整数固定小数点リテラルの指定 文の最後のステートメント以外のステートメントとしての STOP のコーディング

表 61. ***IBM* 拡張言語エレメント** (続き)	
言語領域	**拡張エレメント**
STRING ステートメント	INTO 句で指定されたデータ項目の参照変更
SUBTRACT ステートメント	18 桁より大きいオペランドの合成
UNSTRING ステートメント	送り出しフィールドの参照変更
WRITE ステートメント	INVALID KEY 句と NOT ON INVALID KEY 句 行順次形式 相対ファイルに対する、置き換えられるレコード内の文字位置の数以外の文字位置の数の書き込み 単一の WRITE ステートメントでの ADVANCING PAGE 句と END-OF-PAGE 句の指定 相対ファイルまたは索引付きファイルに対する、INVALID KEY 句と適当な例外プロシージャーの省略
組み込み関数	INTEGER-OF-DATE および INTEGER-OF-DAY 関数における、DATEPROC および INTDATE コンパイラー・オプションの影響 関数: • 434 ページの『ADD-DURATION』 • 436 ページの『CONVERT-DATE-TIME』 • 440 ページの『DATE-TO-YYYYMMDD』 • 440 ページの『DATEVAL』 • 442 ページの『DAY-TO-YYYYDDD』 • 442 ページの『DISPLAY-OF』 • 444 ページの『EXTRACT-DATE-TIME』 • 445 ページの『FIND-DURATION』 • 452 ページの『NATIONAL-OF』 • 460 ページの『SUBTRACT-DURATION』 • 462 ページの『TEST-DATE-TIME』 • 465 ページの『TRIML』 • 465 ページの『TRIMR』 • 466 ページの『UNDATE』 • 469 ページの『YEAR-TO-YYYY』 • 469 ページの『YEARWINDOW』
LENGTH 関数	関数への引数としてのポインター、ADDRESS OF 特殊レジスター、または LENGTH OF 特殊レジスターの指定

表 61. ***IBM* 拡張言語エレメント** *(続き)*	
言語領域	**拡張エレメント**
コンパイラー指示ステートメント	以下のステートメント: • BASIS • PROCESS (CBL) • *CONTROL および *CBL • DELETE • EJECT • INSERT • READY または RESET TRACE • SERVICE LABEL • SERVICE RELOAD • SKIP1、SKIP2、および SKIP3 • TITLE
コンパイラー・ディレクティブ	CALLINTERFACE ディレクティブ
COPY ステートメント	テキスト名修飾子を指定する "OF *library-name*" 構文のオプション テキスト名およびライブラリー名 を指定するためのリテラル SUPPRESS 句 ネストされた COPY ステートメント REPLACING オペランドのワード ・フォーム内の最初または最後の文字としてのハイフンの使用 REPLACING オペランドのワード ・フォーム内での任意の文字の使用 (COBOL 区切り文字を除く)。 [85 COBOL 標準 では、ユーザー定義語の構成で使用される文字のみを受け入れます。]

付録 B コンパイラー限界値

COBOL for Linux には、プログラムおよびクラスの定義において以下の制限があります。

COBOL コンパイラーはコンパイル単位におけるさまざまなメモリー領域のアドレッシングを、この付録で説明されている制限までサポートしますが、完全なアプリケーション (一般に、複数のコンパイル単位で構成されている) は、実行されるアドレス・スペース内で使用可能な私用ストレージの量によって制限されます。 つまり、アプリケーションは、説明されている制限に達する前にストレージを使い果たしてしまう可能性があります。

表 *62.* **コンパイラー限界値**	
言語エレメント	**コンパイラー限界値**
ユーザー定義語の最大長 (例えば、データ名、ファイル名、クラス名)	30 バイト
プログラムのサイズ	NOTEST あり: 999,999 行
リテラルの数	4,194,303
リテラルの全長	4,194,303 バイト
予約語テーブルの項目の数	1536
COPY REPLACING . . . BY . . . (COPY ステートメントごとの項目数)	制限なし
COPY ライブラリーの数	制限なし
IDENTIFICATION DIVISION	
ENVIRONMENT DIVISION	
構成セクション	
SPECIAL-NAMES 段落	
簡略名 IS	18
UPSI-*n* . . . (スイッチ数)	0 から 7
英字名 IS . . .	制限なし
リテラル THRU . . . または ALSO . . .	256
入出力セクション	
FILE-CONTROL 段落	
SELECT ファイル名 . . .	最大 65,535 のファイル名を外部名に割り当て可能
ASSIGN システム名 . . .	制限なし
RECORD KEY の長さ	制限なし
RESERVE 整数 (バッファー)	255
I-O-control 段落	
RERUN ON システム名 . . .	32,767
RERUN 整数 RECORDS	16,777,215
SAME RECORD AREA	255

表 62. **コンパイラー限界値** (続き)	
言語エレメント	**コンパイラー限界値**
SAME RECORD AREA FOR ファイル名 . . .	255
SAME SORT/MERGE AREA	制限なし
MULTIPLE FILE ファイル名 . . .	制限なし
DATA DIVISION	
77 データ項目サイズ	999,999,999 バイト
01-49 データ項目サイズ	999,999,999 バイト
01 + 77 の合計 (データ項目)	制限なし
88 条件名 . . .	制限なし
66 RENAMES . . .	制限なし
PICTURE 節、文字ストリング の文字数	50
PICTURE 節、数字項目の数字桁数	ARITH(COMPAT) が有効な場合: 18 ARITH(EXTEND) が有効な場合: 31
PICTURE 節、数字編集文字位置	249
ピクチャー記号の複製 ( )	999,999,999 バイト
ピクチャー記号の複製 (編集)	32,767
ピクチャー記号の複製 ( )、クラス DBCS 項目	499,999,999 バイト
ピクチャー記号の複製 ( )、クラス国別項目	499,999,999 バイト
基本項目サイズ	999,999,999 バイト
OCCURS 整数	999,999,999 バイト
ODO の合計数	4,194,303
テーブルのサイズ	999,999,999 バイト
テーブル・エレメントのサイズ	999,999,999 バイト
ASCENDING または DESCENDING KEY . . . (OCCURS 節ごと)	12 KEYS
キーの全長 (OCCURS 節ごと)	256 バイト
INDEXED BY . . . (OCCURS 節ごとの索引名)	12
クラスまたはプログラムごとの指標 (指標名) の合計数	65,535
相対指標のサイズ	32,765
FILE SECTION	

表 62. **コンパイラー限界値** (続き)	
言語エレメント	**コンパイラー限界値**
FD レコード記述項目	1,048,576 バイト サイズ制限の詳細については、526 ページの『ファイル入出力制限』を参照してください。
FD ファイル名 . . .	65,535
LABEL データ名 . . . (オプショナル節がない場合)	255
ラベル・レコード長	80 バイト
BLOCK CONTAINS 整数	2,147,483,647 バイト
RECORD CONTAINS 整数	1,048,576 バイト サイズ制限の詳細については、526 ページの『ファイル入出力制限』を参照してください。
LINAGE 節値	99,999,999
SD ファイル名 . . .	65,535
DATA RECORD データ名 . . .	制限なし
LINKAGE SECTION	
合計サイズ	2,147,483,646 バイト
LOCAL-STORAGE SECTION	
合計サイズ	2,147,483,646 バイト
WORKING-STORAGE SECTION	
外部属性のない項目の合計サイズ	2,147,483,646 バイト
外部属性のある項目の合計サイズ	2,147,483,646 バイト
PROCEDURE DIVISION	
PROCEDURE DIVISION USING 識別子 . . .	32,767
プロシージャー名	1,048,575
ステートメントごとの添え字付きデータ名	32,767
行ごとのステートメント (TEST)	7
ADD *ID* . . .	制限なし
ALTER プロシージャー名-*1* TO プロシージャー名-*2* . . .	4,194,303
CALL . . . BY CONTENT *ID*	2,147,483,647 バイト
CALL *ID* または リテラル USING *ID* または リテラル . . .	500
CALL リテラル . . .	4,194,303
実行単位のアクティブ・プログラム	32,767
呼び出される名前の数 (DYN オプション)	制限なし

表 62. **コンパイラー限界値** (続き)	
言語エレメント	**コンパイラー限界値**
CANCEL *ID* または リテラル . . .	制限なし
CLOSE ファイル名 . . .	制限なし
COMPUTE *ID* . . .	制限なし
DISPLAY *ID* または リテラル . . .	制限なし
DIVIDE *ID* . . .	制限なし
ENTRY USING *ID* または リテラル . . .	制限なし
EVALUATE . . . サブジェクト	64
EVALUATE . . . WHEN 節	256
GO プロシージャー名 . . . DEPENDING	255
INSPECT TALLYING および REPLACING 節	制限なし
MERGE ファイル名 ASC または DES KEY . . .	制限なし
マージ・キー合計長	4,092 バイト
MERGE USING ファイル名 . . .	16
MOVE *ID* または リテラル TO *ID*. . .	制限なし
MULTIPLY *ID* . . .	制限なし
OPEN ファイル名 . . .	制限なし
PERFORM	4,194,303
PERFORM . . . TIMES *ID* または リテラル	999,999,999
SEARCH . . . WHEN . . .	制限なし
SET 指標 または *ID* . . . TO	制限なし
SET 指標 . . . UP/DOWN	制限なし
SORT ファイル名 ASC または DES KEY	制限なし
ソート・キー合計長	4,092 バイト
SORT USING ファイル名 . . .	16
STRING *ID* . . .	制限なし
STRING DELIMITED *ID* または リテラル . . .	制限なし
UNSTRING DELIMITED *ID* または リテラル . . .	255
UNSTRING INTO *ID* または リテラル . . .	制限なし
USE . . . ON ファイル名 . . .	制限なし
最大サイズ *ID* の XML PARSE ステートメント	999,999,999 バイト

ファイル入出力制限

このセクションでは、さまざまなタイプのファイルのファイル入出力制限を示します。

行順次ファイル

- 最小レコード・サイズ : 0
- 最大レコード・サイズ: 2**64 -2
- 最大レコード数: 最大なし
- 最大レコード・キー値: 適用外
- ファイルに割り振られた最大バイト数: 2**64 -1

SFS ファイル

- 最大レコード・サイズ: 2**32 -1
- 最大レコード数: 2**64
- 最大レコード・キー値: 2**32 -1
- ファイルに割り振られた最大バイト数: 64GB

SdU ファイル

- 最小レコード・サイズ : 1 バイト
- 最大レコード・キー長: 255 バイト
- 最大相対キー値: 2**31 -1
- 最大レコード・サイズ: 2**64 -2
- 最大レコード数: 2**64
- 最大レコード・キー値: 2**64 -1
- ファイルに割り振られた最大バイト数: 2**64 -1

STL ファイル

- 最小レコード・サイズ : 0
- 最大レコード・キー長: 255 バイト
- 代替キーの最大数: 253
- 最大相対キー値: 2**31 -1
- 最大レコード・サイズ: 2**32 -1
- 最大レコード数: 2**64
- 最大レコード・キー値: 2**32 -1
- ファイルに割り振られた最大バイト数: 2**64 -1

RSD ファイル

- 固定長レコードおよび可変長レコード
- 最小レコード・サイズ : 0
- 最大レコード・サイズ: 2**64 -2
- 最大レコード数: 最大なし
- 最大レコード・キー値: 2**64 -1
- ファイルに割り振られた最大バイト数: 2**64 -1

QSAM ファイル

- 最大レコード・サイズ :
  - 固定長レコード: 2**64

  - 可変長レコード: 2**16 -4
- 最大レコード数: 最大なし
- 最大レコード・キー値: 2**64
- ファイルに割り振られた最大バイト数: 2**64

Db2 ファイル

次にリストされている制限は、簡易表およびデフォルトの 4 KB データベース・ページ・サイズを想定しています。

- 最小レコード・サイズ : 0 バイト
- 最大行長さ: 4005 bytes
- 固定長レコードの最大レコード・サイズ:
  - 索引付きレコード: 4005 bytes
  - Small 形式相対または順次レコード: 4001 バイト
  - Large 形式相対または順次レコード: 3997 バイト

  キーは索引付きファイル・レコード内の固定位置になければならないため、可変長データ・セグメントで終了する索引付きファイルには別の制約があります。
- 可変長レコードの最大レコード・サイズ:
  - VARCHAR カラムの場合、制限は固定長レコードよりも 4 バイト少なくなります。これは、VARCHAR タイプのデータが行内で直接維持され、4 バイト記述子が先頭に付くためです。
  - BLOB などのその他の可変長タイプでは、記述子のみが行に格納されます。データは別に格納され、最大サイズの 2 GB を持ちます。
- 相対および順次ファイルの最大レコード数:
  - 小ファイル・モードの場合: 2**32
  - 大ファイル・モードの場合: 2**64
- 代替キーの最大数: 500
- 最大レコード・キー値:
  - 小ファイル・モードの場合: 2**32 -1
  - 大ファイル・モードの場合: 2**64 -1
- ファイルに割り振られた最大バイト数: 64 GB

Db2 ファイルの制限については、*Db2 (Linux、UNIX、および Windows 版) Database Administration Concepts and Configuration Reference* 内の *SQL* および *XML* 制限を参照してください。.

Encina SFS ファイルの制限の詳細については、*TXSeries® for Multiplatforms: CICS Application Programming Guide* を参照してください。

ARITH の詳細については、*COBOL for Linux on x86* プログラミング・ガイド内の *ARITH* を参照してください。

ADDR の詳細については、*COBOL for Linux on x86* プログラミング・ガイド 内の *ADDR* を参照してください。

付録 C ソース変換ユーティリティー (scu)

ソース変換ユーティリティー (scu) は、COBOL ソース・プログラムを IBM 以外のソース形式または自由形式のソース形式から COBOL for Linux でコンパイルできる形式に変換する時に役立つスタンドアロン Linux プログラムです。

scu で以下の変換を実行します。

- タブ拡張を含め、オプションの制御で空白文字を真のスペースに変換する
- 行終了文字を正規化する
- 必要に応じて、標識領域、領域 A、または領域 B など、適切な領域内で開始するために項目を位置変更する
- CBL(PROCESS) ステートメント用の特殊位置合わせを確実にする
- オプションで、カラム 1 から 6 のシーケンス番号をブランクにし、カラム 73 から 80 のシリアル番号を除去する
- 不規則な固定ソース形式入力をデフォルトで変換し、オプションで自由形式入力を変換する

さらに、以下のような構文や意味の修正も実行します。

- 引用符付きのリテラルの前後に抜けていたスペースを追加する
- レベル 01 になっている OCCURS 節をレベル 02 に調整する
- 非浮動小数点リテラルを浮動小数点定数の表記に変換する
- 余分のピリオドや間違った場所に入っているピリオドを修正する
- データ型で必要な場合に、SET ステートメントを MOVE に変換する
- "<>" を "NOT =" に変換する
- VALUES を VALUE に置き換える
- レベル 01 を領域 A に移動する
- RECORD SEQUENTIAL を LINE SEQUENTIAL に置き換える

制約事項 :

- scu で SET ステートメントを MOVE に変換できるのは、以下のデータ型の場合に限られます。
  - COMP、COMP-4、BINARY
  - COMP-3、PACKED-DECIMAL
  - COMP-5
  - DISPLAY
  - NATIONAL (非リテラル)
- SET ステートメントが 2 行以上にまたがっている場合、scu は、その SET ステートメントを間違った方法で MOVE に変換することがあります。
- scu が VALUES を VALUE に置き換えるのは、VALUES の後にリテラルが続き、VALUES とリテラルが同じ行にある場合に限られます。
- scu が非浮動小数点リテラルを浮動小数点定数の表記に変換するのは、リテラルと VALUE (または VALUES) が同じ行にある場合に限られます。
- scu は、REPLACE ステートメントと、COPY ステートメントの REPLACING 句を無視します。

ヒント :

- 領域 A と領域 B の詳細については、41 ページの『第 6 章 参照形式』を参照してください。
- scu は通常、初期変換を実行してから、構文/意味エラーを修正します (-N オプションとコピーブックは例外です)。ただし、scu は、変換時に重大な問題 (非 COBOL 標準特殊行など) を検出すると、エラー・

メッセージを生成して構文/意味のフェーズをスキップします。-N オプションとコピーブックの例外は以下のとおりです。

- -N を指定した場合は、scu によって初期変換だけが実行されます。
- -G オプションを指定した場合は、メイン・ソース・ファイルでエラーが修正される時にコピーブックでも構文/意味エラーが修正されます。scu によるコピーブックの処理の詳細については、『コピーブックと scu』を参照してください。

出力ファイルには、scu が提供できるすべての修正を取り込んだ、互換性のある COBOL コードが組み込まれます。-o オプションで出力ファイル名を指定しなければ、デフォルトの出力ファイルは *filename*.scu.cbl になります (*filename* は接尾語なしの元のファイル名)。例えば、ソース abc.def.cob の出力ファイルは abc.def.scu.cbl になります。

scu は、プログラム変換の成功または失敗を示す戻りコードを生成します。戻りコードとそれぞれの説明を以下の表にまとめます。

表 *63. scu* の戻りコード

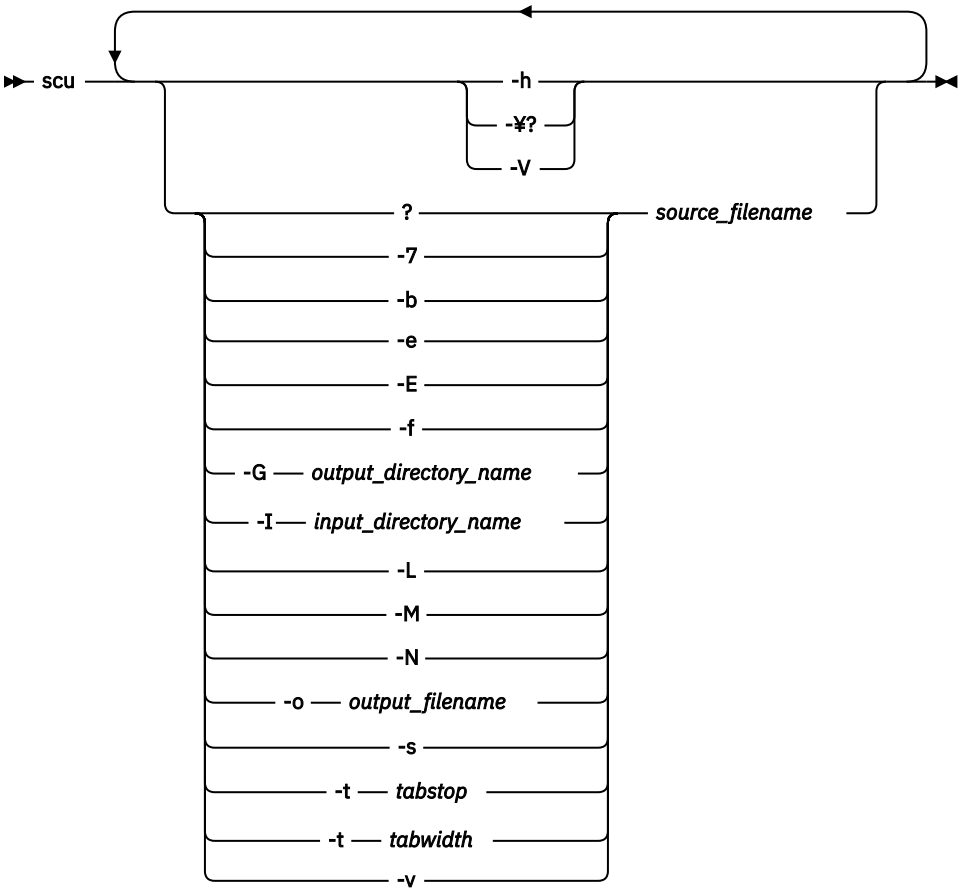
戻りコード	説明
0	scu が正常に実行されたことを示します。
1 - 4	正常終了コードとして将来利用するために予約されています。
5	一般障害が発生したことを示します。
6	コピー・ファイルを開けなかったことを示します。

重要 : scu は、ソース・プログラムを COBOL for Linux でコンパイルできる形式に変換しますが、コードに含まれる非互換箇所をすべて検出したり修正したりできるわけではありません。IBM は、scu による変更にエラーがないことや、変更によって期待どおりのコンパイルと実行が可能になることを保証してはいません。scu によって生成された結果を確認し、期待どおりに変更されているかどうかを確かめる必要があります。scu が修正しようとしたコードをさらに変更することが必要な場合もあります。

ソース変換ユーティリティー (scu) のオプション

ソース・プログラム変換のためのソース変換ユーティリティー (scu) では、さまざまなオプションを使用できます。

scu コマンドの構文は、以下のとおりです。



オプションの説明:

-7

-7 オプションを指定した場合、行の先頭に特殊文字があると、その文字が検出されて第 7 桁 (固定形式または拡張形式の標識域) に移動します。この場合の特殊文字とは、アスタリスク '*'、スラッシュ '/'、ドル記号 '$'、'D' の後にスペースが入った文字列、'd' の後にスペースが入った文字列です。移動する特殊文字の後の文字の処理は、その位置によって異なります。

- 後続の文字が第 2 桁から第 7 桁にある場合は、特殊文字が第 7 桁に入るよう行全体が右に移動します。
- 後続の文字が領域 A または B にある場合、それらの文字はそのまま残ります。ただし、第 73 桁以降に文字がある場合は例外です。第 73 桁以降に文字がある場合は、それらの文字が領域 B の先頭まで左に移動することがあります。

注 :

- ドル記号 '$' の場合、scu は、その行で手操作による介入が必要であるという趣旨のエラーを生成します。
- 行の先頭文字が特殊文字 '-' になっている場合、その特殊文字が第 7 桁に移動するのは、-f オプションと -7 オプションを両方指定した場合と、'-' が第 1 桁にある状況で -f オプションを指定した場合に限られます。
- 特殊文字が第 7 桁にない場合や第 7 桁に移動しない場合、scu は、それらの文字を通常の (特殊ではない) 文字として処理します。

-b

末尾ブランクを削除します。

-e

入力ファイルが 1 行 252 文字の拡張ソース形式かどうかを指定します。このオプションを使用すると、scu は、拡張形式とデフォルトの固定形式を区別して、ソース・コードを正しく変換できるようになります。

-E

scu が出力 (変換後のソース) をデフォルトの 72 桁 (固定形式) に限定しないよう指定します。scu は、必要に応じて各行を最大 252 桁の長さに拡張できます。

