Enterprise COBOL for z/OS and OS/390

SC88-9117-00
(英文原典：SC27-1408-00)

言語解説書

バージョン 3.1

Enterprise COBOL for z/OS and OS/390

SC88-9117-00
(英文原典：SC27-1408-00)

言語解説書

バージョン 3.1

お願い

本書の情報および本書がサポートする製品をご使用になる前に、579ページの付録H、『特記事項』を必ずお読みください。

本書は、IBM Enterprise COBOL for z/OS and OS/390 バージョン 3.1 (プログラム番号 5655-G53)、および新版において特に断りのないかぎり、それ以降のすべてのリリースおよびモディフィケーション・レベルにも適用されます。製品のレベルに合った正しい版をご使用ください。

本書において、日本では発表されていない IBM製品（機械およびプログラム）、プログラミング、およびサービスについて言及または説明する場合があります。しかし、このことは、IBM がこのようなIBM 製品、プログラミング、およびサービスを、必ずしも日本で発表する意図であることを示すものではありません。

本マニュアルに関するご意見やご感想は、次のURLからお送りください。今後の参考にさせていただきます。

`http://www.ibm.com/jp/manuals/main/mail.html`

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

`http://www.ibm.com/jp/manuals/` の「ご注文について」をご覧ください。

(URLは、変更になる場合があります)

原　典： SC27-1408-00
Enterprise COBOL for z/OS and OS/390
Language Reference
Version 3 Release 1

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷　2002.3

©Copyright International Business Machines Corporation 1991, 2001. All rights reserved.

©Copyright IBM Japan 2002

目次

本書について x
IBM 拡張機能 x
廃止される言語エレメント x
構文表記法 xi
DBCS の表記 xii
謝辞 xiii
変更の要約 xiv
Enterprise COBOL 第 1 版 (2001 年 11 月) xiv

COBOL 言語の構造 1

文字 2
文字エンコード・ユニット 3

文字ストリング 5
1 バイト文字から構成される COBOL ワード 5
DBCS 文字が含まれる COBOL ワード 6
ユーザー定義語 6
システム名 8
関数名 8
予約語 8
形象定数 9
特殊レジスター 11
リテラル 25
PICTURE 文字ストリング 34
コメント 34

区切り文字 36

セクションと段落 39
文、ステートメント、および記入項目 39

参照フォーマット 41
シーケンス番号域 41
標識域 41
領域 A 42
領域 B 44
領域 A または領域 B 46

名前のスコープ 48
名前のタイプ 48
外部および内部リソース 50
名前の解決 51

データ名、コピー・ライブラリー、および手続き部名の参照 53

© Copyright IBM Corp. 1991, 2001

参照の固有性 53

制御の移動 64

2000 年言語拡張および日付フィールド 66
2000 年言語拡張の構文 66
用語と概念 67

COBOL ソース単位構造 71

COBOL プログラムの構造 72
ネストされたプログラム 74

COBOL クラス定義構造 77

COBOL メソッド定義構造 81

見出し部 83

見出し部 84
PROGRAM-ID 段落 86
CLASS-ID 段落 88
FACTORY 段落 89
OBJECT 段落 90
METHOD-ID 段落 90
オプションの段落 91

環境部 93

構成セクション 94
SOURCE-COMPUTER 段落 95
OBJECT-COMPUTER 段落 96
SPECIAL-NAMES 段落 97
ALPHABET 文節 100
SYMBOLIC CHARACTERS 文節 103
CLASS 文節 103
CURRENCY SIGN 文節 104
DECIMAL-POINT IS COMMA 文節 105
REPOSITORY 段落 105

入出力セクション 108
FILE-CONTROL 段落 109
SELECT 文節 112
ASSIGN 文節 112
RESERVE 文節 115
ORGANIZATION 文節 116

PADDING CHARACTER 文節 118
RECORD DELIMITER 文節 119
ACCESS MODE 文節 119
RECORD KEY 文節 121
ALTERNATE RECORD KEY 文節 122
RELATIVE KEY 文節 123
PASSWORD 文節 124
FILE STATUS 文節 125
I-O-CONTROL 段落 126
RERUN 文節 127
SAME AREA 文節 128
SAME RECORD AREA 文節 129
SAME SORT AREA 文節 130
SAME SORT-MERGE AREA 文節 130
MULTIPLE FILE TAPE 文節 130
APPLY WRITE-ONLY 文節 131

データ部 133

データ部の概要 134
ファイル・セクション 135
ワーキング・ストレージ・セクション 135
ローカル・ストレージ・セクション 137
リンケージ・セクション 137
データ単位 138
データの関係 139

データ部 - ファイル記述項目 147
ファイル・セクション 149
EXTERNAL 文節 150
GLOBAL 文節 151
BLOCK CONTAINS 文節 151
RECORD 文節 153
LABEL RECORDS 文節 155
VALUE OF 文節 156
DATA RECORDS 文節 156
LINAGE 文節 156
RECORDING MODE 文節 158
CODE-SET 文節 160

データ部 - データ記述記入項目 161
フォーマット 1 161
フォーマット 2 162
フォーマット 3 162
レベル番号 162
BLANK WHEN ZERO 文節 163
DATE FORMAT 文節 164
EXTERNAL 文節 169

GLOBAL 文節 170
JUSTIFIED 文節 171
OCCURS 文節 171
PICTURE 文節 177
REDEFINES 文節 192
RENAMES 文節 196
SIGN 文節 198
SYNCHRONIZED 文節 199
USAGE 文節 206
VALUE 文節 214

手続き部 221

手続き部の構造 223
メソッド手続き部の要件 223
手続き部のヘッダー 224
宣言部分 227
プロシージャー 228
算術式 230
条件式 235
ステートメントのカテゴリー 257
ステートメントによる操作 261

手続き部のステートメント 274
ACCEPT ステートメント 274
ADD ステートメント 278
ALTER ステートメント 281
CALL ステートメント 283
CANCEL ステートメント 290
CLOSE ステートメント 292
COMPUTE ステートメント 296
CONTINUE ステートメント 298
DELETE ステートメント 299
DISPLAY ステートメント 301
DIVIDE ステートメント 304
ENTRY ステートメント 307
EVALUATE ステートメント 308
EXIT ステートメント 312
EXIT METHOD ステートメント 313
EXIT PROGRAM ステートメント 314
GOBACK ステートメント 315
GO TO ステートメント 316
IF ステートメント 318
INITIALIZE ステートメント 320
INSPECT ステートメント 323
INVOKE ステートメント 332
MERGE ステートメント 340
MOVE ステートメント 346

MULTIPLY ステートメント 352
OPEN ステートメント 354
PERFORM ステートメント 360
READ ステートメント 370
RELEASE ステートメント 377
RETURN ステートメント 379
REWRITE ステートメント 381
SEARCH ステートメント 384
SET ステートメント 391
SORT ステートメント 398
START ステートメント 405
STOP ステートメント 408
STRING ステートメント 409
SUBTRACT ステートメント 414
UNSTRING ステートメント 417
WRITE ステートメント 425
| XML PARSE ステートメント 433

組み込み関数 439

組み込み関数 441
関数の指定 441
関数定義 448
ACOS 452
ANNUITY 453
ASIN 454
ATAN 455
CHAR 456
COS 457
CURRENT-DATE 458
DATE-OF-INTEGER 459
DATE-TO-YYYYMMDD 460
DATEVAL 461
DAY-OF-INTEGER 463
DAY-TO-YYYYDDD 464
| DISPLAY-OF 465
FACTORIAL 467
INTEGER 468
INTEGER-OF-DATE 469
INTEGER-OF-DAY 470
INTEGER-PART 471
LENGTH 472
LOG 473
LOG10 474
LOWER-CASE 475
MAX 476
MEAN 477
MEDIAN 478

MIDRANGE . . . 479
MIN . . . 480
MOD . . . 481
NATIONAL-OF . . . 482
NUMVAL . . . 483
NUMVAL-C . . . 484
ORD . . . 486
ORD-MAX . . . 487
ORD-MIN . . . 488
PRESENT-VALUE . . . 489
RANDOM . . . 490
RANGE . . . 491
REM . . . 492
REVERSE . . . 493
SIN . . . 494
SQRT . . . 495
STANDARD-DEVIATION . . . 496
SUM . . . 497
TAN . . . 498
UNDATE . . . 499
UPPER-CASE . . . 500
VARIANCE . . . 501
WHEN-COMPILED . . . 502
YEAR-TO-YYYY . . . 503
YEARWINDOW . . . 504

コンパイラー指示ステートメント . . . 505

コンパイラー指示ステートメント . . . 506
BASIS . . . 506
CBL (PROCESS) ステートメント . . . 508
*CONTROL (*CBL) ステートメント . . . 509
COPY ステートメント . . . 511
DELETE ステートメント . . . 518
EJECT ステートメント . . . 519
ENTER ステートメント . . . 520
INSERT ステートメント . . . 520
READY または RESET TRACE ステートメント . . . 521
REPLACE ステートメント . . . 522
SERVICE LABEL ステートメント . . . 526
SERVICE RELOAD ステートメント . . . 526
SKIP1/2/3 ステートメント . . . 526
TITLE ステートメント . . . 528
USE ステートメント . . . 529

付録 . . . 535

付録A. IBM 拡張 536

付録B. コンパイラーの制限事項 551

付録C. EBCDIC および ASCII の照合シーケンス 556
EBCDIC 照合シーケンス 556
米国英語 ASCII コード・ページ (ISO 646) 560

付録D. ソース言語のデバッグ 564
デバッグ行のコーディング 564
デバッグ・セクションのコーディング 564
DEBUG-ITEM 特殊レジスター 565
コンパイル時スイッチの活動化 565
オブジェクト時スイッチの活動化 565

付録E. 予約語 566

付録F. ASCII に関する考慮事項 573
環境部 573
データ部 575
手続き部 575

付録G. 業界仕様 577

付録H. 特記事項 579
プログラミング・インターフェース情報 579
商標 579

付録I. リソース・リスト 581

用語集 585

索引 604

本書について

本書は、IBM Enterprise COBOL for z/OS and OS/390 (「Enterprise COBOL」と呼びます) の構文を紹介したものです。

本書は、*Enterprise COBOL プログラミング・ガイド* と併せて使用してください。

IBM 拡張機能

IBM 拡張機能は、通常、ANSI および ISO の COBOL 標準に機能、構文、または規則を追加するものです。拡張機能には、規則の小さな緩和から、大きな機能の追加までさまざまなものがあり、これには、XML サポート、Unicode サポート、 Java とのインターオペラビリティーのためのオブジェクト指向 COBOL、DBCS 文字の処理などが含まれます。

536ページの付録A、『IBM 拡張』に、IBM 拡張機能のリストがあります。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、付録および *Enterprise COBOL プログラミング・ガイド* で解説されているコンパイラー・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、標準 COBOL 85 言語エレメントのうち、ANSI および ISO の標準 (本書では*標準 COBOL* と呼びます) の次バージョンからは削除される予定のエレメントを指します。

注: このことは、これらの言語エレメントが Enterprise COBOL の将来のリリースから除去されることを意味するわけでは**ありません**。

ANSI および ISO の COBOL 標準の次バージョンから削除される予定の言語エレメントを、以下に示します。

- ALTER ステートメント
- AUTHOR 段落
- コメント項目
- DATA RECORDS 文節
- 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 文節
- MEMORY SIZE 文節
- MULTIPLE FILE TAPE 文節
- RERUN 文節
- REVERSED 句
- SECURITY 段落
- 分割モジュール

© Copyright IBM Corp. 1991, 2001

- 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 文節
- 形象定数の ALL リテラル (数字または数字編集項目と関連している場合、および 1 より大きい長さと関連している場合)

構文表記法

本書に示されている構文は、以下に定義されている構造に従って記述されています。

- 構文図は、左から右、上から下へと線をたどってください。

  ▶▶── 記号は、構文図の開始点を表しています。

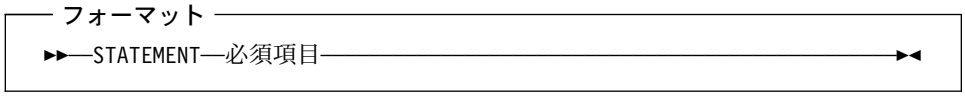
  ──▶ 記号は、構文図が次の線に続くことを示しています。

  ▶── 記号は、構文図が前の線から続いていることを示しています。

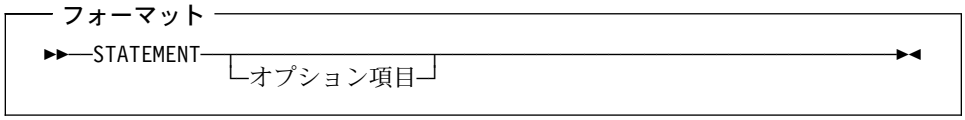
  ──▶◀ 記号は、構文図の終わりを示しています。

  構文が完全なステートメントではなく、構文上の構成単位を示している図は、▶── 記号で始まり、──▶ 記号で終わります。

- 必須項目は、水平線 (主経路) と同じ高さに示されます。

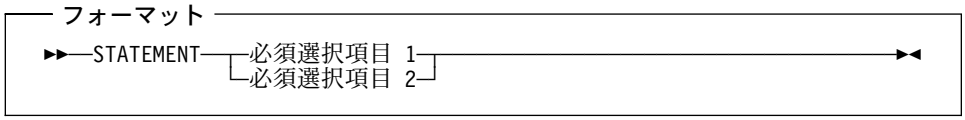


- オプション項目は、主経路の下側に示されます。

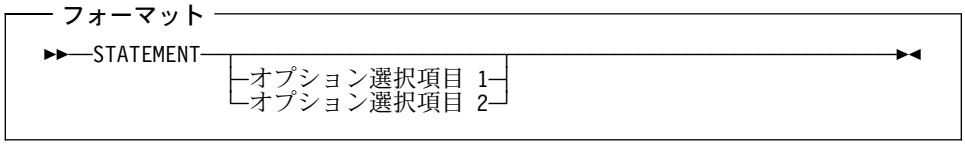


- 複数の項目から選択できる場合には、それらの項目は縦方向に重ねて示されます。

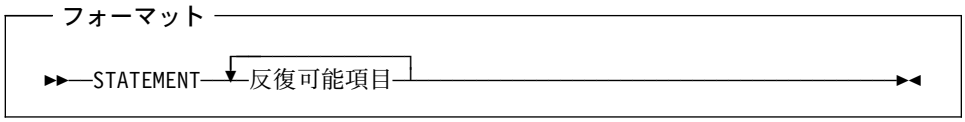
  それらの項目のうち 1 つを選択**しなければならない**場合には、それらの項目のうちの 1 つが主経路と同じ高さに示されます。



  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幹線の下に示されます。



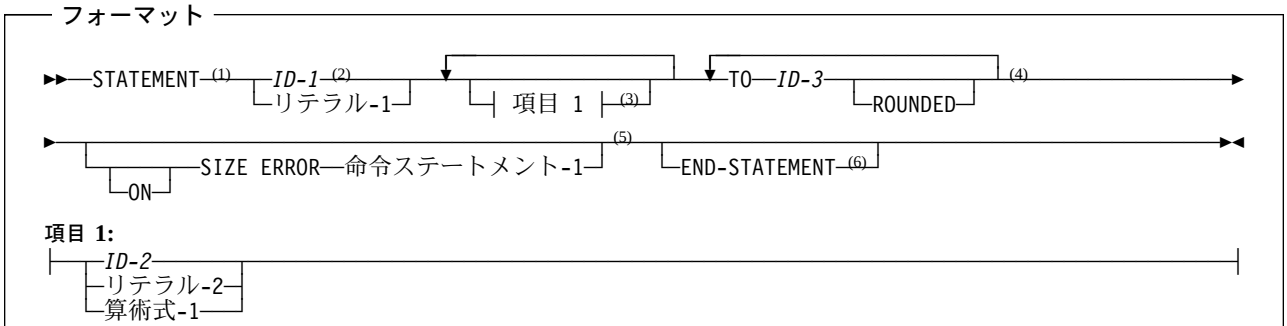
- 幹線から分岐して左へ戻る矢印は、反復可能な項目を示しています。



  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。

- 変数は日本語または小文字で示されます (たとえば 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。



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 区切り文字で区切ることによって、各記入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記入選択項目は、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、各記入項目を分ける 1 つまたは複数の COBOL 区切り文字によって繰り返すことができます。各記入項目にはキーワード ROUNDED を割り当てることができます。

5 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。

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 バイト文字ストリング (DBCS) 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単一バイト 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。

2 バイト文字は、**D1D2D3** という形式で示されています。DBCS 表現のローマ字 (英字) 文字は、**.A.B.C** という形式で示されています。文字の前にあるドット (ピリオド) は、16 進値 X'42' を表します。

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、その作成資料の序論部分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質問は、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、
>
> 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
>
> の著者および著作権所有者は、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。

変更の要約

Enterprise COBOL における、COBOL (OS/390 および VM 版) からの主な変更点を以下に示します。本書では、言語に対する変更のうち、今回の版で導入されたものについては、左側の余白に縦棒を付けて示してあります。

以前の COBOL に対する変更の履歴については、「移行ガイド」を参照してください。

Enterprise COBOL 第 1 版 (2001 年 11 月)

- オブジェクト指向の構文による COBOL と Java の相互運用性 (インターオペラビリティー) により、COBOL プログラムで Java クラスのインスタンスを生成したり、Java オブジェクトのメソッドを呼び出すことができるようになりました。また、Java または COBOL でインスタンスを生成することができ、かつ、Java または COBOL で呼び出すことができるメソッドを持つ Java クラスを COBOL プログラムで定義することができます。
- Java Native Interface (JNI) によって提供されるサービスを呼び出すことで、Java の追加機能を取得することができます。また、JNI へのアクセスを容易にするためのコピーブック JNI.cpy および特殊レジスター JNIENVPTR が提供されています。
- 高速 XML パーサーを含む XML サポートによってプログラムでインバウンド XML メッセージを利用でき、これらの XML メッセージが整形式 (well-formed) かどうかをチェックすることができます。また、メッセージの内容を COBOL データ構造へ変換できます。Unicode UTF-16 またはいくつかの単一バイト EBCDIC (拡張 2 進化 10 進コード) または ASCII コード・ページでエンコードされた XML ファイルがサポートされます。
- 個別の変換ステップなしで、CICS ステートメントを含むプログラムをコンパイルできるようになりました。
  - コンパイラー・オプション CICS が追加され、組み込み CICS トランスレーターの使用と CICS オプションの指定が可能になりました。
- NATIONAL データ型および国別 (N、NX) リテラル、文字変換の組み込み関数 DISPLAY-OF および NATIONAL-OF、およびコンパイラー・オプション NSYMBOL と CODEPAGE により提供される Unicode がサポートされるようになりました。
  - 国別リテラル、英数字および DBCS のデータ項目とリテラルのエンコードに使用するコード・ページを指定するためのコンパイラー・オプション CODEPAGE が追加されました。
  - コンパイラー・オプション NSYMBOL が追加され、これにより以下の制御が可能になりました。
    1. ピクチャー記号 N が DBCS 文字位置を表現するか国別文字位置 (Unicode UTF-16 の場合) を表現するか
    2. 開始の区切り文字 N' または N" が DBCS リテラルであるか国別リテラルであるか
- マルチスレッド化: POSIX スレッドおよび非同期シグナルのサポートにより、COBOL プログラムが含まれるアプリケーションを 1 つのプロセス内でマルチスレッドで実行できるようになりました。

  - コンパイラー・オプション THREAD が追加され、言語環境プログラムのエンクレーブで複数の POSIX スレッドまたは PL/I サブタスクと共にプログラムを実行できるようになりました。

- 4 バイトのデータ項目 FUNCTION-POINTER に COBOL または COBOL 以外の入り口点のアドレスを指定することができ、これにより、C の関数ポインターとのインターオペラビリティーが向上しました。
- BY CONTENT または BY VALUE を指定する CALL ステートメントの引き数リストにおける、ADDRESS OF の使用に関する規則が緩和されました。この引き数はこれまでリンケージ・セクション項目に限定されていましたが、リンケージ・セクション、ワーキング・ストレージ・セクション、ローカル・ストレージ・セクションのいずれでも指定できるようになりました。この機能により、ポインターのパラメーターを持つ C/C++ の関数とのインターオペラビリティーが向上します。
- バイナリー・データ項目のための VALUE 文節が追加されました。これにより、該当する場合には、数字リテラルは、PICTURE 文節内での 9 の数で暗黙的に指定される値に制限されることなく、ネイティブ・バイナリー表現の容量と同じ大きさまでの値を持つことができます。
- このバージョンでは以下のサポートは提供されなくなりました (Enterprise COBOL コンパイラーおよび実行時プログラム 移行ガイドを参照)。
  - SOM ベースのオブジェクト指向構文およびサービス
  - VM/CMS 環境のサポート
  - コンパイラー・オプション CMPR2、ANALYZE、FLAGMIG、TYPECHK、および IDLGEN

COBOL 言語の構造

文字 2
文字エンコード・ユニット 3

文字ストリング 5
1 バイト文字から構成される COBOL ワード 5
DBCS 文字が含まれる COBOL ワード 6
ユーザー定義語 6
システム名 8
関数名 8
予約語 8
形象定数 9
特殊レジスター 11
リテラル 25
PICTURE 文字ストリング 34
コメント 34

区切り文字 36

セクションと段落 39
文、ステートメント、および記入項目 39

参照フォーマット 41
シーケンス番号域 41
標識域 41
領域 A 42
領域 B 44
領域 A または領域 B 46

名前のスコープ 48
名前のタイプ 48
外部および内部リソース 50
名前の解決 51

データ名、コピー・ライブラリー、および手続き部名の参照 53
参照の固有性 53

制御の移動 64

2000 年言語拡張および日付フィールド 66
2000 年言語拡張の構文 66
用語と概念 67

© Copyright IBM Corp. 1991, 2001

文字

COBOL 言語の最も基本的で最小の単位は、**文字**です。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**区切り文字**が形成されます。文字ストリングおよび区切り文字を使用して、Enterprise COBOL を形成するワード、リテラル、句、文節、ステートメント、および文が形成されます。

文字ストリングおよび区切り文字を形成するときに使用されるデフォルトの文字セットは、3ページの表1 に示す*基本文字セット* です。

この基本文字セットは、IBM 2 バイト文字 (DBCS) で拡張されています。DBCS 文字は、隣接する 2 バイトで 1 つの文字を表現します。DBCS 文字が入っている文字ストリングは、**DBCS 文字ストリング**と呼ばれます。DBCS 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、およびコメント記入項目の内容には、コンピューターの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。リテラル、コメント行、およびコメント記入項目の内容には、DBCS 文字も使用できます。

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、DBCS 文字、および国別文字を含むことができます。国別文字は Unicode UTF-16 で表現されます。

NSYMBOL (NATIONAL) コンパイラー・オプションが有効なときは、開始の区切り文字 N" や N' で識別されるリテラルは国別リテラルであり、適用されているコード・ページ (デフォルトのコード・ページまたは CODEPAGE コンパイラー・オプションで指定されたコード・ページ) で有効な任意の 1 バイト文字または 2 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 Unicode UTF-16 文字として保管されます。

詳細については、6ページの『DBCS 文字が含まれる COBOL ワード』、30ページの『DBCS リテラル』、および 32ページの『国別リテラル』を参照してください。

© Copyright IBM Corp. 1991, 2001

表 1. Enterprise COBOL 基本文字セット

文字	意味
b	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
=	等号
$	通貨符号 [1]
,	コンマ
;	セミコロン
.	小数点またはピリオド
"	引用符 [2]
(	左括弧
)	右括弧
>	より大きい
<	より小さい
:	コロン
'	アポストロフィ
A ～ Z	英字 (大文字)
a ～ z	英字 (小文字)
0 ～ 9	数字

注:

1 通貨符号は、有効なコード・ページの設定に関係なく、値 X'5B' 持つ文字です。割り当てられる図形文字は、ドル記号または国内通貨符号です。

2 引用符は値 X'7F' を持つ文字です。

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、コード化文字セット内の 1 つのコード・ポイントです。1 バイト EBCDIC 文字セットの場合、エンコード・ユニットは 1 バイトです。EBCDIC 2 バイト文字セットの場合、エンコード・ユニットは 2 バイトです。EBCDIC 文字セットと ASCII 文字セットの場合、*図形文字* は常に 1 つのエンコード・ユニットで表現されるので、ユーザーはエンコード・ユニットについて考慮する必要がありません。

Unicode UTF-16 の場合、1 つの文字エンコード・ユニットは 2 バイトで構成されます。多くの場合、UTF-16 文字は 1 つのエンコード・ユニットで表現されます。1 バイトまたは 2 バイトの EBCDIC コード・ページまたは ASCII コード・ページから UTF-16 に変換されたそれぞれの文字は、1 つのエンコード・ユニットで表現されます。しかし、UTF-16 に含まれる文字のサブセットは、複数のエンコード・ユニットで表現されます。一部の図形文字は、2 つのエンコード・ユニット (4 バイト) からなる*サロゲート・ペア* で表現されます。また、1 つの基本文字と 1 つ以上の*結合マーク* (4 バイト以上であり、2 バイトずつ増分される) からなる*複合シーケンス* で表現される図形文字もあります。

Enterprise COBOL の規則において、*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。多くの場合、これは図形文字と同じです。データにサロゲート・ペアまたは複合シーケンス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。

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。文字ストリングは区切り文字によって区切られます。

区切り文字とは、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。区切り文字は、36ページの『区切り文字』の項で詳しく説明します。

文字ストリングと特定の区切り文字は、**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 1 行の文字位置 8 ～ 72 にある 1 つの文字または連続した一連の文字です。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 EBCDIC で記述することができ、一部の文字ストリングについては DBCS を使用できます (コンパイラーは ASCII や Unicode で記述されたソース・コードを処理することができません)。

以下を形成する際には、単一バイト EBCDIC または DBCS (あるいは両方) の文字ストリングを使用することができます。

- COBOL ワード
- リテラル
- PICTURE 文字ストリング (単一バイト EBCDIC の文字ストリングのみ)
- コメント・テキスト

1 バイト文字から構成される COBOL ワード

COBOL ワードとは、ユーザー定義語、システム名、または予約語を形成する 30 文字以下の文字ストリングです。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ものから選択されます。

- A ～ Z
- a ～ z
- 0 ～ 9
- - (ハイフン)

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。ユーザー定義語にはすべて (セクション名、段落名、セグメント番号、およびレベル番号を除く) 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。セグメント番号とレベル番号は固有である必要はありません。セグメント番号またはレベル番号の指定は、その他のセグメント番号またはレベル番号と同じにすることができます。

COBOL ワードでは、英小文字はそれぞれ対応する英大文字と等価であるとみなされますが、英数字リテラルや国別リテラルの場合は違います。

ソース・プログラム内では、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。

© Copyright IBM Corp. 1991, 2001

- 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、その語が現れる文節または句の文脈によって判別されます。

DBCS 文字が含まれる COBOL ワード

以下に、DBCS 文字からユーザー定義語を形成する場合の規則を示します。

シフトアウト / シフトイン文字の使用

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

値の範囲

DBCS ユーザー定義語には、どちらのバイトでも、X'41' ～ X'FE' の値の文字を含めることができます。

含まれる文字

DBCS のユーザー定義語は DBCS 文字のみを含むことができ、1 つ以上の DBCS 文字を含んでいる必要があります。DBCS 文字とは、A ～ Z、a ～ z、0 ～ 9、およびハイフン (これらの文字の DBCS 表現では最初のバイトに X'42' が入ります) からなるセットに**含まれない**文字です。

DBCS のユーザー定義語には、単一バイト EBCDIC 文字に対応する文字と、単一バイト EBCDIC 文字に対応しない文字を使用することができます。単一バイト EBCDIC 文字に対応する DBCS 文字は、COBOL ユーザー定義語の通常の規則に従います。つまり、文字 A ～ Z、a ～ z、0 ～ 9、およびハイフン (-) を使用することができます。ハイフンは、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。対応する単一バイト EBCDIC 文字のない DBCS 文字を DBCS ユーザー定義語で使用することができます。

継続規則

DBCS 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**。

英字の大文字 / 小文字

完全な形式での表現。

最大長

14 文字。

ユーザー定義語

Enterprise COBOL では、次のようなユーザー定義語のセットをサポートしています。2 番目の列は、特定のセットのワードで DBCS 文字を使用できるかどうかを示しています。

	DBCS 文字が使用可能か
英字名	はい
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

オブジェクト指向クラス名	いいえ
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
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

レベル番号および優先順位番号については、それぞれの語が 1 桁または 2 桁の整数でなければなりません。

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。優先順位番号とレベル番号以外については、1 つのセット内の各ユーザー定義語は、53ページの『データ名、コピー・ライブラリー、および手続き部名の参照』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記入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、ステートメントと記入項目によって参照できます。

- 英字名
- クラス名
- 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、それが現れる文節またはステートメントの中に記述されます。

システム名

システム名とは、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。システム名には次の 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 3 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。

コンピューター名は DBCS 文字で記述することができますが、その他のシステム名を DBCS 文字で記述することはできません。

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、449ページの表56 を参照してください。

予約語

予約語とは、COBOL ソース・プログラム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。Enterprise COBOL 予約語は、566ページの付録E、『予約語』にリストされています。

予約語には、次の 6 つのタイプがあります。

- キーワード
- オプショナル語
- 形象定数
- 特殊文字ワード
- 特殊オブジェクト ID
- 特殊レジスター

キーワード
キーワードとは、一定の文節、記入項目、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幹線上に大文字で示されています。

オプショナル語
オプショナル語とは、文節、記入項目、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

9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 2 種類あります。これらは単一バイトで表示されるときにのみに特殊文字として認識されます。

- **算術演算子: + - / * ****

  230ページの『算術式』を参照してください。

- **比較演算子: < > = <= >=**

  235ページの『条件式』を参照してください。

特殊オブジェクト ID

COBOL は 2 つの特殊オブジェクト ID である SELF と SUPER を提供します。これらは手続き部のメソッドで使用されます。

SELF

メソッドの手続き部で使用できる特殊オブジェクト ID です。SELF は、現在実行中のメソッドを呼び出すために使用するオブジェクトのインスタンスを指します。

SELF は、構文図で明示的に示された個所でのみ指定できます。

SUPER

INVOKE ステートメントのオブジェクト ID として、メソッドの手続き部で使用できる特殊オブジェクト ID です。このように使用するとき、SUPER は現在実行中のメソッドを呼び出すために使用するオブジェクトのインスタンスを指します。呼び出されるメソッドの解決では、現在実行中のメソッドのクラス定義で宣言されたメソッド、およびそのクラスから派生したクラスで定義されたメソッドは無視されます。そのため、呼び出されたメソッドは親元クラスから継承されます。

特殊レジスター

11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。形象定数用の予約語とその意味は、次のとおりです。

ZERO/ZEROS/ZEROES

文脈に応じて、数値ゼロ (0) か、文字ゼロ (0) の 1 つまたは複数のオカレンスを表します。

英数字を指定する必要がある文脈で形象定数 ZERO/ZEROS/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'0030') 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 つまたは複数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文脈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文脈で使用された場合には DBCS リテラルとして扱われます。また、国別文字を指定する必要があ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。DBCSスペース文字は値 X'4040' を持ち、国別スペース文字は値 NX'0020' を持ちます。

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 つまたは複数のオカレンスを表します。EBCDIC 照合シーケンスでは、その文字は X'FF' です。その他の照合シーケンスでは、実際に使用される文字は、使用する照合シーケンスによって決まります。HIGH-VALUE は英数字リテラルとして扱われます。

国別文字値を指定する必要がある文脈では、HIGH-VALUE/HIGH-VALUES を使用できません。

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 つまたは複数のオカレンスを表します。EBCDIC 照合シーケンスでは、その文字は X'00' です。その他の照合シーケンスでは、実際に使用される文字は、使用する照合シーケンスによって決まります。LOW-VALUE は英数字リテラルとして扱われます。

国別文字値を指定する必要がある文脈では、LOW-VALUE/LOW-VALUES を使用できません。

QUOTE/QUOTES
以下の 1 つまたは複数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

または

- 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/QUOTES は、英数字を指定する必要がある文脈で使用された場合には英数字を表し、国別文字を指定する必要がある文脈で使用された場合には国別文字を表します。引用符の国別文字値は NX'0022' です。アポストロフィの国別文字値は NX'0027' です。

英数字リテラルを囲むのに、QUOTE または 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、INVOKE、STOP、または STRING の各ステートメントでは、形象定数 ALL 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 文節で、シンボリック文字の値として指定された 1 つまたは複数の文字を表します。

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。(たとえば、受け取り項目が国別データ項目である MOVE ステートメントでは、国別文字を使用でき

| ます。なぜなら、送り出し項目が英数字であり、受け取り項目が国別データ項目である
| 場合は暗黙の変換が定義されるからです。)

NULL/NULLS
USAGE IS POINTER、USAGE IS PROCEDURE-POINTER、USAGE IS FUNCTION-POINTER、USAGE IS OBJECT REFERENCE、または ADDRESS OF 特殊レジスターで定義されるデータ項目に、有効なアドレスが入っていないことを示すために使用される値を表します。NULL は、構文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。たとえば、データ名-1 が 5 文字のデータ項目の場合は、次のステートメントのどれでもデータ名-1 が 5 つのスペースで埋められます。

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

| Enterprise 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、ZEROES) だけを使用することができます。形象定数は、関数の引き数として機能する算術式の中以外では、関数の引き数として使用できません。

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 文節で指定されるか、またはデータ項目と関連付けられている場合は (たとえば、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 (たとえば CALL、INVOKE、STOP、STRING、または UNSTRING の各ステートメントの中で)、その文字ストリングの長さは 1 文字です。

特殊レジスター

特殊レジスターとは、コンパイラーによ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 THREAD オプションを指定してコンパイルされたプログラム、あるいはメソッドでは、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

最初にプログラムを CALL したとき、プログラムの CANCEL 後に最初にそのプログラムを CALL したとき、あるいはメソッドの起動を行ったときに、コンパイラーは特殊レジスターを初期値に初期設定します。

以下の 4 つの場合、
- INITIAL 属性を持つプログラム
- RECURSIVE 属性を持つプログラム
- THREAD オプションを指定してコンパイルされたプログラム
- メソッド

では、次の特殊レジスターは各プログラムまたはメソッド・エントリーの初期値にリセットされます。

- 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 4 つの場合、ADDRESS OF 特殊レジスターに設定された値は、そのプログラムまたはメソッド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 時または INVOKE 時に設定されていた値のままです)。

その他の事柄が明示的に制限されていない場合は、データ名または ID が特殊レジスターの暗黙の定義 (このセクションの後の部分で指定されます) と同じ定義を持っている個所では、特殊レジスターを使用することができます。

特別の禁止がない限り、英数字引き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、リンケージ・セクションのレコードのアドレスを参照します。これは、送り出し項目としても受け取り項目としても使用することができます。

ADDRESS OF 特殊レジスターは、相互に再定義するレコードの場合を除き、リンケージ・セクションの各レコード (01 または 77) ごとに存在します。この場合には、ADDRESS OF 特殊レジスターが同じように再定義されます。

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。

更新後の各フィールドの内容は次のようになります。

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
DEBUG-NAME に添え字または指標が付けられている場合、各レベルのオカレンス番号がそれぞれの DEBUG-SUB-n の中に入れられます。項目に添え字または指標が付けられて**いない**場合は、このフィールドはスペースのままです。プログラムで 3 レベルを超

える添え字または指標付けが使用されている場合は、DEBUG-ITEM 特殊レジスターを参照してはなりません。

DEBUG-CONTENTS
データは、表2 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/MERGE 入出力プロシージャーの中のプロシージャー名-n	SORT/MERGE ステートメント	プロシージャー名-n	“SORT INPUT” “SORT OUTPUT” “MERGE OUTPUT” (いずれか該当するもの)
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 *	プロシージャー名-n	“FALL THROUGH”
最初の宣言型プロシージャーの最初の実行	最初の宣言型プロシージャーの行番号	最初の宣言型プロシージャー	“START PROGRAM”

注:

* 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。

JNIENVPTR

JNIENVPTR 特殊レジスターは、Java Native Interface (JNI) 環境ポインターを参照します。JNI 環境ポインターは、Java 呼び出し可能サービスを呼び出すときに使用します。

JNIENVPTR は、暗黙的に USAGE POINTER として定義されます。JNIENVPTR を受け取りデータ項目として指定することはできません。

JNIENVPTR および JNI 呼び出し可能サービスの使用については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

LENGTH OF

LENGTH OF 特殊レジスターには、ID によって使用されるバイト数が入れられます。

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。

注: DBCS および国別データ項目の場合、各文字は 2 バイトのストレージを占有します。

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、手続き部のどこであっても、LENGTH OF を使用することができます。LENGTH OF 特殊レジスターには、次のような暗黙の定義があります。

```
USAGE IS BINARY PICTURE 9(9)
```

ID によって参照されるデータ項目が GLOBAL 文節を含む場合には、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、整数引き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 関数への引き数として LENGTH OF 特殊レジスターが使用された場合、LENGTH OF で指定された引き数とは無関係に、その結果は必ず 4 になります。

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

LENGTH OF は、以下のものにすることは**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 で渡す必要があります。

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、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。

```
MOVE LENGTH OF A TO B
DISPLAY LENGTH OF A, A
ADD LENGTH OF A TO B
CALL "PROGX" USING BY REFERENCE A BY CONTENT LENGTH OF A
```

注: 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。国別データ項目の場合、LENGTH 関数から戻される長さは国別文字位置の数であり、バイト数ではありません。したがって、国別項目の場合、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。

LINAGE-COUNTER

LINAGE 文節を含む各 FD 記入項目ごとに、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、次のどちらかです。

- LINAGE 文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 を持ちます。
- LINAGE 文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。

詳細については、156ページの『LINAGE 文節』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 は手続き部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。(425ページの『WRITE ステートメント』を参照。)

順次ファイルのファイル記述項目が LINAGE 文節および EXTERNAL 文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 文節および GLOBAL 文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。

整数引き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。

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。
- 制御が呼び出し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プログラムに渡されます。呼び出しプログラムが COBOL プログラムの場合、呼び出

しプログラム側の 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
```

RETURN-CODE 特殊レジスターの設定方法の例は次のとおりです。

```
COMPUTE RETURN-CODE = 8
```

または

```
MOVE 8 to RETURN-CODE.
```

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。

注: RETURN-CODE 特殊レジスターは、呼び出されたメソッドまたは CALL...RETURNING を使用するプログラムからの値を戻しません。詳細については、332 ページの『INVOKE ステートメント』または 283ページの『CALL ステートメント』を参照してください。

整数引き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 特殊レジスターには、言語環境プログラム呼び出し可能サービスへのサービス呼び出しからの戻りコード情報は入りません。詳細については、*Enterprise COBOL プログラミング・ガイド* および*言語環境プログラム プログラミング・ガイド* を参照してください。

SHIFT-OUT と SHIFT-IN

SHIFT-OUT と SHIFT-IN の 2 つの特殊レジスターは、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"
```

これらの特殊レジスターは、印刷不能文字を使用することなく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。

英数字引き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。

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。DBCS ユーザー定義語を定義しているとき、および 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。

次に示すのは、SHIFT-OUT と SHIFT-IN の使用法の例です。

```
DATA DIVISION.
WORKING-STORAGE.

01   DBCSGRP.
       05  SO       PIC X.
       05  DBCSITEM PIC G(3) USAGE DISPLAY-1
       05  SI       PIC X.
  ⋮

PROCEDURE DIVISION.

       MOVE SHIFT-OUT TO SO
       MOVE G"<D1D2D3>" TO DBCSITEM
       MOVE SHIFT-IN TO SI
       DISPLAY DBCSGRP
```

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-CONTROL

SORT-CONTROL 特殊レジスターは、以下のように暗黙的に定義される英数字データ項目の名前です。

```
01    SORT-CONTROL GLOBAL PICTURE X(8) USAGE DISPLAY VALUE "IGZSRTCD"
```

このレジスターには、ソートまたはマージの操作のパフォーマンスを向上させるために使用する制御ステートメントを保持したデータ・セットの DD 名が含まれています。

SORT-CONTROL 特殊レジスターによって識別されるデータ・セットに DD ステートメントを指定することができます。この指定により、Enterprise COBOL は実行時にそのデータ・セットをオープンしようとします。エラーが検出されると、通知メッセージによって診断されます。

英数字引き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することに注意してください。

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

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。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
```

SORT-CORE-SIZE はソート制御ファイル内の MAINSIZE 制御ステートメントまたは RESINV 制御ステートメントの代わりに使用することができます。

'MAINSIZE=' オプション制御ステートメント・キーワードは、正の値の SORT-CORE-SIZE と等価です。

'RESINV=' オプション制御ステートメント・キーワードは、負の値の SORT-CORE-SIZE と等価です。

'MAINSIZE=MAX' オプション制御ステートメント・キーワードは、値が +999999 または +99999999 の SORT-CORE-SIZE と等価です。

整数引き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。

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

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。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
```

SORT-FILE-SIZE は、ソート制御ファイルの 'FILSZ=Ennn' 制御ステートメントと等価です。

整数引き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。

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-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。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"
```

SORT-MESSAGE 特殊レジスターを使用することによって、ソート・ユーティリティーが SYSOUT データ・セットの代わりに使用するデータ・セットの DD 名を指定できます。

このような場合、SORT-MESSAGE 内の DD 名はソート制御ファイルの中の 'MSGDDN=' 制御ステートメントで指定される名前と等価です。

英数字引き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。

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

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する 2 進データ項目の名前です。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
```

SORT-MODE-SIZE は、ソート制御ファイルの 'SMS=' 制御ステートメントと等価です。

整数引き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。

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-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
```

ソート / マージ処理の完了時に、このデータ項目に戻りコード 0 (成功) または 16 (不成功) が入れられます。ソート / 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、メッセージが端末装置に表示されます。

すべてのレコードが処理される前にソート / 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。SORT または MERGE 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。

整数引き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。

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

TALLY 特殊レジスターは、2 進データ項目の名前で、次のように定義されています。

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

TALLY の内容を参照したり変更したりすることができます。

整数引き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。

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

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
```

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)
```

たとえば、コンパイルが 1995 年 4 月 27 日午後 2 時 4 分に始まった場合、WHEN-COMPILED には値として 04/27/9514.04.00 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。

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。

注: この特殊レジスターに納められたコンパイル日時は、日時の組み込み関数 WHEN-COMPILED を使用すればアクセス可能です (502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 PARSE ステートメントで識別された処理プロシージャーと XML パーサーとの間の状況のやりとりに使用します。XML パーサーは、各イベントの処理プロシージャーへの制御権移動前およびパーサーの終了時に XML-CODE を設定します。処理プロシージャーから XML パーサーに制御を戻す前に、XML-CODE をリセットすることができます。

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。例外コードについては、*Enterprise COBOL プログラミング・ガイド* 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。標準イベント後に、パーサーに対して、EXCEPTION イベントを発生させずに終了するよう指示します。

- ゼロ (0) 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。

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、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最新の値が入っています。

XML-EVENT

XML-EVENT 特殊レジスターは、XML PARSE ステートメントで識別された処理プロシージャーに対して、XML パーサーからイベント情報を通知するために使用します。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 特殊レジスターを表3に示す XML イベントの名前に設定します。

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 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 (1/2). 特殊レジスター XML-EVENT および XML-TEXT (または XML-NTEXT) 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	属性値の定義済みエンティティー参照に対応する単一の文字。
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。= の左側のストリングです。
ATTRIBUTE-NATIONAL-CHARACTER	XML PARSE ステートメントの ID-1 で指定された XML 文書のタイプに関係なく、XML-TEXT は空であり、XML-NTEXT には (数値) 文字参照に対応する単一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・テキスト。
CONTENT-CHARACTER	要素の内容の定義済みエンティティー参照に対応する単一の文字。
CONTENT-CHARACTERS	開始タグと終了タグに囲まれた、要素の内容。内容にエンティティー参照または別の要素が含まれているときは、この値は要素の内容のサブストリングである場合があります。

表 3 (2/2). 特殊レジスター XML-EVENT および XML-TEXT (または XML-NTEXT) の内容

XML イベント (XML-EVENT の内容)	XML-TEXT または XML-NTEXT の内容
CONTENT-NATIONAL-CHARACTER	XML PARSE ステートメントの ID-1 で指定された XML 文書のタイプに関係なく、XML-TEXT は空であり、XML-NTEXT には (数値) 文字参照に対応する単一の国別文字が入ります。[1]
DOCUMENT-TYPE-DECLARATION	開始文字シーケンス “<!DOCTYPE” および終了文字シーケンス ">" 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値。
END-OF-CDATA-SECTION	常にストリング "]]>" が入ります。
END-OF-DOCUMENT	ヌル。長さはゼロ。
END-OF-ELEMENT	終了要素タグまたは空要素タグの名前。
EXCEPTION	例外が検出された地点までの、正常にスキャンされた文書部分。[2] 特殊レジスター XML-CODE は例外を識別するエラー・コードに設定されます。
PROCESSING-INSTRUCTION-DATA	処理命令の残りの部分。末尾の空白文字は含まれますが、終了シーケンス "?>" および先頭の空白文字は含まれません。
PROCESSING-INSTRUCTION-TARGET	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	引用符またはアポストロフィで囲まれた、XML 宣言内のスタンドアロン宣言値。
START-OF-CDATA-SECTION	常にストリング "<![CDATA[" が入ります。
START-OF-DOCUMENT	文書全体。
START-OF-ELEMENT	開始要素タグまたは空要素タグの名前。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 “&” や ";" を除く)。
UNKNOWN-REFERENCE-IN-ATTRIBUTE	エンティティー参照名 (区切り文字 “&” や ";" 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宣言値。現在は常に "1.0" です。

表 3 (1/2). 特殊レジスター XML-EVENT および XML-TEXT (または XML-NTEXT) の内容

XML イベント (XML-EVENT の内容)	XML-TEXT または XML-NTEXT の内容

注:

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 は長さゼロになるか、文書のエンコード宣言値のみが入ります。XML 例外コードの詳細については、*Enterprise COBOL プログラミング・ガイド* 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 の文書フラグメントに設定されます。

XML-NTEXT は基本の国別データ項目であり、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、*国別文字位置* 0 ～ 8,388,607 です。最大バイト長は 16,777,214 です。

注: 対応する COBOL データ記述記入項目はありません。

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移動します。

- XML PARSE ステートメントのオペランドが国別データ項目である場合
- 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、両方同時に長さゼロの値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。XML-NTEXT 用の LENGTH OF 特殊レジスターの値は、XML-NTEXT に含まれている国別文字の数ではなく、バイト数です。

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 構文解析中に定義され、英数字のクラスからなる文書フラグメントに設定されます。

XML-TEXT は基本の英数字データ項目であり、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 ～ 16,777,215 バイトです。

注: 対応する COBOL データ記述記入項目はありません。

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。

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ると、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、処理プロシージャーに制御権を移動します。

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、両方同時に長さゼロの値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。(9ページの『形象定数』を参照。) リテラルのタイプには、**英数字**、**DBCS**、**国別**、および**数値**があります。

英数字リテラル

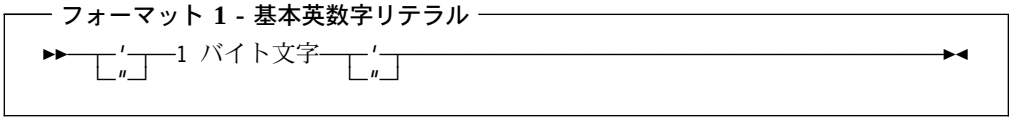
Enterprise COBOL では、次のフォーマットの英数字リテラルをサポートしています。

フォーマット 1 - 基本英数字リテラル
フォーマット 2 - DBCS 文字を含む英数字リテラル
フォーマット 3 - 英数字リテラルの 16 進表記
フォーマット 4 - 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、単一バイト EBCDIC 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。

基本英数字リテラルのフォーマットを以下に示します。



終了の区切り文字は、開始の区切り文字と一致している必要があります。

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 ("") またはアポストロフィ (') で表現する必要があります。たとえば、次のようになります。

```
"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。たとえば、次のようになります。

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

アポストロフィは、APOST/QUOTE コンパイラー・オプションの指定に関係なく、リテラル区切り文字として使用できます。

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。

英数字リテラルの最大長は 160 バイトです。

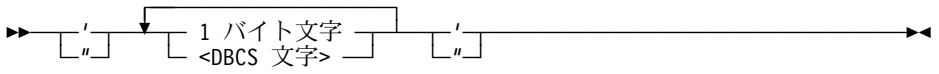
英数字リテラルは、**英数字**データ・クラスおよびカテゴリーに属します (データ・クラスおよびカテゴリーについては、142ページの『データのクラスとカテゴリー』で解説しています)。

DBCS 文字を含む英数字リテラル

DBCS コンパイラー・オプションが有効な場合、英数字リテラルに含まれている文字 X'0E' および X'0F' は、DBCS 文字用のシフト・コードとして認識されます。つまり、対になったシフト・コードではさまれた文字は DBCS 文字として認識されます。NODBCS オプションを付けてコンパイルした英数字リテラルとは異なり、英数字リテラルの中の DBCS 文字には追加の構文規則が適用されます。

DBCS 文字が含まれている英数字リテラルは、以下のフォーマットになります。

フォーマット 2 - DBCS 文字を含む英数字リテラル



" または '
開始と終了の区切り文字です。終了の区切り文字は、開始の区切り文字と一致している必要があります。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。

< シフトアウト制御文字 (X'0E') を表します。

> シフトイン制御文字 (X'0F') を表します。

シフトアウト制御文字とシフトイン制御文字はリテラルの一部であり、対になっている必要があります。シフトアウト制御文字とシフトイン制御文字の間は、0 または偶数バイトにする必要があります。

リテラルの DBCS 部分では、シフト・コードをネストすることはできません。

リテラルに含まれる単一バイト文字に関する構文規則は、基本英数字リテラルの規則に従います。リテラルに含まれる DBCS 文字に関する構文規則は、DBCS リテラルの規則に従います。DBCS 文字を含む英数字リテラルに関する移動および比較の規則は、DBCS 文字を含まない英数字リテラルに関する規則と同じです。

DBCS 文字を含む英数字リテラルの長さは、シフト制御文字を含むそのリテラルのバイト数です。最大長は領域 B の 1 行で使用できるスペースまでです。DBCS 文字を含む英数字リテラルは、次の行に継続できません。

DBCS 文字を含む英数字リテラルは、英数字のカテゴリーに属しています。

DBCS 文字が含まれる英数字リテラルは、次のような場合には使用できません。

- 次の中でリテラルを使用する場合

  ALPHABET 文節
  ASSIGN 文節
  CALL ステートメントのプログラム ID
  CANCEL ステートメント
  CLASS 文節
  CURRENCY SIGN 文節
  END PROGRAM マーカー
  ENTRY ステートメント
  PADDING CHARACTER 文節
  PROGRAM-ID 段落
  RERUN 文節
  STOP ステートメント

- オブジェクト指向クラスの外部クラス名として
- BASIS ステートメントの中で基本名として
- COPY ステートメントの中でテキスト名として
- COPY ステートメントの中でライブラリー名として

Enterprise COBOL ステートメントは、シフト・コードや文字コードに影響されずに、DBCS 文字を含む英数字リテラルを処理します。バイト単位で処理するステートメント (たとえば STRING および UNSTRING) を使用すると、単一バイトの EBCDIC 文字と DBCS 文字の無効な組み合わせのストリングが生じることがあります。バイト単位で処理を行うステートメントの中で DBCS 文字を含む英数字リテラルおよびデータ項目を使用する方法については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

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を次に示します。

フォーマット 3 - 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 進数は、'0' ～ '9'、'a' ～ 'f'、および 'A' ～ 'F' の文字で表します。16 進数字は、1 バイト文字セット (EBCDIC または ASCII) で 1 つの文字を表現します。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。DBCS の文字ストリングを 16 進表記で表現するときは、シフトアウト制御文字 (X'0E') とシフトイン制御文字 (X'0F') で囲む必要があります。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。

DBCS コンパイラー・オプションは、16 進数で表記した英数字リテラルの処理には影響しません。

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。英数字リテラルの 16 進表記は、英数字リテラルを指定できるであればどこでも使用できます。

ヌル終了英数字リテラル

英数字リテラルにはヌル終了を使用することができます。フォーマットは次のとおりです。

フォーマット 4 - ヌル終了英数字リテラル

Z" または Z'
英数字リテラルのヌル終了表記の開始の区切り文字

ヌル終了リテラルの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 '
英数字リテラルのヌル終了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。

リテラルの内容には 1 バイト文字または 2 バイト文字 (あるいは両方) を使用することができますが、値 X'00' を持つ 1 バイト文字は使用できません。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、ヌル終了英数字リテラルの内容には、DBCS 文字が含まれる英数字リテラル (フォーマット 2) 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 1 バイト文字または 2 バイト文字 (あるいは両方) のストリングの長さは、0 ～ 159 バイトになります。リテラルの実際の長さには終わりのヌル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、英数字リテラルを指定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。)

数字リテラル

数字リテラルとは、0 ～ 9 の数字、符号 (+ 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。

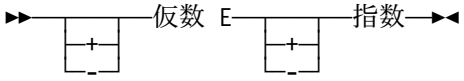
- ARITH(COMPAT) コンパイラー・オプションが有効な場合は、1 ～ 18 桁が使用できます。ARITH(EXTEND) コンパイラー・オプションが有効な場合は、1 ～ 31 桁が使用できます。
- 符号文字は 1 つだけ使用できます。符号文字を付ける場合、それはリテラルの左端の文字でなければなりません。リテラルに符号が付かない場合、そのリテラルは正の値です。
- 小数点は 1 つだけ使用できます。小数点を入れる場合は、想定小数点として扱われます (つまり、リテラル中の 1 桁をとりません)。小数点は、右端の文字として以外であればリテラル中のどこでも置けます。

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。標準データ・フォーマットの文字で表される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。

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は、次の形式で記述します。



- 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 ～ 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は 0.54E-78 ～ 0.72E+76 の範囲内でなければなりません。この範囲に入らない値に関しては、E レベルの診断メッセージが作成され、値はそれぞれ 0 または 0.72E+76 のいずれかに置き換えられます。

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、142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)。

DBCS リテラル

DBCS リテラルのフォーマットおよび規則を以下に示します。

DBCS リテラルのフォーマット

```
G"<D1D2D3>"
G'<D1D2D3>'
N"<D1D2D3>"
N'<D1D2D3>'
```

G" G' N" 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、32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。

開始の区切り文字の直後には、シフトアウト制御文字が必要です。

開始の区切り文字 N" または N' が含まれているリテラルの場合、組み込みの引用符やアポストロフィを DBCS リテラル内で DBCS 文字の 1 つとして指定するときは、2 つの DBCS 引用符またはアポストロフィで 1 つの組み込み DBCS 引用符または組み込み DBCS アポストロフィを表現します。1 つの組み込み DBCS 引用符またはアポストロフィが見つかると、E レベルのコンパイラー・メッセージが出され、2 つ目の組み込み DBCS 引用符またはアポストロフィがあるとみなされます。

<
シフトアウト制御文字 (X'0E') を表します。

>
シフトイン制御文字 (X'0F') を表します。

" 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。

終了の区切り文字は、シフトイン制御文字の直後に置く必要があります。

D1D2D3

どちらのバイトの場合でも、D1D2D3 は X'00' ～ X'FF' の範囲の 1 つ以上の文字です。リテラルの内容には任意の値を指定できますが、実行時にその値が有効かどうかは、CODEPAGE コンパイラー・オプションについて有効な CCSID によって決まります。

最大長

28 文字。

継続規則

複数行にまたがって続けることはできません。

DBCS リテラルを使用できる場合

DBCS リテラルは次のような場合に使用することができます。

- データ部
  - DBCS データ記述記入項目の VALUE 文節の中。DBCS リテラルを VALUE 文節の中でデータ項目として指定した場合、リテラルの長さは、そのデータ項目の持つ PICTURE 文節によって指示されたサイズを超えることはできません。暗黙的あるいは明示的に、DBCS データ項目を USAGE DISPLAY-1 として定義すると、DBCS データ項目が 1 文字あたり 2 バイトの文字形式で保管されることが指定されます。
  - ファイル記述項目の VALUE OF 文節の中
- 手続き部
  - CALL ステートメントの BY CONTENT を渡される引き数として
  - DISPLAY ステートメントおよび EVALUATE ステートメントの中
  - 国別項目が明示的な MOVE ステートメントにおける受け取り項目である場合の送り出し項目として
  - DBCS 項目が以下のステートメントにおける受け取り項目である場合の送り出し項目として
    - INITIALIZE
    - INSPECT
    - MOVE
    - STRING
    - UNSTRING
  - 被比較数が DBCS 項目、国別項目、またはグループ項目である場合の比較条件で
  - 形象定数 ALL に含まれるリテラルとして
  - NATIONAL-OF 組み込み関数への引き数として
- コンパイラー指示ステートメント COPY、REPLACE、および TITLE

国別リテラル

Enterprise COBOL は、次の国別リテラル・フォーマットをサポートしています。

基本国別リテラル
国別リテラルの 16 進表記

基本国別リテラル

基本国別リテラルのフォーマットと規則は、以下のとおりです。

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。

注: 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、30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。

開始の区切り文字 (引用符またはアポストロフィ) として使用されている文字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。たとえば、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・プログラム表現です。文字データには、CODEPAGE コンパイラー・オプションで指定される CCSID (Coded Character Set ID) でエンコードされた EBCDIC 1 バイト文字および 2 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。

リテラルの内容に 2 バイト文字を使用する場合は、2 バイト文字をシフトアウト制御文字とシフトイン制御文字で区切る必要があります。

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 DBCS 文字は 1 つの文字位置としてカウントされます。DBCS 文字用の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。

リテラルの内容に DBCS 文字が含まれている場合、リテラルの長さはソース行の領域 B 内の使用可能な位置に制限されます。

継続規則 リテラルの内容に DBCS 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 DBCS 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・プログラム表現は、実行時の使用のために自動的に UTF-16 へ変換されます (たとえば、リテラルが国別データ項目に移動されたときや、国別データ項目と比較されたとき)。

国別リテラルの 16 進表記

国別リテラルの 16 進表記形式のフォーマットと規則は、以下のとおりです。

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' ～ '9'、'a' ～ 'f'、および 'A' ～ '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 ～ 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 (CCSID 01200) 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個所

国別リテラルは、次のような個所に使用できます。

- 国別データ項目に関連付けられた VALUE 文節または国別条件変数の条件名に関連付けられた VALUE 文節の中
- 形象定数 ALL に含まれるリテラルとして

- 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
- METHOD-ID 段落、END-METHOD マーカー、および INVOKE ステートメントのメソッド名として
- CALL ステートメントの BY CONTENT で渡される引き数として
- INVOKE ステートメントまたは CALL ステートメントの BY VALUE で渡される引き数として
- DISPLAY ステートメントおよび EVALUATE ステートメントの中
- INVOKE ステートメントまたは CALL ステートメントの RETURNING 句の中
- 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き数リストの中

  DISPLAY-OF、LENGTH、LOWER-CASE、MAX、MIN、ORD-MAX、ORD-MIN、REVERSE、または UPPER-CASE。
- 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、Enterprise COBOL 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、区切り文字としてのスペース、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 文節記号の図は、179ページの表10 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てできる文字ストリングです。プログラムの実行には影響しません。コメントには次の 2 種類があります。

コメント項目 (見出し部)
この形式については、91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。

コメントを形成する文字ストリングには、DBCS 文字、または DBCS 文字と単一バイト EBCDIC 文字との組み合わせを使用できます。

DBCS ストリングを含む複数コメント行も可能です。コメント行への DBCS 文字の埋め込みは、行単位で行う必要があります。DBCS 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。コメント行では、DBCS ストリングの有効性に関する構文検査は行われません。

区切り文字

区切り文字とは、表4 にあるように 1 つまたは複数の連続した文字からなるストリングです。

表 4. 区切り文字として使用する文字

区切り文字	意味
␣	スペース
,␣	コンマ
.␣	ピリオド
;␣	セミコロン
(	左括弧
)	右括弧
:	コロン
"␣	引用符
'␣	アポストロフィ
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[1]
N'	国別リテラルの開始の区切り文字[1]
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

注:

1 NSYMBOL(DBCS) コンパイラー・オプションが有効な場合、N" と N' は DBCS リテラルの開始の区切り文字です。

区切り文字に関する規則

以下の説明では、各区切り文字を {} で囲みます。スペースが区切り文字またはそ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 {␣}

スペースは、以下の場合を除き、任意の区切り文字の直前または直後に置くことができます。

- 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、区切り文字とはみなされません。

ピリオド {.␣}、コンマ {,␣}、セミコロン {;␣}

区切り文字のコンマは 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。区切り文字のピリオドは 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。区切り文字のセミコロンは 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。

© Copyright IBM Corp. 1991, 2001

区切り文字のピリオドは、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。区切り文字のコンマと区切り文字のセミコロンは、区切り文字のスペースが使用される場合は、どこでも使用できます。

- **見出し部**では、それぞれの段落が区切り文字のピリオドで終わっていなければなりません。
- **環境部**では、SOURCE-COMPUTER、OBJECT-COMPUTER、SPECIAL-NAMES、および I-O-CONTROL 段落は区切り文字の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記入項目が区切り文字のピリオドで終わっていなければなりません。
- **データ部**では、ファイル (FD)、ソート / マージ・ファイル (SD)、およびデータ記述記入項目がそれぞれ区切り文字のピリオドで終わっていなければなりません。
- **手続き部**では、区切り文字のコンマまたは区切り文字のセミコロンで、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。それぞれの文とそれぞれのプロシージャーは、区切り文字のピリオドで終わっていなければなりません。

括弧 { (} ... {) }
疑似テキストの中を除き、括弧は左右の括弧の釣り合いが取れた形で使用しなければなりません。括弧は、添え字、関数の引き数のリスト、参照修飾子、算術式、または条件を区切ります。

コロン { : }
コロンは区切り文字の 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、直後に区切り文字 (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、対になって使わなければなりません。アポストロフィは、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、直後に区切り文字 (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。アポストロフィは、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'}
これらの開始の区切り文字は、直前に区切り文字 (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。リテラルは、開始の区切り文字に引用符が使用されていた場合は引用符で終了し、開始の区切り文字にアポストロフィが使用されていた場合はアポストロフィで終了します。

DBCS リテラル区切り文字 {G"}、{G'}
これらの開始の区切り文字は、直前に区切り文字 (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。リテラルは、開始の区切り文字に引用符が使用されていた場合は引用符で終了し、開始の区切り文字にアポストロフィが使用されていた場合はアポストロフィで終了します。

国別リテラル区切り文字 {N"}、{N'}、{NX"}、{NX'}
これらの開始の区切り文字は、直前に区切り文字 (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。リテラルは、開始の区切り文字に引用符が使用されていた場合は引用符で終了し、開始の区切り文字にアポストロフィが使用されていた場合はアポストロフィで終了します。

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、直後に区切り文字 (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、対になって使わ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。(511ページの『COPY ステートメント』を参照。)

注: 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。

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、および記入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。記入項目は、文節と句に細分されます。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、228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、および記入項目

関連する規則が他に特に明記していない限り、それぞれに必須の文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。オプションの文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文節やステートメントに関しても同様に適用されます。

文法上の階層構造は、次のとおりです。

- 見出し部
  - 段落
    - 記入項目
      - 文節

- 環境部
  - セクション
    - 段落
      - 記入項目
        - 文節
          - 句

- データ部
  - セクション
    - 記入項目
      - 文節
        - 句

- 手続き部
  - セクション
    - 段落
      - 文
        - ステートメント
          - 句

記入項目

記入項目は、区切り文字のピリオドで終わる一連の文節からなります。記入項目は、見出し部、環境部、およびデータ部において指定できます。

© Copyright IBM Corp. 1991, 2001

文節

文節とは、記入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。文節は、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、区切り文字のピリオドで終わる 1 つまたは複数のステートメントの列です。文は、手続き部で指定できます。

ステートメント

ステートメントとは、COBOL 動詞とそのオペランドの有効な組み合わせのことです。ステートメントは、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、手続き部で指定することができます。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。

- 257ページの『命令ステートメント』
- 259ページの『条件ステートメント』
- 48ページの『名前のスコープ』
- 506ページの『コンパイラー指示ステートメント』

句

プログラムの中のそれぞれの文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。

参照フォーマット

COBOL プログラムは、COBOL 参照フォーマットに従って記述**されなければなりません**。図1 に示したのは、COBOL ソース行の参照フォーマットです。

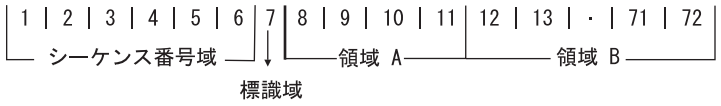


図 1. COBOL ソース行の参照フォーマット

次の各領域は、72 文字行の場合について説明したものです。

シーケンス番号域
桁 1 ～ 6

標識域
桁 7

領域 A
桁 8 ～ 11

領域 B
桁 12 ～ 72

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

標識域は、次のような指定を行うために使用します。

- 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
- テキストがコメントとして扱われること
- デバッグ行

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。標識域に置かれたスラッシュ (“/”) は、そのソース・リストに関して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、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

© Copyright IBM Corp. 1991, 2001

領域 A

次の項目は、領域 A から始めなければなりません。

- 部のヘッダー
- セクションのヘッダー
- 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、END CLASS、および END METHOD 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、部の始まりを示す区切り文字のピリオドで構成されます。

```
IDENTIFICATION DIVISION.

ENVIRONMENT DIVISION.

DATA DIVISION.

PROCEDURE DIVISION.
```

手続き部のヘッダーで USING 句を指定する場合を除き、部のヘッダーのすぐ後には区切り文字の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。以下に例を示します。

```
INPUT-OUTPUT SECTION.
```

データ部では、セクション・ヘッダーは記入項目の開始を示します。以下に例を示します。

```
FILE SECTION.

LINKAGE SECTION.

WORKING-STORAGE SECTION.
```

セクション・ヘッダーは、そのすぐ後に区切り文字の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。

環境部では、段落は段落ヘッダーとそれに続く 1 つまたは複数の記入項目から構成されます。以下に例を示します。

```
OBJECT-COMPUTER. コンピューター名
```

手続き部では、段落は段落名とそれに続く 1 つまたは複数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。これは領域 A の中で開始され、その後にスペースが続く必要があります。(149ページの『ファイル・セクション』を参照。）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区切り文字の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・プログラムの宣言部分の始まりと終わりを示すキーワードです。

手続き部では、キーワード DECLARATIVES および END DECLARATIVES はそれぞれ領域 A で開始し、その後にすぐ区切り文字の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。(227ページの『宣言部分』を参照。)

終了マーカー

「終了」マーカーとは、ワードの組み合わせとその直後に置かれる区切り文字のピリオドであり、COBOL プログラム、メソッド定義、クラス定義、ファクトリー定義、またはオブジェクト定義の終わりを示します。以下に例を示します。

```
END PROGRAM program-name.
END CLASS class-name.
END METHOD "method-name".
END OBJECT.
END FACTORY.
```

プログラムの場合

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。

クラスの場合

クラス名は、対応する CLASS-ID 段落のクラス名と一致している必要があります。

メソッドの場合

メソッド名は、対応する METHOD-ID 段落のメソッド名と一致している必要があります。

OBJECT 段落の場合

OBJECT 段落のヘッダーや終了マーカーには名前がありません。構文は単に **END OBJECT** となります。

FACTORY 段落の場合

FACTORY 段落のヘッダーや終了マーカーには名前がありません。構文は単に **END FACTORY** となります。

領域 B

次の項目は、領域 B から始めなければなりません。

- 記入項目、文、ステートメント、文節
- 継続行

記入項目、文、ステートメント、文節

最初の記入項目、文、ステートメント、または文節は、ヘッダーまたはその後に続く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。連続する文または記入項目は、その前の文または記入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 つの記入項目や文の中では、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。出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『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、記入項目、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。