-f

入力ソースが自由形式であることを指定します。デフォルトの固定 (互換) ソース形式では、 実行可能 COBOL ソース・コードが領域 A (第 8 桁から第 11 桁) と領域 B (第 12 桁から第 72 桁) に入り、標識が第 7 桁に入ります。オプションで、-e を指定すると、入力ファイルのために領域 B が第 252 桁まで拡張されます。-f オプションを指定した場合、scu は、第 1 桁から第 6 桁にある COBOL ソース・コードを内容に応じて標識桁、領域 A、領域 B のいずれかに移動します。

-f オプションを指定すると、特殊文字で始まる自由形式ソース行が以下のいずれかの方法で処理されます。

- 特殊文字が第 1 桁 (自由形式の標識域) にある場合は、その文字が第 7 桁 (固定形式の標識域) に移動します。
- 特殊文字が第 2 桁から第 11 桁にある場合は、その文字が第 12 桁 (領域 B の先頭) に移動します。

注 :

- 自由形式ソース・ファイルの特殊文字は、'*'、'/'、'$'、'-'、'D' の後にスペースが入った文字列、'd' の後にスペースが入った文字列です。
- デフォルトでは、-f オプションは指定されず、自由形式の行の先頭にある特殊文字が第 12 桁 (領域 B の先頭) に移動するのは、その文字が第 8 桁から第 11 桁 (領域 A) にある場合に限られます。その他の桁にある特殊文字はそのまま残ります。第 7 桁に特殊文字以外の文字があると、その文字は消去されてブランクになります。
- -f オプションと -7 オプションを組み合わせて使用すると、第 1 桁から第 6 桁にある特殊文字が第 7 桁に移動します。-7 オプションを単独で指定した場合と動作が似ていますが、この場合は特殊文字に '-' が含まれます。

-G*<copybook output directory name>*

コピー・ファイルを修正して、指定のディレクトリーに配置します。ディレクトリー名で修飾されたコピー・ファイルの場合は、そのディレクトリー名が指定のコピー・ファイル・ディレクトリーのサブディレクトリーとして保持されます。 -G オプションを指定しない場合は、メイン・ソース・ファイルだけが修正されます。

注 : -G と *<copybook output directory name>* の間にスペースを挿入しないでください。

-h

scu で使用可能な機能に関する基本的なヘルプを表示します。-¥? を指定することによっても、-h の場合と同じヘルプ情報を表示できます。 詳細なヘルプ情報については、man scu コマンドを実行して scu の man ページを参照してください。

-I*<copybook input directory name>*

ライブラリー名も SYSLIB も指定しない場合に、コピーブックの検索に使用するディレクトリーに指定のパスを追加します。

注 :

- このオプションは、小文字の l ではなく大文字の I です。
- 各 -I オプションに使用できるパスは 1 つだけです。 複数のパスを追加するには、-I オプションを必要な数だけ使用してください。
- -I と *<copybook input directory name>* の間にスペースを挿入しないでください。
- -I のディレクトリーからコピー・ファイルを取得し、scu で修正して -G のディレクトリーに保管した場合は、-I を指定し、同じ -G のディレクトリーを指定することによって、そのコピー・ファイル

を再び選択できます。そのようにすれば、scu は、同じメイン・ソース・ファイルについても別のメイン・ソース・ファイルについても後続の実行処理で修正版のコピー・ファイルを使用します。

-L

レベル番号が領域 A にある場合に、01 と 77 以外のレベル番号を領域 B に字下げします。

-M

修正した各行の末尾に scu 修正コード (SCU0001 など) を生成し、出力ファイルの下部に簡略説明と要約を表示します。互換性のある標準の COBOL を入力として使用し、-M オプションを指定すると、互換性のない未使用の桁 (第 82 桁以降) に、行が変更されたことを示す scu 修正コードが追加されます。さらに、それぞれの修正コードに関連した修正情報の要約も表示されます。修正コードと要約は、構文/意味の変更について表示され、初期変換については表示されません。例えば、-f を指定して、自由形式のファイルを固定形式 (80 桁) や拡張形式に変換すると、行は変更されますが、scu 修正コードは生成されません。

修正コードの番号に基づいて、メッセージの重要度を判別できます。

表 *64. scu* メッセージの重大度

修正コードの範囲	重大度	説明
SCU0001 から SCU1999	通知	scu は、この修正でそれ以上の変更は必要ないと判断しています。
SCU2000 から SCU3999	警告	scu は、さらに変更が必要な場合もあると判断しています。例えば、OCCURS をレベル 01 からレベル 02 に変更する修正では、関連した変更がさらに必要になる場合があります。
SCU8000 から SCU8999	エラー	scu は、この修正を完了するにはさらに変更が必要であると判断しています。
SCX0001 から SCX8999	未解決のエラー	問題が検出されましたが、scu はその問題を修正できません。SCX*nnnn* エラー・コードが SCU*nnnn* 修正コードに対応しています。
SCX9000 から SCX9999	無視されたエラー	問題が検出されましたが、scu はその問題を修正しようとしません。

注 : scu は、コードに含まれる非互換箇所をすべて検出したり修正したりできるわけではありません。

scu 修正コードとそれに対応するエラー修正の例を以下に示します。

```
SCU0001 fix for IGYDS0001-W: Add missing space(s).
SCU0004 fix for IGYPS0019-W: Extra and misplaced periods in COBOL source. Scu removes the
extra periods.
SCU1002 fix for IGYGR1080-S: Non-floating point literal is assigned to floating point data
item. Scu converts it to floating point constant notation.
SCU1005 fix for IGYPS2024-S: SET used in place of MOVE. Scu converts SET stmt to MOVE stmt.
SCU1006 fix for IGYPS2094-S: "<>" converted to "NOT = ".
SCU1008 fix for IGYDS0017-E: "01" not in Area A. Scu moves 01 to Area A.
SCU3001 fix for IGYDS1063-E: OCCURS clause in level 01. Scu changes it to level 02 and adds
a dummy 01.
SCU8001 fix for IGYDS0093-S: RECORD SEQUENTIAL not supported. Scu replaces RECORD
SEQUENTIAL with LINE SEQUENTIAL.
SCX9001 specified for an 02 level data item following an 01 level OCCURS that has been
changed
to an 02 level OCCURS with fix code SCU3001
```

-N

scu は、構文/意味の変更なしで初期変換だけを実行し、出力を標準出力として書き出します。

-o *<output filename>*
ソース・ファイルの出力ファイル名を指定します。出力ファイルを既存のディレクトリーで修飾することもできます。例えば、scu -o/dirname1/abc.modified.cbl abc.cbl というコマンドを実行すると、出力ファイル abc.modified.cbl が /dirname1 ディレクトリーに保存されます。 デフォルトでは、出力ファイルは現行ディレクトリーに保存されます。 -o オプションを指定しなければ、ソース・ファイルの出力ファイルは abc.scu.cbl になります。

-s
第 1 桁から第 6 桁をブランクにして第 73 桁でソース行を切り捨てることによって、前後のシーケンス番号を削除します。

-t *<tabwidth>*
ソース・コードで使用されているタブ幅を scu に渡して、変換後のデータの桁が正しくなるようにします。 タブ位置の前で検出されるタブ文字は、その直後の文字がタブ位置に移動するように十分な数のスペースに置き換えられます。 デフォルトのタブ幅は 8 です。

-t *<tabstop>,...*
変換のためのタブ・ストップを scu に渡します。2 つ以上のタブ・ストップを指定する場合は、コンマで区切ります。最後のタブ位置の後で検出されるタブ文字は、1 つのスペース文字に置き換えられます。

-v
詳細出力を有効にします。ソース変換、構文/意味検査、修正の実行時に、エラーと修正に関する情報が STDERR に送られるようになります。

-V
scu のバージョン情報を表示します。

コピーブックと scu:

scu が構文/意味エラーを修正しようとする前に、すべてのコピーブックを変換しておくことをお勧めします。現時点では、scu がコピーブックに対して自動的に変換を実行することはないからです。-7、-b、-e、-E、-f、-L、-s、-t などの変換オプションと一緒に -N オプションを指定することによって、最初にコピーブックで scu を実行できます。その後、メイン・ソース・ファイルで scu を実行する時に、変換後のコピーブックが入っているコピーブック・ディレクトリーを -I で指定して構文/意味エラーを処理できます。

付録 D EBCDIC および ASCII の照合シーケンス

照合シーケンスは、データのソート、マージ、および比較を行って、索引編成のファイルを処理するために、コード化文字セット内または COBOL アルファベット内の文字の順序を定義します。

この付録では、1 バイト EBCDIC (拡張 2 進化 10 進交換コード) および 1 バイト ASCII (ASCII コード) 文字セット用の昇順照合シーケンスを示します。 照合シーケンスは、文字セット内の文字の序数 (1 との相対) によって定義されます。

EBCDIC 照合シーケンスについて示されている記号とそれに関連した意味は、CCSID 1140 で定義された EBCDIC コード・ページに定義されているものです。 記号と意味はその他の EBCDIC コード・ページでは異なる場合がありますが、照合シーケンスは同じです。

EBCDIC 照合シーケンス

EBCDIC 照合シーケンスは、EBCDIC で文字が定義される順序です。

下の表は、1 バイト EBCDIC コード・ページ 1140 の照合シーケンスを示しています。

省略符号 (...) は、先行序数と後続序数間の範囲の序数を省略していることを示します。

表 *65.* ***EBCDIC*** **照合シーケンス**

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 (より小さい)	76	4C
78	(	左括弧	77	4D
79	+	正符号	78	4E
80	|	垂直バー、論理和	79	4F
81	&	アンパーサンド	80	50
. . .				
91	!	感嘆符	90	5A
92	$	ドル記号	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	¬	論理否定	95	5F
97	-	負符号、ハイフン	96	60
98	/	スラッシュ	97	61
. . .				
108	,	コンマ	107	6B

表 65. ***EBCDIC* 照合シーケンス** *(続き)*				
序数	**記号**	**意味**	**10 進表記**	**16 進表記**
109	%	パーセント記号	108	6C
110	_	下線	109	6D
111	>	不等号 (より大きい)	110	6E
112	?	疑問符	111	6F
. . .				
122	`	抑音符号	121	79
123	:	コロン	122	7A
124	#	番号記号、ポンド記号	123	7B
125	@	単価記号	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
. . .				
160	€	ユーロ通貨符号	159	9F

表 65. ***EBCDIC* 照合シーケンス** *(続き)*

序数	記号	意味	10 進表記	16 進表記
. . .				
162	~	波形記号	161	A1
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
. . .				
177	^	脱字記号	176	B0
. . .				
188	[	左大括弧	187	BA
189	]	右大括弧	188	BB
. . .				
193	{	左中括弧	192	C0
194	A		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
. . .				
209	}	右中括弧	208	D0
210	J		209	D1
211	K		210	D2
212	L		211	D3
213	M		212	D4
214	N		213	D5
215	O		214	D6

表 65. ***EBCDIC* 照合シーケンス** *(続き)*

序数	記号	意味	10 進表記	16 進表記
216	P		215	D7
217	Q		216	D8
218	R		217	D9
. . .				
225	¥	円記号	224	E0
. . .				
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
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
. . .				

米国英語 ASCII コード・ページ

ASCII 照合シーケンスは、ASCII で文字が定義される順序です。

以下の表は、米国英語 ASCII コード・ページの照合シーケンスを示しています。 照合シーケンスとは、ANSI INCITS 4、7 ビット情報交換用米国標準コード (7 ビット ASCII)、および *ISO/IEC 646*、*7* ビットの情報交換用符号化文字集合 の国際参照バージョンに定義されている文字の順序です。

省略符号 (. . .) は、先行序数と後続序数間の範囲の序数を省略していることを示します。

表 66. ***ASCII* 照合シーケンス**

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	$	ドル記号	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	+	正符号	43	2B
45	,	コンマ	44	2C
46	-	ハイフン、負符号	45	2D
47	.	ピリオド、小数点	46	2E
48	/	スラッシュ、ソリダス	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
61	<	不等号 (より小さい)	60	3C
62	=	等号	61	3D
63	>	不等号 (より大きい)	62	3E
64	?	疑問符	63	3F

表 66. ***ASCII* 照合シーケンス** (続き)

序数	記号	意味	10 進表記	16 進表記
65	@	アットマーク	64	40
66	A		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	\	バックスラッシュ、逆固相線	92	5C
94	]	右大括弧	93	5D
95	^	脱字記号	94	5E
96	_	下線	95	5F
97	`	抑音符号	96	60
98	a		97	61

表 66. ***ASCII* 照合シーケンス** (続き)

序数	記号	意味	10 進表記	16 進表記
99	b		98	62
100	c		99	63
101	d		100	64
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	|	垂直バー	124	7C
126	}	右中括弧	125	7D
127	~	波形記号	126	7E

付録 E ソース言語のデバッグ

いくつかの COBOL 言語エレメントはデバッグ機能を実装します。

COBOL 言語エレメントには、次のようなものがあります。

- デバッグ行
- デバッグ・セクション
- DEBUG-ITEM 特殊レジスター
- コンパイル時スイッチ (WITH DEBUGGING MODE 節)
- オブジェクト時スイッチ

デバッグ行

デバッグ行 は、コンパイル時スイッチが活動化しているときに限りコンパイルされるステートメントです。 デバッグ行を使用すると、例えば、 プロシージャー内のある位置においてデータ項目の値を検査することができます。

プログラムの中にデバッグ行を指定するには、7 桁目 (標識域) に D と記入します。 デバッグ行は連続して記述することもできますが、継続している各デバッグ行の 7 桁目に D を記入する必要があります。 文字ストリングを 2 つの行にまたがって切り離すことはできません。

すべてのデバッグ行は、 そのデバッグ行がコンパイルされるかコメントとして扱われるかに関係なく、 プログラムが構文上正しくなるように記述しなければなりません。

デバッグ行は OBJECT-COMPUTER 段落より後の場合は、 プログラム内のどこにでも書き込むことができます。

デバッグ行が領域 A と領域 B にスペースしか含んでいない場合、ブランク行として扱われます。

デバッグ・セクション

デバッグ・セクションは、最外部のプログラムでのみ可能です。 ネストされているプログラム内では無効になります。 デバッグ・セクションは、 ネストされたプログラムに含まれるプロシージャーによって起動されることはありません。

デバッグ・セクションは、宣言型プロシージャーです。 宣言型プロシージャーについての説明は、493 ページの『USE ステートメント』にあります。 デバッグ・セクションは、例えば、あるプロシージャーを繰り返し実行 させる PERFORM ステートメントによって呼び出すことができます。 関連付けられたプロシージャー名 のデバッグ宣言セクションは、 繰り返すたびに 1 回実行されます。

デバッグ・セクションは、 コンパイル時スイッチとオブジェクト時スイッチの両方がアクティブである場合にのみ 実行されます。

デバッグ機能は、命令ステートメントの中で 命令ステートメントが互いに分離して現れるたびに、それぞれ個別のステートメントの始まりであると認識します。

デバッグ・セクションの外側にあるステートメントから、デバッグ・セクションの中で定義されたプロシージャーは参照できません。

DEBUG-ITEM 特殊レジスターは、 デバッグ宣言型プロシージャーの中からのみ参照することができます。

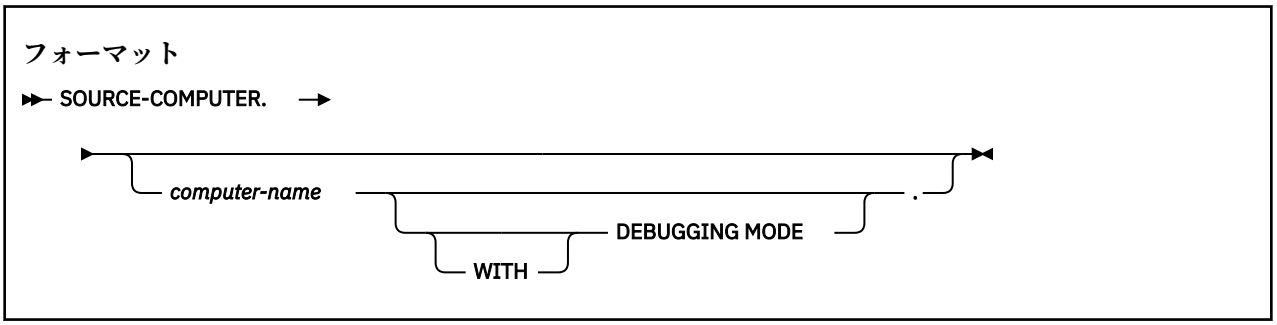
DEBUG-ITEM 特殊レジスター

DEBUG-ITEM 特殊レジスターは、デバッグ・セクションの実行の原因となった条件に関する情報を、デバッグ宣言型プロシージャーに提供します。

DEBUG-ITEM 特殊レジスターの詳細については、16 ページの『DEBUG-ITEM』を参照してください。

コンパイル時スイッチの活動化

コンパイル時スイッチは、デバッグ行とデバッグ・セクションをアクティブな状態にします。 コンパイル時スイッチをアクティブな状態に切り換えるには、 構成セクションの SOURCE COMPUTER 段落で WITH DEBUGGING MODE を指定します。



WITH DEBUGGING MODE

WITH DEBUGGING MODE を指定すると、すべてのデバッグ・セクションとデバッグ行がコンパイルされます。

WITH DEBUGGING MODE を指定しない場合は、すべてのデバッグ・セクションとデバッグ行はコメントとして扱われます。

使用上の注意: COPY ステートメントをデバッグ行として組み込む場合、「D」を COPY ステートメントの最初の行に使用する必要があります。 コンパイラーでは、コピーしたテキストをデバッグ行として扱っています。 COPY ステートメントは、WITH DEBUGGING MODE の指定の有無にかかわらず実行されます。

オブジェクト時スイッチの活動化

オブジェクト時スイッチは、ランタイム・オプションの DEBUG または NODEBUG が指定されている場合に設定されます。 (NODEBUG は IBM のデフォルト値です。)

形式について詳しくは、 「*COBOL for Linux on x86* プログラミング・ガイド」の 『*DEBUG*』 を参照してください。

USE FOR DEBUGGING 宣言型プロシージャーは、DEBUG を指定した場合は有効になり、NODEBUG を指定した場合に動作を禁止されます。

デバッグ行 (7 桁目の「D」行、または「d」行) は、DEBUG オプション、または NODEBUG オプションの影響を受けません。デバッグ行は、 コンパイルされると常にアクティブになっています。

SOURCE-COMPUTER 段落の中で WITH DEBUGGING MODE が指定されていない 場合、 オブジェクト時スイッチは、オブジェクト・プログラムの実行に影響しません。

プログラム実行時スイッチをアクティブまたは非アクティブにするために、ソース単位を再コンパイルする必要はありません。

付録 F 予約語

予約語 とは、COBOL ソース単位であらかじめ定義された意味を持っている文字ストリングです。

以下の表は、COBOL for Linux において予約されている語、および COBOL for Linux の将来のリリースで予約される可能性があるため使用してはならない語を示しています。

- 予約済み の欄に *X* というマークのあるワードは、COBOL for Linux でインプリメントされている機能用として予約されています。 これらのワードはユーザー定義名として使用する場合、S-level メッセージのフラグが付きます。
- 標準のみ の *X* の欄にあるマークのあるワードは、85 COBOL 標準 予約語のうち COBOL for Linux にはインプリメントされていない機能用のものです。 これらのワードをユーザー定義名として使用すると、S-LEVEL メッセージのフラグが付きます。
- 今後予定されている 予約語 の欄に *X* というマークのある語は、COBOL for Linux の将来のリリースで予約される可能性のある語です。 IBM ではこれらの語をユーザー定義名として使用しないことをお勧めします。 これらのワードをユーザー定義名として使用すると、I-LEVEL メッセージのフラグが付きます。

  この欄には 2002 COBOL 標準 の予約語も含まれています。

表 *67.* **予約語**

ワード	予約済み	標準のみ	今後予定されている予約語
+ 算術演算子 - 単項正 (加算)	X		
- 算術演算子 - 単項負 (減算)	X		
* 算術演算子 - 乗算	X		
/ 算術演算子 - 除算	X		
** 算術演算子 - 指数	X		
> 関係演算子 - より大	X		
< 関係演算子 - より小	X		
= 関係演算子 - COMPUTE における等号および代入演算子	X		
== COPY および REPLACE ステートメントにおける疑似テキスト区切り文字	X		
>= 関係演算子 - 以上	X		
<= 関係演算子 - 以下	X		
<> 関係演算子 - 等しくない		X	
*> コメント標識	X		
>> コンパイラー・ディレクティブ標識		X	
ACCEPT	X		
ACCESS	X		
ACTIVE-CLASS			X
ADD	X		
ADDRESS	X		

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
ADVANCING	X		
AFTER	X		
ALIGNED			X
ALL	X		
ALLOCATE	X		
ALPHABET	X		
ALPHABETIC	X		
ALPHABETIC-LOWER	X		
ALPHABETIC-UPPER	X		
ALPHANUMERIC	X		
ALPHANUMERIC-EDITED	X		
ALSO	X		
ALTER	X		
ALTERNATE	X		
AND	X		
ANY	X		
ANYCASE			X
APPLY	X		
ARE	X		
AREA	X		
AREAS	X		
ASCENDING	X		
ASSIGN	X		
AT	X		
AUTHOR	X		
B-AND			X
B-NOT			X
B-OR			X
B-XOR			X
BASED			X
BASIS	X		
BEFORE	X		
BEGINNING	X		
BINARY	X		

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
BINARY-CHAR			X
BINARY-DOUBLE			X
BINARY-LONG			X
BINARY-SHORT			X
BIT			X
BLANK	X		
BLOCK	X		
BOOLEAN	X		
BOTTOM	X		
BY	X		
CALL	X		
CANCEL	X		
CBL	X		
CD		X	
CF		X	
CH		X	
CHARACTER	X		
CHARACTERS	X		
CLASS			X
CLASS-ID			X
CLOCK-UNITS		X	
CLOSE	X		
COBOL	X		
CODE	X		
CODE-SET	X		
COL			X
COLLATING	X		
COLS			X
COLUMN		X	
COLUMNS			X
COM-REG	X		
COMMA	X		
COMMON	X		
COMMUNICATION		X	

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
COMP	X		
COMP-1	X		
COMP-2	X		
COMP-3	X		
COMP-4	X		
COMP-5	X		
COMPUTATIONAL	X		
COMPUTATIONAL-1	X		
COMPUTATIONAL-2	X		
COMPUTATIONAL-3	X		
COMPUTATIONAL-4	X		
COMPUTATIONAL-5	X		
COMPUTE	X		
CONDITION			X
CONFIGURATION	X		
CONSTANT			X
CONTAINS	X		
CONTENT	X		
CONTINUE	X		
CONTROL		X	
CONTROLS		X	
CONVERTING	X		
COPY	X		
CORR	X		
CORRESPONDING	X		
COUNT	X		
CRT			X
CURRENCY	X		
CURSOR			X
DATA	X		
DATA-POINTER			X
DATE	X		
DATE-COMPILED	X		
DATE-WRITTEN	X		

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
DAY	X		
DAY-OF-WEEK	X		
DBCS	X		
DE		X	
DEBUG-CONTENTS	X		
DEBUG-ITEM	X		
DEBUG-LINE	X		
DEBUG-NAME	X		
DEBUG-SUB-1	X		
DEBUG-SUB-2	X		
DEBUG-SUB-3	X		
DEBUGGING	X		
DECIMAL-POINT	X		
DECLARATIVES	X		
DEFAULT	X		
DELETE	X		
DELIMITED	X		
DELIMITER	X		
DEPENDING	X		
DESCENDING	X		
DESTINATION		X	
DETAIL		X	
DISABLE		X	
DISPLAY	X		
DISPLAY-1	X		
DIVIDE	X		
DIVISION	X		
DOWN	X		
DUPLICATES	X		
DYNAMIC	X		
EC			X
EGCS	X		
EGI		X	
EJECT	X		

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
ELSE	X		
EMI		X	
ENABLE		X	
END	X		
END-ACCEPT			X
END-ACCEPT	X		
END-ADD	X		
END-CALL	X		
END-COMPUTE	X		
END-DELETE	X		
END-DISPLAY			X
END-DIVIDE	X		
END-EVALUATE	X		
END-EXEC	X		
END-IF	X		
END-INVOKE			X
END-JSON			X
END-MULTIPLY	X		
END-OF-PAGE	X		
END-PERFORM	X		
END-READ	X		
END-RECEIVE		X	
END-RETURN	X		
END-REWRITE	X		
END-SEARCH	X		
END-START	X		
END-STRING	X		
END-SUBTRACT	X		
END-UNSTRING	X		
END-WRITE	X		
END-XML	X		
ENDING	X		
ENTER	X		
ENTRY	X		

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
ENVIRONMENT	X		
EO			X
EOP	X		
EQUAL	X		
ERROR	X		
ESI		X	
EVALUATE	X		
EVERY	X		
EXCEPTION	X		
EXCEPTION-OBJECT			X
EXEC	X		
EXECUTE	X		
EXIT	X		
EXTEND	X		
EXTERNAL	X		
FACTORY	X		
FALSE	X		
FD	X		
FILE	X		
FILE-CONTROL	X		
FILLER	X		
FINAL		X	
FIRST	X		
FLOAT-EXTENDED			X
FLOAT-LONG			X
FLOAT-SHORT			X
FOOTING	X		
FOR	X		
FORMAT			X
FREE	X		
FROM	X		
FUNCTION	X		
FUNCTION-ID			X
FUNCTION-POINTER	X		

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
GENERATE	X		
GET			X
GIVING	X		
GLOBAL	X		
GO	X		
GOBACK	X		
GREATER	X		
GROUP		X	
GROUP-USAGE	X		
HEADING		X	
HIGH-VALUE	X		
HIGH-VALUES	X		
I-O	X		
I-O-CONTROL	X		
ID	X		
IDENTIFICATION	X		
IF	X		
IN	X		
INDEX	X		
INDEXED	X		
INDICATE		X	
INHERITS	X		
INITIAL	X		
INITIALIZE	X		
INITIATE		X	
INPUT	X		
INPUT-OUTPUT	X		
INSERT	X		
INSPECT	X		
INSTALLATION	X		
INTERFACE			X
INTERFACE-ID			X
INTO	X		
INVALID	X		

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
INVOKE			X
IS	X		
JSON-CODE			X
JUST	X		
JUSTIFIED	X		
KANJI	X		
KEY	X		
LABEL	X		
LAST		X	
LEADING	X		
LEFT	X		
LENGTH	X		
LESS	X		
LIMIT		X	
LIMITS		X	
LINAGE	X		
LINAGE-COUNTER	X		
LINE	X		
LINE-COUNTER		X	
LINES	X		
LINKAGE	X		
LOCAL-STORAGE	X		
LOCALE	X		
LOCK	X		
LOW-VALUE	X		
LOW-VALUES	X		
MEMORY	X		
MERGE	X		
MESSAGE		X	
METHOD			X
METHOD-ID			X
MINUS			X
MODE	X		
MODULES	X		

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
MORE-LABELS	X		
MOVE	X		
MULTIPLE	X		
MULTIPLY	X		
NATIONAL	X		
NATIONAL-EDITED	X		
NATIVE	X		
NEGATIVE	X		
NESTED			X
NEXT	X		
NO	X		
NOT	X		
NULL	X		
NULLS	X		
NUMBER		X	
NUMERIC	X		
NUMERIC-EDITED	X		
OBJECT			X
OBJECT-COMPUTER	X		
OBJECT-REFERENCE			X
OCCURS	X		
OF	X		
OFF	X		
OMITTED	X		
ON	X		
OPEN	X		
OPTIONAL	X		
OPTIONS			X
OR	X		
ORDER	X		
ORGANIZATION	X		
OTHER	X		
OUTPUT	X		
OVERFLOW	X		

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
OVERRIDE	X		
PACKED-DECIMAL	X		
PADDING	X		
PAGE	X		
PAGE-COUNTER		X	
PASSWORD	X		
PERFORM	X		
PF		X	
PH		X	
PIC	X		
PICTURE	X		
PLUS		X	
POINTER	X		
POSITION	X		
POSITIVE	X		
PRESENT			X
PRINTING		X	
PROCEDURE	X		
PROCEDURE-POINTER	X		
PROCEDURES	X		
PROCEED	X		
PROCESSING	X		
PROGRAM	X		
PROGRAM-ID	X		
PROGRAM-POINTER			X
PROPERTY			X
PROTOTYPE			X
PURGE		X	
QUEUE		X	
QUOTE	X		
QUOTES	X		
RAISE			X
RAISING			X
RANDOM	X		

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
RD		X	
READ	X		
READY	X		
RECEIVE		X	
RECORD	X		
RECORDING	X		
RECORDS	X		
RECURSIVE	X		
REDEFINES	X		
REEL	X		
REFERENCE	X		
REFERENCES	X		
RELATIVE	X		
RELEASE	X		
RELOAD	X		
REMAINDER	X		
REMOVAL	X		
RENAMES	X		
REPLACE	X		
REPLACING	X		
REPORT		X	
REPORTING		X	
REPORTS		X	
REPOSITORY	X		
RERUN	X		
RESERVE	X		
RESET	X		
RESUME			X
RETRY			X
RETURN	X		
RETURN-CODE	X		
RETURNING	X		
REVERSED	X		
REWIND	X		

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
REWRITE	X		
RF		X	
RH		X	
RIGHT	X		
ROUNDED	X		
RUN	X		
SAME	X		
SCREEN			X
SD	X		
SEARCH	X		
SECTION	X		
SECURITY	X		
SEGMENT		X	
SEGMENT-LIMIT	X		
SELECT	X		
SELF	X		
SEND		X	
SENTENCE	X		
SEPARATE	X		
SEQUENCE	X		
SEQUENTIAL	X		
SERVICE	X		
SET	X		
SHARING			X
SHIFT-IN	X		
SHIFT-OUT	X		
SIGN	X		
SIZE	X		
SKIP1	X		
SKIP2	X		
SKIP3	X		
SORT	X		
SORT-CONTROL	X		
SORT-CORE-SIZE	X		

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
SORT-FILE-SIZE	X		
SORT-MERGE	X		
SORT-MESSAGE	X		
SORT-MODE-SIZE	X		
SORT-RETURN	X		
SOURCE		X	
SOURCE	X		
SOURCE-COMPUTER	X		
SOURCES			X
SPACE	X		
SPACES	X		
SPECIAL-NAMES	X		
SQL	X		
SQLIMS	X		
STANDARD	X		
STANDARD-1	X		
STANDARD-2	X		
START	X		
STATUS	X		
STOP	X		
STRING	X		
SUB-QUEUE-1		X	
SUB-QUEUE-2		X	
SUB-QUEUE-3		X	
SUBTRACT	X		
SUM		X	
SUPER	X		
SUPPRESS	X		
SYMBOLIC	X		
SYNC	X		
SYNCHRONIZED	X		
SYSTEM-DEFAULT			X
TABLE		X	
TALLY	X		

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
TALLYING	X		
TAPE	X		
TERMINAL		X	
TERMINATE		X	
TEST	X		
TEXT		X	
THAN	X		
THEN	X		
THROUGH	X		
THRU	X		
TIME	X		
TIMES	X		
TITLE	X		
TO	X		
TOP	X		
TRACE	X		
TRAILING	X		
TRUE	X		
TYPE	X		
TYPEDEF		X	
UNIT	X		
UNIVERSAL			X
UNLOCK		X	
UNSTRING	X		
UNTIL	X		
UP	X		
UPON	X		
USAGE	X		
USE	X		
USER-DEFAULT			X
USING	X		
VAL-STATUS			X
VALID			X
VALIDATE			X

表 67. **予約語** (続き)

ワード	予約済み	標準のみ	今後予定されている予約語
VALIDATE-STATUS			X
VALUE	X		
VALUES	X		
VARYING	X		
WHEN	X		
WHEN-COMPILED	X		
WITH	X		
WORDS	X		
WORKING-STORAGE	X		
WRITE	X		
WRITE-ONLY	X		
XML	X		
XML-CODE	X		
XML-EVENT	X		
XML-NTEXT	X		
XML-SCHEMA	X		
XML-TEXT	X		
ZERO	X		
ZEROES	X		
ZEROS	X		

付録 G コンテキストに依存した語

コンテキストに依存した語とは、指定された一般形式でのみ予約される COBOL ワードです。コンテキストに依存した語が一般形式で許可される場所で使用された場合、その語はキーワードとして処理され、それ以外の場合はユーザー定義語として処理されます。

表 *68.* コンテキストに依存した語

コンテキストに依存した語	**言語構成要素またはコンテキスト**
CYCLE	EXIT ステートメント
INITIALIZED	ALLOCATE ステートメント
NAME	XML GENERATE ステートメント
PARAGRAPH	EXIT ステートメント
RECURSIVE	PROGRAM-ID 段落
YYYYDDD	ACCEPT ステートメント
YYYYMMDD	ACCEPT ステートメント

関連参照
10 ページの『ユーザー定義語』

545 ページの『付録 F 予約語』

付録 H ロケールの考慮事項

ロケールは、情報処理のための言語固有および文化固有の規約の集合です。 ロケールによって、言語および国/地域別環境に関する情報が実行時に使用可能となり、同じプログラムで、国または地域ごとに異なる方法でデータを表示または処理できます。

ロケールおよび特定の言語および文化用にロケールを設定する詳細については、*COBOL for Linux on x86* プログラミング・ガイド内の ロケールの設定を参照してください。

コンパイル時およびランタイム・ロケール

一般に、COBOL ランタイムは、COBOL アプリケーションがアクティブ化されたときに有効になるロケールを決定します。 ただし、次の処理は、コンパイル時に有効なロケールに基づいています。

- ユーザー定義の名前およびリテラル値の評価

  コンパイラーは、コンパイル時に有効なロケールによって示されるコード・ページを使用してソース・プログラムを評価します。 これは、ユーザー定義の名前およびリテラル値の評価を含みます。

  リテラル値がコード・ページ変換が必要であるようにデータ項目と関連付けられているとき、ランタイムに有効なコード・ページはデータ項目に使用されます。 ネイティブ ASCII エンコード方式のクラス英数字の項目の場合、ランタイムに有効なロケールによって示されるコード・ページが使用されます。 EBCDIC エンコード方式のクラス英数字の項目の場合、ランタイムに有効な EBCDIC コード・ページが使用されます。
- 照合シーケンスの評価

  COLLSEQ(LOCALE) コンパイラー・オプションまたは NCOLLSEQ(LOCALE) コンパイラー・オプションが有効なとき、コンパイル時ロケールは次の値を決定します。

  – 以下の言語エレメントの THRU 句内のリテラルによって指定される文字の範囲:
    - 条件名 VALUE 節
    - EVALUATE ステートメント
    - SPECIAL-NAMES パラグラフの ALPHABET 節
    - SPECIAL-NAMES パラグラフの CLASS 節
  – SYMBOLIC CHARACTERS 節内に指定される文字の順序位置

次のセクションは、COBOL ランタイム処理におけるロケールの効果を説明します。

コード・ページ

ランタイム・ロケールは、次のことを決定するために使用されます。

- ネイティブ (非 EBCDIC) エンコード方式を持つ英数字および DBCS データ項目をコード化するコード・ページ。 有効な EBCDIC コード・ページは、CHAR(EBCDIC) コンパイラー・オプションが指定され、EBCDIC_CODEPAGE 環境変数は設定されず、データ項目が NATIVE 句なしに説明されるときに使用されます。
- UPPER-CASE および LOWER-CASE 組み込み関数のケース・マッピング
- コード・ページ引数が省略されたときの DISPLAY-OF 組み込み関数の出力コード・ページ。 有効な EBCDIC コード・ページは、CHAR(EBCDIC) コンパイラー・オプションが指定され、EBCDIC_CODEPAGE 環境変数は設定されず、データ項目が NATIVE 句なしに説明されるときに使用されます。
- コード・ページ引数が省略されたときの NATIONAL-OF 組み込み関数のソース・コード・ページ。 有効な EBCDIC コード・ページは、CHAR(EBCDIC) コンパイラー・オプションが指定され、EBCDIC_CODEPAGE 環境変数は設定されず、データ項目が NATIVE 句なしに説明されるときに使用されます。
- ACCEPT ステートメントでの使用 NATIONAL のデータ項目の受け取りのためのデータ変換のためのコード・ページ

- DISPLAY ステートメントでの使用 NATIONAL のデータ項目から表示するためのデータ変換のためのコード・ページ

2 つのランタイム・コード・ページを有効にすることができます。1 つはネイティブ・データ (非 EBCDIC データ)および 1 つはホスト・データ (EBCDIC データ) 用。 これらのランタイム・コード・ページの決定方法の詳細については、*COBOL for Linux on x86* プログラミング・ガイド内の文字データのコード・ページの指定を参照してください。

照合シーケンス

一般に、COBOL for Linux は、COLLSEQ(LOCALE) コンパイラー・オプションまたは NCOLLSEQ(LOCALE) コンパイラー・オプションが有効になったときに、ランタイム・ロケールによって定義される照合シーケンスを使用します。 COLLSEQ(LOCALE) は、英数字および DBCS データ項目に影響します。
NCOLLSEQ(LOCALE) は、USAGE NATIONAL で記述される項目に影響します。

COLLSEQ(BINARY) または NCOLLSEQ(BINARY) が指定されている場合、バイナリー照合シーケンスが使用されます。

いくつかのケースでは、COLLSEQ または NCOLLSEQ コンパイラー・オプションの設定に関わらず言語規則が非ロケール照合シーケンスの使用を指定します。 例えば、次のように指定します。

- バイナリー照合シーケンスは、常に索引付きファイル・キーに使用されます。
- OBJECT-COMPUTER パラグラフの PROGRAM COLLATING SEQUENCE 節に指定される照合シーケンスは、英数字比較に使用されます。
- OBJECT-COMPUTER パラグラフの PROGRAM COLLATING SEQUENCE 節に指定される照合シーケンスは、COLLATING SEQUENCE 句が SORT または MERGE ステートメントに指定されない限り、英数字キーを持つ SORT または MERGE ステートメントに使用されます。
- SORT または MERGE ステートメントの COLLATING SEQUENCE 句内に指定される照合シーケンスは、英字キーおよび 英数字キーに使用されます。

サポートされているロケール

COBOL for Linux は、特定の言語、国、およびコード・ページの組み合わせのロケールをサポートします。システム環境変数は、特定のロケール・カテゴリーに対して有効なランタイム・ロケールを識別します。詳細については、*COBOL for Linux on x86* プログラミング・ガイド内の 環境変数を使用したロケールの指定を参照してください。

付録 I 業界仕様

COBOL for Linux では、さまざまな業界標準がサポートされています。

- ISO COBOL 標準
  - ISO 1989:1985、プログラミング言語 - *COBOL*

    ISO 1989:1985 は、ANSI INCITS 23-1985 (R2001)、プログラミング言語 - *COBOL* と同一。
  - ISO/IEC 1989/AMD1:1992、プログラミング言語 - *COBOL:* 組み込み関数モジュール

    ISO/IEC 1989/AMD1:1992 は、ANSI INCITS 23a-1989 (R2001)、プログラミング言語 - *COBOL* 用の組み込み関数モジュール と同一。
  - ISO/IEC 1989/AMD2:1994、プログラミング言語 - *COBOL* 用の修正および説明のための改訂

    ISO/IEC 1989/AMD2:1994 は、ANSI INCITS 23b-1993 (R2001)、プログラミング言語 - *COBOL* 用の修正のための改訂 と同一。

    すべての必要なモジュールは、標準で定義された最高水準でサポートされています。

    次のような標準のオプショナル・モジュールがサポートされます。

    - 組み込み関数 (1 ITR 0,1)
    - デバッグ (1 DEB 0,2)

    以下に示すような標準のオプショナル・モジュールはサポートされていません。

    - 報告書作成プログラム
    - 通信
    - デバッグ (2 DEB 0,2)
    - セグメント化 (2 SEG 0,2)
  - ISO/IEC 1989:2002, *Information technology - Programming languages - COBOL* (部分サポート)

    ISO/IEC 1989:2002 is identical to ANSI INCITS 1989-2002 (R2013), *Information technology - Programming languages COBOL*
  - ISO/IEC 1989:2014, *Information technology - Programming languages, their environments and system software interfaces - Programming language COBOL* (部分サポート)

    ISO/IEC 1989:2014 is identical to ANSI INCITS 1989-2014, *Information technology - Programming languages, their environments and system software interfaces - Programming language COBOL*
- ANSI COBOL 標準
  - ANSI INCITS 23-1985 (R2001)、プログラム言語 - *COBOL*
  - ANSI INCITS 23a-1989 (R2001)、プログラム言語 - *COBOL* 用の組み込み関数モジュール
  - ANSI INCITS 23b-1993 (R2001)、プログラミング言語 - *COBOL* 用の修正のための改訂

    すべての必要なモジュールは、標準で定義された最高水準でサポートされています。

    次のような標準のオプショナル・モジュールがサポートされます。

    - 組み込み関数 (1 ITR 0,1)
    - デバッグ (1 DEB 0,2)
    - セグメント化 (2 SEG 0,2)

    以下に示すような標準のオプショナル・モジュールはサポートされていません。

    - 通信
    - デバッグ (2 DEB 0,2)

- *ISO/IEC 646*、*7* ビットの情報交換用符号化文字集合 の国際参照バージョン
- 「米国標準規格 *X3.4-1977*、情報交換用コード」に定義されている 7 ビット・コード化文字セット。
- *SPIRIT (Service Provider's Requirements for Information Technology), Part 6—COBOL Language Profile* (Network ManagementForum 刊行)。
- *MIA (Multivendor Integration Architecture), technical requirements* (NTT 仕様)

COBOL for Linux には、COBOL 標準に関連して次のような制約事項があります。

- 演算式でゼロによる除算が発生したときに、ON SIZE ERROR 句が指定されていないと、処理が異常終了します。
- 分割モジュールのオーバーレイ機能はサポートされません。

上記の標準をサポートするために必要なコンパイラー・オプションおよびランタイム・オプションの仕様については、「*COBOL for Linux on x86* プログラミング・ガイド」の を参照してください。

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 Director of Licensing IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
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 の将来の方向性および指針に関するすべての記述は、予告なく変更または撤回される場合があります。これらは目標および目的を提示するものにすぎません。

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. .