以下のものは継続できません。

DBCS ユーザー定義語
DBCS リテラル
DBCS 文字を含む英数字リテラル
DBCS 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すべての文字は、同じ行に置かれている必要があります。たとえば、Z"、G"、N"、NX"、または X" がこれに該当します。

疑似テキスト区切り文字の区切り文字を構成する両方の文字である “==” は、同じ行に置かれている必要があります。

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 DBCS 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。

DBCS 文字が含まれていない英数字リテラルおよび国別リテラルには、次の規則が適用されます。

- 終了の引用符のない英数字リテラルまたは国別リテラルが継続される行に含まれている場合、その継続される行の末尾 (72 桁目まで) 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。継続行の標識域には、ハイフンを指定しなければならず、最初のブランク以外の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。
- 次の行に継続する英数字リテラルの最後の文字が 72 桁目の単一引用符である場合、継続行は 2 つの連続する引用符で開始しなければなりません。これによって、単一引用符がリテラルの値の一部となります。

  英数字リテラルの継続される行にある最後の文字が、領域 B にある単一引用符である場合、継続行は単一の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) 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、スペースが続く単一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、単一引用符にスペースを続けて終了します。

以下の例では、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。すべての文字は単一バイト EBCDIC で入力されますが、それぞれの EBCDIC 文字は、そのリテラルの値で対応する国別文字表現に変換されます。

```
|...+.*..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 文節として有効ではありません。

注: リテラルのそれぞれの連続セグメントの長さが領域 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、連続セグメントの最後の文字が 72 桁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

次の項目は、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。

- レベル番号
- コメント行
- コンパイラー指示ステートメント
- デバッグ行
- 疑似テキスト

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 ～ 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。これは後にスペースまたは区切り文字のピリオドが続かなければなりません。詳細については、162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 (/) を持つ行のことです。コメントは、その行の領域 A および領域 B 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、プログラム定義、メソッド定義、またはクラス定義の任意の場所に置くことができます。見出し部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 (たとえば、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 (/) で開始する必要があります。

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、 *Enterprise COBOL プログラミング・ガイド* 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。

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、CBL (PROCESS)、*CBL (*CONTROL)、DELETE、EJECT、INSERT、SKIP1/2/3、および TITLE も、領域 A および領域 B のどちらでも開始できます。

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、環境部 (OBJECT-COMPUTER 段落の後)、データ部、および手続き部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。

95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 か領域 B のどちらかで開始します。しかし、ある行の標識域 (第 7 桁) にハイフンがあり、その後に開始の疑似テキスト区切り文字が続く場合は、その行の領域 A はブランクになっていなければならず、継続行に関する規則がテキスト・ワードの形成に適用されます。

ブランク行

ブランク行とは、第 7 ～ 72 桁にスペース以外に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。

名前のスコープ

COBOL リソースとは、COBOL プログラムの中でユーザー定義語を介して参照されるリソースです。COBOL リソースを識別するために名前を使用することができます。ここでは、COBOL 名とそのスコープについて解説します。また、名前を参照できる範囲と、その使いやすさおよびアクセス可能性を説明します。

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、プログラムで宣言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。

プログラムの場合

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。

- グローバル名が宣言されるプログラムの中
- グローバル名を宣言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、データ記述記入項目で GLOBAL 文節を使用します。GLOBAL 文節の使用の詳細については、151ページの『GLOBAL 文節』を参照してください。

ローカル名を使用して、それが宣言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、データ記述記入項目にあるデータ名、ファイル名、レコード名、または条件名の宣言に GLOBAL 文節が含まれない場合、その名前はローカル名です。

メソッドの場合

メソッドで宣言された名前は、すべて暗黙的にローカル名です。

クラスの場合

クラス定義で宣言された名前は、そのクラス定義に含まれるすべてのメソッドに対してグローバル名になります。

OBJECT 段落の場合

OBJECT 段落のデータ部で宣言した名前は、その OBJECT 段落に含まれるすべてのメソッドに対してグローバル名になります。

FACTORY 段落の場合

FACTORY 段落のデータ部で宣言した名前は、その FACTORY 段落に含まれるすべてのメソッドに対してグローバル名になります。

注: 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 文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かを示しています。

© Copyright IBM Corp. 1991, 2001

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 文節が、データ名を宣言するデータ記述記入項目か、そのデータ記述記入項目が従属している別の記入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 文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 文節がそのレコード名を宣言するレコード記述で指定されている場合、あるいはファイル・セクション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 文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。

条件名
条件名は条件変数に値を関連付けます。

データ記述記入項目が、GLOBAL 文節を指定する別の記入項目に従属している場合は、条件名はそのデータ記述記入項目で宣言されます。

構成セクションの中で宣言される条件名は、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。詳細については、75ページの『プログラム名の命名規則』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。詳細については、75ページの『プログラム名の命名規則』を参照してください。

メソッド名
メソッド名はメソッドに名前を割り当てます。メソッド名は、英数字リテラルまたは国別リテラルの内容として指定する必要があります。

セクション名
セクション名は手続き部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
段落名は手続き部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、一貫してライブラリー名修飾を持たないテキスト名に定義されたものとして扱われます。

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、511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・プログラムに組み込む COPY テキストの名前を指定します。詳細については、511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、環境部の SPECIAL-NAMES 段落で特定の文字セットや照合シーケンス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。

(データの) クラス名
クラス名は、環境部の 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。

オブジェクト指向クラス名
オブジェクト指向クラス名は、クラスまたはサブクラスに名前を割り当てます。

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。

記号名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。

GLOBAL 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 GLOBAL 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。

外部および内部リソース

アクセス可能データ項目には、通常、保管されるデータの表記が含まれている必要があります。ファイル結合子には、通常、保管されるファイルに関する特定の情報が含まれている必要があります。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またはメソッドに対して**外部**または**内部**のどちらか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またはメソッドではなく、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またはメソッド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またはメソッド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またはメソッド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。

作業域セクション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記入項目に EXTERNAL 文節があると、それによって外部属性が与えられます。あるデータ記述記入項目によって記述されているデータ項目があり、そのデータ記述記入項目が、外部レコードを記述しているデータ記述記入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またはメソッドの内部データの一部になります。

実行単位内の 2 つのプログラムまたはメソッドは、次のような状況では、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するどのプログラムまたはメソッドからでも参照することができます。
- あるプログラムが別のプログラムに含まれている場合、それらの両方のプログラムは、含まれている方のプログラムか、あるいはプログラムを直接または間接に含んでいるプログラムの中で、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することができます。

実行単位内の 2 つのプログラムまたはメソッドは、次のような状況では、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、どのプログラムまたはメソッドからでも参照できる場合。(ただし、そのプログラムまたはメソッドがそのデータ・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に含まれている場合、それら両方のプログラムは、含まれている方のプログラムか、あるいはプログラムを直接または間接に含んでいるプログラムの中で、グローバル属性を処理するデータを参照できます。

EXTERNAL 文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・ファイル記述記入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記入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またはメソッドに対して内部です。EXTERNAL 文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

名前がプログラムで指定されている場合と、クラス定義で指定されている場合では、名前解決の規則が異なります。

プログラム内の名前

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、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 (これらの規則はクラスおよび含まれているメソッドにも適用されます)。

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。この名前のセットを使用すると、1 つまたは複数のリソースが識別されるまで、修飾の通常の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、次のことが当てはまります。

   - 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 を含んでいる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。

クラス定義内の名前

クラス定義では、次の単位の中でリソースを定義することができます。

ファクトリー・データ部
オブジェクト・データ部
メソッド・データ部

オブジェクト定義のデータ部おいてリソースがある名前で定義されている場合、そのオブジェクト定義のインスタンス・メソッドに同じ名前のリソースが定義されていないときは、その名前へのインスタンス・メソッドからの参照は、オブジェクト・データ部のリソースへの参照になります。

ファクトリー定義のデータ部おいてリソースがある名前で定義されている場合、そのファクトリー定義のファクトリー・メソッドに同じ名前のリソースが定義されていないときは、その名前へのファクトリー・メソッドからの参照は、ファクトリー・データ部のリソースへの参照になります。

メソッド内にリソースが定義されている場合、そのメソッド内での同じリソース名への参照はすべて、そのメソッド内のリソースへの参照になります。

1 つのメソッド・データ部、オブジェクト・データ部、ファクトリー・データ部の中で、同じ名前が複数のリソースに関連付けられている場合は、参照の修飾と固有性に関する標準規則が適用されます。

データ名、コピー・ライブラリー、および手続き部名の参照

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。ここでは、修飾に関する規則とデータを明示的または暗黙的に参照する場合の規則について説明します。

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、添え字付け、または参照変更を行うことができます。

別個のプログラムの中で一定の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た場合、および修飾だけを使用して別個のプログラムの 1 つの中の参照が固有に名前の付いたリソースを区別することができない場合、名前のスコープを制限する特定の規則が適用されます。その規則は、識別されたリソースが、参照が含まれているプログラムで記述されたリソースになることを確認します。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

ある名前が複数の階層で存在している場合、その階層の高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。その高位レベルの名前は**修飾子**と呼ばれ、そ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 つまたは複数の句を指定します。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。たとえば、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 MASTER-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 MASTER-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 MASTER-RECORD OF MASTER-FILE は有効な指定ですが、不必要です。

© Copyright IBM Corp. 1991, 2001

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 文節では修飾しては**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。

データ属性の指定

明示的なデータ属性とは、プログラマーが実際の COBOL プログラムのコーディングで指定するデータ属性のことです。

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 文節は必ずしも指定する必要はありません。それを省略し、PICTURE 文節に 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 文節で記号 N が使用されている場合、NSYMBOL(DBCS) コンパイラー・オプションが有効なときは USAGE DISPLAY-1 がデフォルトであり、NSYMBOL(NATIONAL) コンパイラー・オプションが有効なときは USAGE NATIONAL がデフォルトです。これは、暗黙の属性です。

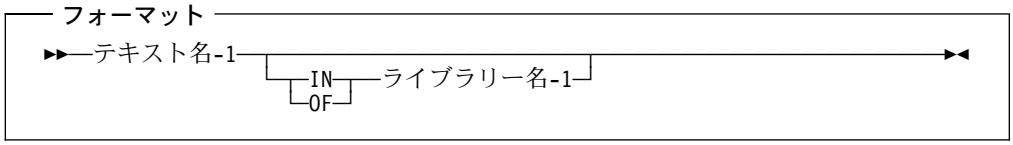
同一の名前

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。複数のリソースに同じ名前を付けても、プログラムは別のプログラムで記述されている同じ名前のリソースを参照しないで、たとえ、それが異なるタイプのユーザー定義語であったとしても、そのプログラムが記述しているリソースを参照します。

これらの同じ規則がクラスおよびそれに含まれているメソッドにも適用されます。

COPY ライブラリーの参照

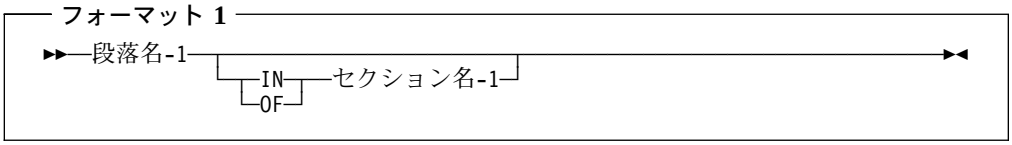
フォーマット

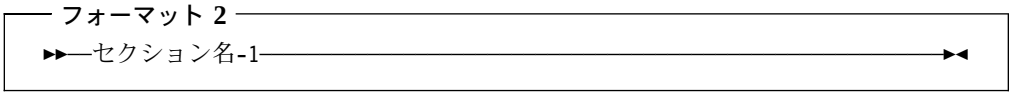


ライブラリー名-1 が指定されていない場合、SYSLIB がライブラリー名とみなされます。

COPY ライブラリーの参照に関する規則については、511ページの『COPY ステートメント』を参照してください。

手続き部の名前の参照





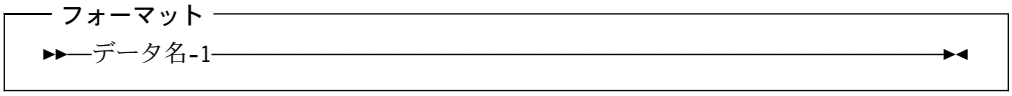
プログラムで明示的に参照される手続き部の名前は、セクション内で固有でなければなりません。セクション名は、228ページの『プロシージャー』で説明しているように、段落名に対して使用できる最高でしかも唯一の修飾子であり、参照される場合は固有でなければなりません。

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。

データ部の名前の参照

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。



データ名-1

任意のデータ記述記入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。

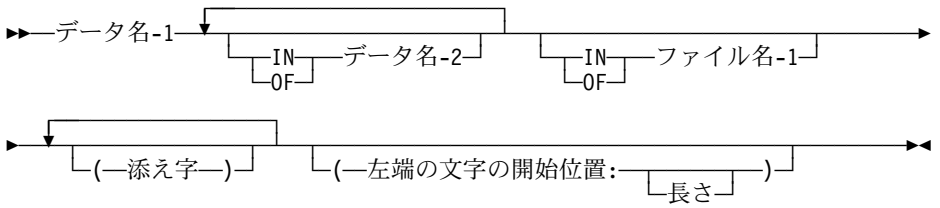
ID

本書の構文図で使用される場合、**ID** という語は、データ名または関数 ID、および参照の固有性の必要に応じて修飾子、添え字、および参照修飾子を伴った有効な組み合わせを表しています。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、58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、61ページの『参照変更』を参照してください。

フォーマット 1

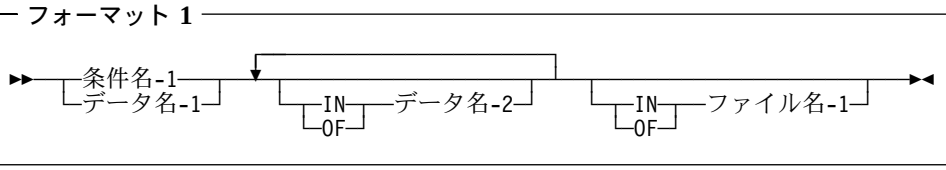


データ名-1、データ名-2
レコード名を指定できます。

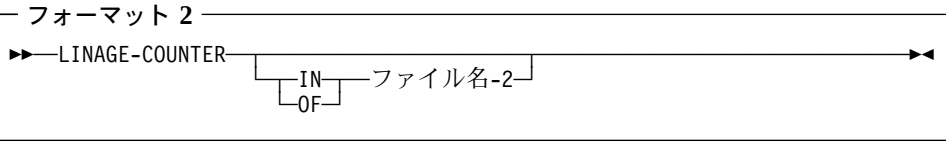
ファイル名-1
データ部の記入項目 FD または SD と一致している必要があります。

このプログラムの中ではファイル名-1 は固有でなければなりません。

フォーマット 1



フォーマット 2



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記入項目によって参照できます。

ファイル名-1
データ部の記入項目 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 文節を含んでいるファイル記入項目がソース・プログラム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
データ部の記入項目 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。

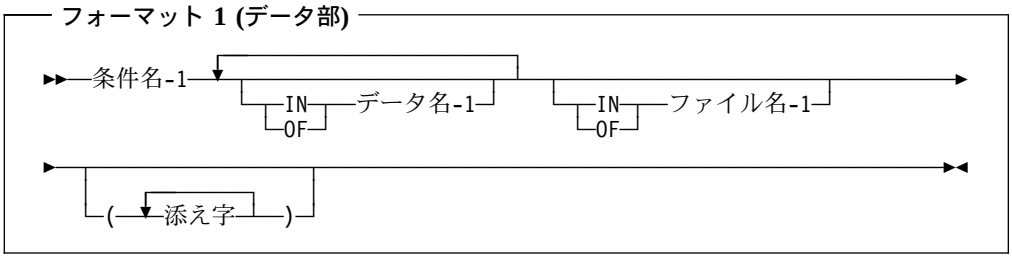
修飾によってデータ名を固有なものにできない場合には、データ名が重複しないようにしてください。

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

同じデータ部の中では、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記入項目に GLOBAL 文節を含めてはなりません。

明示的に参照されるデータ部の名前は、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。データ階層の最高のレベルの名前が参照される場合、それは固有でなければなりません。これは、レベル標識 (ファイル・セクションの FD または SD)、またはレベル番号 01 に関連するデータ項目です。02 ～ 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



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記入項目によって参照できます。

明示的に参照される場合は、名前のスコープ規則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。

以降の章の一般的なフォーマットにおいて「条件名」は、必要に応じて修飾または添え字が付けられた条件名を表します。

データ名-1

レコード名を指定できます。

ファイル名-1
データ部の記入項目 FD または SD と一致している必要があります。

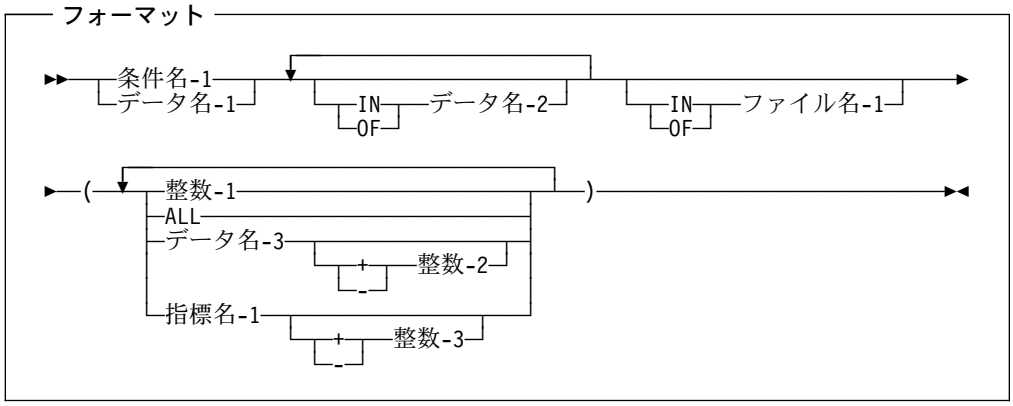
このプログラムの中ではファイル名-1 は固有でなければなりません。

簡略名-1
簡略名として使用できる値の詳細については、97ページの『SPECIAL-NAMES 段落』を参照してください。

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です。

フォーマット



条件名-1
条件名-1 の条件変数には OCCURS 文節が含まれているか、または OCCURS 文節が含まれているデータ記述記入項目に従属していなければなりません。

データ名-1
OCCURS 文節が含まれているか、または OCCURS 文節が含まれているデータ記述記入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。

データ名-3
整数を表す基本数字項目でなければなりません。

データ名-3 は修飾できます。データ名-3 をウィンドウ化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記入項目に対応します。

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 文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。複数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。

たとえば、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。62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は、その項目がテーブル・エレメントか、**または**テーブル・エレメントに関連付けられた項目あるいは条件名である場合以外は、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 文節の中
- KEY が OCCURS 文節の句になっている場合

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 文節で指定されている項目の最大オカレンス数です。

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数である必要はありません。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。たとえば、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、次のものが含まれます。

```
    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 つまたは複数の指標名をデータ記述記入項目に OCCURS 文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 文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記入項目はありません。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、シリアル検索や非シリアル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設定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、174ページの『INDEXED BY 句』を参照してください。

相対添え字付け

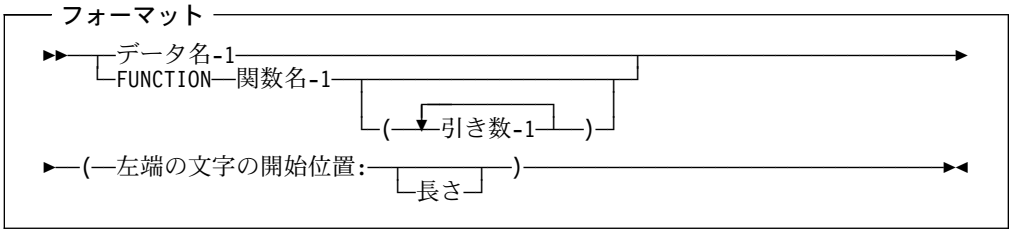
相対添え字付けでは、テーブル・エレメントの名前の後に、データ名または指標名の後に演算子 + または - が続き、さらに正の符号付き整数リテラルまたは符号なし整数リテラルが続く形式の添え字を指定します。

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



データ名-1

使用法が DISPLAY、DISPLAY-1、または NATIONAL であるデータ項目を参照しなければなりません。

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 をウィンドウ化日付フィールドにすることはできません。

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。

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 0 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、ウィンドウ化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、0 でない正の整数でなければなりません。

長さ の評価結果は、ウィンドウ化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 -1 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 + 1 - *左端の文字の開始位置* に等しくなります。

注: 使用法 DISPLAY-1 および NATIONAL の場合、それぞれの文字位置は 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、使用法 DISPLAY-1 および NATIONAL における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。使用法 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使用法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 のデータ記述記入項目に SIGN IS SEPARATE 文節がある場合は、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、数字、数字編集、英字、または英数字編集として記述されている場合、そのデータ項目は、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 とその引き数 (引き数がある場合) 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 文節を伴わない基本データ項目とみなされます。

関数が参照変更される場合、この固有のデータ項目は、国別の関数の場合は国別のクラスおよびカテゴリーに属し、使用法 NATIONAL を持ちます。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、使用法 DISPLAY を持ちます。データ名-1 が参照変更される場合、この固有のデータ項目は、データ名-1 によって参照されるデータ項目に対して定義されているクラス、カテゴリーおよび使用法と同じものに属します。一方、データ名-1 のカテゴリーが数字、数字編集、外部浮動小数点、または英数字編集である場合は、その固有のデータ項目は英数字のクラスおよびカテゴリーに属します。

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

次の例では、WHOLE-NAME によって参照されたデータ項目の最初の 10 文字を、FIRST-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  FIRST-NAME  PIC X(10).
 ⋮

    MOVE WHOLE-NAME(1:10) 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.
```

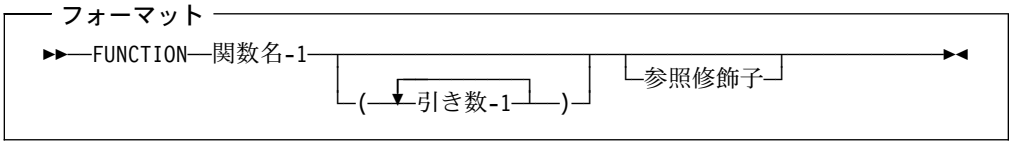
次の例では、TAB の 3 番目のオカレンスの 4 番目と 5 番目の文字を、SUFFIX という変数に転送します。

```
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、関数の評価の結果としてデータ項目を固有なものとして参照する文字ストリングと区切り文字からなる構文的に正しい文字ストリングのことです。



引き数-1

引き数-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。

詳細については、441ページの『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

カテゴリーが英数字または国別である関数についてのみ指定することができます。

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 ID または国別 ID の使用が可能で、かつ、関数への参照が特に禁止されていない個所であれば、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、および許容値など) を持つように要求され、その定義と指定された特定の引き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。

整数または数字の関数を参照する関数 ID は、算術式が使用できる個所であれば、どこでも使用できます。

制御の移動

手続き部では、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。(下記の注を参照。) 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構造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構造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。(257 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。)

注: 「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。

- そのプログラムに手続き部がない場合。
- 宣言セクションの最後のステートメントの後であり、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。
- プログラムまたはメソッド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。

© Copyright IBM Corp. 1991, 2001

- 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。
- COBOL メソッドの外に制御を移す EXIT METHOD ステートメントの後である場合。
- END PROGRAM マーカーまたは END METHOD 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。

同様に、制御がメソッドの手続き部の最後に到達し、次の実行可能ステートメントがない場合は、暗黙の EXIT METHOD ステートメントが実行されます。

2000 年言語拡張および日付フィールド

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 ～ 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 ～ 1999 の年を表すという想定を取り除く、ウィンドウ化と呼ばれる手法を使用して行われます。ウィンドウ化では 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。

たとえば、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、1960-2059 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、164ページの『DATE FORMAT 文節』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、166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、以下の言語エレメントを Enterprise COBOL に導入します。

- データ記述記入項目における **DATE FORMAT** 文節。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は、*Enterprise COBOL プログラミング・ガイド* を参照してください。

© Copyright IBM Corp. 1991, 2001

注: COBOL プログラムが DATEPROC コンパイラー・オプションを使用して (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

本書では、2000 年言語拡張を参照する場合に以下の用語を使用しています。

日付フィールド

日付フィールドは、次のいずれかにすることができます。

- データ記述記入項目が DATE FORMAT 文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、および DAY YYYYDDD。
- 特定の算術演算の結果 (詳細については、232ページの『日付フィールドでの算術演算』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**と**ウィンドウ化日付フィールド**の両方を指します。

ウィンドウ化日付フィールド

ウィンドウ化日付フィールドとは、**ウィンドウ化西暦年**を含む日付フィールドです。ウィンドウ化西暦年は、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。

注: 拡張日付フィールドの主な用途は、ウィンドウ化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。たとえば、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 句において YY または YYYY の前に 1 つまたは複数の 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。

日付フォーマット

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 文節または DATEVAL 組み込み関数引き数-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および組み込み関数によって暗黙的に (詳細については、67ページの『日付フィールド』を参照)

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、それが COBOL のどの部で使用される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 つのデータ・フィールドには互換性があります。

- 日付フォーマットが同じである。
- 両方ともウィンドウ化日付フィールドである (一方がウィンドウ化西暦年である 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。

ウィンドウ化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ます。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ります。年号部分については、ウィンドウ化でも拡張することも可能です。たとえば、DATE FORMAT YYXXX のウィンドウ化日付フィールドは、以下と互換性があります。

- DATE FORMAT YYXXX の別のウィンドウ化日付フィールド

- DATE FORMAT YYYYXXX の拡張日付フィールド

非日付データ

非日付データは、次のいずれかです。

- 日付記述記入項目が DATE FORMAT 文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。たとえば、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 プログラマーが使用できる世紀ウィンドウには、次のものがあります。

1. ウィンドウ化日付フィールドの場合は、YEARWINDOW コンパイラー・オプションによって指定される。
2. ウィンドウ化組み込み関数 DATE-TO-YYYYMMDD、DAY-TO-YYYYDDD、および YEAR-TO-YYYY の場合は、引き数-2 によって指定される。
3. 言語環境プログラム呼び出し可能サービスの場合は、CEESCEN で指定される。

COBOL ソース単位構造

COBOL プログラムの構造 72
ネストされたプログラム 74

COBOL クラス定義構造 77

COBOL メソッド定義構造 81

© Copyright IBM Corp. 1991, 2001

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 に含まれる場合、同じくプログラム B を含むプログラムがプログラム A に他に存在しない場合、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は、74ページの『ネストされたプログラム』および *Enterprise COBOL プログラミング・ガイド* 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 コンパイラーを操作した結果生じた機械語です。

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 つまたは複数のオブジェクト・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、記入項目、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。

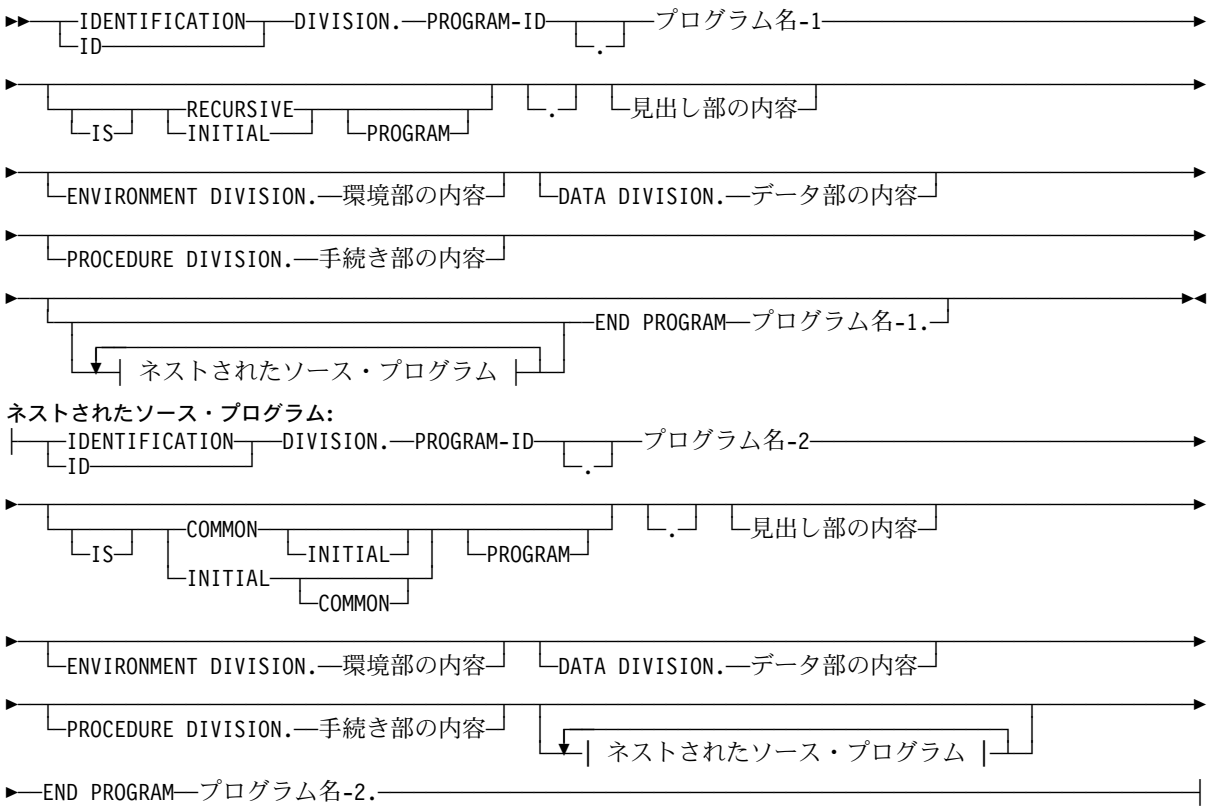
- 見出し部
- 環境部
- データ部
- 手続き部

COBOL ソース・プログラムの終了は、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。

次に、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記入項目とステートメントのフォーマットを示します。

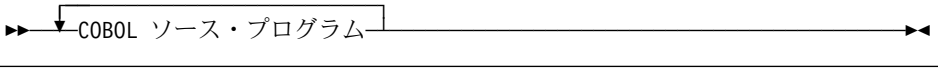
© Copyright IBM Corp. 1991, 2001

フォーマット - COBOL 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。次に示すのは、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記入項目とステートメントのフォーマットです。

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**含めることがあります。

COBOL プログラムに他の COBOL プログラムを組み込むことができます。含まれている (またはネストされた) プログラムは、その中にさらに他のプログラムを含むことがあります。含まれているプログラムは、別のプログラムに**直接的**にまたは**間接的**に含まれることがあります。

THREAD オプションを使用してコンパイルされたプログラムの場合、ネストされたプログラムはサポートされていません。図2 に、直接または間接に含まれたプログラムのある、ネストされたプログラム構造を示します。

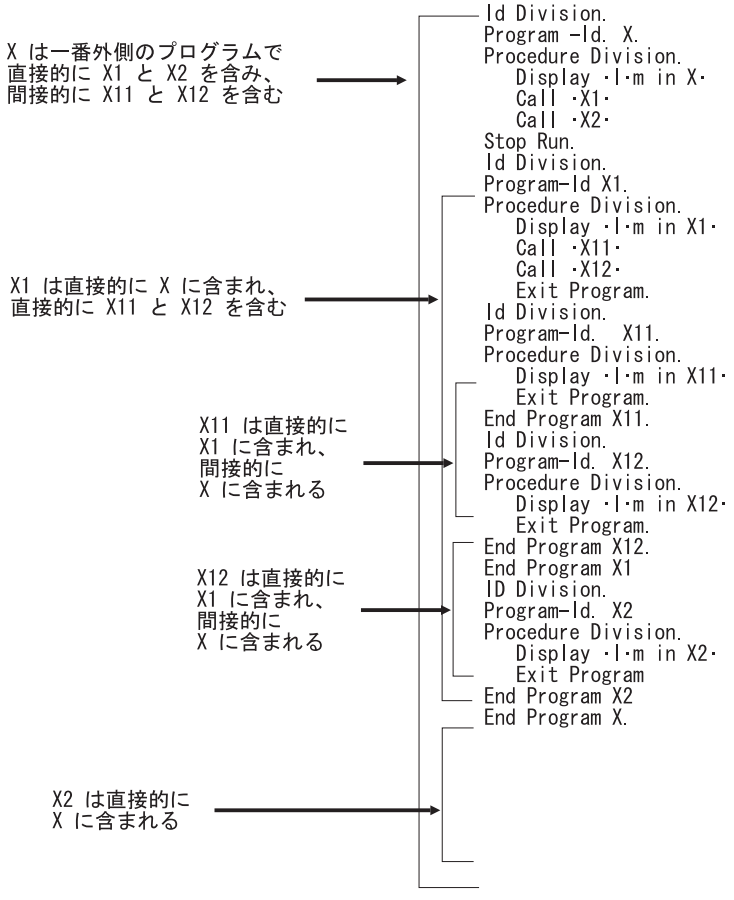


図 2. 直接的および間接的にプログラムを含むネストされたプログラムの構造

プログラム名の命名規則

プログラム名は、プログラムの見出し部の PROGRAM-ID 段落で指定されます。プログラム名が参照されるのは、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規定されます。