プライバシー・ポリシーに関する考慮事項:

サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。 IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。 ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項をご確認ください。

この「ソフトウェア・オファリング」は、Cookie もしくはその他のテクノロジーを使用して個人情報を収集することはありません。

この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、 お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。

このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および「IBM Software Products and Software-as-a-Service Privacy Statement」(http://www.ibm.com/software/info/product-privacy) を参照してください。

プログラミング・インターフェース情報

この「言語解説書」には、プログラムを作成するユーザーが COBOL for Linux 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書かれています。

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。

Intel は、Intel Corporation または子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。

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。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

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。

用語集

この用語集に記載されている用語は、COBOL における意味に従って定義されています。 これらの用語は、他の言語では同じ意味を持つことも、持たないこともあります。

glossary.html

この用語集には、以下の資料からの用語および定義が記載されています。

- 「*ANSI INCITS 23-1985, Programming languages - COBOL*」(「*ANSI INCITS 23a-1989, Programming Languages - COBOL - Intrinsic Function Module for COBOL*」および「*ANSI INCITS 23b-1993, Programming Languages - Correction Amendment for COBOL*」で改訂)
- 「*ISO 1989:1985, Programming languages - COBOL*」は「*ISO/IEC 1989/AMD1:1992, Programming languages - COBOL: Intrinsic function module*」に改訂されました。
- 「*ANSI X3.172-2002, American National Standard Dictionary for Information Systems*」
- *INCITS/ISO/IEC 1989-2002, Information technology - Programming languages - COBOL*
- *INCITS/ISO/IEC 1989:2014, Information technology - Programming languages, their environments and system software interfaces - Programming language COBOL*

米国標準規格 (ANS) の定義の前にはアスタリスク (*) を付けています。

A

簡略複合比較条件 (* abbreviated combined relation condition)
連続する一連の比較条件の中で、 共通のサブジェクトまたは共通のサブジェクトと共通の比較演算子を明示的に省略したことにより生ずる複合条件。

異常終了 (abend)
プログラムの異常終了。

アクセス・モード (* access mode)
ファイル内のレコードを操作するに当たって使用する方式。

実際の小数点 (* actual decimal point)
10 進小数点文字のピリオド (.) またはコンマ (,) によるデータ項目における小数点位置の物理表現。

実際の文書エンコード (actual document encoding)
XML 文書のエンコード・カテゴリーで、以下のいずれかとなる。XML パーサーは文書の最初の数バイトを調べて判別する。

- ASCII
- EBCDIC
- UTF-8
- UTF-16 (ビッグ・エンディアンまたはリトル・エンディアンのいずれか)
- これ以外のサポートされないエンコード
- 認識不能なエンコード

Linux ネイティブ・ファイル・システム (Linux native file system)
エンコード・ファイルまたはバイナリー・ストリーム・ファイルを直接サポートする任意のローカルまたはネットワーク・ファイル・システム。

Linux ネイティブ・ファイル・システムは、行順次ファイルを直接サポートし、他のすべての COBOL ファイル・タイプのファイル・ストアとして使用される。

英字名 (* alphabet-name)
ENVIRONMENT DIVISION の SPECIAL-NAMES 段落のユーザー定義語であり、特定の文字セットまたは照合シーケンス (あるいはその両方) に名前を割り当てるもの。

英字 (* alphabetic character)
英字またはスペース文字。

英字データ項目 (alphabetic data item)
記号 A のみを含む PICTURE 文字ストリングが記述されたデータ項目。英字データ項目は USAGE DISPLAY を持ちます。

英数字 (* alphanumeric character)
コンピューターの 1 バイト文字セットの任意の文字。

英数字位置 (alphanumeric character position)
「文字位置 *(character position)*」を参照。

英数字データ項目 (alphanumeric data item)
暗黙的または明示的に USAGE DISPLAY として記述された、カテゴリー英数字、英数字編集、または数字編集を持つデータ項目を 指す一般的な呼び方。

英数字編集データ項目 (alphanumeric-edited data item)
少なくとも 1 つの記号 A または X のインスタンスおよび少なくとも 1 つの単純挿入記号 B、0、または / を含んでいる、PICTURE 文字ストリングで記述されたデータ項目。 英数字編集データ項目は USAGE DISPLAY を持ちます。

英数字関数 (* alphanumeric function)
コンピューターの英数字セットからの 1 つ以上の文字のストリングで値が構成されている関数。

英数字グループ項目 (alphanumeric group item)
GROUP-USAGE NATIONAL 節なしで定義されたグループ項目。 INSPECT、STRING、および UNSTRING などの操作の場合、英数字グループ項目は、実際のグループの内容にかかわらず、その内容すべてが USAGE DISPLAY として記述されているかのように処理されます。 グループ内の基本項目を処理する必要のある操作 (MOVE CORRESPONDING、ADD CORRESPONDING、または INITIALIZE など) の場合、英数字グループ項目はグループ・セマンティクスを使用して処理されます。

英数字リテラル (alphanumeric literal)
次のセットからの開始区切り文字を有するリテラル。'、"、X'、X"、Z'、または Z"。 この文字ストリングには、コンピューターの有する文字セットの任意の文字を含めることができる。

代替レコード・キー (* alternate record key)
基本レコード・キー以外のキーで、 その内容が索引付きファイルの中のレコードを識別するもの。

米国規格協会 (American National Standards Institute: ANSI)
米国で認定された組織が自発的工業規格を作成して維持する手順を設定する組織であり、製造業者、消費者、および一般の利害関係者で構成される。

引数 (argument)
(1) ID、リテラル、算術式、または関数 ID で、これにより関数の評価に使用する値を指定する。 (2) CALL ステートメント の USING 句のオペランドであり、呼び出されたプログラムに値を渡すのに使用されます。

算術演算 (* arithmetic operation)
ある算術ステートメントが実行されることにより、 またはある算術式が計算されることにより生じるプロセスで、 そこで指定された引数に対して数学的に正しい解が求められる。

算術演算子 (* arithmetic operator)
次に示す集合に属する 1 文字、または 2 文字で構成された固定した組み合わせ。

文字	意味
+	加算
-	減算
*	乗算
/	除算
**	指数

算術ステートメント (* arithmetic statement)
算術演算を実行させるステートメント。 算術ステートメントには、ADD、COMPUTE、DIVIDE、MULTIPLY、および SUBTRACT の各ステートメントがある。

配列 (array)
データ・オブジェクトで構成される集合体。それぞれのオブジェクトは添え字付けによって一意的に参照できる。 配列は、COBOL ではテーブルに類似する。

昇順キー (* ascending key)
データ項目を比較する際の規則に一致するように、最低のキー値から始めて最高のキー値へとデータを順序付けている値に即したキー。

ASCII
情報交換用米国標準コード。 7 ビットのコード化文字をベースとする 1 つのコード化文字セット (パリティー・チェックを含む 8 ビット) を 使用する標準コードであり、データ処理システム、データ通信システム、および関連装置の間での情報交換に使用される。 ASCII セットは、制御文字と図形文字から構成されている。

IBM は、ASCII に対する拡張 (文字 128 から 255) を定義している。

ASCII ベース・マルチバイト・コード・ページ (ASCII-based multibyte code page)
UTF-8、EUC、または ASCII DBCS コード・ページ。 各 ASCII ベース・マルチバイト・コード・ページは、1 バイト文字とマルチバイト文字の両方を含む。 1 バイト文字のエンコードは ASCII エンコードである。

ASCII DBCS
「*2 バイト ASCII (double-byte ASCII)*」を参照。

割り当て名 (assignment-name)
COBOL ファイルの編成を識別する名前で、 システムがこれを認識する際に使用する。

想定小数点 (* assumed decimal point)
データ項目の中に実際には小数点のための文字が入っていない小数点位置。 想定小数点には、論理的な意味があり、物理的には表現されない。

AT END 条件 (AT END condition)
次のような特定の条件のもとで、READ、RETURN、または SEARCH ステートメントを実行した場合に引き起こされる条件。

- 順次アクセス・ファイルに対して READ ステートメントを実行中に、そのファイル内に次の論理レコードが存在しない場合、または相対レコード番号中の有効数字の桁数が相対キー・データ項目のサイズより大きい場合、またはオプションの入力ファイルが使用可能でない場合。
- RETURN ステートメントの実行中に、関連するソート・ファイルまたはマージ・ファイルについての次の論理レコードが存在しない場合。
- SEARCH ステートメントの実行中に、関連する WHEN 句のいずれかで指定された条件を満足することなく、検索操作が終了した場合。

B

基本文字セット (basic character set)
言語のワード、文字ストリング、および区切り文字の作成時に使用される基本的な文字セット。 基本文字セットは 1 バイトの文字でインプリメントされる。 拡張文字セットには DBCS、UTF-8、または EUC 文字が含まれる。これは、コメント、リテラル、およびユーザー定義語で使用できる。

85 COBOL 標準 における「*COBOL 文字セット (COBOL character set)*」と同義。

ビッグ・エンディアン (big-endian)
メインフレームおよび Linux ワークステーションがバイナリー・データおよび UTF-16 文字を保存するときに使用するデフォルト形式。 この形式では、バイナリー・データ項目の最下位バイトが最高位アドレス にあり UTF-16 文字の最下位バイトが最高位アドレスにあります。リトル・エンディアン *(little-endian)* と比較。

バイナリー項目 (binary item)
2 進表記 (基数 2 の数体系) で表される数値データ項目。等価の 10 進数は、10 進数字 0 から 9 に演算符号を加えたもので構成される。 項目の左端のビットは、演算符号。

二分探索 (binary search)
二分探索の各段階では、 データ・エレメント集合が 2 つに分割される。 数が奇数の場合は適切なアクションが取られる。

ブロック (* block)
通常は 1 つ以上の論理レコードで構成される物理的データ単位。 大容量記憶ファイルの場合、ある論理レコードの一部がブロックに入ることがある。 ブロックのサイズは、そのブロックが含まれているファイルのサイズと直接関係はなく、 そのブロックに含まれているか、そのブロックにオーバーラップしている論理レコード のサイズとも直接関係はない。 「物理レコード *(physical record)*」と同義。

ブール条件 (boolean condition)
ブール条件は、ブール・リテラルが true であるか false であるかを決定する。 ブール条件は定数条件式でのみ使用できる。

ブール・リテラル (boolean literal)
true 値を示す B'1'、または false 値を示す B'0' のどちらか。 ブール・リテラルは定数条件式でのみ使用できる。

停止点 (breakpoint)
通常は命令によって指定されるコンピューター・プログラムの場所であり、プログラムの実行は外部からの介入またはモニター・プログラムによって割り込まれる場合がある。

バッファー (buffer)
入力データまたは出力データを一時的に保持するために使用されるストレージの一部分。

組み込み関数 (built-in function)
組み込み関数 *(intrinsic function)* を参照。

バイト (byte)
特定の数のビット (通常 8 ビット) から成るストリングであり、1 つの単位として処理され、1 つの文字または制御機能を表す。

byte order mark (BOM)
UTF-16 または UTF-32 テキストの先頭に使用して、後続テキストのバイト・オーダーを示す Unicode 文字。バイト・オーダーには、「ビッグ・エンディアン (big-endian)」または「リトル・エンディアン (little-endian)」がある。

バイトコード (bytecode)
Java コンパイラーによって生成され、Java インタープリターによって実行される、マシンから独立したコード。(Oracle)

C

呼び出し先プログラム (called program)
CALL ステートメントの対象となるプログラム。 呼び出し先プログラムと呼び出し側プログラムが実行時に結合されて、1 つの実行単位が作成される。

呼び出し側プログラム (* calling program)
別のプログラムへの CALL を実行するプログラム。

ケース構造 (case structure)
結果として生じた多数のアクションの中から選択を行うために、一連の条件をテストするプログラム処理ロジック。

CCSID
コード化文字セット *ID (coded character set identifier)* を参照。

世紀ウィンドウ (century window)
2 桁年号が固有に決まる 100 年間のこと。 COBOL プログラマーが使用できる世紀ウィンドウには、いくつかのタイプがある。

- ウィンドウ表示日付フィールドについては、YEARWINDOW コンパイラー・オプションを使用する。
- ウィンドウ操作組み込み関数 DATE-TO-YYYYMMDD、DAY-TO-YYYYDDD、および YEAR-TO-YYYY については、引数*-2 (argument-2)* によって世紀ウィンドウを指定する。

文字 (* character)
言語のそれ以上分割できない基本単位。

文字エンコード・ユニット (character encoding unit)
コード化文字セット内の 1 つのコード・ポイントに相当するデータの単位。 1 つ以上の文字エンコード・ユニットを使用して、コード化文字セットの文字が表現される。 エンコード・ユニット とも呼ばれる。

USAGE NATIONAL の場合、文字エンコード・ユニットは、 UTF-16 の 2 バイト・コード・ポイントに対応している。

USAGE DISPLAY の場合、文字エンコード・ユニットは、1 つのバイトに対応している。

USAGE DISPLAY-1 の場合、文字エンコード・ユニットは、 DBCS 文字セットの 2 バイト・コード・ポイントに対応している。

文字位置 (character position)
1 文字を保持または表示するために必要な物理ストレージまたは表示スペースの量。 この用語はどのような文字のクラスにも適用される。 文字の特定のクラスについては、以下の用語が適用される。

- 英数字文字位置。USAGE DISPLAY を使用して表される DBCS 文字。
- *DBCS* 文字位置。USAGE DISPLAY-1 を使用して表される DBCS 文字。
- 国別文字位置。USAGE NATIONAL を使用して表される文字。UTF-16 の 文字エンコード・ユニットと同義。

文字セット (character set)
テキスト情報を表すために使用されるエレメントの集合。 ただし、コード化表現は想定されていません。 コード化文字セット *(coded character set)* も参照。

文字ストリング (character string)
COBOL ワード、リテラル、PICTURE 文字ストリング、またはコメント記入項目を形成する一連の連続した文字。 文字ストリングは区切り文字で区切らなければならない。

チェックポイント (checkpoint)
ジョブ・ステップを後で再始動することができるように、 ジョブとシステムの状況に関する情報を記録しておくことができるところ。

クラス (* class)
ゼロ、1 つ、または複数のオブジェクトの共通の動作およびインプリメンテーションを定義するエンティティー。同じ具体化を共用するオブジェクトは、同じクラスのオブジェクトとみなされる。 クラスは階層として定義でき、あるクラスを別のクラスから継承することができる。

クラス条件 (* class condition)
項目の内容がすべて英字であるか、すべて数字であるか、 すべて DBCS であるか、すべて漢字であるか、あるいはクラス名の定義においてリストされた文字だけで構成されるかという命題で、それに関して真の値を判別することができる。

クラス名 (データの) (* class-name (of data))
ENVIRONMENT DIVISION の SPECIAL-NAMES 段落で定義されるユーザー定義語であり、真理値を定義できる命題に名前を割り当てる。データ項目の内容は、クラス名の定義にリストされている文字だけで構成される。

節 (* clause)
記入項目の属性を指定するという目的で順番に並べられた連続する COBOL 文字ストリング。

COBOL 文字セット (COBOL character set)
COBOL 構文を作成する際に使用される文字セット。 完全な COBOL 文字セットは、以下の文字で構成される。

文字	意味
0,1, . . . ,9	数字
A,B, . . . ,Z	英大文字
a,b, . . . ,z	英小文字
	スペース
+	正符号
-	負符号 (-) (ハイフン)
*	アスタリスク

文字	意味
/	斜線 (スラッシュ)
=	等号
$	通貨符号
,	コンマ
;	セミコロン
.	ピリオド (小数点、終止符)
"	引用符
'	アポストロフィ
(	左括弧
)	右括弧
>	より大きい
<	より小さい
:	コロン
_	下線

COBOL ワード (* COBOL word)
ワード *(word)* を参照。

コード・ページ (code page)
すべてのコード・ポイントに図形文字および制御機能の意味を割り当てるもの。 例えば、あるコード・ページでは、8 ビット・コードに対して 256 コード・ポイントに文字と意味を割り当て、別のコード・ページでは、7 ビット・コードに対して 128 コード・ポイントに文字と意味を割り当てることができる。 ワークステーション上の英語の IBM コード・ページは IBM-1252 で、ホストは IBM-1047 である。

コード・ポイント (code point)
コード化文字セット (コード・ページ) に定義する固有のビット・パターン。 コード・ポイントには、グラフィック・シンボルおよび制御文字が割り当てられる。

コード化文字セット (coded character set)
文字セットを設定し、その文字セットの文字とコード化表現との間の関係を設定する明確な規則の集まり。 コード化文字セットの例として、ASCII もしくは EBCDIC コード・ページで、または Unicode 対応の UTF-16 エンコード・スキームで表す文字セットがある。

コード化文字セット ID (CCSID) (coded character set identifier (CCSID))
特定のコード・ページを識別する 1 から 65,535 までの IBM 定義番号。

照合シーケンス (* collating sequence)
コンピューターに受け入れ可能な文字のシーケンスで、 ソート、マージ、比較、および索引付きファイルの順次処理を目的として順序付けしたもの。

列 (* column)
印刷行または参照形式行におけるバイト位置。列は、行の左端の位置から始めて行の右端の位置まで、1 から 1 ずつ増やして番号が付けられる。 列は 1 つの 1 バイト文字を保持する。

複合条件 (* combined condition)
2 つ以上の条件を AND または OR 論理演算子で結合した結果生じる条件。 条件 *(condition)* および 複合否定条件 *(negated combined condition)* も参照。

コメント記入項目 (* comment-entry)
ドキュメンテーションに使用される IDENTIFICATION DIVISION 内の項目で、実行に影響しない。

コメント行 (comment line)
行の標識区域のアスタリスク (*) と、または、プログラム・テキスト区域 (区域 A および区域 B) の最初の文字ストリングとしてのアスタリスクと大なり記号 (*>) と、 その行の区域 A および区域 B に続くコ

ンピューターの文字セットの任意の文字によって表されるソース・プログラム行。 コメント行は、文書化にのみ役立つ。 行の標識区域では斜線 (/)、そしてその行の区域 A および B ではコンピューター文字セットの任意の文字で表される特殊形式のコメント行があると、コメントの印刷前に改ページが行われる。

共通プログラム (* common program)
別のプログラムに直接的に含まれているにもかかわらず、その別のプログラムに直接的または間接的に含まれている任意のプログラムから呼び出すことができるプログラム。

互換性のある日付フィールド (compatible date field)
互換 という用語の意味は、日付フィールドに適用される場合、それが COBOL のどの部で使用されるかによって異なる。

- DATA DIVISION: 2 つの日付フィールドが同一の USAGE を持ち、以下の条件の少なくとも 1 つを満たしている場合、それらの日付フィールドは互換性があります。
  - 日付フォーマットが同じである。
  - ともにウィンドウ表示日付フィールドであり、一方がウィンドウ表示西暦年 DATE FORMAT YY だけで構成される。
  - ともに拡張日付フィールドであり、一方が拡張西暦年 DATE FORMAT YYYY だけで構成される。
  - 一方が DATE FORMAT YYXXXX で、他方が YYXX の形式である。
  - 一方が DATE FORMAT YYYYXXXX で、他方が YYYYXX の形式である。

  ウィンドウ表示日付フィールドは、拡張日付グループであるデータ項目に従属することができる。 2 つの日付フィールドに互換性があると言われるのは、従属日付フィールドが USAGE DISPLAY を持ち、グループ拡張日付フィールドの開始より 2 バイト後で始まっており、2 つのフィールドが以下の少なくとも 1 つの条件を満たしている場合である。
  - 従属日付フィールドの DATE FORMAT パターンが、グループ日付フィールドの DATE FORMAT パターンと同じ数の X を持つ。
  - 従属日付フィールドが DATE FORMAT YY を持つ。
  - グループ日付フィールドが DATE FORMAT YYYYXXXX を持ち、従属日付フィールドが DATE FORMAT YYXX を持つ。
- PROCEDURE DIVISION: 2 つの日付フィールドが、ウィンドウ表示または拡張できる年部分を除いて、同じ日付形式を持っている場合、それらのフィールドは互換性があります。例えば、DATE FORMAT YYXXX という形式のウィンドウ表示日付フィールドは、以下のものと互換性がある。
  - DATE FORMAT YYXXX という形式の別のウィンドウ表示日付フィールド。
  - DATE FORMAT YYYYXXX という形式の拡張日付フィールド。

コンパイル (* compile)
(1) 高水準言語で表現されたプログラムを、中間言語、アセンブリー言語、または コンピューター言語で表現されたプログラムに変換すること。 (2) あるプログラミング言語で書かれたコンピューター・プログラムから、プログラムの全体的なロジック構造を利用することによって、または 1 つの記号ステートメントから複数のコンピューター命令を作り出すことによって、またはアセンブラーの機能のようにこれら両方を使用することによって、マシン言語プログラムを生成すること。

コンパイル変数 (compilation variable)
ある特定のリテラル値のシンボル名、あるいは DEFINE ディレクティブまたは DEFINE コンパイラー・オプションによって指定されたコンパイル時演算式のシンボル名。

コンパイル時 (* compile time)
COBOL コンパイラーによって、COBOL ソース・コードが COBOL オブジェクト・プログラムに変換される時間。

コンパイル時演算式 (compile-time arithmetic expression)
DEFINE ディレクティブおよび EVALUATE ディレクティブに、または定数条件式に指定されている算術式のサブセット。 コンパイル時演算式における、正規演算式との違い:

- 指数演算子を指定することはできません。

- オペランドはすべて、整数リテラルか、すべてのオペランドが整数リテラルである演算式でなければなりません。
- 式はゼロによる除算が発生しないように指定する必要があります。

コンパイラー (compiler)
高水準言語で記述されたソース・コードをマシン言語のオブジェクト・コードに変換するプログラム。

コンパイラー指示ステートメント (compiler-directing statement)
コンパイル時にコンパイラーに特定の処置を行わせるステートメント。標準コンパイラー指示ステートメントには、 COPY、REPLACE、および USE がある。

コンパイラー指示 (compiler directive)
コンパイル時にコンパイラーに特定の処置を行わせる指示。 COBOL for Linux は、CALLINTERFACE コンパイラー指示、および条件付きコンパイルのコンパイラー指示 (DEFINE、EVALUATE、および IF) をサポートします。

複合条件 (* complex condition)
1 つ以上の論理演算子が 1 つ以上の条件に基づいて作動する条件。 条件 *(condition)*、単純否定条件 *(negated simple condition)*、および 複合否定条件 *(negated combined condition)* も参照。

複合 ODO (complex ODO)
次のような OCCURS DEPENDING ON 節の特定の形式。

- 可変位置項目またはグループ: DEPENDING ON オプションを指定した OCCURS 節によって記述されたデータ項目の後に、非従属データ項目またはグループが続く。 グループは英数字グループでも国別グループでも構いません。
- 可変位置テーブル: DEPENDING ON オプションを指定した OCCURS 節によって記述されたデータ項目の後に、OCCURS 節に よって記述された非従属データ項目が続く。
- 可変長エレメントを持つテーブル: OCCURS 節によって記述されたデータ項目に、DEPENDING ON オプションを指定した OCCURS 節によって記述された従属データ項目が含まれている。
- 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。

コンポーネント (component)
(1) 関連ファイルからなる機能グループ化。 (2) オブジェクト指向プログラミングでは、特定の機能を実行し、他のコンポーネントやアプリケーションと連携するように設計されている、再使用可能なオブジェクトまたはプログラム。 JavaBeans は、コンポーネントを作成するための、Oracle が提供するアーキテクチャーである。

コンピューター名 (* computer-name)
プログラムがコンパイルまたは実行されるコンピューターを識別するシステム名。

条件 (例外) (condition (exception))
アプリケーションの通常のプログラミングされたフローを変えるもの。 条件は、ハードウェアまたはオペレーティング・システムによって検出され、その結果、割り込みが起こる。 このほかにも、条件は言語特定の生成コードまたは言語ライブラリー・コードによっても検出できる。

条件 (式) (condition (expression))
ある真の値が決定される実行時のデータの状況。 条件という用語が本書で 一般形式の「条件」(条件-*1*、条件-*2*、...) の中、またはこれに関連して使用された場合は、 . .) 次のいずれかである。オプションとして括弧で囲まれた単純条件からなる条件式、あるいは、単純条件、論理演算子、および括弧の構文的に正しい組み合わせ (真理値を判別できる) からなる複合条件。 単純条件 *(simple condition)*、複合条件 *(complex condition)*、単純否定条件 *(negated simple condition)*、複合条件 *(combined condition)*、および 複合否定条件 *(negated combined condition)* も参照。

条件式 (* conditional expression)
EVALUATE、IF、PERFORM、または SEARCH ステートメントの中で指定される単純条件または複合条件。単純条件 *(simple condition)* および 複合条件 *(complex condition)* も参照。

条件句 (* conditional phrase)
ある条件ステートメントが実行された結果得られる条件の真理値の判別に基づいてとられるべき処置を指定する句。

条件ステートメント (* conditional statement)
条件の真理値を判別することと、オブジェクト・プログラムの次の処理がこの真理値によって決まることを指定するステートメント。

条件変数 (* conditional variable)
1 つまたは複数の値を持つデータ項目であり、これらの値が、そのデータ項目に割り当てられた条件名を持つ。

条件名 (* condition-name)
条件変数が想定できる値のサブセットに名前を割り当てるユーザー定義語。または、インプリメントする人が定義したスイッチまたは装置の状況に割り当てられるユーザー定義語。

条件名条件 (* condition-name condition)
真理値を判別できる命題で、かつ、条件変数の値が、その条件変数と関連する条件名に属する一連の値のメンバーである命題。