- 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。

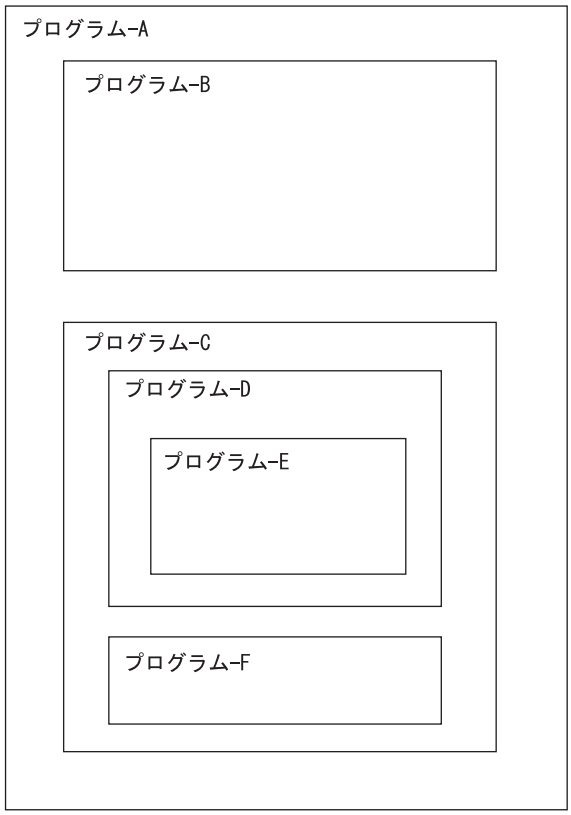
  どのプログラムを呼び出すかを判別するために使用するメカニズムは以下のとおりです。

  – 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、独立して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-D をプログラム-C が参照することができます。なぜなら、プログラム-C はプログラム-D を含んでいるからです。また、プログラム-C に含まれる任意のプログラムもプログラム-D を参照することができます。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 で参照することはできますが、プログラム-E、プログラム-A、またはプログラム-B のステートメントによって参照することはできません。

COBOL クラス定義構造

Enterprise COBOL は、オブジェクト指向型の構文をサポートするため、COBOL プログラムと Java プログラムの相互協調処理が容易になります。

Enterprise COBOL オブジェクト指向構文を使用して、以下のことを行うことができます。

- メソッドとデータを COBOL でインプリメンテーションした状態で、クラスを定義する。
- Java クラスまたは COBOL クラスのインスタンスを作成する。
- Java オブジェクトまたは COBOL オブジェクトにメソッドを呼び出す。
- Java クラスまたは別の COBOL クラスから継承したクラスを書き込む。
- 多重定義メソッドを定義して呼び出す。

Java のオブジェクトの基本的な機能は、COBOL 言語から直接アクセスできます。COBOL のプログラマーは、*Enterprise COBOL プログラミング・ガイド* で説明するように、Java Native Interface (JNI) のサービスを呼び出すことによって、追加の機能を利用することができます。

Java プログラムは、マルチスレッド化することができ、Java の相互協調処理を行うには非同期シグナルの許容が必要です。したがって、これらの Java プログラムと COBOL を混在させるには、*Enterprise COBOL プログラミング・ガイド* で説明するように、THREAD コンパイラー・オプションによって提供されるスレッド使用可能化を使用する必要があります。

Java の String データは実行時に Unicode で表現されます。国別データ型の Enterprise COBOL によって提供される Unicode サポートを使用すると、COBOL プログラムと Java の String データをやりとりできるようになります。

以下に、Java との相互運用性 (インターオペラビリティー) を実現するためにオブジェクト指向 COBOL で使用するエンティティーおよび概念を示します。

クラス
ゼロ、1 つ、または複数のオブジェクト・インスタンスの操作および状態を定義し、複数のオブジェクト・インスタンスが共用する共通オブジェクト (ファクトリー・オブジェクト) の操作および状態を定義するエンティティー。

COBOL INVOKE ステートメントの NEW オペランドを使用して、または Java クラス・インスタンス生成式を使用して、オブジェクト・インスタンスを作成します。

オブジェクト・インスタンスは、もはや使用されない状態になったときに、Java ランタイム・システムのガーベッジ・コレクションによって自動的に解放されます。個々のオブジェクトを明示的に解放することはできません。

インスタンス・メソッド
サポートされる操作の 1 つを、クラスのオブジェクト・インスタンスに定義するプロシージャー・コード。COBOL クラスが導入するインスタンス・メソッドは、クラス定義の OBJECT 段落内で定義されます。

COBOL インスタンス・メソッドは、Java の public 非静的メソッドと同義です。

© Copyright IBM Corp. 1991, 2001

特定のオブジェクト・インスタンスの実行は、インスタンス・メソッド上で、COBOL INVOKE ステートメント、または Java メソッド呼び出し式を使用して行います。

インスタンス・データ
個々のオブジェクト・インスタンスの状態を定義するデータ。COBOL クラス内のインスタンス・データは、クラス定義の OBJECT 段落のデータ部のワーキング・ストレージ・セクションで定義されます。

COBOL インスタンス・データは、Java クラスの private 非静的メンバー・データと同義です。

オブジェクトの状態には、継承されたクラスによって導入されたインスタンス・データの状態も含まれます。それぞれのインスタンス・オブジェクトごとに、そのクラス定義で定義されているインスタンス・データのコピーと、継承されたクラスで定義されているインスタンス・データのコピーがあります。

COBOL オブジェクト・インスタンス・データへのアクセスは、データを定義するクラス定義に定義されている、COBOL インスタンス・メソッド内からのみ行うことができます。

オブジェクト・インスタンス・データの初期化は VALUE 文節を使用して行うことができます。または、インスタンス・メソッドを書き込んで、カスタム初期化を行うことができます。

ファクトリー・メソッド、静的メソッド
サポートされる操作の 1 つを、クラスの共通ファクトリー・オブジェクトに定義するプロシージャー・コード。COBOL ファクトリー・メソッドは、クラス定義の FACTORY 段落内で定義されます。ファクトリー・メソッドは、クラスの個々のインスタンス・オブジェクトに関連付けられるのではなく、クラスに関連付けられます。

COBOL ファクトリー・メソッドは、Java の public 静的メソッドと同義です。

COBOL からの COBOL ファクトリー・メソッドの実行は、クラス名を第 1 オペランドとして指定する INVOKE ステートメントを使用して行います。Java プログラムからの COBOL ファクトリー・メソッドの実行は、静的メソッド呼び出し式を使用して行います。

ファクトリー・メソッドは、データが同一のクラス定義で記述されていたとしても、そのクラスのインスタンス・データを直接処理することはできません。ファクトリー・メソッドは、インスタンス・メソッドを呼び出して、インスタンス・データを処理する必要があります。

COBOL ファクトリー・メソッドは一般的に、オブジェクト・インスタンスを作成するカスタマイズ・メソッドを定義するために使用されます。たとえば、カスタマイズ・ファクトリー・メソッドをコーディングすることができます。これによって、初期値をパラメーターとして受け入れ、INVOKE ステートメントの NEW オペランドを使用してインスタンス・オブジェクトを作成し、インスタンス・オブジェクトの初期化に使用するために、これらの初期値を引き数として渡してカスタマイズ・インスタンス・メソッドを呼び出します。

ファクトリー・データ、静的データ
個々のオブジェクト・インスタンスにではなく、クラスに関連付けられるデータ。
COBOL ファクトリー・データは、クラス定義の FACTORY 段落内のデータ部のワーキング・ストレージ・セクションで定義されます。

COBOL ファクトリー・データは、Java の private 静的データと同義です。

1 つのクラスに対して 1 つのファクトリー・データのコピーがあります。ファクトリー・データは、そのクラスのみに関連付けられ、クラスのすべてのオブジェクト・インスタンスによって共用されます。ファクトリー・データは、特定のインスタンス・オブジェクトに関連付けられてはいません。たとえば、ファクトリー・データ項目は、作成されたインスタンス・オブジェクトの数を保持するために使用される場合があります。

COBOL ファクトリー・データへのアクセスは、同じクラス定義に定義された COBOL ファクトリー・メソッド内でのみ行うことができます。

継承

継承とは、クラス定義 (継承側クラス) が、別のクラス定義 (被継承クラス) に書き込まれているメソッド、データ記述、およびファイル記述を取得するメカニズムです。継承関係にある 2 つのクラスがともに認識されるとき、継承側クラスはサブクラス (派生クラスまたは子クラス) であり、被継承クラスはスーパークラス (親クラス) です。また、継承側クラスは、親クラスがその親クラスから継承した、メソッド、データ記述、およびファイル記述を間接的に取得することもできます。

COBOL クラスは、正確に 1 つの親クラスから継承する必要があります。これは、COBOL または Java でインプリメンテーションすることができます。

すべての COBOL クラスは、java.lang.Object クラスから、直接的または間接的に継承する必要があります。

インスタンス変数

OBJECT 段落のデータ部で定義される個々のデータ項目。

Java Native Interface (JNI)

非 Java プログラムとの相互協調処理を実現するために設計された Java の機能。

Java Native Interface (JNI) 環境ポインター

JNI サービスの呼び出しに使用する JNI 環境構造のアドレスを取得するために使用するポインター。JNI 環境ポインターを参照するために、COBOL 特殊レジスター JNIENVPTR が用意されています。

オブジェクト・リファレンス

個々のオブジェクトを識別して参照するために使用される情報が含まれているデータ項目。オブジェクト・リファレンスは、Java クラスまたは COBOL クラスのインスタンスであるオブジェクトを参照することができます。

サブクラス

別のクラスから継承するクラス。被継承クラスの派生クラスまたは子クラスとも呼ばれます。

スーパークラス

別のクラスによって継承されるクラス。継承側クラスの親クラスとも呼ばれます。

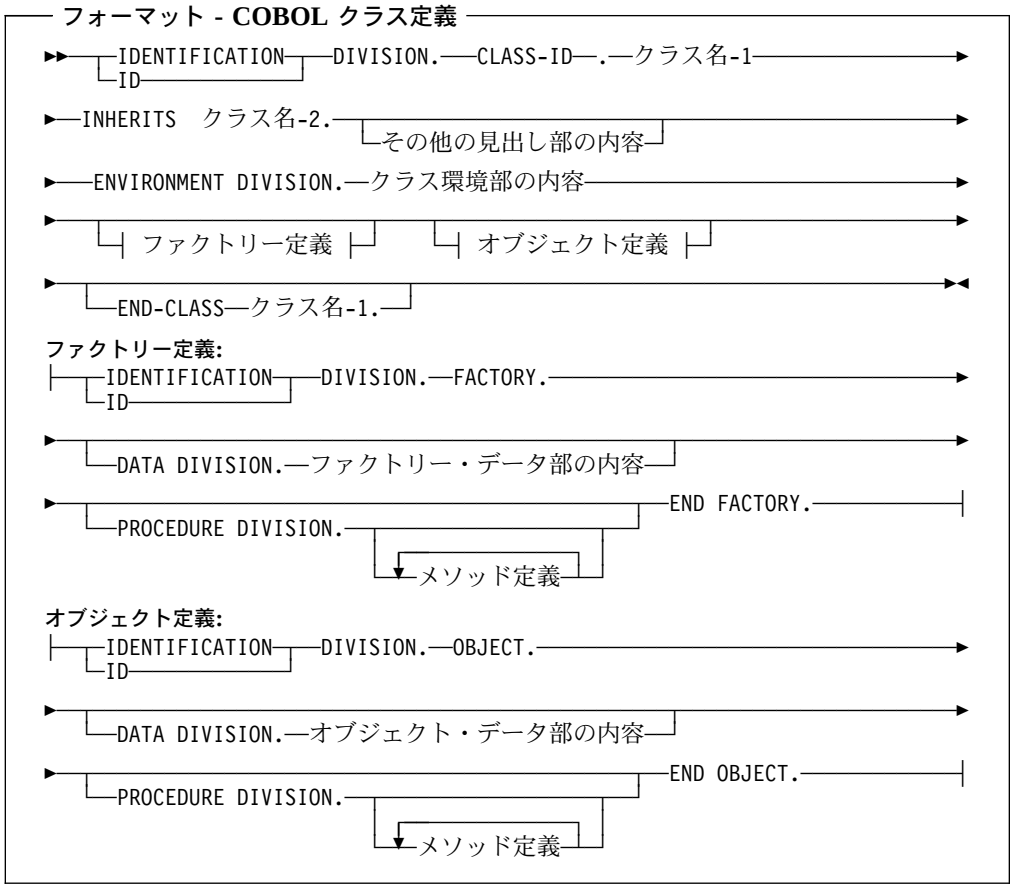
COPY ステートメントおよび REPLACE ステートメントと END CLASS マーカーを除き、COBOL クラス定義のステートメント、記入項目、段落、およびセクションは、以下の構造に分類されます。

- 見出し部
- 環境部 (構成セクションのみ)
- ファクトリー定義
  - 見出し部
  - データ部
  - 手続き部

        - メソッド定義 (複数の場合もある)
- オブジェクト定義
  - 見出し部
  - データ部
  - 手続き部
    - メソッド定義 (複数の場合もある)

COBOL クラス定義の終了は、END CLASS マーカーによって示されます。

COBOL クラス定義のフォーマットは、以下のとおりです。



END CLASS
クラス定義の終了を指定します。

COBOL メソッド定義構造

COBOL メソッド定義ではメソッドを記述します。メソッド定義の指定は、(FACTORY 段落および OBJECT 段落) クラス定義内でのみ行うことができます。

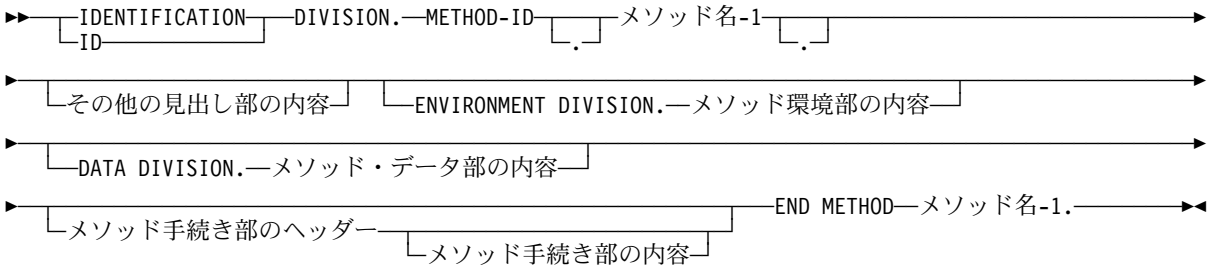
COPY および REPLACE ステートメントと END METHOD マーカーを除き、COBOL メソッド定義のステートメント、記入項目、段落、およびセクションは、次の 4 つの部に分類されます。

- 見出し部
- 環境部 (入出力セクションのみ)
- データ部
- 手続き部

COBOL メソッド定義の終了は END METHOD マーカーによって示されます。

COBOL メソッド定義のフォーマットは、以下のとおりです。

フォーマット - COBOL メソッド定義



METHOD-ID
メソッド定義を識別します。詳細は、90ページの『METHOD-ID 段落』を参照してください。

メソッド手続き部のヘッダー
手続き部の開始を示し、メソッド・パラメーターと戻り項目 (ある場合) を識別します。詳細は、224ページの『手続き部のヘッダー』を参照してください。

END METHOD
メソッド定義の終了を指定します。

オブジェクト定義に定義されたメソッドは、インスタンス・メソッドです。指定されたクラス内のインスタンス・メソッドは、以下のデータにアクセスすることができます。

- そのクラスの OBJECT 段落のデータ部に定義したデータ (インスタンス・データ)
- そのインスタンス・メソッドのデータ部に定義したデータ (メソッド・データ)

インスタンス・メソッドは、親クラスに定義されたインスタンス・データ、それ自体のクラスに定義されたファクトリー・データ、またはそのクラスの別のメソッドに定義されたメソッド・データに、直接的にアクセスすることはできません。これらのデータにアクセスするには、メソッドを呼び出す必要があります。

© Copyright IBM Corp. 1991, 2001

ファクトリー定義に定義されているメソッドは、ファクトリー・メソッドです。指定されたクラス内のファクトリー・メソッドは、以下のデータにアクセスすることができます。

- そのクラスの FACTORY 段落のデータ部に定義したデータ (ファクトリー・データ)
- そのファクトリー・メソッドのデータ部に定義したデータ (メソッド・データ)

ファクトリー・メソッドは、親クラスに定義されたファクトリー・データ、それ自体のクラスに定義されたインスタンス・データ、またはそのクラスの別のメソッドに定義されたメソッド・データに、直接的にアクセスすることはできません。これらのデータにアクセスするには、メソッドを呼び出す必要があります。

メソッドは COBOL プログラムおよびメソッドから呼び出すことができ、Java プログラムから実行することができます。メソッドは、それ自体を直接的または間接的に呼び出す INVOKE ステートメントを実行できます。したがって、COBOL メソッドは暗黙的に再帰的です (COBOL プログラムの再帰がサポートされるのは RECURSIVE 属性が PROGRAM-ID 段落に指定されている場合だけであるのとは異なります)。

見出し部

見出し部 84
PROGRAM-ID 段落 86
CLASS-ID 段落 88
FACTORY 段落 89
OBJECT 段落 90
METHOD-ID 段落 90
オプションの段落 91

© Copyright IBM Corp. 1991, 2001

見出し部

見出し部は、すべての COBOL ソース・プログラム、ファクトリー定義、オブジェクト定義、およびメソッド定義の最初の部でなければなりません。プログラム、クラス、またはメソッドに名前を付け、ファクトリー定義およびオブジェクト定義を指定します。プログラム、クラス、またはメソッド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、見出し部の最初の段落は PROGRAM-ID 段落でなければなりません。他の段落はオプションであり、任意の順序で指定することができます。

クラス IDENTIFICATION DIVISION
クラスの場合、見出し部の最初の段落は CLASS-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。

ファクトリー IDENTIFICATION DIVISION
ファクトリー IDENTIFICATION DIVISION には、FACTORY 段落ヘッダーのみを含めることができます。

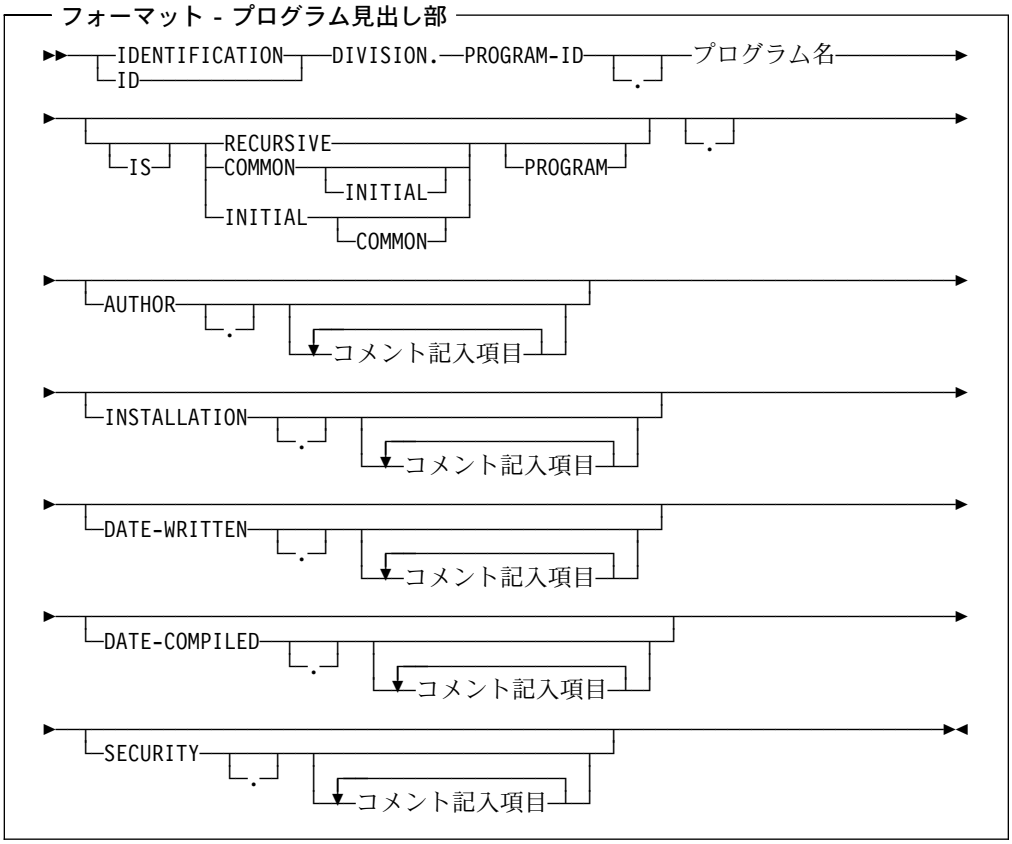
オブジェクト IDENTIFICATION DIVISION
オブジェクト IDENTIFICATION DIVISION には、OBJECT 段落ヘッダーのみを含めることができます。

メソッド IDENTIFICATION DIVISION
メソッドの場合、見出し部の最初の段落は METHOD-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、以下のとおりです。

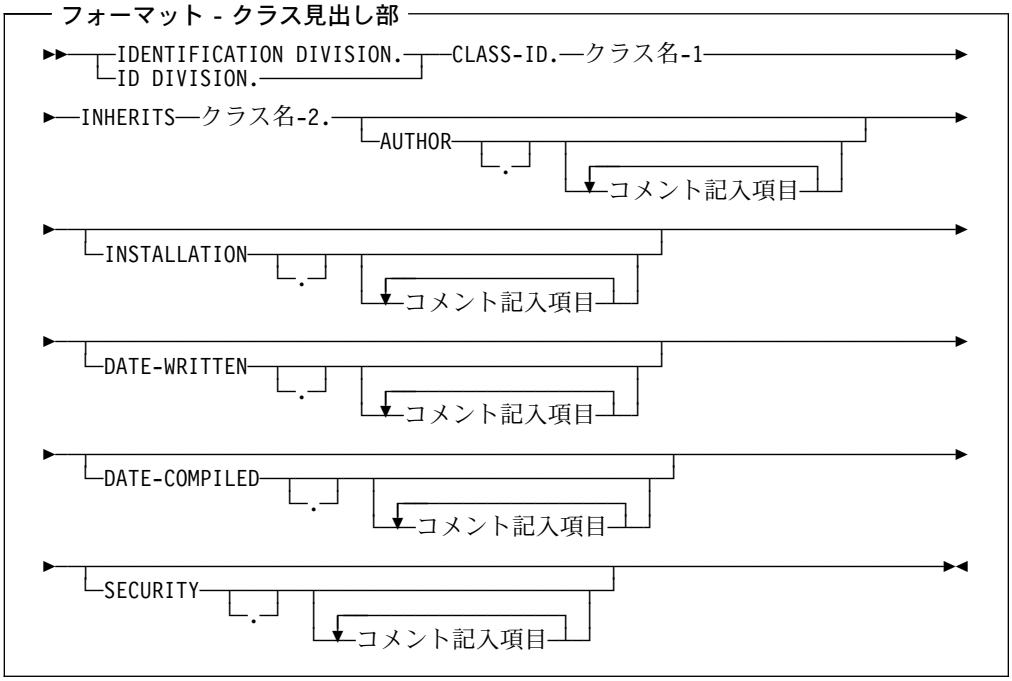
© Copyright IBM Corp. 1991, 2001

フォーマット - プログラム見出し部



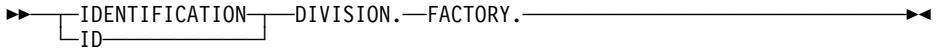
クラス IDENTIFICATION DIVISION のフォーマットは、以下のとおりです。

フォーマット - クラス見出し部



ファクトリー IDENTIFICATION DIVISION のフォーマットは、以下のとおりです。

フォーマット - ファクトリー見出し部

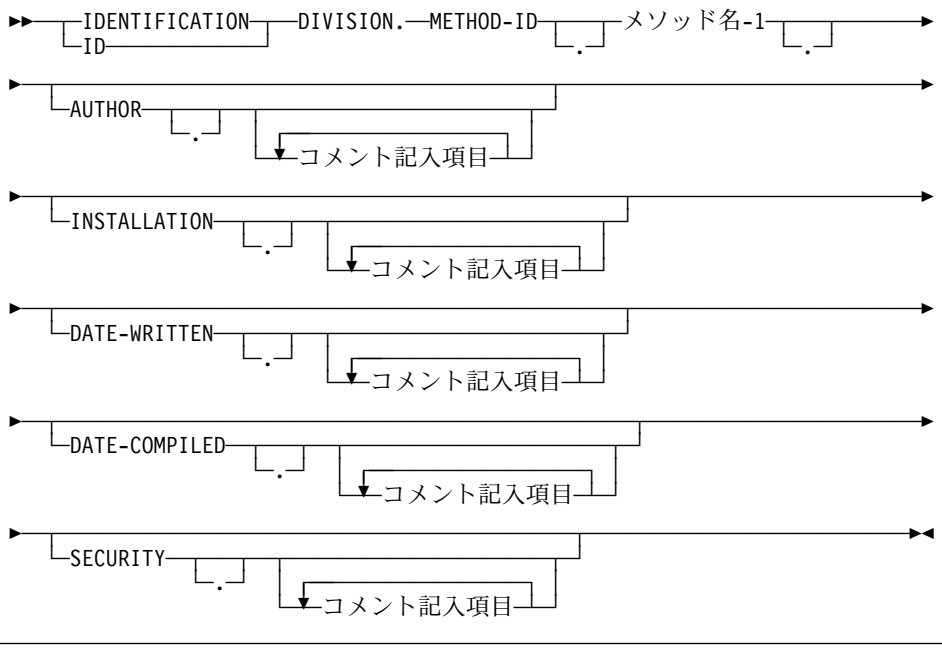


オブジェクト IDENTIFICATION DIVISION のフォーマットは、以下のとおりです。

フォーマット - オブジェクト見出し部

メソッド IDENTIFICATION DIVISION のフォーマットは、以下のとおりです。

フォーマット - メソッド見出し部



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、見出し部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 (COMPAT)

名前は長さが最大 30 文字です。

ユーザー定義語として指定するときは、ハイフンと 0 ～ 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として使用することはできません。

英数字リテラルにおけるプログラム名の規則は同じですが、最外部プログラムの名前には拡張文字 $、#、および @ を含めることができる点で異なっています。

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、その長さは最大 160 文字です。

ハイフンと英数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として使用することはできません。

PGMNAME (LONGMIXED)

プログラム名はリテラルとして指定する必要があります。

名前は長さが最大 160 文字です。

プログラム名は X'41' ～ X'FE' の範囲の任意の文字から構成することができます。

PGMNAME コンパイラー・オプションと、コンパイラーが名前を処理する方法については、 *Enterprise COBOL プログラミング・ガイド* 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、オプションの文節。

RECURSIVE 文節は、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、ネストされた副プログラムを含むことはできません。

RECURSIVE 文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-1 に再帰的に再入させることができます。RECURSIVE 文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ワーキング・ストレージ・セクションは、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ローカル・ストレージ・セクションは、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ファイル・セクションの 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

RECURSIVE 文節は、THREAD オプションを使用してコンパイルされたプログラムが必須です。

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ていて、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 文節は、ネストされたプログラムでのみ使用できます。プログラム名の規則の詳細については、 *Enterprise COBOL プログラミング・ガイド* 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プログラムがその初期状態に置かれることを指定します。 INITIAL 属性は、THREAD オプションを使用してコンパイルされたプログラムではサポートされていません。

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にある場合、次のことが実行されます。

- プログラムのワーキング・ストレージ・セクションにある内部データが初期設定されます。VALUE 文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 文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、75ページの『プログラム名の規則』を参照してください。

CLASS-ID 段落

CLASS-ID 段落は、クラスの名前を指定し、選択された属性をそのクラスに割り当てます。CLASS-ID 段落は必須であり、クラス見出し部の最初の段落でなければなりません。

クラス名-1
クラスを識別するユーザー定義語。クラス名-1 はオプションとして、クラス定義の構成セクションの Repository 段落に記入項目を持つことができます。

INHERITS
クラス名-1 がクラス名-2 (親クラス) のサブクラス (または派生クラス) であることを定義する文節。クラス名-1 は、直接的にも間接的にもクラス名-1 から継承することはできません。

クラス名-2
クラス名-1 によって継承されるクラスの名前。クラス名-2 は、クラス定義の構成セクションの REPOSITORY 段落で指定する必要があります。

一般規則

クラス名-1 およびクラス名-2 は、1 バイト文字を使用した COBOL ユーザー定義語の標準形成規則に従っていなければなりません。

Java パッケージに含まれているクラス名の指定の詳細について、またはクラス名への非 COBOL 命名規則の使用については、105ページの『REPOSITORY 段落』を参照してください。

1 つのクラス定義を一連のプログラムに含めたり、他の複数のクラス定義を単一のコンパイル・グループに含めたりすることはできません。各クラスは、別々のソース・ファイルとして指定する必要があります。すなわち、クラス定義をバッチ・コンパイルに組み込むことはできません。

継承

クラスのインスタンスで使用可能なメソッドは、それから直接的または間接的に派生した任意のサブクラスのインスタンスでも使用できます。サブクラスは、親 (または親元) クラスでは存在していない新規のメソッドを導入したり、親クラスから継承したメソッドをオーバーライドしたりすることができます。サブクラスが親クラスから継承した既存メソッドをオーバーライドするときには、サブクラスはそのメソッドの新規具体化を定義し、それが継承した具体化に置き換わります。

クラス名-1 のインスタンス・データは、クラス名-1 のワーキング・ストレージ・セクションで宣言されたデータと共にクラス名-2 で宣言したインスタンス・データです。しかし、インスタンス・データは常に、それを導入するクラスに専用されていることに注意してください。

継承のセマンティクスは、Java によって定義されます。すべてのクラスは、直接的または間接的に java.lang.Object クラスから取り出されなければなりません。

Java は単一の継承をサポートします。すなわち、クラスは、複数の親からは*直接的に* 継承することはできません。クラス定義の INHERITS 句には、1 つのクラス名しか指定できません。

FACTORY 段落

ファクトリー IDENTIFICATION DIVISION は、ファクトリー定義を導入します。ファクトリー定義とは、クラスのファクトリー・オブジェクトを定義する、クラス定義の部分です。ファクトリー・オブジェクトは、クラスのすべてのオブジェクト・インスタンスが共用する単一の共用オブジェクトです。

ファクトリー定義には、ファクトリー・データとファクトリー・メソッドが含まれます。

OBJECT 段落

オブジェクト IDENTIFICATION DIVISION は、オブジェクト定義を導入します。オブジェクト定義とは、クラスのインスタンス・オブジェクトを定義する、クラス定義の部分です。

オブジェクト定義には、オブジェクト・データおよびオブジェクト・メソッドが含まれます。

METHOD-ID 段落

METHOD-ID 段落は、メソッドの名前を指定し、選択された属性をそのメソッドに割り当てます。METHOD-ID 段落は必須であり、メソッド見出し部の最初の段落でなければなりません。

メソッド名-1

メソッドの名前が含まれる、英数字リテラルまたは国別リテラル。名前は、Java メソッド名の形成規則に従っていなければなりません。メソッド名は、変換せずに直接使用します。メソッド名は、大文字小文字を区別して処理されます。

メソッド・シグニチャー

メソッドの *シグニチャー* は、手続き部 USING 句で指定されているように、メソッドの名前と、メソッドの仮パラメーターの数および型で構成されます。

メソッド多重定義、オーバーライド、および隠蔽

COBOL メソッドは、Java 言語の規則に基づいて、*多重定義*、*オーバーライド*、*または隠蔽* になる可能性があります。

メソッド多重定義

クラスに定義されるメソッド名は固有である必要はありません。(「クラスに定義される」メソッドには、クラス定義によって導入されるメソッドと、親クラスから継承されたメソッドとがあります。)

クラスに対して定義されるメソッド名には、固有のシグニチャーが必要です。クラスに対して定義されている 2 つのメソッドの名前が同一で、シグニチャーが異なる場合、これらの 2 つのメソッドは *多重定義されている* と言います。

メソッド戻り値のタイプがある場合、それはメソッド・シグニチャーには組み込まれません。

クラスが 2 つのメソッドを定義するときは、同一のシグニチャーと異なる戻り値タイプを使ったり、または、同一のシグニチャーを使いながら、一方には戻り値を指定し、もう一方には戻り値を指定しないで、定義してはなりません。

多重定義メソッド定義に関する規則、および多重定義メソッドの起動の解決は、Java の対応規則に基づきます。

メソッドのオーバーライド (インスタンス・メソッド)

サブクラス内のインスタンス・メソッドは、2 つのメソッドのシグニチャーが同一である場合には、親クラスから継承される、同じ名前のインスタンス・メソッドを*オーバーライド* します。

メソッドが、親クラスで定義されたインスタンス・メソッドをオーバーライドするとき、メソッド戻り値 (手続き部 RETURNING データ名) の有無は、これらの 2 つのメソッドで一貫していなければなりません。さらに、メソッド戻り値を指定するときには、オーバーライドされる側の戻り値とオーバーライドする側のメソッドは、データ型が同一でなければなりません。

インスタンス・メソッドは、COBOL 親クラス内のファクトリー・メソッド、または Java 親クラス内の静的メソッドをオーバーライドすることはできません。

メソッドの隠蔽 (ファクトリー・メソッド)

クラスのスーパークラス内で、同一シグニチャーでなければアクセス可能であるメソッドが、同一シグニチャーであるためにアクセスが不可能となるとき、ファクトリー・メソッドはこのクラスのスーパークラス内の同一のシグニチャーを持つすべてのメソッドを*隠蔽* する、と言います。ファクトリー・メソッドは、インスタンス・メソッドを隠蔽することはできません。

オプションの段落

見出し部にある以下のオプションの段落は、省略可能です。

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
プログラムがコンパイルされた日付。

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**コメント記入項目**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、領域 B の中で 1 行または複数行を記述できます。

段落名 DATE-COMPILED とそれに関連付けられたコメント記入項目は、ソース・コード・リストにおいて次のように現在日付を伴って出力されます。

```
DATE-COMPILED. 04/27/95.
```

コメント記入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。コメント記入項目では、標識域 (第 7 桁) にハイフンを入れることはできません。

DBCS 文字ストリングをコメント記入項目としてプログラムの見出し部に含めることができます。DBCS ストリングを含むコメント記入項目は、複数行にすることができます。

DBCS ストリングには、シフトアウト制御文字を先頭に、シフトイン制御文字を末尾にそれぞれ付ける必要があります。以下に例を示します。

```
AUTHOR.       <.A.U.T.H.O.R.-.N.A.M.E>, XYZ CORPORATION
DATE-WRITTEN. <.D.A.T.E>
```

複数行にわたるコメント記入項目で DBCS 文字を使用する場合、シフトアウトおよびシフトイン文字の対が各行ごとに必要です。

DBCS ストリングに関しては、5ページの『文字ストリング』に説明があります。

環境部

構成セクション 94
SOURCE-COMPUTER 段落 95
OBJECT-COMPUTER 段落 96
SPECIAL-NAMES 段落 97
ALPHABET 文節 100
SYMBOLIC CHARACTERS 文節 103
CLASS 文節 103
CURRENCY SIGN 文節 104
DECIMAL-POINT IS COMMA 文節 105
REPOSITORY 段落 105

入出力セクション 108
FILE-CONTROL 段落 109
SELECT 文節 112
ASSIGN 文節 112
RESERVE 文節 115
ORGANIZATION 文節 116
PADDING CHARACTER 文節 118
RECORD DELIMITER 文節 119
ACCESS MODE 文節 119
RECORD KEY 文節 121
ALTERNATE RECORD KEY 文節 122
RELATIVE KEY 文節 123
PASSWORD 文節 124
FILE STATUS 文節 125
I-O-CONTROL 段落 126
RERUN 文節 127
SAME AREA 文節 128
SAME RECORD AREA 文節 129
SAME SORT AREA 文節 130
SAME SORT-MERGE AREA 文節 130
MULTIPLE FILE TAPE 文節 130
APPLY WRITE-ONLY 文節 131

© Copyright IBM Corp. 1991, 2001

構成セクション

構成セクションは、プログラムおよびクラスのオプショナルのセクションであり、そこにはプログラム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環境部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記入項目は、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。

クラス構成セクション

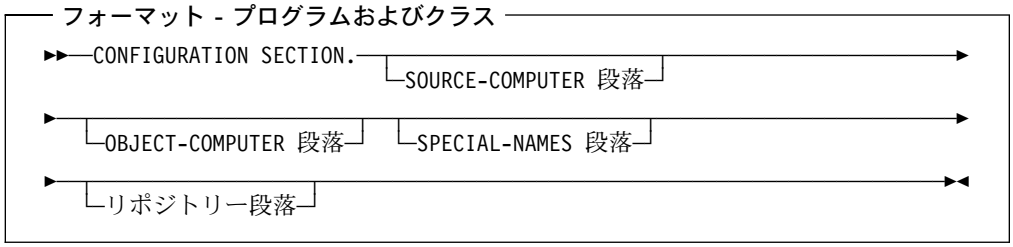
構成セクションをクラス定義の環境部で指定します。REPOSITORY 段落は、クラス定義の環境部で指定することができます。

クラス構成セクションの中の記入項目は、そのクラスによって導入されるすべてのメソッドを含む、クラス定義全体に適用されます。

メソッド構成セクション

入出力セクションは、メソッド構成セクションで指定することができます。記入項目は、構成セクションが指定されているメソッドにのみ適用されます。

フォーマット - プログラムおよびクラス



構成セクションでは、次のことが可能です。

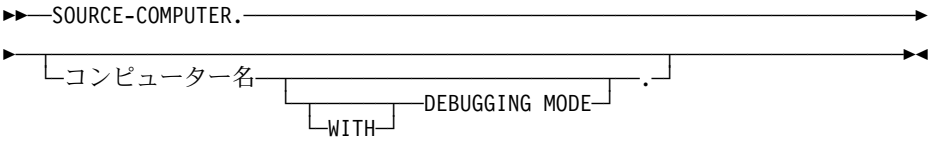
- IBM 定義の環境名をユーザー定義の簡略名に関係付ける。
- 照合シーケンスを指定する。
- 通貨符号値、および PICTURE 文節でその通貨符号値を表すために使用する通貨記号を指定する。
- PICTURE 文節と数字リテラルのコンマとピリオドの機能を交換する。
- 英字名を文字セットまたは照合シーケンスに関係付ける。
- シンボリック文字を指定する。
- 文字の集合にクラス名を関係付ける。
- オブジェクト指向クラス名を外部クラス名に関連付けて、クラス定義またはプログラムで使用できるクラス名を識別する。

© Copyright IBM Corp. 1991, 2001

SOURCE-COMPUTER 段落

SOURCE-COMPUTER 段落は、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-390
```

WITH DEBUGGING MODE
ソース・プログラム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アクティブにします。

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、たとえば、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、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 文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、環境部 (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。

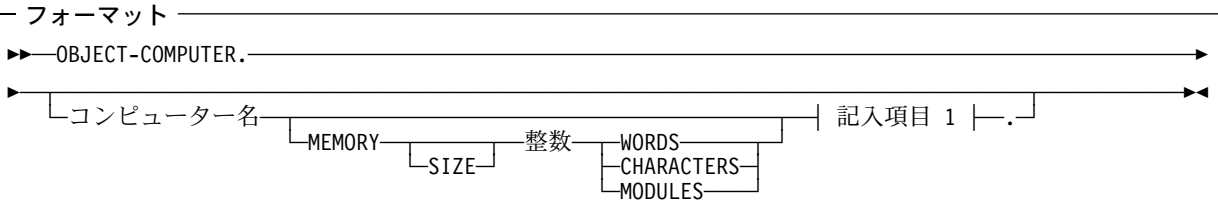
デバッグ行で領域 A および領域 B にスペースしかない場合、それはブランク行とみなされます。

WITH DEBUGGING MODE 文節を除き、SOURCE-COMPUTER 段落は構文チェックがなされます。ただし、これはプログラムの実行には何も影響しません。

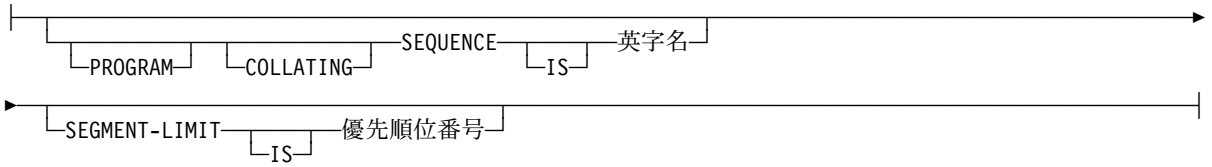
OBJECT-COMPUTER 段落

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

フォーマット



記入項目 1:



コンピューター名
システム名。以下に例を示します。

```
IBM-390
```

MEMORY SIZE
オブジェクト・プログラムを実行するのに必要な主記憶域の容量。MEMORY SIZE 文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
ワード数、文字数、またはモジュール数を表します。

PROGRAM COLLATING SEQUENCE IS
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。

英字名
照合シーケンス。

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るのでない限り、PROGRAM COLLATING SEQUENCE 句は、英数字のマージ・キーまたはソート・キーにも適用されます。

PROGRAM COLLATING SEQUENCE 文節は DBCS または国別データ項目には適用されません。

PROGRAM COLLATING SEQUENCE 文節を省略した場合には、EBCDIC 照合シーケンスが使用されます。(556ページの付録C、『EBCDIC および ASCII の照合シーケンス』を参照。)

SEGMENT-LIMIT IS

特定の永続セグメントは、**固定部分**セグメントの論理特性を保持しながら、独立セグメントによってオーバーレイすることができます。(固定部分セグメントは、固定永続セグメントと固定オーバーレイ可能セグメントから構成されます。)

優先順位番号

1 ～ 49 の範囲の整数。

SEGMENT-LIMIT を指定した場合は、次のようになります。

- 固定**永続**セグメントは、指定した優先順位番号より低い優先順位番号のセグメントです。
- 固定**オーバーレイ可能**セグメントは、指定した優先順位番号以上で 49 以下の優先順位番号のセグメントです。

たとえば、SEGMENT-LIMIT IS 25 と指定した場合は、次のようになります。

- 優先順位番号として 0 ～ 24 の番号のセクションが固定**永続**セグメントです。
- 優先順位番号として 25 ～ 49 の番号のセクションが固定**オーバーレイ可能**セグメントです。

SEGMENT-LIMIT を省いた場合、優先順位番号として 0 ～ 49 の番号のすべてのセクションが固定永続セグメントとなります。

THREAD オプションを使用してコンパイルされたプログラムの場合、セグメント化はサポートされていません。

PROGRAM COLLATING SEQUENCE 文節を除き、OBJECT-COMPUTER 段落は構文チェックの対象となりますが、これはプログラムの実行には何も影響しません。

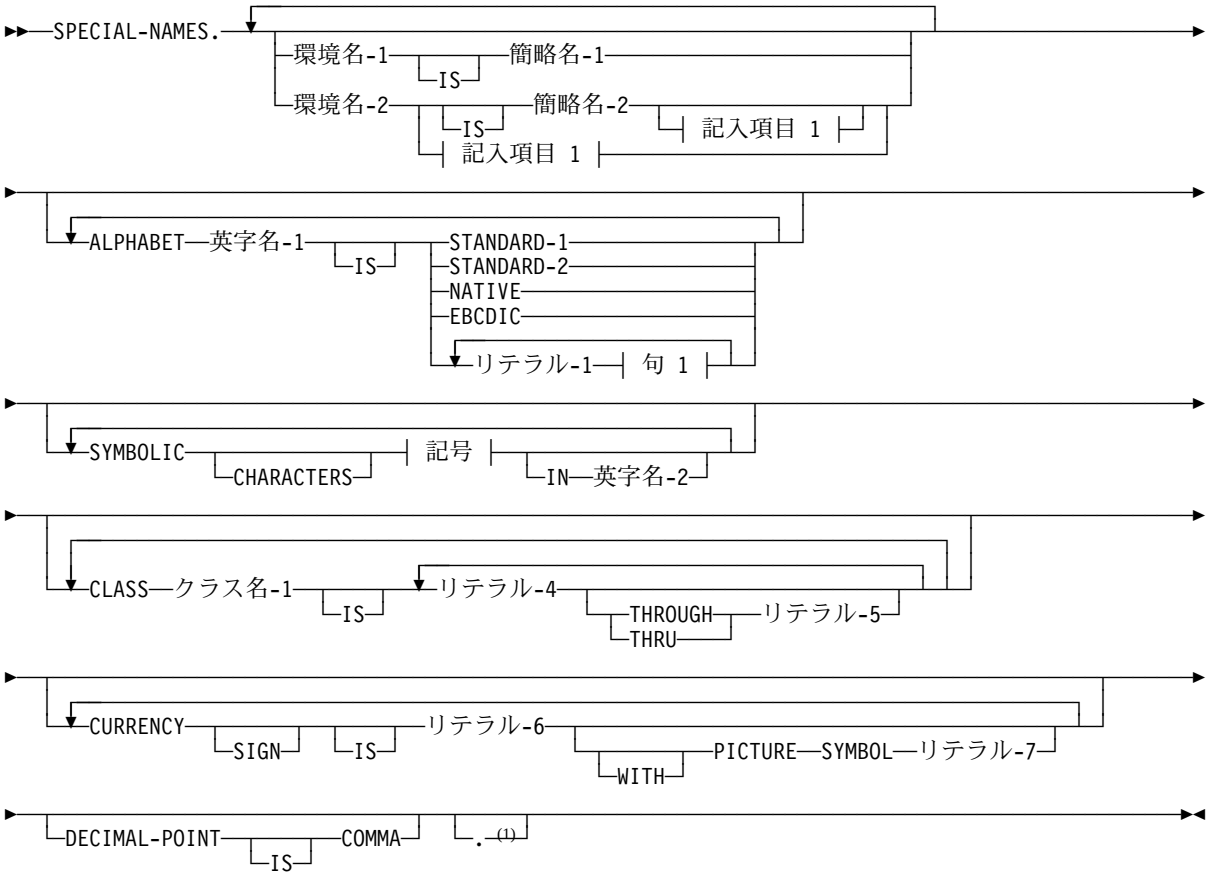
SPECIAL-NAMES 段落

SPECIAL-NAMES 段落の機能は次のとおりです。

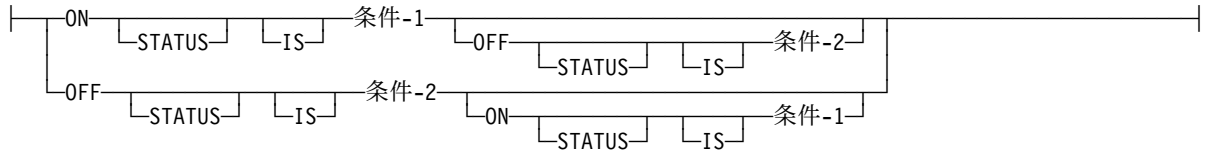
- IBM 指定の環境名をユーザー定義の簡略名に関係付ける。
- 英字名を文字セットまたは照合シーケンスに関係付ける。
- シンボリック文字を指定する。
- 文字の集合にクラス名を関係付ける。
- 1 つまたは複数の通貨符号値を指定して、それぞれの通貨符号値を表すピクチャー記号を PICTURE 文節に定義する。
- PICTURE 文節と数字リテラルのコンマと小数点の機能を指定する。

SPECIAL-NAMES 段落の文節は、任意の順序で指定できます。

フォーマット

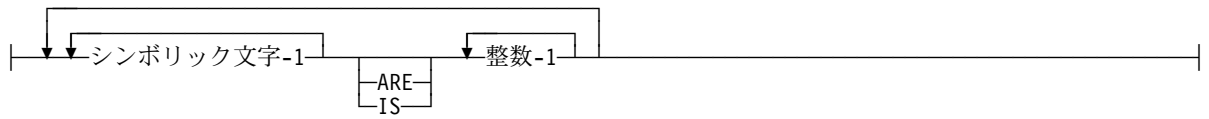


記入項目 1:



句 1:

記号:



注:

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

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 として有効な指定は、次のとおりです。

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 ～ C12	それぞれチャネル 1 ～ 12 にスキップ	WRITE ADVANCING
CSP	行送りの抑止	WRITE ADVANCING
S01 ～ S05	せん孔装置のポケット選択 1 ～ 5	WRITE ADVANCING
AFP-5A	Advanced Function Printing	WRITE ADVANCING

環境名-2
1 バイトのユーザー・プログラム状況標識 (UPSI) スイッチ。環境名-2 に指定できるのは UPSI-0 ～ 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。たとえば、手続き部の冒頭で 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。(251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およびオフ状況 (またはそのどちらか) に関連付けることができます。

手続き部では、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。

他のプログラムを中に含むプログラムの SPECIAL-NAMES 段落で指定された条件名は、その中に含まれるどのプログラムからも参照することができます。

ALPHABET 文節

ALPHABET 文節は、指定した文字コード・セットまたは照合シーケンスに英字名を関係付ける手段を提供します。

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

ALPHABET 英字名-1 IS

英字名-1 は、以下のどちらかを使用した場合、**照合シーケンス**を指定します。

- OBJECT-COMPUTER 段落の PROGRAM COLLATING SEQUENCE 文節
- 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、以下のどちらかを指定した場合、**文字コード・セット**を指定します。

- FD 記入項目 CODE-SET 文節
- SYMBOLIC CHARACTERS 文節

STANDARD-1

ASCII 文字セットを指定します。

STANDARD-2

国際標準 646 で定義された ISO 7 ビット・コードの国際参照バージョンである、情報処理交換用の 7 ビット・コード化文字セットを指定します。

NATIVE

固有文字コード・セットを指定します。英字名文節を省略すると、EBCDIC とみなされます。

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。

- これらのリテラルは英数字または数値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。
- 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。

556ページの付録C、『EBCDIC および ASCII の照合シーケンス』に、単一バイトの 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が示されています。

- 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。文字セット内の文字のうち指定されていない文字の相互の相対的な順序は、そのまま変わりません。
- 1 つの英字名文節の中では、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。たとえば、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 文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高位文字として、すべての文字および上記の例の 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低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 文節の中でその英字名を参照することは**できません** (160ページの『CODE-SET 文節』を参照)。

リテラル-1、**リテラル-2**、および**リテラル-3** には、シンボリック文字の形象定数を指定することはできません。

浮動小数点リテラルは、ユーザー指定の照合シーケンスの中では使用することはできません。

SYMBOLIC CHARACTERS 文節

SYMBOLIC CHARACTERS シンボリック文字-1

これは、1 つまたは複数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 文節の中では一度しか使用できません。シンボリック文字として DBCS ユーザー定義語を使用できます。

1 バイト文字セットを適用できるのは、SYMBOLIC CHARACTERS 句だけです。表されるそれぞれの文字は、英数字です。

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 文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 文節の中で 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。

  注: 順序位置には、1 から始まる番号が付けられます。

CLASS 文節

CLASS クラス名-1 IS

ある名前を、その文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。クラス名は、クラス条件の中でのみ参照できます。この文節でそのリテラルの値によって指定された文字は、このクラス名を構成する文字の排他的集合を定義します。

CLASS 文節のクラス名を DBCS ユーザー定義語にすることができます。

リテラル-4、リテラル-5

カテゴリー数値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。数字リテラルの場合は符号なしの 1 以上の整数で、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、単一バイト 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。

数値リテラルは、浮動小数点を指定することはできません。

英数字リテラルの場合、そのリテラルは実際の単一バイト EBCDIC 文字、または単一バイト ASCII 文字です。

リテラル-4 と **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 文節

CURRENCY SIGN 文節は、PICTURE 文節の文字ストリングに*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側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。たとえば、'$'、'EUR'、'FRF'、'F'、'HK$'、'HKD'、または X'9F' (一部のホスト・ベースのコード・ページでユーロ通貨符号 € を表す 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、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

CURRENCY SIGN 文節は、通貨符号値、および PICTURE 文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 文節を含めることができます。各 CURRENCY SIGN 文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 文節記号とは異なり、通貨記号は大文字小文字が区別されます。たとえば、'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 を形象定数にすることはできません。また、DBCS リテラル、またはヌル終了リテラルにすることはできません。

PICTURE SYMBOL 段落が指定されて**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。
- 以下のものにすることはできません。
  - 数字 0 ～ 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 + - , .* / ; ( ) " = '

- 英小文字 f、h、i、j、k、l、m、o、q、t、u、w、y のいずれか。

PICTURE SYMBOL 段落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、通貨符号値を表すための通貨記号を指定します。
- 1 つまたは複数の文字で構成できます。
- 以下のものを含めることはできません。
  - 数字 0 ～ 9
  - 特殊文字 + - . ,

PICTURE SYMBOL リテラル-7

PICTURE 文節でリテラル-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 は単一の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 を以下のものにすることはできません。

- 形象定数
- 数字 0 ～ 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 文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 文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は、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

CURRENCY SIGN 文節を使用する場合、NUMVAL-C 組み込み関数が使用できないことがあります。詳細については、484ページの『NUMVAL-C』を参照してください。

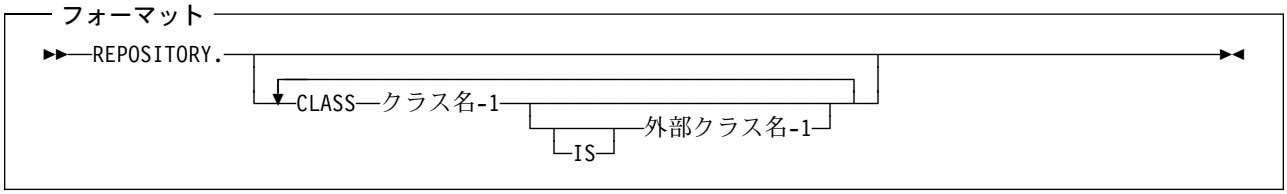
DECIMAL-POINT IS COMMA 文節

DECIMAL-POINT IS COMMA

PICTURE 文字ストリングと数字リテラルの中で、ピリオドとコンマの機能を交換します。

REPOSITORY 段落

REPOSITORY 段落をプログラムまたはクラス定義で使用すると、そのプログラムまたはクラス定義で参照しようとする、すべてのオブジェクト指向クラスを識別することができます。さらにオプションとして、REPOSITORY 段落はクラス名と外部クラス名との関連を定義します。



クラス名-1
クラスを識別するユーザー定義語。

外部クラス名-1
COBOL プログラムが、Java 形成規則を使用して定義されたクラス名によってクラスを定義したりアクセスしたりできるようにするための名前。

外部クラス名-1 は、完全修飾 Java クラス名の形成規則に従って英数字リテラルとして指定する必要があります。クラスが Java パッケージに含まれている場合には、*外部クラス名-1* は、そのパッケージの完全修飾名を指定し、その後に "."、続いて、Java クラスの単純名を付ける必要があります。

Java クラス名の形成規則については、Gosling 他著の「Java 言語仕様、第 2 版」を参照してください。

一般規則

1. すべての参照されるクラス名には、その参照が含まれている COBOL プログラムまたはクラス定義の REPOSITORY 段落に記入項目を持つ必要があります。指定されたクラス名は、指定された REPOSITORY 段落で一度だけ指定できます。
2. プログラム定義において、REPOSITORY 段落は最外部プログラムでのみ指定することができます。
3. COBOL クラス定義の REPOSITORY 段落にはオプションとして、クラス自体の名前の記入項目を含めることができます。ただし、この記入項目は必須ではありません。そのような記入項目を使用すると、非 COBOL 文字を使用する外部クラス名、または、COBOL クラスが Java パッケージに含まれているときは完全パッケージ修飾クラス名を指定する外部クラス名、あるいはその両方の外部クラス名を指定することができます。
4. クラス REPOSITORY 段落の中の記入項目は、そのクラスによって導入されるすべてのメソッドを含む、クラス定義全体に適用されます。プログラム REPOSITORY 段落の中の記入項目は、それに含まれるプログラムも含めて、プログラム全体に適用されます。

クラスの識別と参照

外部クラス名を使用すると、そのクラスを定義するクラス定義の外側から、指定されたクラスを識別して参照することができます。外部クラス名は、以下のように*外部クラス名-1* または*クラス名-1* (クラスの REPOSITORY 段落で指定されているもの) のいずれかの内容を使用して判別されます。

1. *外部クラス名-1* は、変換せずに直接使用されます。外部クラス名は、大文字小文字を区別して処理されます。
2. *外部クラス名-1* が指定されない場合には、*クラス名-1* が使用されます。クラスを識別し、Java 形成規則に準拠する外部名を作成するため、*クラス名-1* は、以下のように処理されます。
   - 名前は大文字に変換されます。
   - ハイフンは 0 に変換されます。

- 最初の文字が数字でない場合には、次の規則に従って変換されます。
  - 数字 1 ～ 9 は A ～ I に変換されます。
  - 0 は J に変換されます。

クラスは、Java または COBOL でインプリメントすることができます。

Java パッケージに含まれるクラスを参照するときには、*外部クラス名-1* を指定して、完全修飾 Java クラス名を付ける必要があります。

以下に例を示します。