*** CONFIGURATION SECTION**
ENVIRONMENT DIVISION のセクションであり、ソース・プログラムとオブジェクト・プログラムの全体的な仕様を記述する。

CONSOLE
オペレーター・コンソールに関連する COBOL 環境名。

定数条件式 (constant conditional expression)
IF ディレクティブで、または EVALUATE ディレクティブの WHEN 句で使用される可能性がある条件式のサブセット。

定数条件式は、以下の項目のいずれかでなければなりません。

- 両方のオペランドがリテラルであるか、リテラル項のみを含む算術式である比較条件。条件は比較条件の規則に従う必要があり、以下の追加事項があります。
  - オペランドは同じカテゴリーでなければなりません。算術式は数値カテゴリーです。
  - リテラルが指定され、それらが数値リテラルではない場合、関係演算子は “IS EQUAL TO”、“IS NOT EQUAL TO”、“IS =”、“IS NOT =”、または “IS <>” でなければなりません。

  比較条件 *(relation condition)* も参照。
- 定義済み条件。定義済み条件 *(defined condition)* も参照。
- ブール条件。 ブール条件 *(boolean condition)* も参照。
- 前述の単純条件の形式を、AND、OR、および NOT を使用して複合条件に結合することによって形成した複合条件。 簡略複合比較条件を指定することはできません。複合条件 *(complex condition)* も参照。

含まれているプログラム (contained program)
別の COBOL プログラムにネストされている COBOL プログラム。

連続項目 (* contiguous items)
DATA DIVISION 内の連続する記入項目によって記述され、相互に一定の階層関係を持っている項目。

コピーブック (copybook)
一連のコードが含まれたファイルまたはライブラリー・メンバーであり、コンパイル時に COPY ステートメントを使用してソース・プログラムに組み込まれる。 ファイルはユーザーが作成する場合、COBOL によって提供される場合、 または他の製品によって供給される場合とがある。 「コピー・ファイル *(copy file)*」と同義。

カウンター (* counter)
他の数字を使ってその数字分だけ増減したり、 あるいは 0 または任意の正もしくは負の値に変更またはリセットしたりできるようにした、 数または数表現を収めるために使用されるデータ項目。

相互参照リスト (cross-reference listing)
コンパイラー・リストの一部であり、プログラム内においてファイル、フィールド、および標識が定義、参照、および変更される場所に関する情報が入る。

通貨記号値 (currency-sign value)
数字編集項目に保管される通貨単位を識別する文字ストリング。 典型的な例としては、$、USD、EUR などがある。 通貨記号値は、CURRENCY コンパイラー・オプションで定義するか、ENVIRONMENT DIVISION の SPECIAL-NAMES 段落内の CURRENCY SIGN 節によって定義することができる。 CURRENCY SIGN 節が指定されない場合、NOCURRENCY コンパイラー・オプションが有効であれば、ドル記号 ($) がデフォルトの通貨記号値として使用される。 通貨記号 *(currency symbol)* も参照。

通貨記号 (currency symbol)
数字編集項目内の通貨記号値の部分を示すために、PICTURE 節で使用される文字。 通貨記号は、CURRENCY コンパイラー・オプションで定義するか、ENVIRONMENT DIVISION の SPECIAL-NAMES 段落内の CURRENCY SIGN 節によって定義することができる。 CURRENCY SIGN 節が指定されない場合、NOCURRENCY コンパイラー・オプションが有効であれば、ドル記号 ($) がデフォルトの通貨記号値および通貨記号として使用される。 通貨記号と通貨符号値は複数定義可能。 通貨記号値 *(currency sign value)* も参照。

現行レコード (* current record)
ファイル処理において、 ファイルに関連するレコード域の中で使用可能なレコード。

現行ボリューム・ポインター (* current volume pointer)
順次ファイルの現行のボリュームを指している概念上のエンティティー。

D

データ節 (* data clause)
COBOL プログラムの DATA DIVISION のデータ記述記入項目に現れる節で、データ項目の特定の属性を記述する情報を提供する。

データ記述項目 (* data description entry)
COBOL プログラムの DATA DIVISION 内の記入項目であり、レベル番号の後に必要に応じてデータ名が続き、その後に必要に応じて一連のデータ節で構成されるもの。

DATA DIVISION
COBOL プログラムの 1 つの部 (division)。 使用するファイルとファイルに含まれるレコード、必要となる内部 WORKING-STORAGE レコード、COBOL 実行単位内の複数のプログラムで使用可能なデータなど、プログラムで処理するデータを記述する。

データ項目 (* data item)
COBOL プログラムにより、または関数評価の規則により、定義されたデータの単位 (リテラルを除く)。

データ名 (* data-name)
データ記述項目で記述されたデータ項目に名前を割り当てるユーザー定義語。 一般形式で使用された場合、データ名は、その形式の規則で特に許可されていない限り、参照変更、添え字付け、または修飾してはならないワードを表す。

日付フィールド (date field)
次の項目のいずれか。

- データ記述記入項目に DATE FORMAT 節が含まれているデータ項目。
- 次の組み込み関数の 1 つで戻される値。

```
DATE-OF-INTEGER
DATE-TO-YYYYMMDD
DATEVAL
DAY-OF-INTEGER
DAY-TO-YYYYDDD
YEAR-TO-YYYY
YEARWINDOW
```

- ACCEPT ステートメントの概念上のデータ項目 DATE、DATE YYYYMMDD、DAY、および DAY YYYYDDD。
- ある種の算術演算の結果。 詳細については、『日付フィールドを使用する算術計算』(*COBOL for Linux on x86* 言語解説書) を参照してください。

「日付フィールド *(date field)*」という用語は、「拡張日付フィールド *(expanded date field)*」と「ウィンドウ表示日付フィールド *(windowed date field)*」の両方を指す。 非日付 *(nondate)* も参照。

日付形式 (date format)
次のいずれかの方法で指定される 日付フィールドの日付パターン。

- DATE FORMAT 節または DATEVAL 組み込み関数 argument-2 によって明示的に。
- 日付フィールドを戻すステートメントまたは組み込み関数によって暗黙的に。 詳細については、日付フィールド (*COBOL for Linux on x86* 言語解説書) を参照してください。

Db2 ファイル・システム (Db2 file system)
Db2 ファイル・システムは、順次ファイル、索引付きファイル、および相対ファイルをサポートします。Db2 ファイル・システムは CICS との拡張相互協調処理を提供しており、Db2 に保管されている CICS ESDS、KSDS、および RRDS ファイルにバッチ COBOL プログラムがアクセスできるようにします。

DBCS
2 バイト文字セット *(double-byte character set (DBCS))* を参照

DBCS 文字 (DBCS character)
IBM 2 バイト文字セット (DBCS) に定義された任意の文字。

DBCS 文字位置 (DBCS character position)
文字位置 *(character position)* を参照。

DBCS データ項目 (DBCS data item)
少なくとも 1 つの記号 G または少なくとも 1 つの記号 N (NSYMBOL(DBCS) コンパイラー・オプションが有効なとき) を含んでいる PICTURE 文字ストリングで記述されたデータ項目。DBCS データ項目は USAGE DISPLAY-1 を持っています。

デバッグ行 (* debugging line)
行の標識区域に文字 D がある行のこと。

デバッグ・セクション (* debugging section)
USE FOR DEBUGGING ステートメントが含まれているセクション。

宣言文 (* declarative sentence)
区切り記号のピリオドによって終了する 1 つの USE ステートメントから構成されるコンパイラー指示文。

宣言部分 (* declaratives)
PROCEDURE DIVISION の先頭に書き込まれた 1 つ以上の特殊目的セクションの集合であり、その先頭にはキーワード DECLARATIVE が付き、その最後にはキーワード END DECLARATIVES が続いている。宣言部分は、セクション・ヘッダー、USE コンパイラー指示文、および 0 個、1 個、または複数個の関連する段落で構成される。

編集解除 (* de-edit)
項目の編集解除された数値を判別するために、数字編集データ項目からすべての編集文字を論理的に除去すること。

定義済み条件 (defined condition)
コンパイル変数が定義されているかどうかをテストするコンパイル時条件。 定義済み条件は、IF ディレクティブで、または EVALUATE ディレクティブの WHEN 句で指定される。

範囲区切りステートメント (* delimited scope statement)
明示的範囲終了符号を含んでいるステートメント。

区切り文字 (* delimiter)
1 つの文字、または一連の連続する文字であり、文字ストリングの終わりを識別し、その文字ストリングを後続の文字ストリングから区切る。区切り文字は、これを使用して区切られる文字ストリングの一部ではない。

降順キー (* descending key)
データ項目を比較する際の規則に一致するように、 最高のキー値から始めて最低のキー値へとデータを順序付けている値に付けられるキー。

数字 (digit)
0 から 9 までの任意の数字。COBOL では、この用語を用いて他の記号を参照することはない。

桁位置 (* digit position)
1 つの桁を保管するために必要な物理ストレージの大きさ。 この大きさは、データ項目を定義するデータ記述項目に 指定された用途によって異なる。

直接アクセス (* direct access)
前回アクセスしたデータへの参照には依存せず、 プロセスがデータの位置にのみ依存する方法で、 データを記憶装置から取得したり、データを記憶装置に入れる機能。

表示浮動小数点データ項目 (display floating-point data item)
暗黙的または明示的に USAGE DISPLAY として記述されており、外部浮動小数点データ項目を記述する PICTURE 文字ストリングを持っている、データ項目。

部 (* division)
部の本体と呼ばれる、0 個、1 個、または複数個のセクションまたは段落の集合であり、特定の規則に従って形成および結合されたもの。 それぞれの部は、部のヘッダーおよび関連した部の本体で構成される。 COBOL プログラムには、見出し部、環境部、データ部、および手続き部の 4 つの部がある。

部の見出し (* division header)
ワードとその後に続く、部の先頭を示す分離文字ピリオドの組み合わせ。 部のヘッダーは次のとおり。

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
```

Do 構造 (do construct)
構造化プログラミングでは、DO ステートメントを使えば、プロシージャー内の複数のステートメントをグループ化できる。 COBOL では、インライン PERFORM ステートメントが同様に機能する。

do-until
構造化プログラミングにおいて、do-until ループは、少なくとも 1 回は実行され、所定の条件が真になるまで実行される。 COBOL では、TEST AFTER 句を PERFORM ステートメントで使用すれば、同様に機能する。

do-while
構造化プログラミングにおいて、do-while ループは、所定の条件が真である場合、およ び真である間に実行される。 COBOL では、TEST BEFORE 句を PERFORM ステートメントで使用すれば、同様に機能する。

文書タイプ宣言 (document type declaration)
あるクラスの文書に対する文法を規定するマークアップ宣言を含む、または 指示する XML エレメント。この文法は、文書タイプ定義または DTD とも呼ばれる。

文書タイプ定義 (document type definition (DTD))
XML 文書のクラスの文法。 文書タイプ宣言を参照。

2 バイト ASCII (double-byte ASCII)
DBCS 文字および 1 バイト ASCII 文字を含む IBM の文字セット (「ASCII DBCS」とも呼ばれる)。

2 バイト EBCDIC (double-byte EBCDIC)
DBCS 文字および 1 バイト EBCDIC 文字を含む IBM の文字セット (「EBCDIC DBCS」とも呼ばれる)。

2 バイト文字セット (double-byte character set (DBCS))
それぞれの文字が 2 バイトで表現される 1 組の文字。 256 個のコード・ポイントで表現される記号より多くの記号を含んでいる 言語 (日本語、中国語、および韓国語など) は、2 バイト文字セットを必要とする。 各文字に 2 バイトが必要なため、DBCS 文字の入力、表示、および印刷には、DBCS を受け入れ可能なハードウェアおよびサポートされるソフトウェアが必要。

DWARF
DWARF は UNIX International Programming Languages Special Interest Group (SIG) で開発された。言語に依存しないデバッグ情報を提供することにより、さまざまな言語の、統一された方法でのシンボリックなソース・レベル・デバッグのニーズを満たすように設計されている。 DWARF ファイルには、

さまざまなエレメントに編成されたデバッグ・データが含まれている。 詳しくは、「*DWARF/ELF エクステンション ライブラリー・リファレンス*」の『*DWARF program information*』を参照。

動的アクセス (* dynamic access)
1 つの OPEN ステートメントの実行範囲内において、特定の論理レコードを、大容量記憶ファイルからは順次アクセス以外の方法で取り出したりそのファイルに入れたりでき、またファイルからは順次アクセスの方法で取り出せるアクセス・モード。

動的 CALL (dynamic CALL)
DYNAM オプションを使用してコンパイルされたプログラム内の CALL *literal* ステートメント、またはプログラム内の CALL *identifier* ステートメント。

E

EBCDIC (拡張 2 進化 10 進コード) (* EBCDIC (Extended Binary-Coded Decimal Interchange Code))
8 ビット・コード化文字をベースとするコード化文字セット。

EBCDIC 文字 (EBCDIC character)
EBCDIC (拡張 2 進化 10 進コード) セットに含まれているいずれかの記号。

EBCDIC DBCS
「*2 バイト EBCDIC (double-byte EBCDIC)*」を参照。

編集データ項目 (edited data item)
0 の抑止または編集文字の挿入、あるいはその両方を行うことによって変更されたデータ項目。

編集用文字 (* editing character)
次に示す集合に属する 1 文字、または 2 文字で構成される固定した組み合わせ。

文字	意味
	スペース
0	ゼロ
+	正符号
-	負符号
CR	貸方
DB	借方
Z	ゼロの抑止
*	小切手変造防止
$	通貨符号
,	コンマ (小数点)
.	ピリオド (小数点)
/	斜線 (スラッシュ)

エレメント (テキスト・エレメント) (element (text element))
1 つのデータ項目または動詞の記述などのようなテキスト・ストリングの 1 つの論理単位で、その前にエレメント・タイプを識別する固有のコードが付けられたもの。

基本項目 (* elementary item)
論理的にそれ以上細分できないものとして記述されているデータ項目。

CICS SFS ファイル・システム (CICS SFS file system)
SFS ファイル・システム (SFS file system) を参照。

エンコード・ユニット (encoding unit)
文字エンコード・ユニット *(character encoding unit)* を参照。

PROCEDURE DIVISION の終わり (* end of PROCEDURE DIVISION)
COBOL ソース・プログラムにおいて、それ以後にはプロシージャーが存在しない物理的な位置。

プログラム終了マーカー (* end program marker)
語の組み合わせに分離文字ピリオドが続いたもので、COBOL ソース・プログラムの終わりを示す。 プログラム終了マーカーは、次のように記述する。

```
END PROGRAM program-name.
```

項目 (* entry)
分離文字ピリオドで終了させられる連続する節の記述セットであり、COBOL プログラムの IDENTIFICATION DIVISION、ENVIRONMENT DIVISION、または DATA DIVISION に書き込まれる。

環境節 (* environment clause)
ENVIRONMENT DIVISION 記入項目の一部として現れる節。

ENVIRONMENT DIVISION
COBOL プログラムの 4 つの主コンポーネントの 1 つ。 ENVIRONMENT DIVISION では、ソース・プログラムがコンパイルされるコンピューターと、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。 この部では、ファイルの論理概念とそのレコードの間のリンケージ、およびファイルが保管される装置の物理的局面を提供する。

環境名 (environment-name)
IBM が指定する名前であり、システム論理装置、プリンターおよびカード穿孔装置の制御文字、報告書コード、またはプログラム・スイッチ、あるいはそれらの組み合わせを識別する。環境名が ENVIRONMENT DIVISION の簡略名と関連付けられている場合は、その簡略名を、置換が有効な任意の形式で置き換えることができる。

環境変数 (environment variable)
コンピューター環境の一部の局面を定義する多数の変数のいずれかであり、その環境で動作するプログラムからアクセス可能。 環境変数は、動作環境に依存するプログラムの動作に影響を与える。

実行時 (execution time)
実行時 *(run time)* を参照。

実行時環境 (execution-time environment)
ランタイム環境 *(runtime environment)* を参照。

拡張日付フィールド (expanded date field)
拡張 (4 桁) 年を含む日付フィールド。日付フィールド *(date field)* および 拡張西暦年 *(expanded year)* も参照。

拡張西暦年 (expanded year)
4 桁の年だけから構成される日付フィールド。 その値には世紀が含まれる (例えば、1998)。 ウィンドウ表示西暦年 *(windowed year)* と比較。

明示的範囲終了符号 (* explicit scope terminator)
特定の PROCEDURE DIVISION ステートメントの有効範囲を終わらせる予約語。

指数 (exponent)
別の数 (底) をべき乗する指数を示す数。 正の指数は乗算を示し、負の指数は除算を示し、小数の指数は数量の根を示す。 COBOL では、指数式は記号 ** の後に指数を付けて表す。

式 (* expression)
算術式または条件式。

拡張モード (* extend mode)
ファイルに対する EXTEND 句の指定のある OPEN ステートメントが実行されてから、そのファイルに対する REEL または UNIT 句の指定のない CLOSE ステートメントが実行される前までの、ファイルの状態。

Extensible Markup Language
XML を参照。

拡張 (extensions)
85 COBOL 標準 に記述されているものの他に、IBM コンパイラーでサポートされる COBOL 構文およびセマンティクス。

外部コード・ページ (external code page)
ASCII または UTF-8 XML 文書の場合は、現在のランタイム・ロケールが指示するコード・ページ。EBCDIC XML 文書の場合は、次のいずれか。

- EBCDIC_CODEPAGE 環境変数で指定されたコード・ページ
- EBCDIC_CODEPAGE 環境変数が設定されていない場合に現在のランタイム・ロケールとして選択されたデフォルトの EBCDIC コード・ページ。

外部データ (* external data)
プログラムの中で外部データ項目および外部ファイル結合子として記述されるデータ。

外部データ項目 (* external data item)
実行単位の 1 つ以上のプログラムにおいて外部レコードの一部として記述されるデータ項目であり、その項目が記述されている任意のプログラムから参照することができる。

外部データ・レコード (* external data record)
実行単位の 1 つ以上のプログラムにおいて記述される論理レコードであり、そのデータ項目は、それらが記述されている任意のプログラムから参照できる。

外部 10 進数データ項目 (external decimal data item)
ゾーン *10* 進数データ項目 *(zoned decimal data item)* および 国別 *10* 進数データ項目 *(national decimal data item)* を参照。

外部ファイル結合子 (* external file connector)
実行単位の 1 つ以上のオブジェクト・プログラムにアクセス可能なファイル結合子。

外部浮動小数点データ項目 (external floating-point data item)
表示浮動小数点データ項目 *(display floating-point data item)* および 国別浮動小数点データ項目 *(national floating-point data item)* を参照。

外部プログラム (external program)
最外部プログラム。 ネストされていないプログラム。

外部スイッチ (* external switch)
インプリメントする人によって定義され、名前が付けられたハードウェア装置またはソフトウェアで、2 つの選択的な状態のうち一方が存在することを示すために使用される。

F

形象定数 (* figurative constant)
ある予約語を使用することによって参照されるコンパイラー生成の値。

ファイル (* file)
論理レコードの集合。

ファイル属性対立条件 (* file attribute conflict condition)
あるファイルで入出力操作を実行する試みが失敗し、 プログラムの中でそのファイルに対して指定したファイル属性が、 そのファイルの固定属性と一致していないこと。

ファイル節 (* file clause)
DATA DIVISION の記入項目であるファイル記述項目 (FD 記入項目) およびソート・マージ・ファイル記述項目 (SD 記入項目) のいずれかの一部として現れる節。

ファイル結合子 (* file connector)
ファイルに関する情報が入っており、ファイル名と物理ファイルの間のリンケージとして、さらにファイル名とその関連レコード域の間のリンケージとして使用されるストレージ域。

ファイル制御項目 (* file control entry)
SELECT 節と、ファイルの関連物理属性を宣言するすべての従属節。

FILE-CONTROL 段落 (FILE-CONTROL paragraph)
ENVIRONMENT DIVISION 内の段落であり、この中では、特定のソース単位で使用されるデータ・ファイルが宣言される。

ファイル記述項目 (* file description entry)
DATA DIVISION の FILE SECTION の中にある記入項目。レベル標識 FD と、それに続くファイル名、および、必要に応じて、次に続く一連のファイル節から構成される。

ファイル名 (* file-name)
DATA DIVISION の FILE SECTION の中のファイル記述項目またはソート・マージ・ファイル記述項目で記述されるファイル結合子に名前を付けるユーザー定義語。

ファイル編成 (* file organization)
ファイルの作成時に確立される永続的な論理ファイル構造。

ファイル位置標識 (file position indicator)
概念的エンティティーであり、索引付きファイルの場合は参照キー内の現行キーの値、順次ファイルの場合は現行レコードのレコード番号、相対ファイルの場合は現行レコードの相対レコード番号が入っている。あるいは、次の論理レコードが存在しないことを示すか、オプションの入力ファイルが使用可能でないことを示すか、AT END 条件が既に存在していることを示すか、もしくは有効な次のレコードが設定されていないことを示す。

*** FILE SECTION**
DATA DIVISION のセクションであり、ファイル記述項目、ソート・マージ・ファイル記述項目、および関連するレコード記述が入っている。

ファイル・システム (file system)
データ・レコードおよびファイル記述プロトコルの特定のセットに準拠するファイルの集合、およびこれらのファイルを管理する一連のプログラム。

固定ファイル属性 (* fixed file attributes)
ファイルに関する情報であり、ファイルの作成時に設定され、それ以降はファイルが存在する限り変更できない。 これらの属性には、ファイルの編成 (順次、相対、指標付き)、 基本レコード・キー、代替レコード・キー、コード・セット、 最小および最大のレコード・サイズ、レコード・タイプ (固定長、可変長)、 索引付きファイルのキーの照合シーケンス、ブロック化因数、埋め込み文字、 レコード区切り文字がある。

固定長レコード (* fixed-length record)
ファイル記述項目またはソート・マージ記述項目が、 すべてのレコードのバイトの個数が同じであるように要求しているファイルに関連付けられたレコード。

固定小数点項目 (fixed-point item)
PICTURE 節で定義される数値データ項目であり、オプションの符号の位置、その中に含まれる桁数、およびオプションの小数点の位置を指定するもの。 2 進数、パック 10 進数、または外部 10 進数のいずれかのフォーマットをとることができる。

浮動コメント標識 (*>) (floating comment indicators (*>))
浮動コメント標識がプログラムのテキスト域 (領域 A プラス領域 B) 内の最初の文字ストリングである場合は、この行がコメント行であることを示します。 また、浮動コメント標識がプログラムのテキスト領域内の 1 つ以上の文字ストリングの後にある場合は、インライン・コメントを示します。

浮動小数点 (floating point)
実数を 1 対の数表示で表す、数を表記するための形式。 浮動小数点表記では、固定小数点部分 (最初の数表示) と、暗黙浮動小数点の底を指数で表される数だけ累乗して得られる値 (2 番目の数表示) との積が、実数になります。 例えば、数値 0.0001234 の浮動小数点表記は 0.1234 -3 です (ここで、0.1234 は 小数部であり、-3 は指数です)。

浮動小数点データ項目 (floating-point data item)
小数部と指数が入っている数値データ項目。 その値は、小数部に、指数で指定されただけ累乗された数字データ項目の底を乗算することによって得られる。

フォーマット (* format)
データの集合の特定の配列。

機能 (* function)
ステートメントの実行中に参照された時点で決定される値を持つ、 一時的なデータ項目。

関数 ID (* function-identifier)
関数を参照する文字ストリングと分離文字の構文的に正しい組み合わせ。 関数で表現されるデータ項目は、関数名と引数 (ある場合) によって一意的に識 別される。 関数 ID は、参照修飾子を含むことができる。 英数字関数を参照する関数 ID は、 一定の制限に従いつつ ID が指定できる一般フォーマットの中ならばどこにでも指定できる。 整数関数または数字関数を参照する関数 ID は、算術式が指定できる一般フォーマットの中ならばどこにおいても指定できる。

機能名 (function-name)
必要な引数を伴って関数の値を決定する呼び出しを行うメカニズムに付けられる名前を表すワード。

関数ポインター・データ項目 (function-pointer data item)
入り口点を指すポインターを保管できるデータ項目。 USAGE IS FUNCTION-POINTER 節で定義されるデータ項目に、関数入り口点のアドレスが含められる。 一般的に、C および Java プログラムと通信するために使用される。

G

ガーベッジ・コレクション (garbage collection)
参照されなくなったオブジェクト用のメモリーが Java ランタイム・システムによって自動的に解放されること。

GDG
世代別データ・グループ *(GDG)* を参照。

GDS
世代別データ・セット *(GDS)* を参照。

世代別データ・グループ(GDG) (generation data group (GDG))
発生順に関連するファイルの集合。このような各ファイルは、世代別データ・セット *(GDS)* または世代と呼ばれる。

世代別データ・セット (GDS) (generation data set (GDS))
世代別データ・グループ *(GDG)* 内のファイルのうちの 1 つ。このような各ファイルは、グループ内の他のファイルと発生順に関連している。

グローバル名 (* global name)
1 つのプログラムにおいてのみ宣言されるが、そのプログラム、またはそのプログラム内に含まれている任意のプログラムから参照できる名前。 条件名、データ名、ファイル名、レコード名、報告書名、およびいくつかの特殊レジスターが、グローバル名となり得る。

グループ項目 (group item)
(1) 複数の従属データ項目から構成される 1 つのデータ項目。 英数字グループ項目 *(alphanumeric group item)* および 国別グループ項目 *(national group item)* を参照。 (2) 明示的または文脈によって、国別グループまたは英数字グループとして修飾されていない場合、この用語は一般にグループを指す。

グループ区切り文字 (grouping separator)
読みやすさのために数値を何桁かまとめて区切るのに使用される文字。 デフォルトの文字はコンマです。

H

ヘッダー・ラベル (header label)
(1) 記録メディア・ユニットのデータ・レコードの前にある、ラベル。 (2)「ファイル開始ラベル *(beginning-of-file label)*」の同義語。

高位終了 (* high-order end)
文字ストリングの左端の文字。

ホスト英数字データ項目 (host alphanumeric data item)
(XML 文書の場合) NATIVE 句が含まれないデータ記述記入項目を持つ カテゴリー英数字データ項目のうち、CHAR(EBCDIC) オプションを有効にしてコンパイルされたもの。 データ項目のエンコードは、有効な EBCDIC コード・ページである。 このコード・ページは、EBCDIC_CODEPAGE 環境変数が設定されている場合はこの環境変数から決定され、 設定されていない場合は、ランタイム・ロケールに関連付けられたデフォルトのコード・ページから決定される。

I

IBM COBOL 拡張部分 (IBM COBOL extension)
85 COBOL 標準 に記述されているものの他に、IBM コンパイラーでサポートされる COBOL 構文およびセマンティクス。

ICU
International Components for Unicode (ICU) を参照。

IDENTIFICATION DIVISION
COBOL プログラムの 4 つの主コンポーネントの 1 つ。 IDENTIFICATION DIVISION では、プログラム、クラスを識別する。 IDENTIFICATION DIVISION には、作成者名、インストール、または日付を含めることができる。

ID (* identifier)
データ項目に名前を付けるための文字ストリングと分離文字の構文的に正しい組み合わせ。 関数ではないデータ項目を参照するときは、ID は、データ名と、修飾子、添え字、または 参照修飾子 (一意的に参照するために必要な場合) から構成される。 関数であるデータ項目を参照する際には、関数 ID が使われる。

命令ステートメント (* imperative statement)
命令の動詞で開始して、行うべき無条件の処置を指定するステートメント。または明示範囲終了符号によって区切られた条件ステートメント (範囲区切りステートメント)。 1 つの命令ステートメントは、一連の命令ステートメントから構成することができる。

暗黙の範囲終了符号 (* implicit scope terminator)
終了していないステートメントが前にある場合、その範囲を区切る分離文字ピリオド。 または、前にある句の中に含まれるステートメントがある場合、 そのステートメントの範囲の終わりをそれが現れることによって示すステートメントの句。

指標 (* index)
コンピューターのストレージ域またはレジスター。 ここにはテーブル内の個々のエレメントを識別するものを表現する内容が入る。

指標データ項目 (* index data item)
指標名に関連する値を、インプリメントする人が指定した形式で収めることができるデータ項目。

索引付きデータ名 (indexed data-name)
データ名とそれに続く括弧で囲まれた 1 つまたは複数の指標名から構成される ID。

索引付きファイル (* indexed file)
指標付き編成のファイル。

指標付き編成 (* indexed organization)
各レコードがそのレコードの中にある 1 つまたは複数のキー値によって識別される永続的な論理ファイル構造。

指標付け (indexing)
指標名を使用した添え字付け と同義。

索引名 (* index-name)
特定のテーブルに関連付けられた指標に付ける名前を表すユーザー定義語。

初期設定プログラム (* initial program)
実行単位内にプログラムが呼び出されるたびに初期状態に置かれるプログラム。

初期状態 (* initial state)
プログラムが実行単位の中に呼び出された最初期のそのプログラムの状態。

インライン (inline)
ルーチン、サブルーチン、または他のプログラムに分岐せずに、 連続的に実行されるプログラム内の命令。

入力ファイル (* input file)
入力モードでオープンされるファイル。

入力モード (* input mode)
ファイルに対する INPUT 句の指定のある OPEN ステートメントが実行されてから、そのファイルに対する REEL または UNIT 句の指定のない CLOSE ステートメントが実行される前までの、ファイルの状態。

入出力ファイル (* input-output file)
I-O モードでオープンされるファイル。

*** INPUT-OUTPUT SECTION**
ENVIRONMENT DIVISION のセクションであり、オブジェクト・プログラムに必要なファイルおよび外部メディアに名前を付け、実行時にデータの伝送および処理に必要な情報を提供する。

入出力ステートメント (* input-output statement)
個々のレコードに対して操作を行うことにより、またはファイルを 1 つの単位として操作することにより、ファイルの処理を行うステートメント。 入出力ステートメントには、ACCEPT (ID 句付き)、CLOSE、DELETE、DISPLAY、OPEN、READ、REWRITE、SET (TO ON または TO OFF 句付き)、START、および WRITE がある。

入力プロシージャー (* input procedure)
ソートすべき特定のレコードの解放を制御する目的で、SORT ステートメントの実行時に制御が渡されるステートメントの集合。

整数 (* integer)
(1) 小数点の右側に桁位置がない数値リテラル。 (2) DATA DIVISION に定義される数値データ項目であり、小数点の右側に桁位置を含まないもの。 (3) 関数の起こりうるすべての評価の戻り値で、小数点の右側の桁がすべてゼロであることが定義されて いる数字関数。

整数関数 (integer function)
カテゴリーが数字で、その定義が小数点の右側に桁位置を持たない関数。

言語間通信 (ILC) (interlanguage communication (ILC))
異なるプログラム言語で書かれた複数のルーチンが通信できること。 ILC サポートにより、各種言語で書かれたコンポーネント・ルーチンからアプリケーションを簡単に構築することができる。

中間結果 (intermediate result)
連続して行われる算術演算の結果を収める中間フィールド。

内部データ (* internal data)
プログラムの中で記述されるデータで、すべての外部データ項目および外部ファイル結合子を除いたもの。 プログラムの LINKAGE SECTION で記述された項目は、内部データとして扱われる。

内部データ項目 (* internal data item)
実行単位内の 1 つのプログラムの中で記述されるデータ項目。 内部データ項目は、グローバル名を持つことができる。

内部 10 進数データ項目 (internal decimal data item)
USAGE PACKED-DECIMAL または USAGE COMP-3 として記述されており、項目を数値として定義する PICTURE 文字ストリング (記号 9、S、P、または V の有効な組み合わせ) を持っている、データ項目。 「パック *10* 進数データ項目 *(packed-decimal data item)*」と同義。

内部ファイル結合子 (* internal file connector)
実行単位内にあるただ 1 つのオブジェクト・プログラムのみがアクセスできるファイル結合子。

内部浮動小数点データ項目 (internal floating-point data item)
USAGE COMP-1 または USAGE COMP-2 として記述されているデータ項目。 COMP-1 は、単精度浮動小数点データ項目を定義します。 COMP-2 は、倍精度浮動小数点データ項目を定義します。 内部浮動小数点データ項目に関連した PICTURE 節はありません。

International Components for Unicode (ICU)
IBM が後援、サポート、および使用するオープン・ソース開発プロジェクト。ICU ライブラリーは、AIX® および Linux をはじめする、さまざまなプラットフォーム上で、強力かつフル機能の Unicode サービスを提供する。

レコード内データ構造 (* intrarecord data structure)
連続したデータ記述記入項目のサブセットによって定義される、1 つの論理レコードから得られるグループ・データ項目および基本データ項目の集合全体。 これらのデータ記述記入項目には、レコード内データ構造を記述している最初のデータ記述記入項目のレベル番号より大きいレベル番号を持つすべての記入項目が含まれる。

組み込み関数 (intrinsic function)
よく使用される算術関数のような事前定義関数で、組み込み関数参照によって呼び出される。

無効キー条件 (* invalid key condition)
索引付きファイルまたは相対ファイルに関連するキーの特定値が無効であると判別された場合に生じる、実行時の条件。

*** I-O-CONTROL**
ENVIRONMENT DIVISION 段落の名前。この段落では、再実行開始点についてのオブジェクト・プログラム要件、複数データ・ファイルによる同じ区域の共用、および単一入出力装置上の複数のファイル・ストレージが指定される。

I-O-CONTROL 記入項目 (* I-O-CONTROL entry)
ENVIRONMENT DIVISION の I-O-CONTROL 段落内の記入項目であり、プログラム実行中に指定のファイルへのデータの伝送と処理を行うために必要な情報を提供する節が入っている。

入出力モード (* I-O mode)
ファイルに対する I-O 句の指定のある OPEN ステートメントが実行されてから、そのファイルに対する REEL または UNIT 句の指定のない CLOSE ステートメントが実行される前までの、ファイルの状態。

入出力状況 (* I-O status)
入出力操作の結果としての状況を示す 2 文字の値を収める概念上のエンティティー。 この値は、そのファイルについてのファイル制御記入項目で FILE STATUS 節を使用することによって、プログラムに使用可能にされる。

反復構造 (iteration structure)
ある条件が真である間、あるいはある条件が真になるまで、一連のステートメントが繰り返して実行されるプログラムの処理ロジック。

J

J2EE
Java 2 Platform, Enterprise Edition (J2EE) を参照。

Java 2 Platform, Enterprise Edition (J2EE)
エンタープライズ・アプリケーションの開発とデプロイメントのために、Oracle によって定義された環境。 J2EE プラットフォームは、多層の Web ベース・アプリケーションを開発するための機能を提供するサービス、アプリケーション・プログラミング・インターフェース (API)、およびプロトコルで構成されている。 (Oracle)

Java Native Interface (JNI)
Java 仮想マシン (JVM) 内で実行される Java コードが、他のプログラム言語で記述されたアプリケーションおよびライブラリーと連携できるようにするプログラミング・インターフェース。

Java 仮想マシン (JVM) (Java virtual machine (JVM))
コンパイル済みの Java プログラムを実行する中央演算処理装置のソフトウェア・インプリメンテーション。

JSON
JSON (JavaScript Object Notation) とは、単純なデータ交換フォーマットである。

JVM
Java 仮想マシン (JVM) (Java virtual machine (JVM)) を参照。

K

K
記憶容量に関連して使用されるときは、2 の 10 乗。 10 進表記では 1024。

キー (* key)
レコードの位置を識別するデータ項目、またはデータの順序付けを識別するための一連のデータ項目。

参照キー (* key of reference)
索引付きファイルの中のレコードをアクセスするために現在使用されている基本キーまたは代替キー。

キーワード (* keyword)
コンテキスト・センシティブ語または予約語。その語の表示フォーマットがソース単位で使用されるときは、その語は必須である。

キロバイト (KB) (kilobyte (KB))
1 キロバイトは 1024 バイトに相当する。

L

言語名 (* language-name)
特定のプログラミング言語を指定するシステム名。

最後に使われた状態 (last-used state)
内部値がプログラム終了時と同じままで、初期値にリセットされない、プログラムの状態を言う。

文字 (* letter)
以下の 2 つのセットのいずれかに属する文字。

1. 英大文字: 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
2. 英小文字: 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

レベル標識 (* level indicator)
特定のタイプのファイルを識別するか、または階層での位置を識別する 2 つの英字。 DATA DIVISION 内のレベル標識には、CD、FD、および SD がある。

レベル番号 (* level-number)
階層構造におけるデータ項目の位置を示すか、またはデータ記述記入項目の特性を示す、2 桁の数字で表されたユーザー定義語。 1 から 49 までの範囲のレベル番号は、論理レコードの階層構造におけるデータ項目の位置 を示す。 1 から 9 のレベル番号は、1 桁の数字として書き込むことも、 0 の後に有効数字を書き込むこともできる。 レベル番号 66、77、および 88 は、データ記述項目の特性を識別する。

ライブラリー名 (* library-name)
COBOL ライブラリーの名前を表すユーザー定義語。与えられたソース・プログラムをコンパイルするためにコンパイラーが使用するライブラリーを識別する。

ライブラリー・テキスト (* library text)
COBOL ライブラリーの中にある一連のテキスト・ワード、コメント行、 区切り文字のスペース、または区切り文字の疑似テキスト区切り文字。

リリアン日 (Lilian date)
グレゴリオ暦の開始以降の日数。 第 1 日は 1582 年 10 月 15 日、金曜日。 リリアン日フォーマットは、 グレゴリオ暦の考案者であるルイジ・リリオにちなんだ名称。

*** LINAGE-COUNTER**
ページ本体内の現在位置を指す値を収めた特殊レジスター。

リンク (link)
(1) リンク接続 (伝送メディア) と、それぞれがリンク接続の終端にある 2 つのリンク・ステーションの組み合わせ。 1 つのリンクは、マルチポイントまたはトークンリング構成において、複数のリンク間で共用できる。 (2) データ項目あるいは 1 つ以上のコンピューター・プログラムの部分を相互接続すること。例えば、リンケージ・エディターによってオブジェクト・プログラムをリンクして共用ライブラリーを作成すること。

LINKAGE SECTION
呼び出し先のプログラムの DATA DIVISION 内のセクションであり、呼び出し側プログラムから使用可能なデータ項目が記述される。 これらのデータ項目は、呼び出し側プログラムおよび呼び出し先プログラムの両方から参照できる。

リテラル (literal)
ストリングを構成するために配列された文字によって、または形象定数を使用することによって、その値が決められる文字ストリング。

リトル・エンディアン (little-endian)
Intel プロセッサーが 2 進データおよび UTF-16 文字を保管するために使用するデフォルト形式。 この形式では、2 進数データ項目の最上位バイトが最上位のアドレスになり、 UTF-16 文字の最上位バイトが最上位のアドレスになる。 ビッグ・エンディアン *(big-endian)* と比較。

ロケール (locale)
プログラム実行環境の一連の属性であり、文化的に重要な考慮事項を示す。例えば、文字コード・ページ、照合シーケンス、日時形式、通貨表記、数値表記、または言語など。

*** LOCAL-STORAGE SECTION**
DATA DIVISION のセクションであり、VALUE 節で割り当てられた値に応じて、呼び出し単位で割り振りまたは解放が行われるストレージを定義する。

論理演算子 (* logical operator)
予約語 AND、OR、または NOT のいずれか。 条件の形成において、AND または OR、あるいはその両方を論理連結語として使用できる。 NOT は論理否定のために使用することができる。

論理レコード (* logical record)
最も包括的なデータ項目。 レコードのレベル番号は 01。レコードは、基本項目またはグループ項目のどちらでもよい。「レコード *(record)*」と同義。

下位終了 (* low-order end)
文字ストリングの右端の文字。

LSQ ファイル・システム (LSQ file system)
LSQ ファイル・システムでは、LINE SEQUENTIAL ファイルのみがサポートされる。

M

メインプログラム (main program)
プログラムとサブルーチンからなる階層において、プロセス内でプログラムが実行されたときに最初に制御を受け取るプログラム。

Make ファイル (makefile)
アプリケーションに必要なファイルのリストが収められたテキスト・ファイル。 make ユーティリティーはこのファイルを使用して、ターゲット・ファイルを最新の変更で更新する。

大容量記憶 (* mass storage)
データを順次と非順次の 2 つの方法で編成して保管しておくことができるストレージ・メディア。

大容量記憶装置 (* mass storage device)
磁気ディスクなど、大きな記憶容量を持つ装置。

大容量記憶ファイル (* mass storage file)
大容量記憶メディアに格納されたレコードの集合。

MBCS
マルチバイト文字セット *(MBCS) (multibyte character set (MBCS))* を参照。

メガバイト、MB (* megabyte (MB))
1 メガバイトは 1,048,576 バイトに相当する。

マージ・ファイル (* merge file)
MERGE ステートメントによってマージされるレコードの集まり。 マージ・ファイルは、マージ機能により作成され、マージ機能によってのみ使用できる。

簡略名 (* mnemonic-name)
ENVIRONMENT DIVISION において、指定されたインプリメントする人の名前に関連したユーザー定義語。

モジュール定義ファイル (module definition file)
ロード・モジュール内のコード・セグメントを記述するファイル。

マルチバイト文字 (multibyte character)
マルチバイト文字セット内で 2 バイト以上で表わされる文字。 例えば、2 バイト以上で表わされる DBCS 文字または UTF-8 文字。 UTF-16 文字は、UTF-16 がマルチバイト文字セットでないため、マルチバイト文字ではありません。

マルチバイト文字セット(MBCS) (multibyte character set (MBCS))
可変数のバイトで表わされる文字から構成されるコード化文字セット。 例えば、EUC (拡張 UNIX コード)、UTF-8、1 バイト文字および 2 バイト EBCDIC 文字または ASCII 文字の混合物からなる文字セットなど。

マルチタスキング (multitasking)
2 つ以上のタスクの並行実行またはインターリーブド実行を可能にする操作モード。

マルチスレッド化 (multithreading)
コンピューター内で複数のパスを使用して実行を行う並行操作。「マルチプロセッシング *(multiprocessing)*」と同義。

N

名前 (name)
COBOL オペランドを定義する 30 文字を超えないで構成されたワード。

ネーム・スペース (namespace)
XML 名前空間 *(XML namespace)* を参照。

国別文字 (national character)
(1) 国別リテラルまたは USAGE NATIONAL の UTF-16 文字。 (2) UTF-16 で表される任意の文字。

国別文字データ (national character data)
UTF-16 で表されるデータの一般参照。

国別文字位置 (national character position)
文字位置 *(character position)* を参照。

国別データ (national data)
「国別文字データ *(national character data)*」を参照。

国別データ項目 (national data item)
カテゴリー国別、国別編集、または USAGE NATIONAL の数字編集のデータ項目。

国別 10 進数データ項目 (national decimal data item)
暗黙的または明示的に USAGE NATIONAL として記述されており、PICTURE の記号 9、S、P、および V の有効な組み合わせを含んでいる、外部 10 進数データ項目。

国別編集データ項目 (national-edited data item)
少なくとも 1 つの N のインスタンスおよび単純挿入記号 B、0、または / の少なくとも 1 つを含んでいる PICTURE 文字ストリングで記述されている、データ項目。 国別編集データ項目は USAGE NATIONAL を持ちます。

国別浮動小数点データ項目 (national floating-point data item)
暗黙的または明示的に USAGE NATIONAL として記述されており、浮動小数点データ項目を記述する PICTURE 文字ストリングを持っている、外部浮動小数点データ項目。

国別グループ項目 (national group item)
明示的または暗黙的に GROUP-USAGE NATIONAL 節で記述されたグループ項目。 国別グループ項目は、INSPECT、STRING、および UNSTRING などの操作で、カテゴリー国別の基本データ項目として定義されているかのように処理されます。 英数字グループ項目内で USAGE NATIONAL データ項目を定義するのとは対照的に、この処理により、国別文字の埋め込みおよび切り捨てが確実に正しく行われます。 グループ内の基本項目を処理する必要のある操作 (MOVE CORRESPONDING、ADD CORRESPONDING、および INITIALIZE など) の場合、国別グループは グループ・セマンティクスを使用して処理されます。

ネイティブ英数字データ項目 (native alphanumeric data item)
(XML 文書の場合) NATIVE 句で記述されたカテゴリー英数字データ項目、または CHAR(NATIVE) オプションを有効にしてコンパイルされたカテゴリー英数字データ項目。 データ項目のエンコードは、有効なランタイム・ロケールの ASCII または UTF-8 コード・ページである。

固有文字セット (* native character set)
OBJECT-COMPUTER 段落で指定されたコンピューターに関連した、インプリメントする人が定義した文字セット。

固有照合シーケンス (* native collating sequence)
OBJECT-COMPUTER 段落で指定されたコンピューターに関連した、インプリメントする人が定義した照合シーケンス。

複合否定条件 (* negated combined condition)
論理演算子 NOT とその直後に括弧で囲んだ複合条件を続けたもの。 条件 *(condition)* および 複合条件 *(combined condition)* も参照。

単純否定条件 (* negated simple condition)
論理演算子 NOT とその直後に単純条件を続けたもの。 条件 *(condition)* および 単純条件 *(simple condition)* も参照。

ネストされたプログラム (nested program)
他のプログラムの中に直接的に含まれているプログラム。

次の実行可能文 (* next executable sentence)
現在のステートメントの実行完了後に制御が移される次の文。

次の実行可能なステートメント (* next executable statement)
現在のステートメントの実行完了後に制御が移される次のステートメント。

次のレコード (* next record)
ファイルの現行レコードに論理的に続くレコード。

独立項目 (* noncontiguous items)
WORKING-STORAGE SECTION および LINKAGE SECTION 内の基本データ項目で、他のデータ項目と階層上の関係を持たないもの。

非日付データ (nondate)
次の項目のいずれか。

- 日付記述記入項目に DATE FORMAT 節が含まれていないデータ項目
- リテラル
- UNDATE 関数を使用して変換された日付フィールド
- 参照変更された日付フィールド
- 日付フィールド・オペランドを含む特定の算術演算の結果。例えば、2 つの互換日付 フィールドの差

ヌル (null)
無効なアドレスの値をポインター・データ項目に割り当てるために使用される形象定数。 NULL を使えるところならばどこでも、NULLS を使用できる。

数字 (* numeric character)
次のような数字に属する文字。 0、1、2、3、4、5、6、7、8、9。

数値データ項目 (numeric data item)
(1) 記述により内容が数字 0 から 9 より選ばれた文字で表される値に制限されるデータ項目。 符号付きである場合、この項目は +、-、または他の表記の演算符号も含むことができます。 (2) カテゴリー数値、内部浮動小数点、または外部浮動小数点のデータ項目。 数値データ項目は、USAGE DISPLAY、NATIONAL、PACKED-DECIMAL、BINARY、COMP、COMP-1、COMP-2、COMP-3、COMP-4、または COMP-5 を持つことができます。

数字編集データ項目 (numeric-edited data item)
印刷出力の際に使用するのに適したフォーマットの数値データを含むデータ項目。データ項目は、外部 10 進数字の 0 から 9 の数字、小数点、コンマ、通貨符号、符号制御文字、その他の編集記号から構成される。 数字編集項目は、USAGE DISPLAY または USAGE NATIONAL のいずれかで表すことができる。

数字関数 (* numeric function)
クラスとカテゴリーは数字だが、考えられる評価のいくつかにおいて整数関数の要件を満たさないような関数。

数値リテラル (* numeric literal)
1 つ以上の数字から構成されるリテラルで、小数点または代数符号あるいはその両方を含むことができる。 小数点は右端の文字であってはならない。 代数符号がある場合には、それが左端の文字でなければならない。

O

オブジェクト・コード (object code)
コンパイラーまたはアセンブラーからの出力。 それ自体が実行可能なマシン・コードか、 またはその種のコードの作成を目的としての処理に適する。

*** OBJECT-COMPUTER**
ENVIRONMENT DIVISION にある段落の名前であり、ここではオブジェクト・プログラムが実行されるコンピューター環境が記述される。

オブジェクト・コンピューター記入項目 (* object computer entry)
ENVIRONMENT DIVISION の OBJECT-COMPUTER 段落内の記入項目。この記入項目には、オブジェクト・プログラムが実行されるコンピューター環境を記述する節が入っている。

項目のオブジェクト (* object of entry)
COBOL プログラムの DATA DIVISION 記入項目内の一連のオペランドと予約語であり、その記入項目のサブジェクトの直後に続く。

オブジェクト・プログラム (object program)
問題を解決するためにデータと相互に作用することを目的とする実行可能なマシン言語命令とその他の要素の集合または グループ。 このコンテキストでは、オブジェクト・プログラムとは一般に、COBOL コンパイラーがソース・プログラム定義を操作した 結果得られるマシン言語である。 あいまいになる危険がない場合には、オブジェクト・プログラム という用語の代わりに プログラム というワードだけが使用される。

オブジェクト時 (*object time)
オブジェクト・プログラムが実行されるとき。 実行時 *(run time)* と同義。

廃止される言語エレメント (* obsolete element)
2002 COBOL 標準 から削除された 85 COBOL 標準 の COBOL 言語エレメント。

ODBC
Open Database Connectivity (ODBC) を参照。

ODO オブジェクト (ODO object)
次の例では、X が OCCURS DEPENDING ON 節のオブジェクト (ODO オブジェクト) である。