```
Repository.
Class JavaException is "java.lang.Exception"
```

上記のリポジトリー記入項目は完全修飾外部クラス名 "java.lang.Exception" を参照するためのローカル・クラス名 JavaException を定義します。

Java パッケージに含まれる COBOL クラスを定義するときには、そのクラス自体のリポジトリー段落に記入項目を指定し、外部クラス名として完全 Java パッケージ修飾名を付けます。

入出力セクション

環境部の入出力セクションには、次の 2 つの段落があります。

- FILE-CONTROL 段落
- I-O-CONTROL 段落

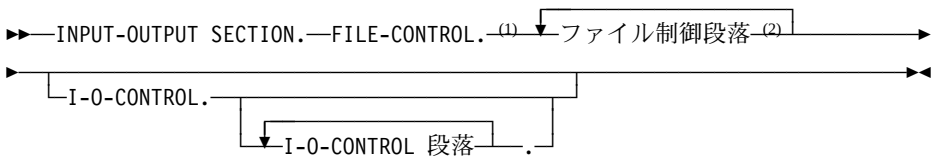
入出力セクションの正確な内容は、使用されているファイル編成とアクセス方式により決まります。116ページの『ORGANIZATION 文節』、および 119ページの『ACCESS MODE 文節』を参照してください。

プログラム入出力セクション
同じ規則がプログラムとメソッドの入出力セクションに適用されます。

クラス入出力セクション
入出力セクションは、クラス定義に対して有効**ではありません**。

メソッド入出力セクション
同じ規則がプログラムとメソッドの入出力セクションに適用されます。

プログラムおよびメソッド



注:
1. 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。
2. 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。

FILE-CONTROL
キーワード FILE-CONTROL は FILE-CONTROL 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、その後に区切り文字のピリオドを付けなければなりません。

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データ・セットに関連付けます。

SELECT 文節を使用し、領域 B から開始する必要があります。後に区切り文字のピリオドを付けなければなりません。109ページの『FILE-CONTROL 段落』を参照してください。

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
外部データ・セットと COBOL プログラムとの間で効率的にデータを伝送するために必要とされる情報を指定します。一連の記入項目は、区切り文字のピリオドで終了しなければなりません。126ページの『I-O-CONTROL 段落』を参照してください。

© Copyright IBM Corp. 1991, 2001

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データ・セットに関連付け、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、以下のとおりです。

- 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

表6 に、Enterprise COBOL 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 6. ファイルのタイプ

ファイル編成	アクセス方式
順次	QSAM、VSAM
相対	VSAM の
索引付き	VSAM の
行順次	固有 [1]

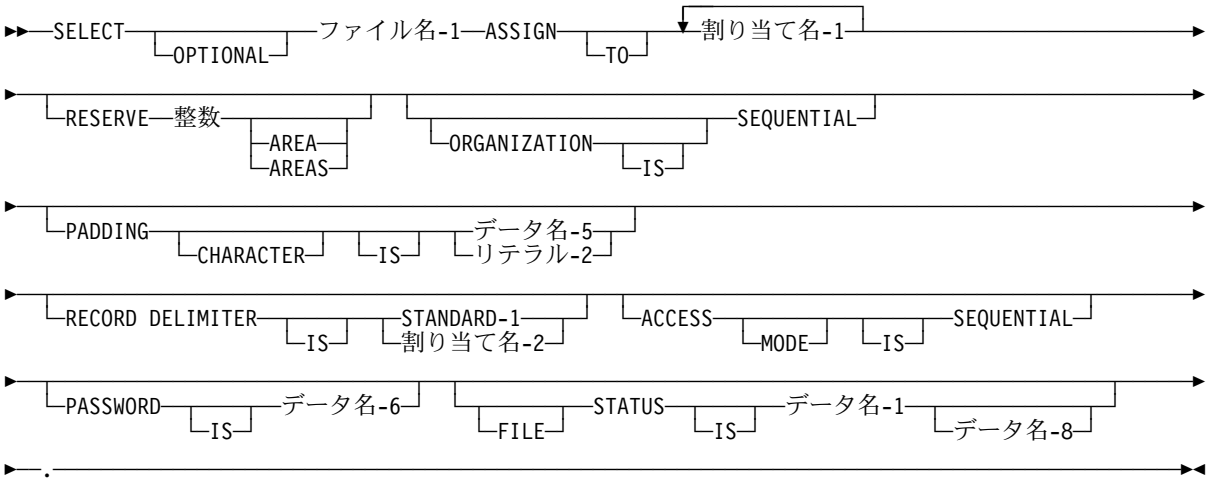
注:

[1] 行順次サポートは、HFS ファイルに限定されます。

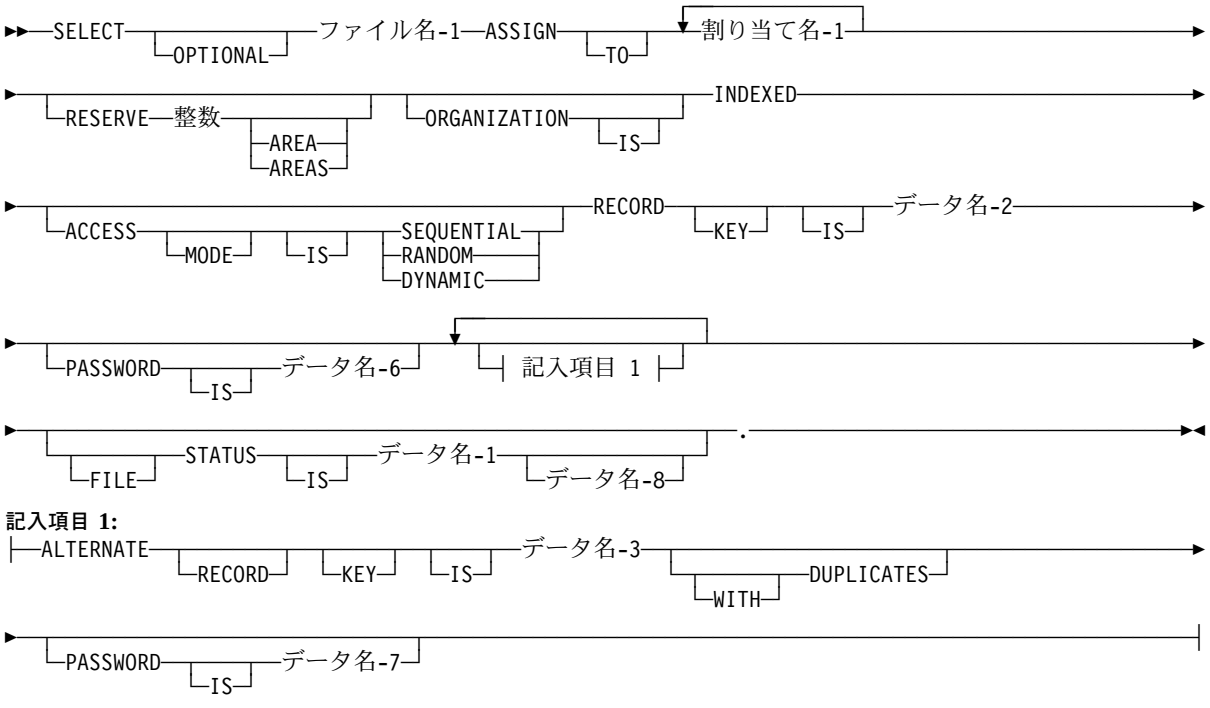
FILE-CONTROL 段落は、"FILE-CONTROL" という語で開始し、後に区切り文字のピリオドが続きます。ここには、データ部の FD 記入項目または SD 記入項目で記述されるそれぞれのファイルに対応して、1 つの (ただ 1 つの) 記入項目を記述する必要があります。

各記入項目内では、SELECT 文節が最初でなければなりません。その他の文節の順序は任意です。ただし、索引付きファイルの場合、PASSWORD 文節を指定する場合は、関連する RECORD KEY または ALTERNATE RECORD KEY データ名の直後に指定しなければなりません。

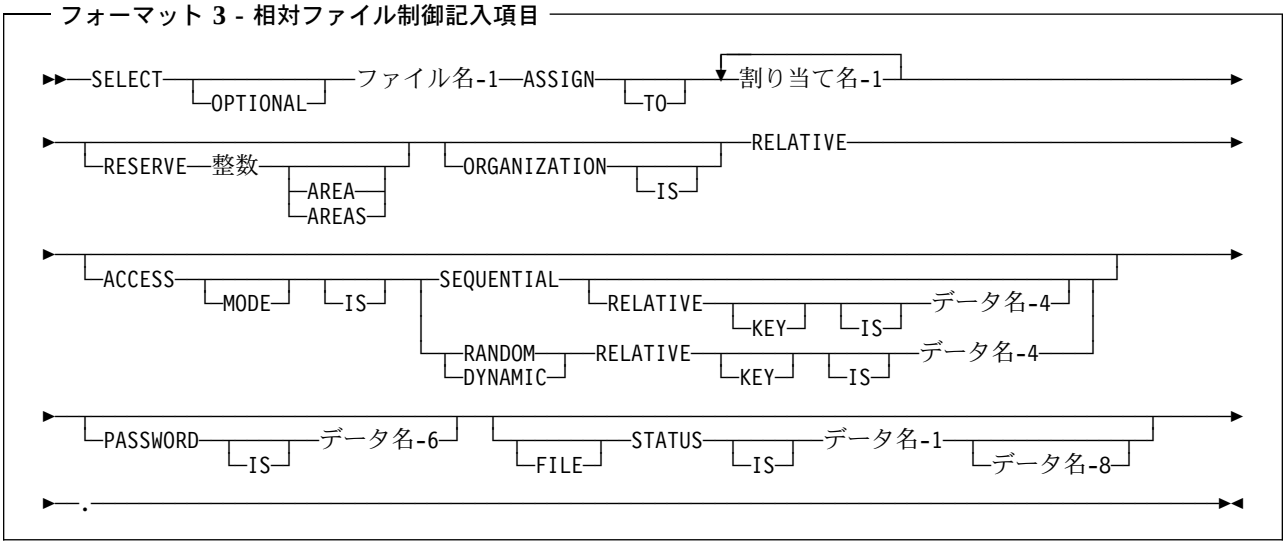
フォーマット 1 - 順次ファイル制御記入項目



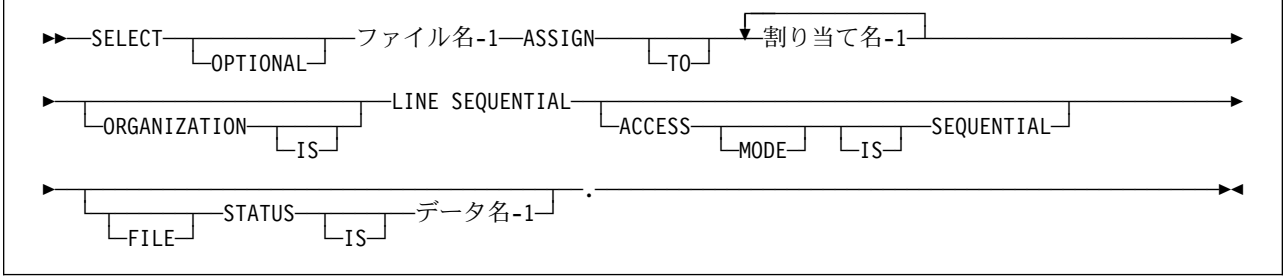
フォーマット 2 - 索引付きファイル制御記入項目



フォーマット 3 - 相対ファイル制御記入項目



フォーマット 4 - 行順次ファイル制御記入項目



SELECT 文節

SELECT 文節は、COBOL プログラムの中で外部データ・セットと関連付けるファイルを選択します。

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。必ず存在しているとは限らない入力ファイルについては、オブジェクト・プログラムが実行されるたびに SELECT OPTIONAL を指定する必要があります。詳細については、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

ファイル名-1
データ部の記入項目 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 文節の後には ASSIGN 文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 文節

ASSIGN 文節は、そのプログラムでのファイルの名前を、実際のデータ・ファイルのための外部名と関連付けます。

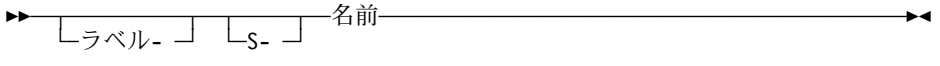
割り当て名-1
外部データ・ファイルを指定します。名前または英数字リテラルの内容として指定することができます。

注: 割り当て名-1 はデータ項目の名前ではありません。また、割り当て名-1 は、データ項目に含めることはできません。割り当て名-1 は、単に文字ストリングです。

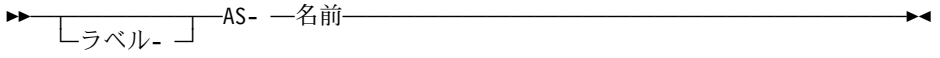
最初の割り当て名の後の割り当て名は、構文チェックされますが、プログラムの実行には何も影響しません。

割り当て名-1 のフォーマットは次のとおりです。

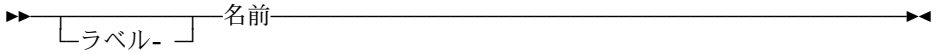
フォーマット - QSAM ファイル



フォーマット - VSAM 順次ファイル



フォーマット - 行順次、VSAM 索引ファイルまたは VSAM 相対ファイル



label-

ファイルの割り当て先の装置と装置クラス (プログラマー用) を記述します。ハイフンで終了する必要があります。ハイフンで終了しない場合、指定値はチェックされません。プログラムの実行には影響しません。このラベルを指定する場合、これはハイフンで終わらなければなりません。

S- QSAM では、S- (編成) フィールドは省略することができます。

AS-

VSAM 順次ファイルでは、AS- (編成) フィールドは必ず指定しなければなりません。

VSAM の索引付きファイルおよび相対ファイルでは、編成フィールドは必ず省いてください。

名前

このファイルの外部名を指定する必須フィールド。

このファイルの DD ステートメントで指定された名前か、ファイル割り振り情報が入った環境変数の名前の、いずれかでなければなりません。環境変数の指定の詳細については、『環境変数の割り当て名』を参照してください。

名前 は、次の形成規則に従っている必要があります。

- 割り当て名-1 がユーザー定義語の場合
  - 名前は 1 ～ 8 文字です。
  - 名前には A ～ Z、a ～ z、0 ～ 9 を含めることができます。
  - 先頭の文字は英字でなければなりません。
- 割り当て名-1 がリテラルの場合
  - 名前は 1 ～ 8 文字です。
  - 名前には A ～ Z、a ～ z、0 ～ 9、@、#、$ を含めることができます。
  - 先頭の文字は英字でなければなりません。

ユーザー定義語とリテラルのどちらの場合も、コンパイラーは*名前* を大文字に変換したものをファイルの DD 名にします。

ソート・ファイルやマージ・ファイルでは、*名前* はコメントとして扱われます。

SELECT 文節の中でファイル名-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の中のすべてのファイル制御記入項目は、ASSIGN 文節の中の割り当て名-1 の指定と矛盾がないようにする必要があります。QSAM ファイルと VSAM 索引付きファイルおよび VSAM 相対ファイルでは、最初の割り当て名-1 に指定された名前が同じでなければなりません。VSAM 順次ファイルでは、AS-名前として指定する必要があります。

環境変数の割り当て名

割り当て名-1 の *名前* 部分は当初、DD 名として扱われます。この DD 名を使用して割り振られたファイルがない場合は、*名前* は環境変数として扱われます。

注: COBOL コンパイラーは自動的に外部ファイル名を大文字に変換するため、環境変数名を定義する場合は大文字のみを使用する必要があります。

この環境変数が存在していて、有効な PATH あるいは DSN オプション (以下で説明) が入っている場合は、ファイルがこのオプションで提供されている情報を使用して動的に割り振られます。

環境変数に有効な PATH あるいは DSN オプションが入っていないか、動的割り振りが失敗した場合は、ファイルのオープンはファイル状況 98 で終わります。

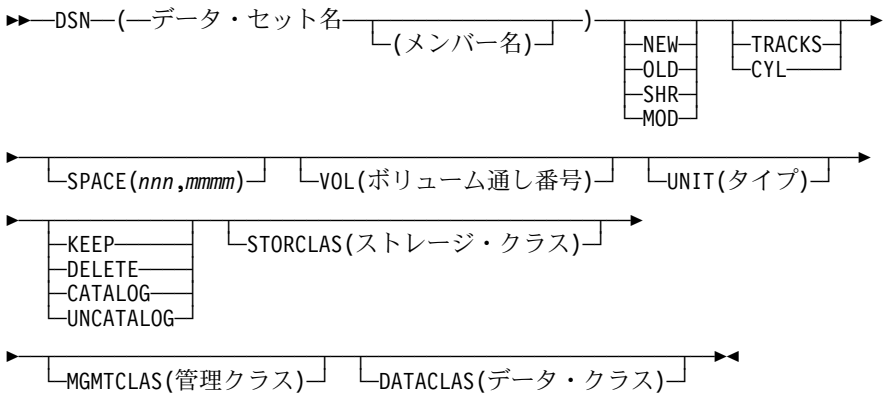
環境変数の内容は、各 OPEN ステートメントでチェックされます。ファイルが直前の OPEN ステートメントで動的に割り振られていて、環境変数の内容が直前の OPEN 以降に変更されている場合は、現在、環境変数に設定されているオプションを使用して動的に割り振る前に、直前の割り振りは動的に割り振り解除されます。

実行単位が終了すると、COBOL ランタイム・システムはすべての自動的に生成された動的割り振りを自動的に割り振り解除します。

QSAM ファイル用の環境変数の内容: QSAM ファイルの場合、環境変数には PATH オプションまたは DSN オプションが次のフォーマットで入っている必要があります。

DSN に続くオプション (たとえば、NEW、TRACKS など) は、コンマあるいは 1 つまたは複数のブランクで分離する必要があります。

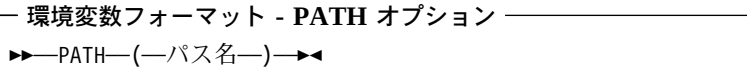
環境変数フォーマット (QSAM ファイル) - DSN オプション



データ・セット名 は完全修飾名でなければなりません。このデータ・セットは一時データ・セットであってはなりません (つまり、& 記号で始まってはなりません)。*データ・セット名* または *メンバー名* の後には、データ・セット属性を任意の順序で指定できます。

データ・セット属性値の指定については、*z/OS MVS JCL 解説書* (GC88-6578) にある DD ステートメントの説明を参照してください。

環境変数フォーマット - PATH オプション

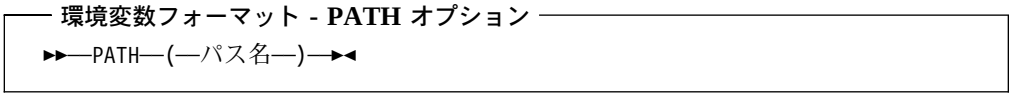


パス名 は、絶対パス名でなければなりません (つまり、スラッシュで始まっている必要があります)。*パス名* の指定の詳細については、*z/OS MVS JCL 解説書* (GC88-6578) にある PATH パラメーターの説明を参照してください。

環境変数の内容に入っている最初と最後のブランクは無視されます。括弧内にブランクを入れることはできません。

行順序ファイル用の環境変数の内容: 行順序ファイルの場合、環境変数には PATH オプションが次のフォーマットで入っている必要があります。

環境変数フォーマット - PATH オプション

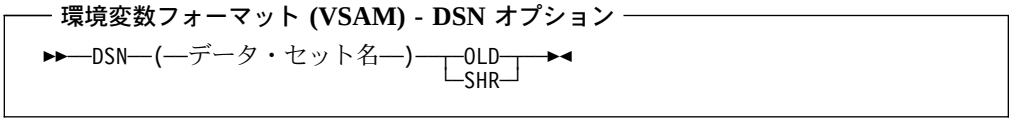


パス名 は、絶対パス名でなければなりません (つまり、スラッシュで始まっている必要があります)。*パス名* の指定の詳細については、*z/OS MVS JCL 解説書* (GC88-6578) にある PATH パラメーターの説明を参照してください。

環境変数の内容に入っている最初と最後のブランクは無視されます。括弧内にブランクを入れることはできません。

索引付き、相対、または順序 VSAM ファイル用の環境変数の内容: 索引付き、相対、または順序 VSAM ファイルの場合、環境変数には DSN オプションが次のフォーマットで入っている必要があります。

環境変数フォーマット (VSAM) - DSN オプション



データ・セット名 には、基本クラスターのデータ・セット名を指定します。*データ・セット名* は完全修飾名で、カタログに登録された事前定義済みの VSAM データ・セットを参照するものでなければなりません。

索引付きファイルに代替索引がある場合は、代替索引パスのそれぞれについて、(上記のように) DSN オプションが入った追加の環境変数を定義する必要があります。これらの環境変数の名前は、代替索引の DD 名に使用したのと同じ命名規則に従う必要があります。つまり、次のようになります。

- 各代替索引パス用の環境変数名は、基本クラスター環境変数名に整数を連結した形で構成されます。最初の代替索引に関連するパスについては、この整数を 1 とし、後続の代替索引ごとに 1 ずつ加えていきます。(基本クラスターの環境変数名が CUST であれば、代替索引の環境変数名は CUST1、CUST2 などになります。)
- 基本クラスターの環境変数名の長さがすでに 8 文字である場合は、代替索引の環境変数名は環境変数名の基本クラスター部分の右側を切り捨てて、連結した長さが 8 文字になるようにします。(たとえば、基本クラスターの環境変数名が DATAFILE であれば、代替クラスターの環境変数名は DATAFIL1、DATAFIL2 などになります。)

環境変数の内容に入っている最初と最後のブランクは無視されます。括弧内にブランクを入れることはできません。

DSN に続くオプション (SHR など) は、コンマあるいは 1 つまたは複数のブランクで分離する必要があります。

RESERVE 文節

RESERVE 文節を使用すると、実行時にファイルに割り振られる入出力バッファーの数を指定することができます。

RESERVE 文節は、行順次ファイルについてはサポートされていません。

RESERVE 文節を省略した場合、実行時のバッファーの数は DD ステートメントから取得されます。何も指定されていない場合は、システムのデフォルト値が採用されます。

SELECT 文節の中でファイル名-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の中のすべてのファイル制御記入項目の値は、RESERVE 文節の中で指定された整数値と同じでなければなりません。

ORGANIZATION 文節

ORGANIZATION 文節は、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、120 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 文節を省略すると、コンパイラーは ORGANIZATION IS SEQUENTIAL とみなします。

SELECT 文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記入項目は、同じ編成で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。ファイルを作成したら、それ以降、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の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、FILE-CONTROL 段落の RECORD KEY 文節によって 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、FILE-CONTROL 段落の ALTERNATE RECORD KEY 文節によって 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。たとえば、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。

書き込みにあたっては、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においては、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捨てられ、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。読み取れない最初の文字がレコード区切り文字の場合は、その文字は捨てられます。レコード区切り文字でない場合、読み取れなかった最初の文字は、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。

行順次ファイルに書き込まれるレコードは、USAGE...DISPLAY や DISPLAY-1 のデータ項目から構成されている必要があります。外部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。

行順序ファイルに入れることができるのは、印刷可能文字と以下の制御文字だけです。

アラーム
バックスペース
用紙送り
改行
復帰
水平タブ
垂直タブ
DBCS シフトアウト
DBCS シフトイン

改行文字はレコード区切り文字として処理されますが、他の制御文字は COBOL がファイルにあるレコードのデータの一部として扱います。

行順次ファイルにおいては、次のものはサポートされません。

- APPLY WRITE ONLY 文節
- CODE-SET 文節
- DATA RECORDS 文節
- LABEL RECORDS 文節
- LINAGE 文節
- OPEN I-O オプション
- PADDING CHARACTER 文節
- RECORD CONTAINS 0 文節
- RECORD CONTAINS 文節 (フォーマット 2。たとえば、RECORD CONTAINS 100 to 200 CHARACTERS)
- RECORD DELIMITER 文節
- RECORDING MODE 文節
- RERUN 文節
- RESERVE 文節
- 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 文節
- WRITE...AFTER ADVANCING *簡略名*
- WRITE...AT END-OF-PAGE
- WRITE...BEFORE ADVANCING

PADDING CHARACTER 文節

PADDING CHARACTER 文節は、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

1 文字の英数字データ項目としてデータ部に定義しなければならず、ファイル・セクションには定義できません。データ名-5 は修飾することができます。

リテラル-2
これは、英数字リテラルでなければなりません。

EXTERNAL ファイルでデータ名-5 が指定されている場合、これは外部データ項目を参照しなければなりません。

PADDING CHARACTER 文節は構文チェックは行いますが、コンパイル時または実行時の検査は実行されないので、この文節がプログラムの実行に影響を与えることはありません。

RECORD DELIMITER 文節

RECORD DELIMITER 文節は、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 文節は構文チェックは行いますが、コンパイル時または実行時の検査は実行されないので、この文節がプログラムの実行に影響を与えることはありません。

ACCESS MODE 文節

ACCESS MODE 文節は、ファイル内のレコードが処理のために使用可能にされる際の方法を定義します。ACCESS MODE 文節を指定しない場合には、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 文節を RELATIVE KEY 文節の前に置く必要はありません。

ACCESS MODE IS SEQUENTIAL
次の 4 つ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
ファイルの中のレコードは、ファイルの中の既存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、109ページの表6 を参照してください。

注: 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・モードのいずれにもアクセスできます。

アクセス・モード

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、レコードは順次にもランダムにも (あるいはその両方で) 処理できます。

EXTERNAL 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記入項目は、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、相対ファイル記入項目では、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記入項目の中の RELATIVE KEY 句は、各場合に応じて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

データ編成の各タイプごとに有効なアクセス・モードは、次のとおりです。

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シーケンスは、最初にレコードが作成されたシーケンスです。

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。取り出したいレコードは、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、現在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。取り出したいレコードは、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。この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 文節

RECORD KEY 文節 (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。

これは、そのファイルと関係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。これは、以下の項目のタイプのいずれかにすることができます。

英数字
数字
数字編集
英数字編集
英字

外部浮動小数点
内部浮動小数点
DBCS
国別

キー・データ項目のカテゴリーには関係なく、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、キーは英数字項目として扱われます。

データ名-2 としてウィンドウ化日付フィールドは使用できません。

データ名-2 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-2 は修飾できます。

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 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 ファイルの場合、同じ EXTERNAL 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、同じ記述を指定しなければなりません。これは強制ではありませんが、データ名-2 は、すべてのレコードで同一相対位置と同一相対長さの同一のデータ記述を持つ必要があります。

ALTERNATE RECORD KEY 文節

ALTERNATE RECORD KEY 文節 (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。

これは、そのファイルと関係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。これは、以下の項目のタイプのいずれかにすることができます。

英数字
数字
数字編集
英数字編集
英字
外部浮動小数点
内部浮動小数点
DBCS
国別

キー・データ項目のカテゴリーには関係なく、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、キーは英数字項目として扱われます。

データ名-3 をウィンドウ化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-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。

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-3 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-3 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。

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 RECORD KEY の左端、あるいはその他の ALTERNATE RECORD KEY がある場合には、その左端の文字位置と同じにすることはできません。

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 ファイルの場合、同じ EXTERNAL 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、同じ記述を指定しなければなりません。これは強制ではありませんが、データ名-3 は、すべてのレコード内で同一相対位置と同一相対長さの同一のデータ記述を持つ必要があります。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。

RELATIVE KEY 文節

RELATIVE KEY 文節 (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 が含まれない無符号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 は修飾できます。

データ名-4 をウィンドウ化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 ステートメントが出

されるとシステムは、RELATIVE KEY データ項目の内容を使用して順次処理を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 ステートメントが出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 文節を指定する必要があります。

EXTERNAL 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の中の 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 文節は、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 文節

PASSWORD 文節は、ファイルへのアクセスを制御します。

データ名-6
データ名-7
パスワード・データ項目。この各項目は、データ部のワーキング・ストレージ・セクションの中で英数字項目として定義されていなければなりません。最初の 8 文字はパスワードとして使用されます。フィールドが満たない場合は 8 文字までブランクが埋め込まれます。各パスワード・データ項目は、外部的に定義されたものと同じでなければなりません。

PASSWORD 文節を指定する場合、オブジェクト時にこのファイルに対する正しいパスワードが PASSWORD データ項目に入っていないと、このファイルは正常にオープンできません。

フォーマット 1 の考慮事項

PASSWORD 文節は、QSAM 順次ファイルでは無効です。

フォーマット 2 とフォーマット 3 の考慮事項

PASSWORD 文節を指定する場合、この文節は、それが関連付けられている RECORD KEY データ名または ALTERNATE RECORD KEY データ名の直後になければなりません。

索引付きファイルにおいて、そのファイルが VSAM に対してあらかじめ完全に定義されている場合は、RECORD KEY のために PASSWORD データ項目に正しいパスワードが入っている限り、ファイル作成時にそのファイルを正常にオープンできます。

その他の種類のファイル処理に関しては、COBOL オブジェクト時にサブルーチンを通じて行われるファイル作成の動的呼び出し処理を含め、このファイルに対するすべての PASSWORD データ項目に有効なパスワードが入っているのでない限り、ファイルは正常にオープンできません。このことは、そのオブジェクト・プログラムの中でそのデータへのすべてのパスが使用されているかどうかに無関係です。

EXTERNAL ファイルの場合、データ名-6 とデータ名-7 は、外部データ項目を参照していなければなりません。関連する各ファイル制御項目の中の PASSWORD 文節は、同じ外部データ項目を参照する必要があります。

FILE STATUS 文節

FILE STATUS 文節は、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 文節を指定すると、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、267ページの『共通の処理機能』を参照。)

データ名-1
状況キー・データ項目は、ワーキング・ストレージ・セクション、ローカル・ストレージ・セクション、またはリンケージ・セクションで、次のいずれかとして定義することができます。

- 2 文字の英数字項目
- 明示的もしくは暗黙のうちに USAGE IS DISPLAY を伴う 2 文字の数字データ項目。これは英数字項目として扱われます。

  注: 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 は修飾できます。

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 文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しています。その定義はファイル・システムとプラットフォームに特定のものなので、データ名-8 の特定の値に依存しているアプリケーションは、プラットフォーム間での移植は不可能です。

データ名-8 は、データ部のワーキング・ストレージまたはリンケージ・セクションに 6 バイトのグループ項目として定義しなければなりません。

データ名-8 は、ファイルが VSAM ファイル (つまり ESDS、KSDS、RRDS) の場合に限り指定します。

VSAM ファイルの場合、6 バイトの VSAM 戻りコードは、次のものから構成されます。

- データ名-8 の最初の 2 バイトには、2 進表記の VSAM **戻りコード**が含まれています。戻りコードのこの部分の値は、VSAM により 0、8、または 12 として定義されています。
- データ名-8 の次の 2 バイトには、2 進表記の VSAM **機能コード**が含まれています。戻りコードのこの部分の値は、VSAM により 0、1、2、3、4、または 5 として定義されています。
- データ名-8 の最後の 2 バイトには、2 進表記の VSAM **フィードバック・コード**が含まれています。そのコード値は 0 ～ 255 です。

VSAM が戻りコードとして 0 以外の値を戻した場合、データ名-8 が設定されます。

VSAM を呼び出すことなく FILE STATUS が戻された場合、データ名-8 は 0 になります。

データ名-1 が 0 に設定さている場合、データ名-8 の内容は未定義です。VSAM 状況戻りコード情報は、現在定義されている COBOL プログラムの FILE STATUS コードに変換しなくても使用することができます。ユーザーは、VSAM で定義されているレベルと同じレベルで、例外条件を識別し取り扱うことができます。

機能コードと**フィードバック・コード**が設定されるのは、**戻りコード**が 0 以外の値に設定される場合だけです。戻りコードが 0 に設定されている場合にこれらのフィールドを参照しても、それらの内容は信頼できるものではありません。

戻りコード、**機能コード**、および**フィードバック・コード**の各フィールドの値は、VSAM によって定義されます。VSAM 定義に対して COBOL では何も追加、削除、または修正しません。詳細については、 *DFSMS Macro Instructions for Data Sets* を参照してください。

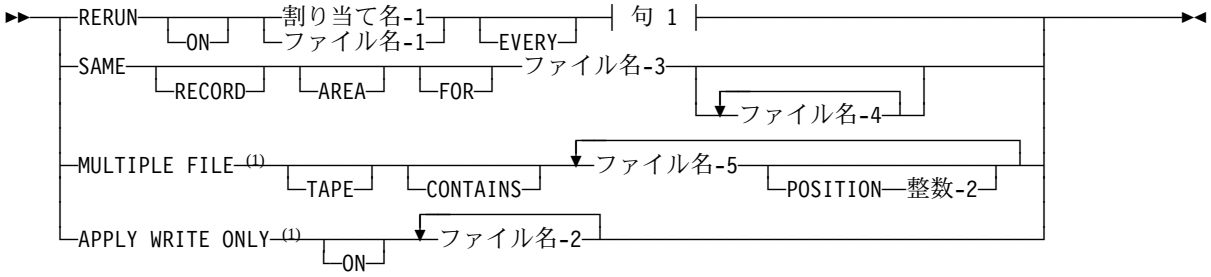
I-O-CONTROL 段落

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べきかを指定し、また、いろいろ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。

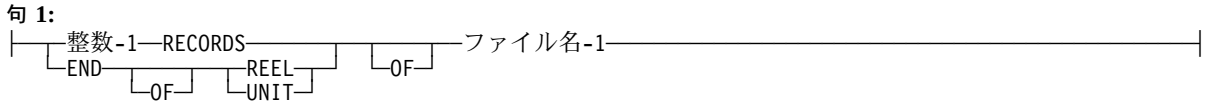
キーワード I-O-CONTROL は、この段落の冒頭に一度だけ使用することができます。I-O-CONTROL というワードは、領域 A で開始し、区切り文字のピリオドを後に付けなければなりません。

I-O-CONTROL 段落の中に文節を記述する場合、その順序は任意です。I-O-CONTROL 段落は、区切り文字のピリオドによって終わります。

順次 I-O-CONTROL 記入項目



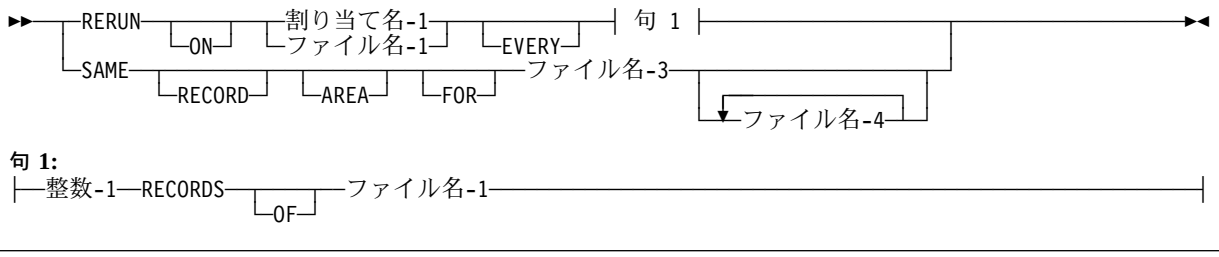
句 1:



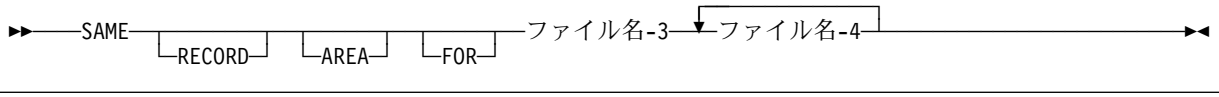
注:

[1] MULTIPLE FILE 文節および APPLY WRITE-ONLY 文節は、VSAM ファイルに対してはサポートされません。

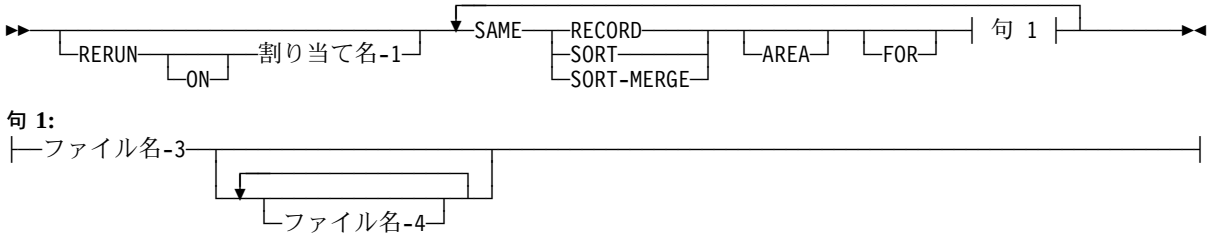
相対および索引付き I-O-CONTROL 記入項目



行順次 I-O-CONTROL 記入項目



ソート / マージ I-O-CONTROL 記入項目



RERUN 文節

RERUN 文節は、チェックポイント・レコードを取ることを指定します。それぞれの句の制約事項に従っていれば、2 つ以上の RERUN 文節を指定することができます。

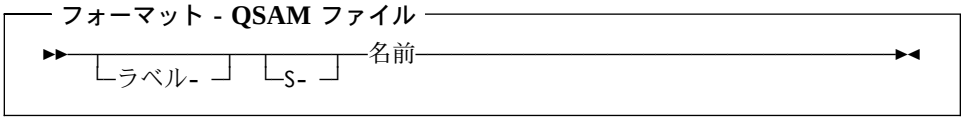
チェックポイント・データ・セット定義と、COBOL 85 標準に完全準拠するために必要なチェックポイント方式については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

次の場合、RERUN 文節を使用しないでください。

- EXTERNAL 属性のファイルに対して
- RECURSIVE 属性のプログラム内で
- THREAD オプションを指定してコンパイルしたプログラム内で
- メソッドで

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・ファイル用の外部データ・セット。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 文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。QSAM ファイルでは、この割り当て名のフォーマットは次のとおりです。



すなわち、これは QSAM ファイルでなければなりません。磁気テープ装置または直接アクセス装置上になければなりません。573ページの付録F、『ASCII に関する考慮事項』も参照してください。

VSAM および QSAM に関する考慮事項:

RERUN 文節で指名されたファイルは、そのファイルが GLOBAL として定義されているものであっても、同じプログラム内で I-O-CONTROL 段落として定義されたファイルでなければなりません。

SORT/MERGE に関する考慮事項:

I-O-CONTROL 段落の中で RERUN 文節を指定する場合、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。この文節を省略すると、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 ステートメントすべてに共通の影響を及ぼします。

EVERY 整数-1 RECORDS

この指定により、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。

注: この文節はサポートされていません。この文節をプログラムでコーディングしても、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。

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 文節

SAME AREA 文節を使用することによって、2 つ以上のファイル (ソート・ファイルやマージ・ファイルではない) が、処理中に同じ主記憶域を使用するということを指定できます。

SAME AREA 文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 FILE-CONTROL 段落中に指定しなければなりません。ファイル名-3 とファイル名-4 で外部ファイル結合子を参照することはできません。

- QSAM ファイルでは、SAME 文節は説明文として扱われます。
- VSAM ファイルの場合、 SAME 文節は SAME RECORD AREA と同等であるかのように扱われます。

1 つのプログラムの中に複数の SAME AREA 文節を指定できます。しかし、以下のことが言えます。

- 複数の SAME AREA 文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 文節の中に SAME AREA 文節のファイル名が 1 つ以上ある場合は、その SAME AREA 文節のすべてのファイル名がその SAME RECORD AREA 文節の中にもなければなりません。ただし、その SAME AREA 文節にないファイル名でも、その SAME RECORD AREA 文節に指定できます。
- SAME AREA 文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 文節

SAME RECORD AREA 文節は、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。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 文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 文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 文節を指定できます。しかし、以下のことが言えます。

- 複数の SAME RECORD AREA 文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 文節の中に SAME AREA 文節のファイル名が 1 つ以上ある場合は、その SAME AREA 文節のすべてのファイル名がその SAME RECORD AREA 文節の中にもなければなりません。ただし、その SAME AREA 文節にないファイル名でも、その SAME RECORD AREA 文節に指定できます。
- SAME AREA 文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 文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 文節を含めることはできません。
- RECORD CONTAINS 0 CHARACTERS 文節を指定する場合、SAME RECORD AREA 文節を指定することはできません。

SAME RECORD AREA 文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 文節

SAME SORT AREA 文節は構文チェックされますが、これは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 FILE-CONTROL 段落中に指定しなければなりません。ファイル名-3 とファイル名-4 で外部ファイル結合子を参照することはできません。

SAME SORT AREA 文節を指定する場合、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 文節を指定できます。しかし、1 つのソート・ファイルを 2 つ以上の文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 文節と 1 つ以上の SAME SORT AREA 文節の両方で指定されている場合は、SAME AREA 文節のすべてのファイルが SAME SORT AREA 文節の中になければなりません。
- SAME SORT AREA 文節で指定されたファイルは、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 文節で指定されたソート・ファイル以外のファイルは、ユーザーがそれらを SAME AREA 文節または SAME RECORD AREA 文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。
- この文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、この文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 文節

SAME SORT-MERGE AREA 文節は、SAME SORT AREA 文節と同じです。

MULTIPLE FILE TAPE 文節

MULTIPLE FILE TAPE 文節 (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。

この文節は 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。この機能は、DD ステートメントの LABEL パラメーターを通じて、システムによって実行されます。

APPLY WRITE-ONLY 文節

APPLY WRITE-ONLY 文節は、標準順次編成で、可変長レコードを含み、ブロック化されているファイルに対するバッファーおよび装置スペースの割り振りを最適化します。この句を指定した場合は、バッファーの使用可能スペースが次のレコードのサイズよりも小さい場合にのみ、そのバッファーが切り捨てられます。指定しない場合は、バッファーの残存スペースがそのファイルの最大レコード・サイズよりも小さい場合に、そのバッファーが切り捨てられます。

APPLY WRITE-ONLY は、QSAM ファイルに対してのみ有効です。

ファイル名-2
各ファイルは標準順次編成でなければなりません。

対応する外部ファイル記述項目間で APPLY WRITE-ONLY 文節に矛盾があってはなりません。APPLY WRITE-ONLY と同じ結果を得るための代替方法については、*Enterprise COBOL プログラミング・ガイド* に記載されているAWO コンパイラー・オプションの説明を参照してください。

データ部

データ部の概要 134
ファイル・セクション 135
ワーキング・ストレージ・セクション 135
ローカル・ストレージ・セクション 137
リンケージ・セクション 137
データ単位 138
データの関係 139

データ部 - ファイル記述項目 147
ファイル・セクション 149
EXTERNAL 文節 150
GLOBAL 文節 151
BLOCK CONTAINS 文節 151
RECORD 文節 153
LABEL RECORDS 文節 155
VALUE OF 文節 156
DATA RECORDS 文節 156
LINAGE 文節 156
RECORDING MODE 文節 158
CODE-SET 文節 160

データ部 - データ記述記入項目 161
フォーマット 1 161
フォーマット 2 162
フォーマット 3 162
レベル番号 162
BLANK WHEN ZERO 文節 163
DATE FORMAT 文節 164
EXTERNAL 文節 169
GLOBAL 文節 170
JUSTIFIED 文節 171
OCCURS 文節 171
PICTURE 文節 177
REDEFINES 文節 192
RENAMES 文節 196
SIGN 文節 198
SYNCHRONIZED 文節 199
USAGE 文節 206
VALUE 文節 214

© Copyright IBM Corp. 1991, 2001

データ部の概要

ここでは、プログラム、オブジェクト定義、ファクトリー定義、およびメソッドのデータ部の構造を概説します。データ部のセクションには、COBOL プログラム、オブジェクト定義、ファクトリー定義、またはメソッド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。データ部はオプショナルです。

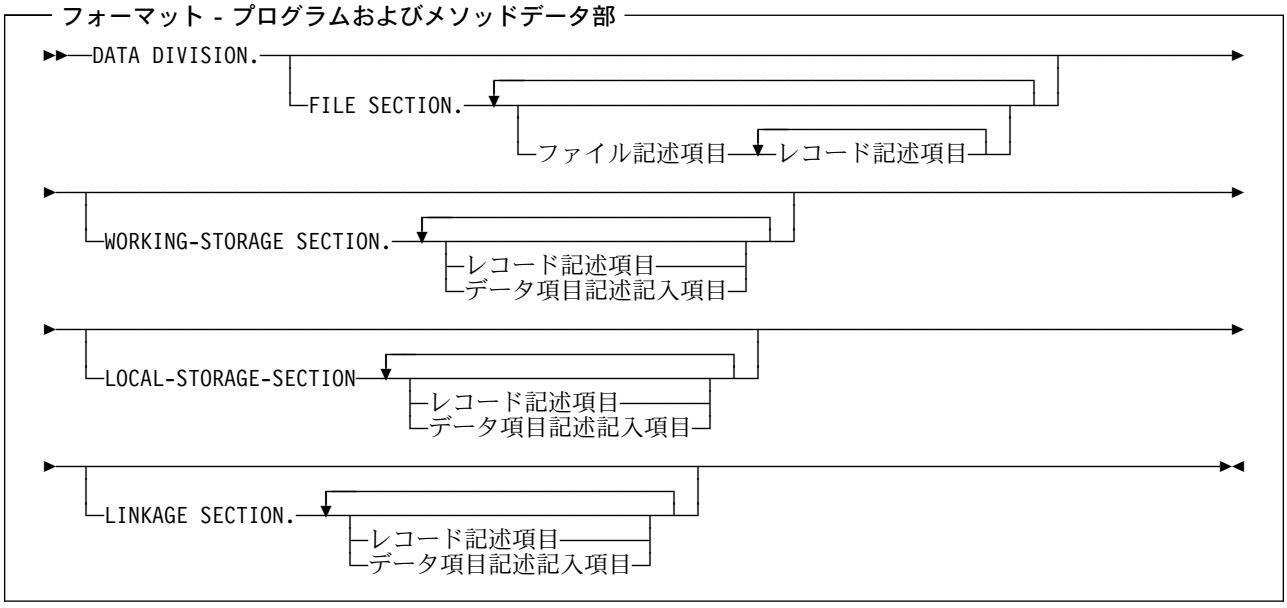
プログラム・データ部
COBOL ソース・プログラムのデータ部は、プログラムにより処理されるすべてのデータを、特定の構造により記述します。

オブジェクト・データ部
オブジェクト・データ部には、インスタンス・オブジェクト・データ (インスタンス・データ) のデータ記述記入項目が含まれています。インスタンス・データは、クラス定義の OBJECT 段落のワーキング・ストレージ・セクションで定義されます。

ファクトリー・データ部
ファクトリー・データ部には、ファクトリー・オブジェクト・データ (ファクトリー・データ) のデータ記述記入項目が含まれています。ファクトリー・データは、クラス定義の FACTORY 段落のワーキング・ストレージ・セクションで定義されます。

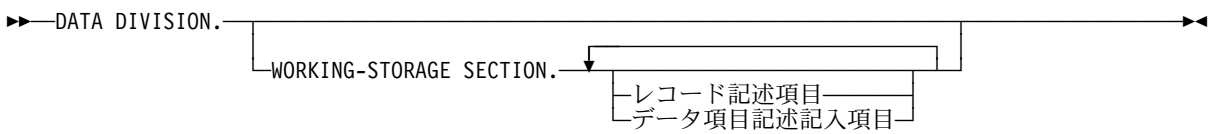
メソッド・データ部
メソッド・データ部には、メソッド内でアクセス可能なデータのデータ記述記入項目が含まれています。メソッド・データ部には、ローカル・ストレージ・セクション、または、ワーキング・ストレージ・セクション、あるいはその両方を含めることができます。*メソッド・データ* という用語は、両方に適用されます。ローカル・ストレージ内のメソッド・データは、メソッドの呼び出しごとに動的に割り振られて初期化されます。ワーキング・ストレージ内のメソッド・データは静的であり、メソッドの呼び出しがあっても持続します。

フォーマット - プログラムおよびメソッドデータ部



© Copyright IBM Corp. 1991, 2001

フォーマット - オブジェクトおよびファクトリー・データ部



ファイル・セクション

ファイル・セクションは、データ・ファイルの構造を定義します。ファイル・セクションは FILE SECTION というヘッダーで開始し、その後に区切り文字のピリオドを付けなければなりません。

ファイル記述項目
ファイル・セクション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文節については、147ページの『データ部 - 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 (1 つまたは複数) を記述する 1 組のデータ記述記入項目 (161ページの『データ部 - データ記述記入項目』を参照)。

ファイル・セクションのレコードは、英字、英数字、DBCS、国別、または数値のクラスのグループ項目として、または基本項目として記述しなければなりません。

2 つ以上のレコード記述項目を指定することができます。その場合、それぞれの記入項目が同一のレコード・ストレージ域の再定義になります。

ファイル・セクションで記述されるデータ域は、そのデータ域を含むファイルがオープンされていない限り、処理のために使用することはできません。

注: メソッド・ファイル・セクションで定義できるのは、EXTERNAL ファイルだけです。単一の実行単位レベルのファイル結合子は、指定された EXTERNAL ファイルの宣言を含むすべてのプログラムおよびメソッドによって共用されます。

ワーキング・ストレージ・セクション

ワーキング・ストレージ・セクションは、データ・ファイルの一部ではなく、プログラムまたはメソッドによって開発され処理されるデータ・レコードを記述します。また、ソース・プログラムまたはメソッド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。

ワーキング・ストレージ・セクションは WORKING-STORAGE SECTION というセクション・ヘッダーで開始し、その後に区切り文字のピリオドを付けなければなりません。

プログラム・ワーキング・ストレージ
プログラム (およびメソッド) の場合ワーキング・ストレージ・セクションには、実行単位全体を通して複数のプログラムおよびメソッド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。ファイル・セクション内のレコード記述に使用されるすべての文節は、VALUE 文節と EXTERNAL 文節 (これらの文節はファイル・セクションの中のレコード記述項目では指定できません) と同じく、ワーキング・ストレージ・セクション内のレコード記述に使用することができます。

メソッド・ワーキング・ストレージ
メソッドのワーキング・ストレージの単一コピーは、メソッドの最初の呼び出し時に静的に割り振られ、実行単位の持続期間中、最後に使用された状態で持続します。メソッドが呼び出される場合は、どのオブジェクト・インスタンスに対してメソッドが呼び出されたかに関係なく常に同一のコピーが使用されます。

VALUE 文節がメソッド・ワーキング・ストレージ・データ項目で指定された場合、そのデータ項目は最初の呼び出しの際に VALUE 文節の値に初期設定されます。

EXTERNAL 属性がメソッドワーキング・ストレージ・セクションのデータ記述記入項目で指定された場合は、そのデータ項目のストレージの単一コピーが実行単位で一度割り振られます。そのストレージは、外部データ項目の定義を含む実行単位内のすべてのプログラムおよびメソッドによって共用されます。

オブジェクト・ワーキング・ストレージ
OBJECT 段落のワーキング・ストレージ・セクションで説明されているデータは、オブジェクト・インスタンス・データです。オブジェクトがインスタンス化されるときに、それぞれのオブジェクト・インスタンスに対して別々のインスタンス・データが静的に割り振られます。インスタンス・データは、Java ランタイム・システムによって解放されるまで、最後に使用された状態で持続します。

オブジェクト・インスタンス・データは、データ宣言に指定した VALUE 文節によって、またはインスタンス・メソッドに指定した論理によって、初期化することができます。

ファクトリー・ワーキング・ストレージ
FACTORY 段落のワーキング・ストレージ・セクションで説明されているデータは、ファクトリー・データです。クラスのファクトリー・オブジェクトが作成されるときに、ファクトリー・データの単一コピーが静的に割り振られます。ファクトリー・データは、実行単位の持続期間中、最後に使用された状態で持続します。

ファクトリー・データは、データ宣言に指定した VALUE 文節によって、またはファクトリー・メソッドに指定した論理によって、初期化することができます。

ワーキング・ストレージ・セクションには、レコード記述項目と、独立したデータ項目のデータ記述記入項目 (**データ項目記述記入項目**) が含まれています。

レコード記述項目
ワーキング・ストレージ・セクション内にあって、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。説明については、161ページの『データ部 - データ記述記入項目』を参照してください。

データ項目記述記入項目
ワーキング・ストレージ・セクション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記入項目の中で定義されます。説明については、161ページの『データ部 - データ記述記入項目』を参照してください。

ローカル・ストレージ・セクション

ローカル・ストレージ・セクションは、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。 呼び出しのたびに、ローカル・ストレージ・セクションで定義されたデータ項目が再度割り振られ、その VALUE 文節で割り当てら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、ローカル・ストレージ・セクションで定義されたデータ項目が、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、ネストされたプログラムが呼び出されるたびに VALUE 文節で割り当てられた値に初期設定されます。

メソッドの場合、ローカル・ストレージ・セクションで定義されたデータ項目が、メソッドの呼び出しごとに割り振られて初期化されます。

ローカル・ストレージ・セクションで定義されたデータ項目は、EXTERNAL 文節を指定することはできません。

ローカル・ストレージ・セクションは、LOCAL-STORAGE SECTION というヘッダーで開始し、その後に分離記号のピリオドを付ける必要があります。

ローカル・ストレージ・セクションは、再帰的プログラム、非再帰的プログラム、およびメソッドで指定することができます。

注: メソッドのローカル・ストレージの内容は、プログラムのローカル・ストレージの内容と同じです。ただし、GLOBAL 属性に効力がない点が違います (メソッドはネストできないため)。

メソッドのローカル・ストレージ・セクションで定義されたデータの分離コピーは、メソッドが呼び出されるたびに作成されます。データに割り振られたストレージは、メソッドが戻ると解放されます。

リンケージ・セクション

リンケージ・セクションは、別のプログラムまたはメソッドで利用できるデータを記述します。

レコード記述項目

説明については、135ページの『ワーキング・ストレージ・セクション』を参照してください。

データ項目記述記入項目

説明については、135ページの『ワーキング・ストレージ・セクション』を参照してください。

リンケージ・セクションのレコード記述項目とデータ項目記述記入項目は、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、プログラムまたはメソッドの中にストレージは確保されません。

リンケージ・セクションでは任意のデータ記述文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 文節を指定することはできません。
- リンケージ・セクションで EXTERNAL 文節を指定することはできません。

リンケージ・セクションで GLOBAL 文節を指定することができます。(GLOBAL 属性はメソッドに対しては効力がないことに注意してください。)

データ単位

データは、以下の概念的単位にまとめられます。

- ファイル・データ
- プログラム・データ
- インスタンス・データ
- ファクトリー・データ
- メソッド・データ

ファイル・データ

ファイル・データはファイルに含まれています。(149ページの『ファイル・セクション』を参照。）**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、データ部で記述します。

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取り扱われるデータの単位です。物理レコードの大きさは、それが収容される特定の入出力装置によって決まります。この大きさは、ファイルに含まれる論理情報の大きさや内容とは必ずしも直接的な関係はありません。

論理レコードは、そのサブディビジョンに論理関係を持つデータの単位です。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 (たとえば、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を記述します。たとえば、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などを記述します。

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、独立したデータ記述記入項目 (**データ項目記述記入項目**) の中で定義することができます。

メソッド・データ

メソッド・データは、メソッドのデータ部で定義されて、そのメソッドのプロシージャー・コードによって処理されます。メソッド・データは、プログラム・データと同じ方法で、論理レコードおよび独立したデータ記述記入項目に編成されます。

ファクトリー・データ

ファクトリー・データは、クラス定義の FACTORY 段落のデータ部で定義され、そのクラスのファクトリー・メソッド内のプロシージャー・コードによって処理されます。ファクトリー・データは、プログラム・データと同じ方法で、論理レコードおよび独立したデータ記述記入項目に編成されます。

実行単位には指定されたクラスに対してファクトリー・オブジェクトが 1 つあります。したがって、そのクラスの実行単位にあるファクトリー・データのインスタンスは 1 つのみです。

インスタンス・データ

インスタンス・データは、クラス定義の OBJECT 段落のデータ部で定義され、そのクラスのインスタンス・データ内のプロシージャー・コードによって処理されます。インスタンス・データは、プログラム・データと同じ方法で、論理レコードおよび独立したデータ記述記入項目に編成されます。

指定されたクラスのそれぞれのオブジェクト・インスタンスごとに、インスタンス・データの別個のコピーが 1 つあります。指定されたクラスには複数のオブジェクト・インスタンスがあってもかまいません。それぞれが独自に、インスタンス・データの別個のコピーを持ちます。

データの関係

プログラムで使用するすべてのデータの関係は、データ部の中で、レベル標識とレベル番号のシステムを使用して定義します。

レベル標識は、その記述記入項目を使用して、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高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、その記述記入項目を使用して、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、162ページの『レベル番号』を参照。)

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るようにすることができます。

たとえば、百貨店の顧客ファイルの場合に、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、取引番号、売上の部門番号、売上の単位数、売上高、直前の収支、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小部分 (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。グループ自体を組み合わせて、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。特殊なレベル番号も使用されます。それらは特別な目的に使用されるデータ項目を識別します。

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記入項目が必要で、その記入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 ～ 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 の記入項目は、グループ項目でも基本項目でも可能です。これは領域 A から始めなければなりません。

02 ～ 49
これらのレベル番号は、レコード内のグループ項目や基本項目を指定します。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。

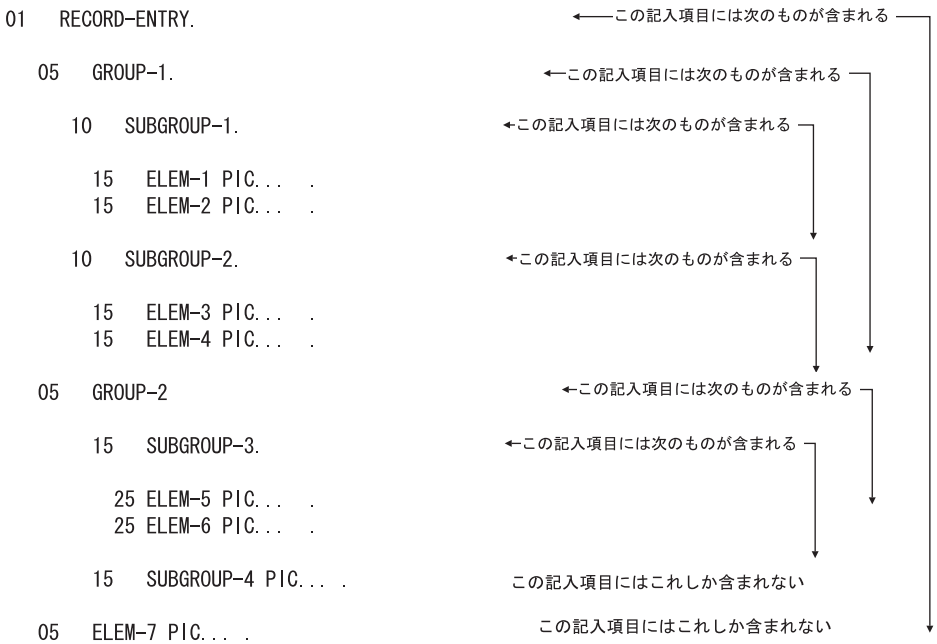
1 つのグループ項目には、そのグループ項目の後にあるすべてのグループ項目と基本項目のうち、そのグループのレベル番号以下のレベル番号が現れるまでのものが含まれます。

あるグループ項目に直接従属するすべての基本項目またはグループ項目には、それらが従属しているそのグループ項目のレベル番号より高い同一のレベル番号を割り当てる必要があります。

141ページの図3 に、これまで説明してきた概念を図示します。レベル-01 の記入項目に直接従属するグループは、すべて同一のレベル番号であることに注意してください。また、あるサブグループの基本項目と別のサブグループの基本項目は、必ずしも同じレベル番号ではなく、基本項目は階層内のどのレベルでも指定することができることに注意してください。

次のように作成された
COBOL レコード記述記入項目は、

以下のように細分化されます。



このレコード記述記入項目のストレージでの配置は次のとおりです。

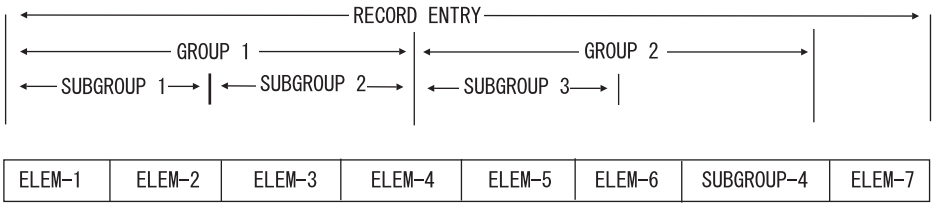


図 3. レコード記述のレベル

Enterprise COBOL は、同じレベルの他のものと等しくない、標準以外のレベル番号を受け入れます。たとえば、次の 2 つのレコード記述項目は同じものとして扱われます。