```
WORKING-STORAGE SECTION.
01  TABLE-1.
    05  X                   PIC S9.
    05  Y OCCURS 3 TIMES
          DEPENDING ON X    PIC X.
```

ODO オブジェクトの値によって、テーブル内の ODO サブジェクトの数が決まる。

ODO 対象 (ODO subject)
上記の例では、Y が OCCURS DEPENDING ON 節の サブジェクト (ODO サブジェクト) である。 テーブル内の ODO サブジェクトの数である Y の値は、X の値によって決まる。

Open Database Connectivity (ODBC)
さまざまなデータベースおよびファイル・システムのデータへのアクセスを可能にするアプリケーション・プログラミング・インターフェース (API) の仕様。

オープン・モード (* open mode)
OPEN ステートメントが実行されてから、REEL および UNIT 句の指定のない CLOSE ステートメントが実行される前までのファイルの状態。 個々のオープン・モードは、OPEN ステートメントの中で、INPUT、OUTPUT、I-O、または EXTEND のいずれかとして指定する。

オペランド (* operand)
(1) オペランドの一般的な定義は、「操作の対象となるコンポーネント」である。 (2) 本書の目的に沿った言い方をすれば、ステートメントや記入項目の形式中に現れる小文字または日本語で書かれた語 (または語群) はオペランドと見なされ、そのオペランドによって指示されたデータに対して暗黙の参照を行う。

演算、操作 (operation)
オブジェクトに関して要求できるサービス。

演算符号 (* operational sign)
値が正であるか負であるかを示すために数値データ項目または数値リテラルに付けられる代数符号。

オプション・ファイル (optional file)
オブジェクト・プログラムが実行されるたびに必ずしも使用可能でなくてもよいものとして宣言されているファイル。

オプショナル・ワード (* optional word)
言語を読みやすくする目的でのみ特定の形式で含められる予約語。 このようなワードが表示されている形式をソース単位内で使用する場合、そのワードの有無はユーザーが選択できる。

出力ファイル (* output file)
出力モードまたは拡張モードのいずれかでオープンされるファイル。

出力モード (* output mode)
OUTPUT または EXTEND 句の指定のある OPEN ステートメントが実行されてから、 REEL および UNIT 句の指定のない CLOSE ステートメントが実行される前までのファイルの状態。

出力プロシージャー (* output procedure)
SORT ステートメントの実行中にソート機能が完了した後で制御が渡されるステートメントの集合、または MERGE ステートメントの実行中に、要求があればマージ機能がマージ済みの順序になっているレコードのうち次のレコードを選択できるようになった後で制御が渡されるステートメントの集合。

オーバーフロー条件 (overflow condition)
ある演算結果の一部が意図した記憶単位の容量を超えたときに起こる条件。

P

パック 10 進数データ項目 (packed-decimal data item)
内部 *10* 進数データ項目 *(internal decimal data item)* を参照。

埋め込み文字 (padding character)
物理レコード内の未使用文字位置を埋めるのに使用される英数字または国別文字。

ページ (page)
データの物理的分離を表す、出力データの垂直分割。 分離は、内部論理要件または出力メディアの外部特性、あるいはその両方に基づいて行われる。

ページ本体 (* page body)
行を記述できる、または行送りすることができる (またはその両方ができる) 論理ページの部分。

段落 (* paragraph)
PROCEDURE DIVISION では、段落名の後に分離文字ピリオドが続き、その後に 0 個以上の文が続く。IDENTIFICATION DIVISION および ENVIRONMENT DIVISION では、 段落ヘッダーの後に 0 個以上の記入項目が続く。

段落ヘッダー (* paragraph header)
予約語の後に分離文字ピリオドが付いたもので、IDENTIFICATION DIVISION および ENVIRONMENT DIVISION において段落の始まりを示すもの。 IDENTIFICATION DIVISION で許可されている段落ヘッダーは次のとおり。

```
PROGRAM-ID. (Program IDENTIFICATION
   DIVISION)
AUTHOR.
INSTALLATION.
DATE-WRITTEN.
DATE-COMPILED.
SECURITY.
```

ENVIRONMENT DIVISION で許可されている段落ヘッダーは次のとおり。

```
SOURCE-COMPUTER.
OBJECT-COMPUTER.
SPECIAL-NAMES.
REPOSITORY. (Program
   CONFIGURATION SECTION)
FILE-CONTROL.
I-O-CONTROL.
```

段落名 (* paragraph-name)
PROCEDURE DIVISION の中の段落を識別し開始するユーザー定義語。

パラメーター (parameter)
呼び出し側プログラムと呼び出し先プログラムの間で受け渡されるデータ。

句 (* phrase)
連続する 1 つ以上の COBOL 文字ストリングを配列したセットで、COBOL プロシージャー・ステートメントまたは COBOL 節の一部を構成する。

物理レコード (* physical record)
ブロック *(block)* を参照。

ポインター・データ項目 (pointer data item)
アドレス値を保管できるデータ項目。 これらのデータ項目は、USAGE IS POINTER 節を使用してポインターとして明示的に定義される。 ADDRESS OF 特殊レジスターは、ポインター・データ項目として暗黙的に定義されている。 ポインター・データ項目は、 他のポインター・データ項目と等しいかどうかを比較したり、 他のポインター・データ項目に内容を移動することができる。

移植する、ポート (port)
(1) 異なるプラットフォームで実行できるようにコンピューター・プログラムを変更すること。 (2) インターネット・プロトコルでは、Transmission Control Protocol (TCP) プロトコルまたは User Datagram Protocol (UDP) プロトコルと高水準のプロトコルまたはアプリケーションの間の特定の論理結合子。 ポートはポート番号によって識別される。

可搬性 (portability)
あるアプリケーション・プラットフォームから別のアプリケーション・プラットフォームに、ソース・プログラムに比較的わずかな変更を加えるだけでアプリケーション・プログラムを移行できる能力。

基本レコード・キー (* prime record key)
索引付きファイルのレコードを固有なものとして識別する内容を持つキー。

優先順位番号 (* priority-number)
セグメンテーションの目的で、PROCEDURE DIVISION 内のセクションを分類するユーザー定義語。 セグメント番号には 0 から 9 までの文字だけしか使用できない。セグメント番号は 1 桁または 2 桁として表すことができる。

プロシージャー (* procedure)
PROCEDURE DIVISION 内にある 1 つの段落または論理的に連続する段落のグループ、あるいは 1 つのセクションまたは論理的に連続するセクションのグループ。

プロシージャー・ブランチ・ステートメント (* procedure branching statement)
ソース・コードの中にステートメントが書かれている順番どおりに次の実行可能ステートメントに制御の移動をせず、別のステートメントに明示的に制御の移動を引き起こすステートメント。 プロシージャー分岐ステートメントは次のとおり。ALTER、CALL、EXIT、EXIT PROGRAM、GO TO、MERGE (OUTPUT PROCEDURE 句付き)、PERFORM および SORT (INPUT PROCEDURE または OUTPUT PROCEDURE 句付き)、XML PARSE。

PROCEDURE DIVISION
COBOL の部の 1 つで、問題を解決するための命令を記述する。

プロシージャー統合 (procedure integration)
COBOL 最適化プログラムの機能の 1 つであり、実行されるプロシージャーまたは含まれ ているプログラムへの呼び出しを単純化する。

PERFORM プロシージャー統合とは、PERFORM ステートメントが、実行されるプロシージャーによって置き換えられるプロセスのこと。 含まれているプログラムのプロシージャー統合とは、含まれているプログラムへの呼び出しがプログラム・コードによって置き換えられるプロセスのこと。

プロシージャー名 (* procedure-name)
PROCEDURE DIVISION の中にある段落またはセクションに名前を付けるために使用される ユーザー定義語。 プロシージャー名は、段落名 (これは修飾することができる) またはセクション名から構成される。

プロシージャー・ポインター (procedure pointer)
入り口点を指すポインターを保管できるデータ項目。 USAGE IS PROCEDURE-POINTER 節を付けて定義したデータ項目が、 プロシージャーへの入り口点のアドレスを収める。

プロシージャー・ポインター・データ項目 (procedure-pointer data item)
入り口点を指すポインターを保管できるデータ項目。 USAGE IS PROCEDURE-POINTER 節で定義されるデータ項目には、プロシージャー入り口点のアドレスが入っている。一般的に、COBOL のプログラムと通信するために使用される。

プロセス (process)
プログラムの全部または一部の実行中に発生する一連のイベント。 複数のプロセスを並行して実行することができ、1 つのプロセス内で実行されるプログラムはリソースを共用することができる。

プログラム (program)
(1) コンピューターによる処理に適した一連の命令。 処理には、コンパイラーを使用してプログラムの実行準備をすることやランタイム環境を使用してプログラムを実行することが含まれます。 (2) 1 つ以上の相互に関係のあるモジュールの論理アセンブリー。 同じプログラムの複数のコピーを異なるプロセスで実行することができます。

プログラム識別記入項目 (* program identification entry)
IDENTIFICATION DIVISION の PROGRAM-ID 段落内の記入項目であり、プログラム名を指定し、選択されたプログラム属性をプログラムに割り当てる節が入っている。

プログラム名 (program-name)
IDENTIFICATION DIVISION およびプログラム終了マーカーにおいて、COBOL ソース・プログラムを識別するユーザー定義語または英数字リテラル。

プロジェクト (project)
ダイナミック・リンク・ライブラリー (DLL) や他の実行可能ファイル (EXE) などのターゲットを作成するのに必要な、データおよびアクションの完全セット。

疑似テキスト (* pseudo-text)
ソース・プログラムまたは COBOL ライブラリーにおいて、疑似テキスト区切り文字によって区切られた一連のテキスト・ワード、コメント行、または区切り文字スペース (疑似テキスト区切り文字を含まない)。

疑似テキスト区切り文字 (* pseudo-text delimiter)
疑似テキストを区切るために使用される隣接した 2 つの等号文字 (==)。

句読文字 (* punctuation character)
以下のセットに属する文字。

文字	意味
,	コンマ
;	セミコロン
:	コロン
.	ピリオド (終止符)
"	引用符
(	左括弧
)	右括弧
	スペース
=	等号

Q

QSAM (待機順次アクセス方式) (QSAM (Queued Sequential Access Method))
基本順次アクセス方式 (BSAM) の拡張版。 この方式を使用する場合、キューは、処理を待機する入力データ・ブロック、 または処理が終了して補助ストレージまたは出力装置への転送を待機する出力データ・ブロックで形成される。

QSAM ファイル・システム (QSAM file system)
QSAM (待機順次アクセス方式) ファイル・システムでは、固定長レコード、可変長レコード、およびスパン・レコードがサポートされ、オプション binary および quote site rdw を指定して (z/OS FTP を使用して) z/OS から AIX または Linux に転送した QSAM ファイルに直接アクセスできる。QSAM ファイルでは、すべての COBOL データ型がレコード内でサポートされる。

修飾されたデータ名 (* qualified data-name)
データ名と、その後に連結語の OF または IN とデータ名修飾子を続けたものが 1 つ以上のセットで続いて構成される ID。

修飾子 (* qualifier)
(1) レベル標識と関連付けられるデータ名または名前であり、参照の際に、別のデータ名 (修飾子に従属する項目の名前) と 一緒に、または条件名と一緒に使用される。 (2) セクション名。 そのセクションの中で指定されている段落名と共に参照する際に使用される。 (3) ライブラリー名。 そのライブラリーと関連付けられたテキスト名と共に参照する際に使用される。

R

ランダム・アクセス (* random access)
キー・データ項目のプログラム指定値を使って、 相対ファイルまたは索引付きファイルから取り出したり、削除したり、 またはそこに入れたりする論理レコードを識別するアクセス・モード。

レコード (* record)
論理レコード *(logical record)* を参照。

レコード域 (* record area)
DATA DIVISION の FILE SECTION 内のレコード記述項目で記述されるレコードを処理する目的で割り振られるストレージ域。 FILE SECTION では、レコード域の現行の文字位置の数は、明示または暗黙の RECORD 節によって決められる。

レコード記述 (* record description)
レコード記述項目 *(record description entry)* を参照。

レコード記述項目 (* record description entry)
特定のレコードに関連したデータ記述項目全体。 「レコード記述 *(record description)*」と同義。

レコード・キー (record key)
索引付きファイル内のレコードを識別する内容を持つキー。

レコード・キー名 (record-key-name)
索引付きファイルに関連付けられているキーを表すユーザー定義語。

レコード名 (* record-name)
COBOL プログラムの DATA DIVISION 内のレコード記述項目で記述されるレコードに名前を付けるユーザー定義語。

レコード番号 (* record number)
編成が順次であるファイル内のレコードの順序数。

記録モード (recording mode)
ファイル内の論理レコードの形式。 記録モードは、F (固定長)、V (可変長)、S (スパン)、または U (不定形式) とすることができる。

再帰 (recursion)
それ自体を呼び出すプログラム、または、それ自体で呼び出したプログラムのいずれかによって直接あるいは間接に呼び出されるプログラム。

再起可能 (recursively capable)
PROGRAM-ID ステートメントで RECURSIVE 属性が指定されていれば、プログラムは再帰可能である (再帰的に呼び出すことができる)。

リール (reel)
ストレージ・メディアの個別部分。その大きさはインプリメントする人によって決定され、1 つのファイルの一部、1 つのファイルの全部、または任意の個数のファイルが収容される。「ユニット *(unit)*」および「ボリューム *(volume)*」と同義。

再入可能 (reentrant)
プログラムまたはルーチンの属性。この属性によって、ロード・モジュールの 1 つのコピーを複数のユーザーが共用できる。

参照形式 (* reference format)
COBOL ソース・プログラムを記述するに際して標準的な方式を提供する形式。

参照変更 (reference modification)
新規のカテゴリー英数字、カテゴリー DBCS、またはカテゴリー国別のデータ 項目を定義する方法であり、USAGE DISPLAY、DISPLAY-1、または NATIONAL データ項目の左端文字および左端文字位置を基準にした長さを指定して定義する方法です。

参照修飾子 (* reference-modifier)
固有のデータ項目を定義する文字ストリングと分離文字の構文的に正しい組み合わせ。 区切り用の左括弧区切り文字、左端の文字位置、区切り文字のコロン、 任意指定の長さ、および区切り用の右括弧区切り文字を含む。

関係 (* relation)
関係演算子 *(relational operator)* または 比較条件 *(relation condition)* を参照。

比較文字 (* relation character)
以下のセットに属する文字。

文字	意味
>	より大きい
<	より小さい
=	に等しい

比較条件 (* relation condition)
ある算術式、データ項目、英数字リテラル、または索引名の値が、他の算術式、データ項目、英数字リテラル、または索引名の 値と特定の関係があるという命題 (それに対して真理値を判別する)。 関係演算子 *(relational operator)* も参照。

比較演算子 (* relational operator)
比較条件の構造で使用される、予約語、比較文字、連続する予約語のグループ、または 連続する予約語と比較文字のグループ。使用できる演算子とそれらの意味は次のとおり。

文字	意味
IS GREATER THAN	より大きい
IS >	より大きい
IS NOT GREATER THAN	より大きくない
IS NOT >	より大きくない
IS LESS THAN	より小さい
IS <	より小さい
IS NOT LESS THAN	より小さくない
IS NOT <	より小さくない
IS EQUAL TO	に等しい
IS =	に等しい
IS NOT EQUAL TO	に等しくない
IS NOT =	に等しくない
IS GREATER THAN OR EQUAL TO	より大きいか等しい
IS >=	より大きいか等しい
IS LESS THAN OR EQUAL TO	より小さいか等しい
IS <=	より小さいか等しい

相対ファイル (* relative file)
相対編成のファイル。

相対キー (* relative key)
相対ファイルの中の論理レコードを識別するための内容を持つキー。

相対編成 (* relative organization)
各レコードが、レコードのファイル内における論理的順序位置を指定する 0 より大きい整数値によって、固有なものとして識別される永続的な論理ファイル構造。

相対レコード番号 (* relative record number)
相対編成ファイル内でのレコードの序数。 この数値は、整数の数字リテラルとして扱われる。

予約語 (* reserved word)
COBOL ソース・プログラムの中で使用することができるが、ユーザー定義語またはシステム名としてプログラムの中で使用されてはならないワードのリスト中に挙げられている COBOL ワード。

リソース (* resource)
オペレーティング・システムの制御下に置かれており、実行中のプログラムによって使用できる機能またはサービス。

結果の ID (* resultant identifier)
算術演算の結果が収められるユーザー定義のデータ項目。

ルーチン (routine)
コンピューターに操作または一連の関連操作を実行させる、COBOL プログラム内の一連のステートメント。

ルーチン名 (* routine-name)
COBOL 以外の言語で記述されたプロシージャーを識別するユーザー定義語。

RSD ファイル・システム (RSD file system)
RSD (レコード順次区切り) ファイル・システムは、順次ファイルをサポートするワークステーション・ファイル・システムである。 RSD ファイルは、固定長または可変長レコードのすべての COBOL データ・タイプをサポートし、ほとんどのファイル・エディターで編集可能であり、他の言語で作成されたプログラムによって読み取ることができる。 このシステムは順次ファイルのみをサポートする。

実行時 (* run time)
オブジェクト・プログラムが実行されるとき。 「オブジェクト時 *(object time)*」と同義。

ランタイム環境 (runtime environment)
COBOL プログラムが実行される環境。

実行単位 (* run unit)
1 つの独立型オブジェクト・プログラム、あるいは COBOL の CALL ステートメントによって相互作用し、実行時に 1 つのエンティティーとして機能する複数のオブジェクト・プログラム。

S

SBCS
1 バイト文字セット *(SBCS) (single-byte character set (SBCS))* を参照。

範囲終了符号 (scope terminator)
PROCEDURE DIVISION の特定のステートメントの終わりを示す COBOL 予約語。これは明示的なもの (例えば、END-ADD など) であることもあれば、暗黙のもの (分離文字ピリオド) であることもある。

セクション (* section)
ゼロ、1 つ、または複数の段落またはエンティティー (セクション本体と呼ばれる) と、その最初のものの前にセクション・ヘッダーが付いているもの。 各セクションは、 セクション・ヘッダーとそれに関連付けられたセクション本体から構成される。

セクション・ヘッダー (* section header)
後ろに分離文字ピリオドが付いたワードの組み合わせで あり、ENVIRONMENT、DATA、または PROCEDURE の各部において、セクションの始まりを示すもの。 ENVIRONMENT DIVISION および DATA DIVISION では、 セクション・ヘッダーは、予約語の後に分離文字ピリオドを続けたものから構成される。 ENVIRONMENT DIVISION で許可されているセクション・ヘッダーは次のとおり。