```
01   EMPLOYEE-RECORD.
     05  EMPLOYEE-NAME.
         10  FIRST-NAME PICTURE  X(10).
         10  LAST-NAME  PICTURE  X(10).
     05  EMPLOYEE-ADDRESS.
         10  STREET     PICTURE  X(10).
         10  CITY       PICTURE  X(10).
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).
```

特殊なレベル番号

特別なレベル番号は、レコードを構造化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 文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、196ページの『RENAMES 文節』を参照。)

77 他の項目の小部分ではなく、それ以上分割されないデータ項目記述記入項目 (ワーキング・ストレージ・セクションまたは リンケージ・セクションの独立した項目) 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記入項目を識別します。(詳細については、214ページの『VALUE 文節』を参照。)

注: プログラムまたはメソッドの中で参照されるワーキング・ストレージ・セクションおよびリンケージ・セクション内にあるレベル-77 とレベル-01 の記入項目には、固有のデータ名を付けなければなりません。なぜなら、これらはどちらも修飾することができないからです。プログラムまたはメソッド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、必ずしも一意に定義する必要はありません。

字下げ

連続するデータ記述記入項目は、先行する記入項目と同じ桁から始めたり、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。

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データおよびリテラルのほとんどは、クラスとカテゴリーに分けられます。

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 IS POINTER、 USAGE IS FUNCTION-POINTER、 USAGE IS PROCEDURE-POINTER、または USAGE IS OBJECT REFERENCE で定義されているデータ項目

関数は、基本データ項目を参照し、その関数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。次の表7 を参照してください。

表8 に示すように、すべての他の基本データ項目には、クラスとカテゴリーがあります。

リテラルには、144ページの表9 に示すように、クラスとカテゴリーがあります。

すべてのグループ項目には、それに従属する基本項目が別のクラスのカテゴリーに属している場合でも、英数字クラスとカテゴリーに属します。

表 7. 関数のクラスとカテゴリー

関数の型	クラスとカテゴリー
英数字	英数字
国別	国別
整数	数字
数字	数字

表 8. データのクラスとカテゴリー

項目のレベル	クラス	カテゴリー
基本項目	英字	英字
	数字	数字
		内部浮動小数点
		外部浮動小数点
	英数字	数字編集
		英数字編集
		英数字
		DBCS
	国別	国別
グループ	英数字	英字
		数字
		内部浮動小数点
		外部浮動小数点
		数字編集
		英数字編集
		英数字
		DBCS
		国別

表 9. 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

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。347ページの『基本移動』を参照)。

数字

これらの受け取り項目では、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、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。

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 文節が指定されている場合、上記の規則は、171ページの『JUSTIFIED 文節』に説明されているように修正されます。

国別

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'0020') 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 文節が指定されている場合、上記の規則は、 171ページの『JUSTIFIED 文節』に説明されているように修正されます。

標準データ・フォーマット

COBOL はデータ記述をできる限りマシンから独立したものにします。したがって、データの性質は、マシン依存のフォーマットではなく、標準データ・フォーマットに関係するように記述されます。

標準データ・フォーマットでは、システムがどのような記数法を使用している場合でも、数字を記述する場合は 10 進数体系を使用し、また文字データを記述する場合はコンピューターの文字セットのすべての文字を使用します。

文字ストリングと項目のサイズ

PICTURE 文節で記述される項目の場合、基本項目のサイズは、その PICTURE 文字ストリングの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、ストレージ内では、その項目が実際に占有する文字位置またはバイト数 (PICTURE 文字ストリングと USAGE 文節の組み合わせから決定される) によって項目のサイズが決められます。

内部浮動小数点項目の場合、その USAGE 文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に移動する場合、コンパイラーは、短い方の項目の PICTURE 文字ストリングで表されている文字数に合わせてデータを切り捨てます。

たとえば、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。詳しい説明については、206ページの『USAGE 文節』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 の詳細については、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

符号付きデータ

Enterprise COBOL で使用される代数符号には、算術演算符号と編集符号の 2 つのカテゴリーがあります。

算術演算符号

算術演算符号は、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 文節、SIGN 文節 (存在する場合)、および関与している操作環境によって決まります。 (内部表現の詳細については、*Enterprise COBOL プログラミング・ガイド* を参照。) 0 は、算術演算符号には関係なく固有の値とみなされます。符号なしフィールドは、常に正または 0 とみなされます。

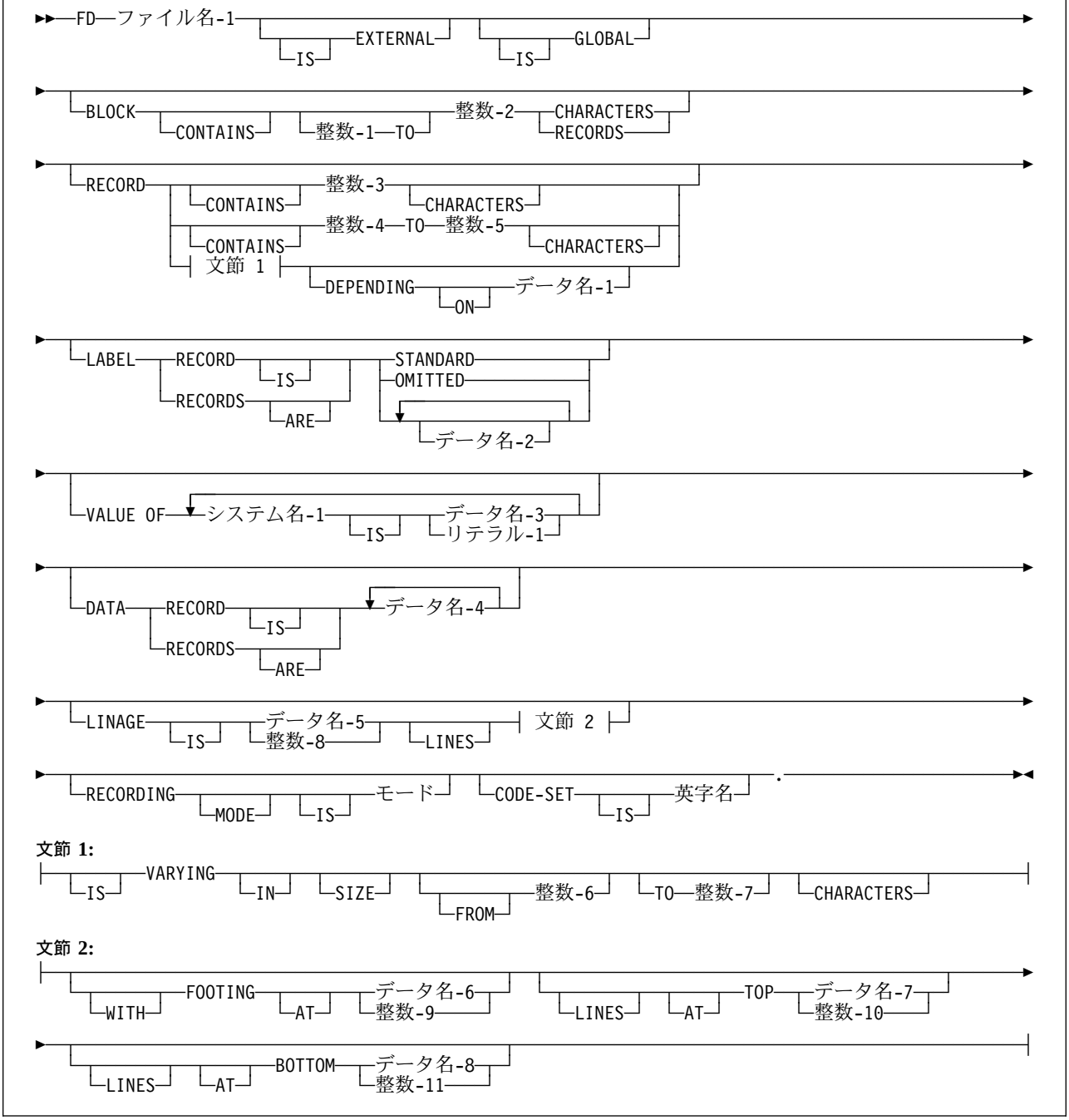
編集符号

編集符号は編集済み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。

データ部 - ファイル記述項目

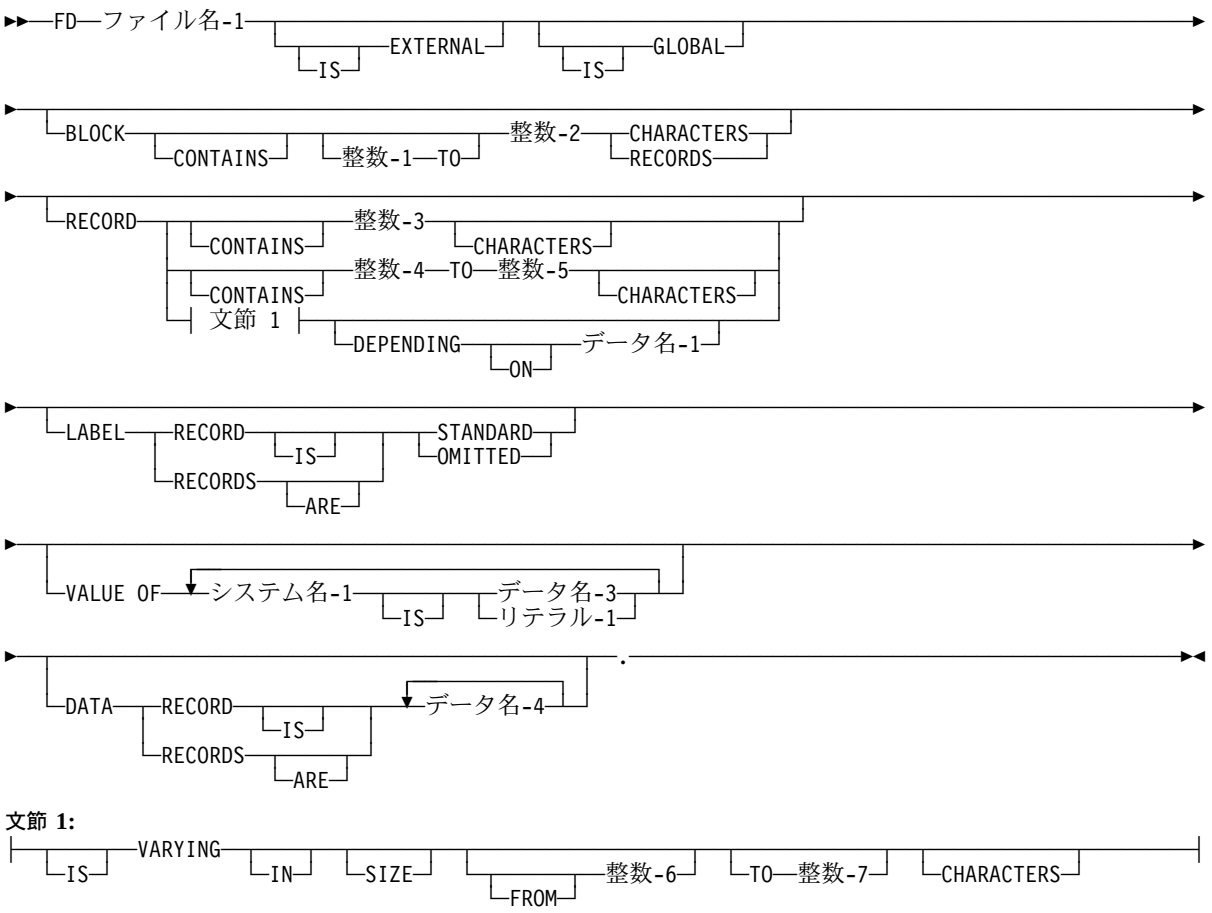
COBOL プログラムで、**ファイル記述 (FD) 記入項目** (またはソート / マージ・ファイルの場合には、**ソート・ファイル記述 (SD) 記入項目**) は、ファイル・セクションの中の最高レベルの編成を表します。FD 記入項目や SD 記入項目の後にオプションの文節をどのような順序で指定するかは、重要なことではありません。

フォーマット 1 - 順次ファイル



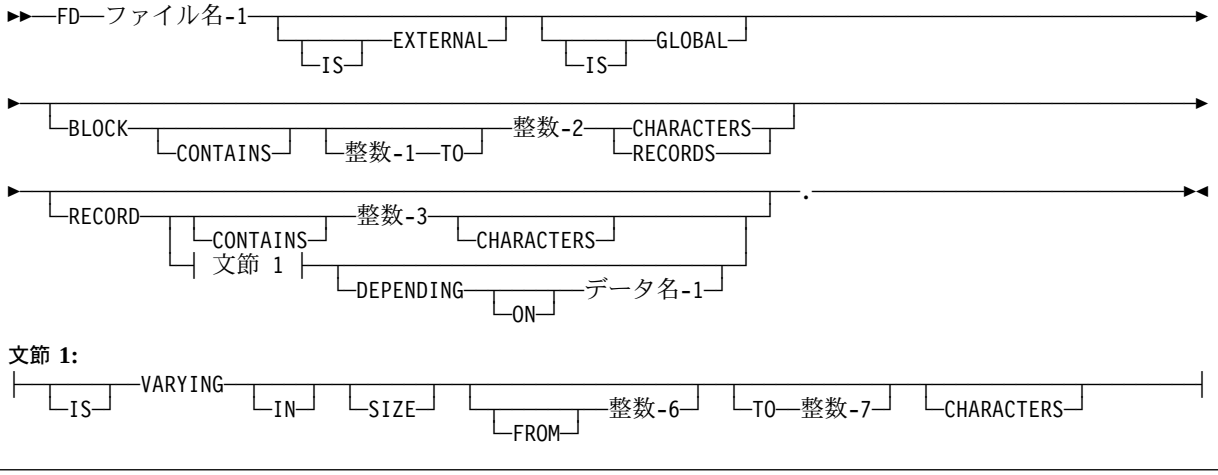
© Copyright IBM Corp. 1991, 2001

フォーマット 2 - 相対 / 索引付きファイル



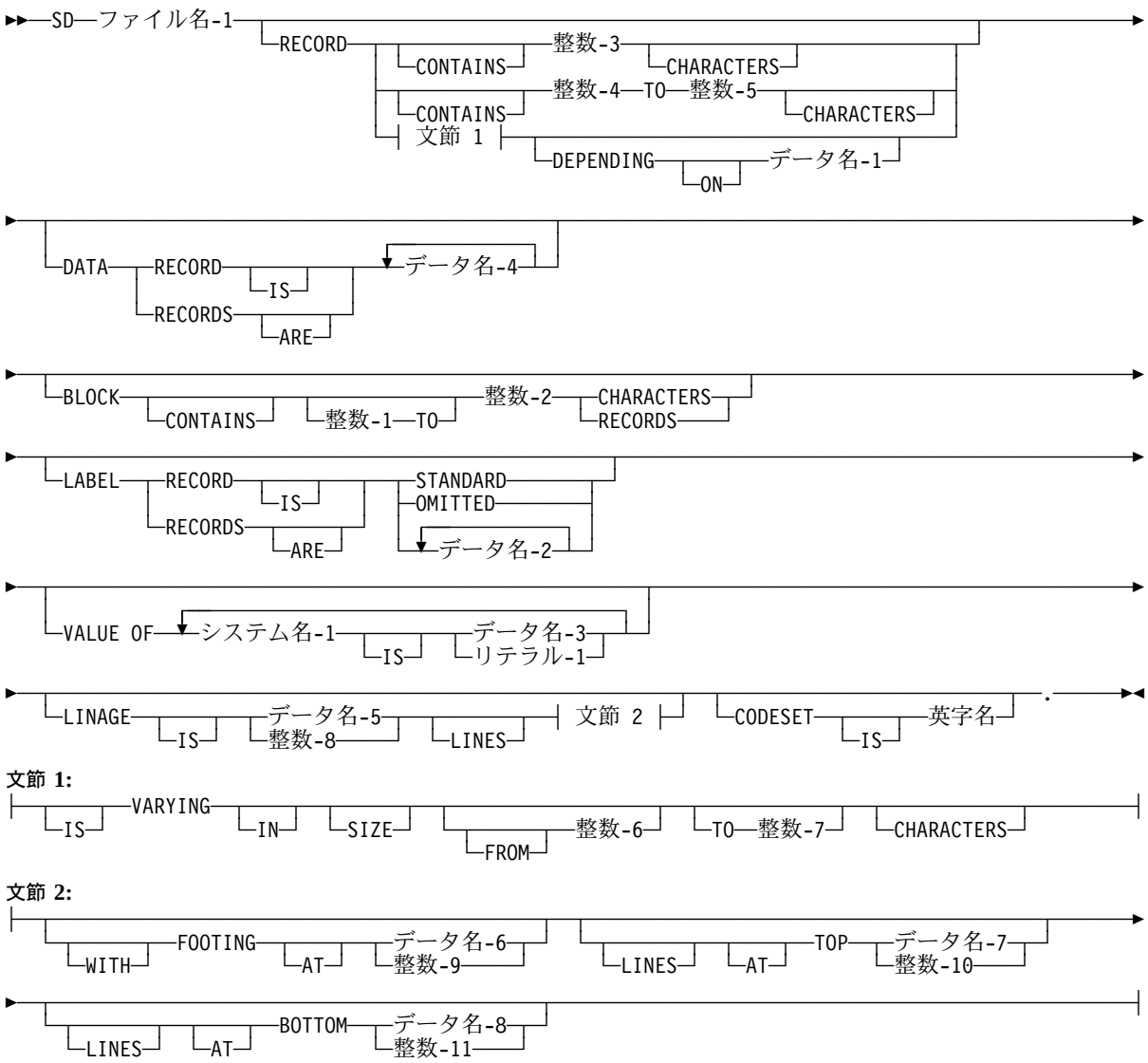
文節 1:

フォーマット 3 - 行順次ファイル



文節 1:

フォーマット 4 - ソート / マージ・ファイル



ファイル・セクション

ファイル・セクションには、入力ファイルと出力ファイルのそれぞれに対するレベル標識が含まれていなければなりません。

- ソート / マージ・ファイル以外のすべてのファイルの場合、ファイル・セクションには必ず FD 記入項目が必要です。
- ソート・ファイルまたはマージ・ファイルの場合、ファイル・セクションには必ず SD 記入項目が必要です。

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 文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。そのファイル名は、プログラムの中で固有でなければなりません。

ファイル名の後には、1 つまたは複数のレコード記述項目が付きます。2 つ以上のレコード記述項目を指定する場合、それぞれの記入項目が同一のストレージ域の再定義になることを意味しています。

ファイル名の後の文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 記入項目の最後の文節の直後には、区切り文字の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 記入項目は、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 記入項目の最後の文節の直後には、区切り文字の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・ファイルで必要なファイル・セクションの記入項目を示しています。

```
SD  SORT-FILE.