```
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
```

DATA DIVISION で許可されているセクション・ヘッダーは次のとおり。

```
FILE SECTION.
WORKING-STORAGE SECTION.
LOCAL-STORAGE SECTION.
LINKAGE SECTION.
```

PROCEDURE DIVISION では、セクション・ヘッダーは、セクション名、その後に続く予約語 SECTION、およびその後の分離文字ピリオドから構成される。

セクション名 (* section-name)
PROCEDURE DIVISION の中にあるセクションに名前を付けるユーザー定義語。

セグメント化 (segmentation)
85 COBOL 標準 分割モジュールに基づく COBOL for Linux の機能。 セグメンテーション機能は、セクション・ヘッダーの優先順位番号を使用して、セクションを固定セグメントまたは独立セグメントに割り当てる。 セグメント種別は、セグメントに含まれるプロシージャーが初期状態の制御を受け取るか、最後に使われた状態の制御を受け取るかに影響を及ぼす。

選択構造 (selection structure)
条件が真であるか偽であるかに応じて、ある一連のステートメントか、または別の一連のステートメントが実行されるというプログラムの処理ロジック。

文 (* sentence)
1 つ以上のステートメントの並びで、その最後のものは、分離文字ピリオドで終了する。

別々にコンパイルされたプログラム (* separately compiled program)
あるプログラムをそこに含まれたプログラムと共に、 他のすべてのプログラムとは別個にコンパイルしたときのそのプログラム。

区切り文字 (* separator)
文字ストリングを区切るために使用される、1 文字または連続する 2 文字。

区切り文字のコンマ (* separator comma)
文字ストリングを区切るために使われる、後ろに 1 つのスペースが続く 1 つのコンマ (,)。

分離文字ピリオド (* separator period)
文字ストリングを区切るために使われる、後ろに 1 つのスペースが続く 1 つのピリオド (.)。

区切り文字のセミコロン (* separator semicolon)
文字ストリングを区切るために使われる、後ろに 1 つのスペースが続く 1 つのセミコロン (;)。

順序構造 (sequence structure)
一連のステートメントが、順序どおりに実行されるプログラムの処理ロジック。

順次アクセス (* sequential access)
ファイル内のレコードの並び方によって規定されている、 論理レコードの連続した前後関係順に、論理レコードをファイルから取り出したり、 ファイルに書き込んだりするアクセス・モード。

順次ファイル (* sequential file)
順次編成のファイル。

順次編成 (* sequential organization)
レコードがファイルに書き込まれるときに確定されたレコードの前後関係によって識別されるような永続的な論理ファイル構造。

逐次探索 (serial search)
最初のメンバーから始めて最後のメンバーで終わるように、 ある集合のメンバーが連続的に検査される探査方法。

SFS ファイル・システム (SFS file system)
CICS の SFS (Structured File Server) ファイル・システムは、順次ファイル・アクセス、相対ファイル・アクセス、およびキー索引付きファイル・アクセスをサポートするレコード単位のファイル・システムである。

共用ライブラリー (shared library)
リンカーによって作成され、少なくとも 1 つのサブルーチンを含み、複数のプロセスで使用できるライブラリー。 プログラムとサブルーチンは従来どおりリンクされるが、異なるサブルーチンに共通する

コードは結合されて 1 つのライブラリー・ファイルに入れられる。このライブラリー・ファイルは実行時にロード可能で、多数のプログラムで共用できる。 この共用ライブラリー・ファイルを識別するキーは、各サブルーチンのヘッダーにある。

符号条件 (* sign condition)
データ項目や算術式の代数値が、0 より小さいか、大きいか、または等しいかという命題で、それに関して真理値が判別できる。

シグニチャー (signature)
ある操作とそのパラメーターの名前。

単純条件 (* simple condition)
以下のセットから選択される任意の単一条件。

- 比較条件
- クラス条件
- 条件名条件
- スイッチ状況条件
- 符号条件

条件 *(condition)* および 単純否定条件 *(negated simple condition)* も参照。

1 バイト文字セット (SBCS)(single-byte character set (SBCS))
各文字が 1 バイトで表現される文字のセット。*ASCII* および *EBCDIC* (拡張 *2* 進化 *10* 進コード*) (EBCDIC (Extended Binary-Coded Decimal Interchange Code))* も参照。

遊びバイト (レコード内) (slack bytes (within records))
複数の基本データ項目を正しく位置合わせするために、コンパイラーによってデータ項目間に挿入されるバイト。 遊びバイトには意味のあるデータは含まれない。 正しい位置合わせを行うために遊びバイトが必要なときは、SYNCHRONIZED 節によって、コン パイラーに遊びバイトを挿入させる。

遊びバイト (レコード間) (slack bytes (between records))
複数の基本データ項目を正しく位置合わせするために、プログラマーによってファイルのブロック化論理レコードの間に挿入されるバイト。 場合によっては、レコード間に遊びバイトを挿入することによってバッファー内で処理されるレコードのパフォーマンスが改善される。

ソート・ファイル (* sort file)
SORT ステートメントによってソートされるレコードの集まり。ソート・ファイルは、ソート機能によってのみ作成され使用される。

ソート・マージ・ファイル記述項目 (* sort-merge file description entry)
DATA DIVISION の FILE SECTION の中にある記入項目。レベル標識 SD と、それに続くファイル名、および、必要に応じて、次に続く一連のファイル節から構成される。

*** SOURCE-COMPUTER**
ENVIRONMENT DIVISION にある段落の名前であり、ここではソース・プログラムがコンパイルされるコンピューター環境が記述される。

コンパイル用コンピューター記入項目 (* source computer entry)
ENVIRONMENT DIVISION の SOURCE-COMPUTER 段落内の記入項目であり、ソース・プログラムがコンパイルされるコンピューター環境を記述する節が入っている。

ソース項目 (* source item)
SOURCE 節によって指定される ID で、印刷可能な項目の値を提供する。

ソース・プログラム (source program)
ソース・プログラムは、他の形式や記号を使用して表現することができるが、本書では、構文的に正しい COBOL ステートメントの集合を常に指している。 COBOL ソース・プログラムは、IDENTIFICATION DIVISION または COPY ステートメントで開始され、指定された場合はプログラム終了マーカーで終了するか、または追加のソース・プログラム行なしで終了する。

ソース単位 (source unit)
COBOL ソース・コードの 1 単位で、個別にコンパイルできる。プログラム。 コンパイル単位 とも呼ばれる。

特殊文字 (special character)
以下のセットに属する文字。

文字	意味
+	正符号
-	負符号 (-) (ハイフン)
*	アスタリスク
/	斜線 (スラッシュ)
=	等号
$	通貨符号
,	コンマ
;	セミコロン
.	ピリオド (小数点、終止符)
"	引用符
'	アポストロフィ
(	左括弧
)	右括弧
>	より大きい
<	より小さい
:	コロン
_	下線

SPECIAL-NAMES
ENVIRONMENT DIVISION にある段落の名前。この段落では、環境名がユーザー指定の簡略名と関連付けられる。

特殊名記入項目 (* special names entry)
ENVIRONMENT DIVISION の SPECIAL-NAMES 段落内の記入項目。この記入項目は、通貨記号を指定したり、小数点を選択したり、シンボリック文字を指定したり、インプリメントする人の名前をユーザー指定の簡略名と関連付けたり、英字名を文字セットまたは照合シーケンスと関連付けたり、クラス名を一連の文字と関連付けたりするための手段を提供する。

特殊レジスター (* special registers)
コンパイラーの生成する特定のストレージ域のことで、その基本的な使用法は、具体的な COBOL 機能を使用したときに作り出される情報を記憶することである。

ステートメント (* statement)
COBOL ソース・プログラムに書かれる、動詞を冒頭に置いた、ワード、リテラル、および区切り記号の構文的に正しい組み合わせ。

STL ファイル・システム (STL file system)
標準言語ファイル・システムは、COBOL 用のネイティブ・ワークステーション・ファイル・システムである。 このシステムは、順次ファイル、相対ファイル、および索引ファイルをサポートする。

構造化プログラミング (structured programming)
コンピューター・プログラムを編成してコーディングするための技法であり、この技法 では、プログラムはセグメントの階層で構成され、それぞれのセグメントには 1 つの 入り口点と 1 つの出口点がある。 制御は、構造の下方へと渡され、 階層内のより上位レベルへの無条件ブランチは行われない。

項目のサブジェクト (* subject of entry)
DATA DIVISION の記入項目内において、レベル標識またはレベル番号の直後に現れるオペランドまたは予約語。

サブプログラム (* subprogram)
呼び出し先プログラム *(called program)* を参照。

添え字 (* subscript)
整数、(オプションで演算子 + または - 付きの整数が続く) データ名、あるいは (オプションで演算子 + または - 付きの整数が続く) 索引名のいずれかによって表されるオカレンス番号。 これによりテーブル内の特定のエレメントを識別します。 可変数の引数を認める関数では、添え字付き ID を関数引数として使用する場合は、添え字に ALL を使用することができる。

添え字付きデータ名 (* subscripted data-name)
データ名とその後の括弧で囲まれた 1 つ以上の添え字から構成される ID。

置換文字 (substitution character)
ソース・コード・ページからターゲット・コード・ページへの変換の際に、ターゲット・コード・ページで定義されていない文字を表すのに使用される文字。

サロゲート・ペア (surrogate pair)
UTF-16 形式のユニコードで、共に 1 つのユニコード図形文字を表すエンコード方式ペアの単位。 ペアの最初の単位は上位サロゲート と呼ばれ、第 2 の単位は下位サロゲート と 呼ばれる。 上位サロゲートのコード値の範囲は、X'D800' から X'DBFF' である。 下位サロゲートのコード値の範囲は、X'DC00' から X'DFFF' である。 サロゲート・ペアは、Unicode 16 ビット・コード化 文字セットに適合する文字を 65,536 文字を超えて提供する。

スイッチ状況条件 (switch-status condition)
オンまたはオフに設定可能な UPSI スイッチが、特定の状況に設定されているという命題で、これに関して真理値を判別することができる。

シンボリック文字 (* symbolic-character)
ユーザー定義の形象定数を指定するユーザー定義語。

構文 (syntax)
(1) 意味や解釈および使用の方法に依存しない、文字同士または文字のグループ同士の間の関係。 (2) 言語における表現の構造。 (3) 言語構造を支配する規則。 (4) 記号相互の関係。 (5) ステートメントの構築にかかわる規則。

SYSADATA
ADATA コンパイラー・オプションが有効な場合に生成される追加のコンパイル情報のファイル。

SYSIN
1 次コンパイラー入力ファイル。

SYSLIB
2 次コンパイラー入力ファイル。LIB コンパイラー・オプションが有効な場合に処理される。

SYSPRINT
コンパイラー・リスト・ファイル。

システム名 (* system-name)
オペレーティング環境と連絡し合うために使用される COBOL ワード。

T

テーブル (* table)
DATA DIVISION の中で OCCURS 節によって定義される、論理的に連続するデータ項目の集合。

テーブル・エレメント (* table element)
テーブルを構成する繰り返し項目の集合に属するデータ項目。

テキスト名 (* text-name)
ライブラリー・テキストを識別するユーザー定義語。

テキスト・ワード (* text word)
以下のいずれかの文字から成る COBOL ライブラリー、ソース・プログラム、または疑似テキスト内のマージン A およびマージン R の間の、1 文字または連続した文字のシーケンス。

- スペース以外の区切り記号、疑似テキスト区切り文字、英数字リテラルの開始と終了の区切り文字。ライブラリー、ソース・プログラム、または疑似テキスト内のコンテキストに関係なく、右括弧文字と左括弧文字は常に テキスト・ワードと見なされる。

- リテラル。英数字リテラルの場合には、そのリテラルの境界となる開始の引用符と終了の引用符を含むリテラル。
- コメント行および区切り記号によって囲まれたワード COPY を除く、その他の連続する一連の COBOL 文字で、区切り記号でもリテラルでもないもの。

スレッド (thread)
プロセスの制御下にあるコンピューター命令のストリーム (プロセス内のアプリケーションによって開始される)。

トークン (token)
COBOL エディターでは、プログラムにおける意味の単位。 トークンには、データ、言語キーワード、ID、またはその他の言語構文の一部を含めることができる。

トップダウン設計 (top-down design)
関連付けられた諸機能が、構造の各レベルで実行されるようにする階層構造を使ったコンピューター・プログラムの設計。

トップダウン開発 (top-down development)
構造化プログラミング *(structured programming)* を参照。

トレーラー・ラベル (trailer-label)
(1) 記録メディア・ユニットのデータ・レコードの後にある、ラベル。 (2)「ファイル終わりラベル *(end-of-file label)*」の同義語。

トラブルシューティング (troubleshoot)
コンピューター・ソフトウェアの使用中に問題を検出し、突き止め、除去すること。

真の値 (* truth value)
2 つの値 (真または偽) のどちらか一方によって、条件評価の結果を表したもの。

U

単項演算子 (* unary operator)
正符号 (+) または負符号 (-)。 算術式の変数や算術式の左括弧の前に置き、それぞれ +1 または -1 を式に乗算する。

Unicode
現代世界の各国の言語で記述されるテキストの交換、処理、表示をサポートする汎用文字エンコード標準。 UTF-8、UTF-16、UTF-32 など、Unicode を表現する複数のエンコード・スキームが ある。 COBOL for Linux では、国別データ・タイプの表記としてリトル・エンディアン・フォーマットの UTF-16 を使用して Unicode をサポートしている。

URI (Uniform Resource Identifier (URI))
リソースを一意に指す文字のシーケンスのことで、COBOL for Linux では、名前空間の ID。 URI 構文は、文書「*Uniform Resource Identifier (URI): Generic Syntax*」で定義されています。.

ユニット (unit)
直接アクセスのモジュールであり、その大きさは IBM によって決められている。

不成功の実行 (* unsuccessful execution)
ステートメントの実行が試みられたが、そのステートメントに指定された操作すべてを実行できなかったこと。 あるステートメントの実行不成功は、そのステートメントによって参照されるデータには影響を及ぼさないが、状況表示には影響を与える可能性がある。

UPSI スイッチ (UPSI switch)
ハードウェア・スイッチの機能を実行するプログラム・スイッチ。 UPSI-0 から UPSI-7 まで、8 つが提供される。

URI
URI を参照。

ユーザー定義語 (* user-defined word)
節やステートメントの形式を満たすためにユーザーが提供する必要のある COBOL ワード。

V

変数 (* variable)
オブジェクト・プログラムの実行によって変更を受ける可能性のある値を持つデータ項目。 算術式で使われる変数は、数字基本項目でなければならない。

可変長項目 (variable-length item)
OCCURS 節の DEPENDING 句で記述された表を含んだグループ項目。

可変長レコード (* variable-length record)
ファイル記述項目またはソート・マージ・ファイル記述項目が、 文字位置の数が可変であるレコードを許容しているファイルに関連付けられているレコード。

可変オカレンス・データ項目 (* variable-occurrence data item)
可変オカレンス・データ項目とは、反復される回数が可変であるテーブル・エレメントを言う。 そのような項目は、そのデータ記述記入項目内に OCCURS DEPENDING ON 節を持っているか、またはそのような項目に従属していなければならない。

可変位置グループ (* variably located group)
同じレコード内の可変長テーブルに続くグループ項目 (可変長テーブルに従属するわけではない)。 グループ項目は、英数字グループでも国別グループでも構いません。

可変位置項目 (* variably located item)
同じレコード内の可変長テーブルに続くデータ項目 (可変長テーブルに従属するわけではない)。

動詞 (* verb)
COBOL コンパイラーまたはオブジェクト・プログラムによってとられる処置を表すワード。

ボリューム (volume)
外部ストレージのモジュール。 テープ装置の場合はリール、直接アクセス装置の場合はユニット。

VSAM ファイル・システム (VSAM file system)
COBOL の順次編成、相対編成、および索引編成をサポートするファイル・システム。

VSAM
STL ファイル・システム または *SFS* ファイル・システム の総称。

W

Web サービス (web service)
特定のタスクを実行し、HTTP や SOAP といったオープン・プロトコルを介してアクセス可能なモジュラー・アプリケーション。

空白文字 (white space)
文書にスペースを挿入する文字。 空白文字には以下のものがある。

- スペース
- 水平タブ
- 復帰
- 改行
- 次の行

Unicode 標準では上記のように呼ばれる。

ウィンドウ表示日付フィールド (windowed date field)
ウィンドウ表示 (2 桁) 年を含む日付フィールド。 日付フィールド *(date field)* および ウィンドウ表示西暦年 *(windowed year)* も参照。

ウィンドウ表示西暦年 (windowed year)
2 桁の年だけから構成される日付フィールド。この 2 桁の年は、世紀ウィンドウを使用して解釈できる。 例えば、10 は 2010 として解釈できる。 世紀ウィンドウ *(century window)* も参照。 拡張西暦年 *(expanded year)* と比較。

ワード (* word)
ユーザー定義語、システム名、予約語、または関数名を形成する、30 文字を超えない文字ストリング。

*** WORKING-STORAGE SECTION**
独立項目または WORKING-STORAGE レコード、あるいはその両方から構成される、WORKING-STORAGE データ項目を記述する DATA DIVISION のセクション。

ワークステーション (workstation)
コンピューターの総称 (パーソナル・コンピューター、3270 端末、インテリジェント・ワークステーション、および UNIX 端末を含む)。ワークステーションはメインフレームまたはネットワークに接続されることがよくある。

ラッパー (wrapper)
オブジェクト指向コードとプロシージャー指向コード間のインターフェースを提供するオブジェクト。 ラッパーを使用すると、他のシステムがプログラムを再利用したり、プログラムにアクセスしたりできるようになる。

X

x
PICTURE 節内の記号であり、コンピューターの有する文字セットの任意の文字を含めることができる。

XML
Extensible Markup Language。 マークアップ言語を定義するための標準メタ言語。SGML から派生した、SGML のサブセットである。 XML では、SGML の複雑で使用頻度の低い部分が省略され、文書タイプを扱うアプリケーションの作成、構造化情報の作成および管理、異種コンピューター・システム間での 構造化情報の伝送および共用がはるかに容易になっている。 XML を使用するときに、SGML で必要とされるような堅固なアプリケーションや 処理は不要である。 XML は、World Wide Web Consortium (W3C) の主導で開発された。

XML データ (XML data)
XML エレメントを持つ階層構造に編成されたデータ。 データ定義は XML エレメント・タイプ宣言で定義される。

XML 宣言 (XML declaration)
使用している XML のバージョンや文書のエンコードなど、XML 文書の特性を指定する XML テキスト。

XML 文書 (XML document)
W3C XML 規格で定義されているとおり正しい形式のデータ・オブジェクト。

XML ネーム・スペース (XML namespace)
W3C XML ネーム・スペース仕様によって定義されたメカニズムで、エレメント名および属性名の集まりの有効範囲を制限する。 一意的に選択された XML 名前空間によって、複数の XML 文書または XML 文書内の複数のコンテキストでエレメント名または属性名が一意的に識別されます。

Y

年フィールド拡張 (year field expansion)
2 桁の年の日付フィールドを、ファイルおよびデータベースの中で完全な 4 桁の年になるように明示的に拡張した後、そのフィールドをプログラムの中で拡張形式で使用する方法。 これは、2 桁の年を使用していたアプリケーションに対して確実に信頼できる日付処理を行う唯一の方法である。

Z

ゾーン 10 進数データ項目 (zoned decimal data item)
暗黙的または明示的に USAGE DISPLAY として記述されており、PICTURE の記号 9、S、P、および V の有効な組み合わせを含んでいる、外部 10 進数データ項目。 ゾーン 10 進数データ項目の内容は、文字 0 から 9 で表され、必要に応じて符号が付きます。 PICTURE ストリングが符号を指定しており、SIGN IS SEPARATE 節が指定されている場合、符号は 文字 + または - として表されます。 SIGN IS SEPARATE が指定されていない場合、符号は、符号位置の最初の 4 ビットをオーバーレイする 1 つの 16 進数字です (先行または末尾)。

#

77 レベル記述記入項目 (77-level-description-entry)
レベル番号 77 を持つ不連続データ項目を記述するデータ記述記入項目。

85 COBOL 標準 (85 COBOL Standard)
以下の標準によって定義された COBOL 言語。

- モジュール「*ANSI INCITS 23-1985, Programming languages - COBOL*」は「*ANSI INCITS 23a-1989, Programming Languages - COBOL - Intrinsic Function Module for COBOL*」に改訂されました。

- 「*ISO 1989:1985, Programming languages - COBOL*」は 「*ISO/IEC 1989/AMD1:1992, Programming languages - COBOL: Intrinsic function module*」に改訂されました。

2002 COBOL 標準 (2002 COBOL Standard)
以下の標準によって定義された COBOL 言語。

- *INCITS/ISO/IEC 1989-2002, Information technology - Programming languages - COBOL*

2014 COBOL 標準 (2014 COBOL Standard)
以下の標準によって定義された COBOL 言語。

- *INCITS/ISO/IEC 1989:2014, Information technology - Programming languages, their environments and system software interfaces - Programming language COBOL*

リソース・リスト

COBOL for Linux 資料

「インストール・ガイド」GC43-5391-00

「言語解説書」SC43-5386-00

「プログラミング・ガイド」SC43-5390-00

サポート

COBOL for Linux の使用に関して問題がある場合は、IBM Support Web サイトを参照してください。このサイトでは最新のサポート情報が提供されています。

関連資料

DB2® for Linux, UNIX, and Windows

以下の資料が IBM 資料にあります。

- コマンド解説書
- データベース: 管理の概念および構成リファレンス
- *Db2 V11.1 for Linux、UNIX、および Windows の SQL* 解説書

TXSeries for Multiplatforms

- *IBM TXSeries for Multiplatforms* 資料

IBM CICS TX on Cloud

- *IBM CICS TX on Cloud* 資料

Unicode および文字表現

- *Unicode*、www.unicode.org/
- *International Components for Unicode: Converter Explorer*、http://demo.icu-project.org/icu-bin/convexp/
- *Character Data Representation Architecture: Reference and Registry*、http://www-01.ibm.com/software/globalization/cdra/

XML

- *Extensible Markup Language (XML)*、www.w3.org/XML/
- *Namespaces in XML 1.0*、www.w3.org/TR/xml-names/
- *Namespaces in XML 1.1*、www.w3.org/TR/xml-names11/
- *XML specification*、www.w3.org/TR/xml/

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: 5737-L11

SC43-5386-00