01  SORT-RECORD  PICTURE X(80).
```

ファイル・セクションのレコードは、英字、英数字、DBCS、国別、または数値のクラスのグループ項目として、または基本項目として記述しなければなりません。

EXTERNAL 文節

EXTERNAL 文節は、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。

ファイル・セクションの中で EXTERNAL 文節は、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の中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 文節の使用は、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 文節の使用については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

GLOBAL 文節

GLOBAL 文節は、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、それを記述し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。

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 文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 文節を指定している場合、あるいはファイル・セクション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 文節を指定している場合、グローバル名になります。(GLOBAL 文節の使用の詳細については、 *Enterprise COBOL プログラミング・ガイド* 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。

1. 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。
2. 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 文節

BLOCK CONTAINS 文節は、物理レコードのサイズを指定します。CHARACTERS 句は、BLOCK CONTAINS 文節で指定する整数は、レコード内のバイト数を反映することを示します。

たとえば、DBCS 文字が 10 文字または国別文字が 10 文字のブロックの場合、BLOCK CONTAINS 文節は `BLOCK CONTAINS 20 CHARACTERS` とします。

ファイル内のレコードがブロック化されていない場合には、BLOCK CONTAINS 文節は省略できます。省略した場合、コンパイラーはレコードがブロック化されていないものとみなします。各物理レコードの内容がそれぞれただ 1 つの完全な論理レコードの場合でも、BLOCK CONTAINS 1 RECORD と指定すると、固定長ブロック化レコードになります。

BLOCK CONTAINS 文節は、関連したファイル制御記入項目が VSAM ファイルを指定している場合には省略できます。ブロック化の概念は VSAM ファイルに対しては意味がありません。文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。

EXTERNAL ファイルに関しては、相互に対応する EXTERNAL ファイルのすべての BLOCK CONTAINS 文節の値は、実行単位内部で一致していなければなりません。この一致はバイトに関するもので、値が CHARACTERS として指定されているか、あるいは RECORDS として指定されているかにはよりません。

整数-1、整数-2

0 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 と整数-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 と整数-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。たとえば、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。

QSAM ファイルでは、BLOCK CONTAINS 0 を指定することができます。BLOCK CONTAINS 0 が QSAM ファイルに指定されている場合には、以下のようになります。

- ブロック・サイズは実行時に DD パラメーターまたはデータ・セット・ラベルによって決定されます。RECORD CONTAINS 0 CHARACTERS 文節を指定し、さらに BLOCK CONTAINS 0 CHARACTERS 文節を指定する (または省略する) と、ブロック・サイズは、実行時に DD パラメーターまたはファイルのデータ・セット・ラベルによって決定されます。出力データ・セットの場合、上記の条件のいずれかが当てはまる場合は、言語環境プログラムの使用する DCB のブロック・サイズ値は 0 になります。ブロック・サイズ値を指定しない場合、オペレーティング・システムはシステムの決定するブロック・サイズ (SDB) を選択する場合があります。SDB に関する詳細は、オペレーティング・システムの仕様を参照してください。

  SYSIN/SYSOUT ファイルの場合は、BLOCK CONTAINS を省略することができます。その場合、ブロック化はオペレーティング・システムによって決定されます。

BLOCK CONTAINS 文節は、SD で指定された際に、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。

BLOCK CONTAINS 文節を、RECORDING MODE U 文節と共に使用することはできません。

RECORD 文節

RECORD 文節を使用する場合、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。

たとえば、DBCS 文字が 10 文字のレコードがある場合、RECORD 文節は `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 文節は `RECORD CONTAINS 20 CHARACTERS` とする必要があります。

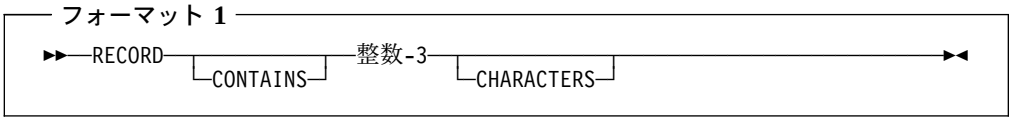
レコード・サイズは、グループ項目のサイズを得る際の規則に従って決定されます。(206ページの『USAGE 文節』および 199ページの『SYNCHRONIZED 文節』を参照。)

RECORD 文節を省略した場合、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。レコード記述内の記入項目の 1 つに OCCURS DEPENDING ON 文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。

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。



整数-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。

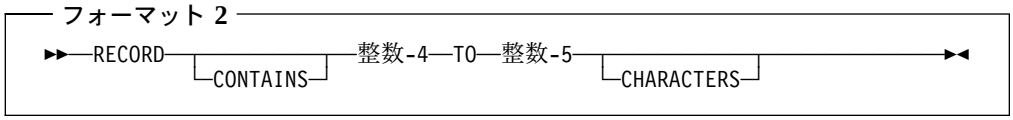
RECORD CONTAINS 0 CHARACTERS 文節を、固定長レコードを含む入力 QSAM ファイルに対して指定することができます。レコード・サイズは、オブジェクト時に、DD ステートメント・パラメーターまたはデータ・セット・ラベルによって決定されます。オブジェクト時に実際のレコード長が 01 のレコード記述よりも大きい場合には、01 のレコード長だけが使用可能です。逆に実際のレコード長がそれよりも短い場合には、実際のレコード長だけが参照できます。それ以外では、初期設定されていないデータまたはアドレッシング例外になる可能性があります。

注: RECORD CONTAINS 0 文節を指定した場合、SAME AREA、SAME RECORD AREA、または APPLY WRITE-ONLY 文節を指定することはできません。

RECORD CONTAINS 0 文節は SD 記入項目については指定しないでください。

フォーマット 2

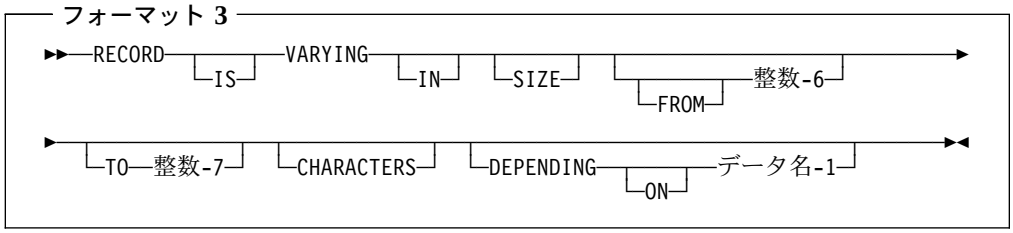
フォーマット 2 は、固定長レコードまたは可変長レコードのバイトの数を指定します。すべての 01 レコード記述項目に示されたレコード長が同一である場合には、固定長レコードになります。フォーマット 2 の RECORD CONTAINS 文節が必要になることはありません。最小と最大のレコード長はレコード記述項目によって決定されるからです。



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 をウィンドウ化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 が指定されている場合は、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オカレンス項目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 が指定されている場合は、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 文節

LABEL RECORDS 文節は、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。

VSAM ファイルでは、LABEL RECORDS 文節は 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。したがって、COBOL のラベル処理は実行されません。

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。

LABEL RECORDS 文節は、SD の下ではコメントとして扱われます。

VALUE OF 文節

VALUE OF 文節は、この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。この文節は、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。

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、ワーキング・ストレージ・セクションに記述しなければなりません。USAGE IS INDEX 文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形象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 文節は、SD で指定された際に、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 文節

DATA RECORDS 文節は構文チェックされますが、この文節は、この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
この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 文節

LINAGE 文節は、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 文節は、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。

整数はすべて無符号でなければなりません。データ名はすべて、無符号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や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。

図4 に、LINAGE 文節の各句の使用法を示します。

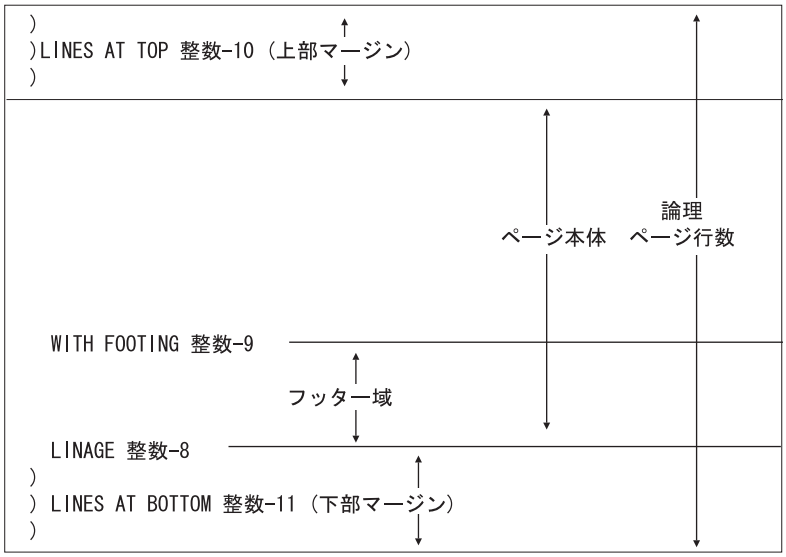


図 4. LINAGE 文節句

LINAGE 文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 と LINES AT BOTTOM の両句またはそのどちらかを省略した場合、上部マージンと下部マージンとして想定される値は 0 です。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、想定される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、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。図4 を参照してくだ

さい。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、およびデータ名-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、データ名-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 文節が含まれている場合には、LINAGE 文節があること。
- 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、およびデータ名-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。

EXTERNAL ファイルの紙送り制御文字の動作については、426ページの『ADVANCING 句』を参照してください。

LINAGE 文節は、SD のもとではコメントとして扱われます。

LINAGE-COUNTER 特殊レジスター

LINAGE-COUNTER 特殊レジスターについては、16ページの『LINAGE-COUNTER』を参照してください。

RECORDING MODE 文節

RECORDING MODE 文節は、QSAM ファイル内の物理レコードのフォーマットを指定します。この文節は、VSAM ファイルでは無視されます。

RECORDING MODE で使用できる値は次のとおりです。

レコード・モード F (固定長)
ファイル内のレコードはすべて同一の長さで、それぞれのレコードは 1 つのブロック内に完全に含まれています。ブロックには複数のレコードを含めることが可能であり、多くの場合、1 ブロックのレコード数は一定しています。このモードには、レコード長フィールドやブロック記述子フィールドはありません。

レコード・モード V (可変長)
レコードは固定長または可変長であり、各レコードは 1 つのブロック内に完全に含まれなければなりません。ブロックには複数のレコードを含めることができます。各データ・レコードにはレコード長フィールドが含まれており、各ブロックにはブロック記述子フィールドが含まれています。これらのフィールドは、データ部には記述されませ

ん。これらのフィールドの長さはそれぞれ 4 バイトであり、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。

レコード・モード U (固定長または可変長)
レコードは固定長か可変長のいずれかです。しかし、各ブロックのレコードは 1 つだけです。レコード長フィールドやブロック記述子フィールドはありません。

注: RECORDING MODE U は、BLOCK CONTAINS 文節を使用している場合には使用できません。

レコード・モード S (スパン)
レコードは、可変長または固定長のどちらかであり、1 ブロックより大きい場合も可能です。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、そのレコードの断片が書き込まれます。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データ部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。

注: レコード・モード S を使用する場合は、BLOCK CONTAINS CHARACTERS 文節が指定されていなければなりません。レコード・モード S は、ASCII ファイルでは使用することができません。

QSAM ファイルで RECORDING MODE 文節が指定されていない場合、Enterprise COBOL コンパイラーはレコード・モードを以下のようにして決めます。

F そのファイルに関連付けられたレベル-01 の最大のレコードが、BLOCK CONTAINS 文節の中に指定されたブロック・サイズ以下である場合、コンパイラーはレコード・モードを F にします。この場合、次のいずれかのようにします。

- RECORD CONTAINS *整数* 文節を使用します。(詳細については、*Enterprise COBOL コンパイラーおよび実行時プログラム 移行ガイド* を参照してください。)
- RECORD 文節を省き、そのファイルに関連したすべてのレベル-01 のレコードが同じサイズであり、OCCURS DEPENDING ON 文節の指定されたレコードがないことを確認します。

V そのファイルに関連付けられたレベル 01 の最も大きなレコードが、BLOCK CONTAINS 文節の中に指定されたブロック・サイズよりも大きくない場合、コンパイラーはレコード・モードを V にします。この場合、次のいずれかのようにします。

- RECORD IS VARYING 文節を使用します。
- RECORD 文節を省き、そのファイルに関連したすべてのレベル-01 のレコードが同じサイズではなく、OCCURS DEPENDING ON 文節の指定されたレコードがあることを確認します。
- RECORD CONTAINS *整数-1* TO *整数-2* 文節を使用します。整数-1 にはそのファイルに関連したレベル-01 のレコードの最小の長さを、整数-2 にはその最大の長さを指定します。これら 2 つの整数の値は違う値であり、異なる長さのレコードまたは OCCURS DEPENDING ON 文節の指定されたレコードのどちらかの最小値と最大値に一致した値でなければなりません。

S 最大のブロック・サイズが最大のレコード・サイズより小さい場合、コンパイラーはレコード・モードを S にします。

U レコード・モード U は、デフォルトでは決して指定されません。RECORDING MODE U 文節は、明示的に使用しなければなりません。

CODE-SET 文節

CODE-SET 文節は、磁気テープ・ファイル上でデータを表現するために使用される文字コードを指定します。CODE-SET 文節を使用すれば、入出力装置上のデータを表すために使用される文字コード規則が英字名によって識別されます。

英字名は、SPECIAL-NAMES 段落の中で、ASCII コードのファイルでは STANDARD-1 として、ISO 7 ビット・コードのファイルでは STANDARD-2 として、EBCDIC コードのファイルでは EBCDIC として、または NATIVE として、定義されている必要があります。NATIVE を指定した場合、CODE-SET 文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。

また CODE-SET 文節は、入出力メディア上の文字コードと内部 EBCDIC 文字セットの間で変換するためのアルゴリズムを指定します。

あるファイルに対して CODE-SET 文節を使用する場合、そのファイルのデータはすべて USAGE DISPLAY でなければならず、また符号付き数字データがある場合は、それを SIGN IS SEPARATE 文節を使用して記述していなければなりません。

CODE-SET 文節を省略する場合、このファイルの文字セットは EBCDIC 文字セットとみなされます。

関連した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た実行単位内のすべての CODE-SET 文節は、同一の文字セットでなければなりません。

CODE-SET 文節は、磁気テープ・ファイルに対してのみ有効です。

CODE-SET 文節は、SD で指定された際に、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。

データ部 - データ記述記入項目

データ記述記入項目は、データ項目の特性を指定します。

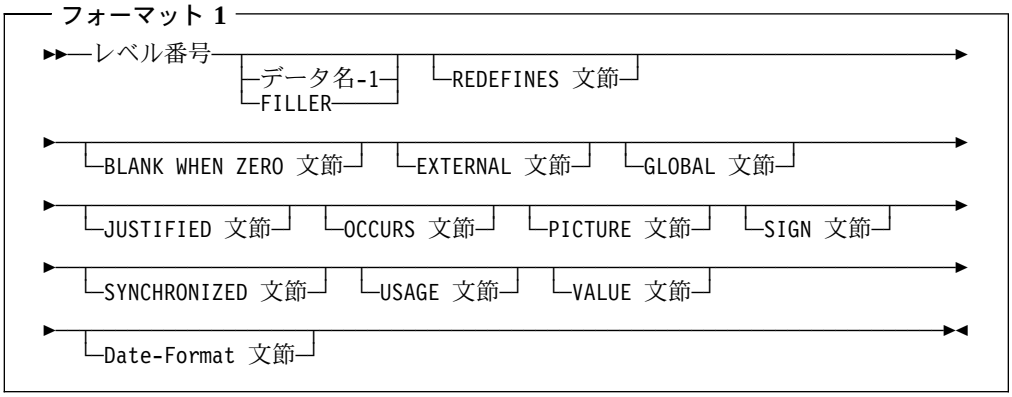
この章では、データ記述記入項目とレコード記述項目 (いくつかのデータ記述記入項目の集まり) のコーディングの仕方について説明します。この章では、データ記述記入項目を指すときもレコード記述項目を指すときも、**データ記述記入項目**という用語を使用します。

独立したデータ項目を定義するデータ記述記入項目は、レコードを構成しません。そのような項目を**データ項目記述記入項目**といいます。

データ記述記入項目の一般的なフォーマットには 3 種類あります。データ記述記入項目は、終わりに区切り文字の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、データ記述記入項目についてデータ部の全セクションで使用されます。



注: 文節はどのような順序でも記述できますが、例外が 2 つあります。

データ名または 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 文節を指定する場合、データ名または 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。データ名または FILLER を指定しない場合、REDEFINES 文節はレベル番号の直後に置かなければなりません。

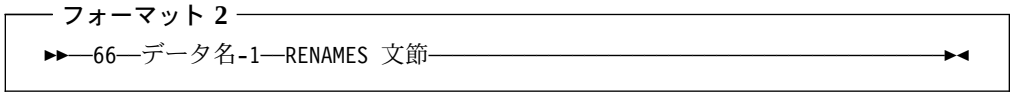
フォーマット 1 のレベル番号としては、01 ～ 49、または 77 のいずれかの番号が使用できます。

文節を区切るためには、スペース、区切り文字のコンマ、または区切り文字のセミコロンが必要です。

© Copyright IBM Corp. 1991, 2001

フォーマット 2

フォーマット 2 は、すでに定義されている項目を再グループ化します。



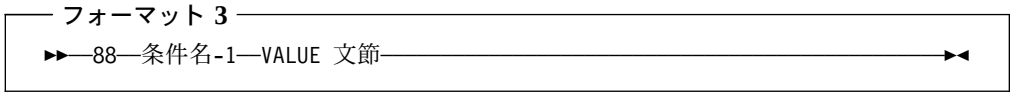
レベル-66 の項目は、他のレベル-66 の項目の名前に変更することはできません。また、レベル-01、レベル-77、またはレベル-88 の項目についても、名前の変更はできません。

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記入項目の直後になければなりません。

詳しい説明は、196ページの『RENAMES 文節』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。



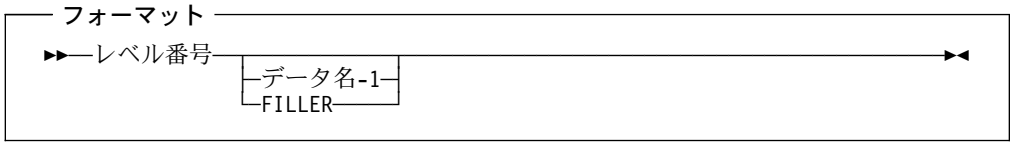
条件名
ある値、値の集合、または値の範囲を条件変数に関係付けるユーザー指定の名前。

条件変数とは、1 つ以上の値を取ることができるデータ項目で、これらの値を順番に条件名と関係付けることができます。

フォーマット 3 は、基本項目とグループ項目の双方を記述するために使用することができます。条件名項目の詳細については、214ページの『VALUE 文節』を参照してください。

レベル番号

レベル番号は、レコード内のデータの階層を指定し、また特別な目的を持つデータ項目を識別します。レベル番号は、データ記述記入項目、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 ～ 49 の整数か、または特別なレベル番号 66、77、または 88 のいずれかの値になります。



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、その後に区切り文字の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文節を付けなければなりません。

レベル番号 02 ～ 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区切り文字の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 ～ 9 は、レベル番号 01 ～ 09 で置き換えることができます。

連続するデータ記述記入項目は、最初の項目と同じ桁から始めることも、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。

詳細については、139ページの『データのレベル』を参照してください。

データ名
これは、記述されるデータを明示的に識別します。

指定すれば、データ名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 文節や EXTERNAL 文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 文節および EXTERNAL 文節を持っているときにも指定しなければなりません。

FILLER
これは、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。

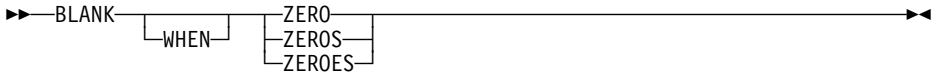
MOVE CORRESPONDING ステートメント、ADD CORRESPONDING ステートメント、または SUBTRACT CORRESPONDING ステートメントの中では、FILLER 項目は無視されます。INITIALIZE ステートメントの中では、FILLER 基本項目は無視されます。

データ名または FILLER 文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 文節

BLANK WHEN ZERO 文節は、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 文節は、基本数字項目または基本数字編集項目に対してのみ指定することができます。これらの項目は、USAGE IS DISPLAY 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。数字項目に対して BLANK WHEN ZERO 文節を指定すると、その項目は数字編集項目とみなされます。

BLANK WHEN ZERO 文節は、レベル 66 の項目またはレベル 88 の項目に対しては指定できません。

PICTURE 記号 S または * と同じ記入項目に対して、BLANK WHEN ZERO 文節を指定してはなりません。

BLANK WHEN ZERO 文節は、次のような項目には使用できません。

- USAGE IS INDEX 文節を使用して記述されている項目
- 日付フィールド
- DBCS 項目
- 国別項目
- 外部浮動小数点項目および内部浮動小数点項目
- USAGE IS POINTER、USAGE IS FUNCTION-POINTER、USAGE IS PROCEDURE-POINTER、または USAGE IS OBJECT REFERENCE で定義されているデータ項目

DATE FORMAT 文節

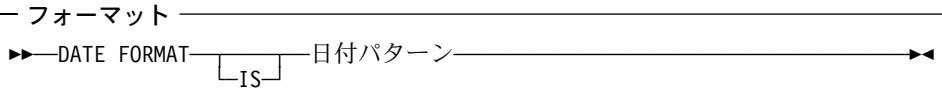
DATE FORMAT 文節は、データ項目がウィンドウ化日付フィールドまたは拡張日付フィールドであることを指定します。

ウィンドウ化日付フィールド
YY を含む DATE FORMAT 文節によって指定されたウィンドウ化 (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 文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 文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 文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。

フォーマット



日付パターン は、ウィンドウ化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 ～ 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	ウィンドウ化 (2 桁) 年。
YYYY	拡張 (4 桁) 年。
X	1 文字。たとえば、1 学期または四半期 (1 ～ 4) などを表す数字。
XX	2 文字。たとえば、月を表す数字 (01 ～ 12)。
XXX	3 文字。たとえば、1 年のうちの何日目かを表す数字 (001 ～ 366)。
XXXX	4 文字。たとえば、月を表す 2 桁 (01 ～ 12) と月の何日かを表す 2 桁 (01 ～ 31)。

日付フィールドおよび関連名の概要については、66ページの『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日付フィールドを使用する方法については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

ウィンドウ化日付フィールドのセマンティクス

ウィンドウ化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。

ウィンドウ化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、ウィンドウ化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、数字ウィンドウ化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化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。たとえばウィンドウ化西暦年の値 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化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。

比較条件のオペランドとして使用された場合の例を以下に示します。

```
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化日付フィールドは、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)

日付トリガー値

DATEPROC(TRIG) コンパイラー・オプションが有効である場合、ウィンドウ化日付フィールドの拡張ではウィンドウ化日付フィールド中の特定のトリガー値または限界値が考慮されます。

英数字ウィンドウ化日付フィールドの場合の特殊値は LOW-VALUE、HIGH-VALUE、および SPACE です。DATE FORMAT 文節に最低 1 つの X が含まれる英数字および数字ウィンドウ化日付フィールド (つまりウィンドウ化西暦年だけではないウィンドウ化日付フィールド) の場合、全桁 0 または全桁 9 の値もトリガーとして扱われます。

全桁 0 の値は、どんな有効日付より前の日付となります。全桁 9 の値の目的は、どんな有効日付よりも後の日付として動作することです。

このような方法でウィンドウ化日付フィールドにトリガーが含まれる場合は、19 や 20 を世紀値として推測するのではなく、トリガー値が拡張日付結果の世紀部分にコピーされるかのようにして拡張されます。

この特殊トリガー拡張は、ウィンドウ化日付フィールドが比較条件のオペランドとして、または算術または MOVE ステートメントの送り出しフィールドとして使用される場合に実行されます。トリガー拡張は、ウィンドウ化日付フィールドが算術式のオペランドとして使用される場合は実行されませんが、算術式のウィンドウ化日付の最終結果には適用されることがあります。

日付フィールドの使用に関する制約事項

以降のページでは、日付フィールドを以下の関連で使用する場合の制約事項をまとめています。

- DATE FORMAT 文節を他の文節と結合する
- 日付フィールドだけから構成されるグループ項目
- 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
- 日付フィールドを引き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。

- 232ページの『日付フィールドでの算術演算』
- 241ページの『日付フィールド』(条件式の中)
- 278ページの『ADD ステートメント』
- 414ページの『SUBTRACT ステートメント』
- 346ページの『MOVE ステートメント』

DATE FORMAT 文節と他の文節との結合

DATE FORMAT 文節と結合することができる USAGE 文節の句は、DISPLAY、COMPUTATIONAL (またはそれと等価の COMPUTATIONAL-4 および BINARY)、および COMPUTATIONAL-3 (またはそれと等価の PACKED-DECIMAL) だけです。DATE FORMAT 文節は、TRUNC(BIN) コンパイラー・オプションが有効である場合、USAGE COMP データ項目には使用できません。

PICTURE 文節の文字ストリングでは、DATE FORMAT 文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。

以下の文節は、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 文節の SEPARATE CHARACTER 句

EXTERNAL 文節は、ウィンドウ化日付フィールド、またはウィンドウ化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。

次の文節を 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。

REDEFINES (192ページを参照)
VALUE (214ページを参照)

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 文節を使用して定義されている場合は、次の制約事項が適用されます。

- 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 文節の*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 文節が指定されている場合、両方の DATE FORMAT 文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。

  1. 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 文節がある場合、DATE FORMAT 文節は、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。
  2. グループ項目が YYXXXX、YYYYXXXX、XXXXYY、または XXXXYYYY の DATE FORMAT 文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 文節はそれぞれ 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)。

3. ウィンドウ化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。
4. 上記の制限事項で説明したように、DATE FORMAT 文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化部分、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。

たとえば、以下は有効なグループ項目を定義しています。

```
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

ウィンドウ化日付フィールドを引き数として受け入れない言語エレメント

ウィンドウ化日付フィールドは、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いてデータ名として。

  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) 記入項目の RECORD IS VARYING DEPENDING ON 文節においてデータ名として。
- データ部データ定義記入項目の OCCURS DEPENDING ON 文節のオブジェクトとして。

- データ部データ定義記入項目の OCCURS 文節の ASCENDING KEY または DESCENDING KEY 句のキーとして。
- 以下のステートメントにおいてデータ名または ID として。

  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。
- INVOKE ステートメントで、メソッドが呼び出されるオブジェクトを指定する ID、またはメソッド名を含む ID として。
- PERFORM ステートメントの TIMES および VARYING 句の ID として (ウィンドウ化日付フィールドは 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 (ウィンドウ化日付フィールドは SEARCH 条件の中では指定*可能*)。
- WRITE ステートメントの ADVANCING 句の ID として。
- 組み込み関数 (UNDATE 組み込み関数を除く) への引き数として。

ウィンドウ化日付フィールドを、MERGE および SORT ステートメントで、いくつかの制限付きで昇順キーまたは降順キーとして使用することができます。詳細については、340ページの『MERGE ステートメント』および 398ページの『SORT ステートメント』を参照してください。

日付フィールドを引き数として受け入れない言語エレメント

以下のステートメントでは、ウィンドウ化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 句の ID として使用する場合を除く)
- MULTIPLY ステートメント (GIVING 句の 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 文節

EXTERNAL 文節は、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またはメソッドではなく、その実行単位に関連付けられているということを指定します。外部データ項目は、そのデータ項目を記述している実行単位内のどのプログラムまたはどのメソッドからでも参照できます。データ項目の個々の記述を使用して別々のプログラムまたはメソッドから構成される外部データ項目に対する参照は、常に同じデータ項目に対するものです。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 文節は、レベル番号が 01 のデータ記述記入項目の中でのみ指定できます。それは、プログラムまたはメソッドのワーキング・ストレージ・セクションにあるデータ記述記入項目でのみ指定できます。リンケージ・セクションまたはファイル・セクションのデータ記述記入項目では指定できません。外部レコードを記述している項目に従属するデータ記述記入項目によって記述されているデータ項目があれば、それにも EXTERNAL 属性が影響します。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。

データ名文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またはどのメソッド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またはメソッドが同じ外部データ・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標準データ・フォーマット文字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 つのプログラムまたはメソッドに、外部レコード全体を再定義する REDEFINES 文節を含むデータ記述記入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またはメソッドで同じように全体を再定義する必要はありません。
- EXTERNAL 文節を使用しても、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。

GLOBAL 文節

GLOBAL 文節は、データ名が、そのデータ名を宣言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宣言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。

あるデータ名がそれによって宣言されているデータ記述記入項目か、そのデータ記述記入項目が従属している別の項目のどちらかに GLOBAL 文節が指定されている場合、そのデータ名はグローバル名です。GLOBAL 文節は、ワーキング・ストレージ・セクション、ファイル・セクション、リンケージ・セクション、ローカル・ストレージ・セクションの中で指定することができます。ただし、レベル番号が 01 のデータ記述記入項目の中でなければなりません。

同じデータ部の中では、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記入項目に GLOBAL 文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。

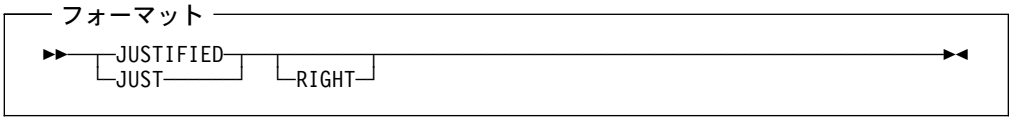
実行単位内の 2 つのプログラムは、次のような場合には共通データを参照することができます。

1. 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。
2. 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。

JUSTIFIED 文節

JUSTIFIED 文節は、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 文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 文節は指定できません。

- 数字、数字編集項目、または英数字編集項目の場合
- 編集済み DBCS 項目の場合
- USAGE IS INDEX 文節を付けて記述されている項目の記述の中
- USAGE IS FUNCTION-POINTER、USAGE IS POINTER、USAGE IS PROCEDURE-POINTER、または USAGE IS OBJECT REFERENCE で定義されているデータ項目
- 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

受け取り項目で JUSTIFIED 文節を指定すると、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は DBCS スペース (X'4040') で埋められます。国別項目の場合、それぞれの未使用位置はデフォルトの Unicode スペース (X'0020') で埋められます。それ以外の場合、それぞれの未使用位置は英数字スペースで埋められます。

JUSTIFIED 文節を省略すると、標準位置合わせの規則に従います (144ページの『位置合わせの規則』を参照)。

JUSTIFIED 文節は、VALUE 文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。

OCCURS 文節

テーブル操作に使用されるデータ部の文節は、OCCURS 文節と USAGE IS INDEX 文節です。USAGE IS INDEX の説明については、206ページの『USAGE 文節』を参照してください。

OCCURS 文節は、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、この文節を使用すると、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 文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 文節の**サブジェクト**は、その OCCURS 文節を含んでいるデータ項目のデータ名です。OCCURS 文節それ自体を除き、そのサブジェクトと共に書かれたデータ記述文節は、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 文節のサブジェクト、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、必ず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
- 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

添え字付けまたは指標付けがなされている場合、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。

添え字付けや指標付けがなされていない場合、サブジェクトはテーブル全体を参照します。

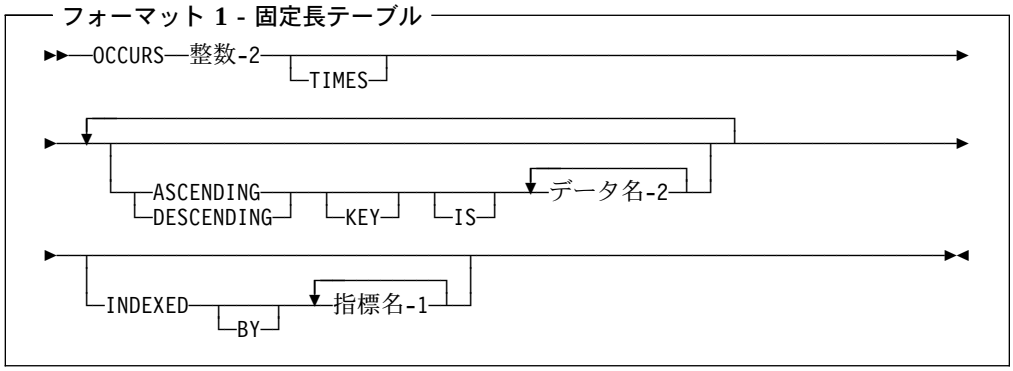
OCCURS 文節は、次のようなデータ記述記入項目では指定できません。

- 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 文節を含む項目に従属することは可能)。192ページの『REDEFINES 文節』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 文節を使用します。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 文節は、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 文節は、OCCURS DEPENDING ON 文節への従属文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。

フォーマット 1 - 固定長テーブル



整数-2
正確なオカレンス数。整数-2 は 0 より大きくなければなりません。

ASCENDING/DESCENDING KEY 句

データは、データ名-2 に含まれる値に従い、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、レベルが高いものから順に指定します。

順序は、オペランドの比較の規則に従って決められます (239ページの『比較条件』を参照)。ASCENDING および DESCENDING KEY データ項目は、OCCURS 文節とテーブル・エレメントの二分探索を実行する SEARCH ALL ステートメントで使用されます。

データ名-2

これは、サブジェクト記入項目の名前、またはサブジェクト記入項目に従属する項目の名前でなければなりません。データ名-2 としてウィンドウ化日付フィールドは使用できません。データ名-2 は修飾できます。

データ名-2 がサブジェクト記入項目を指定している場合、その記入項目の全体が ASCENDING/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 がサブジェクト記入項目を指定しない場合、データ名-2 は次のようになります。

- テーブル記入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 文節を含む他の記入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 文節を含むことはできません。

データ名-2 では、OCCURS DEPENDING ON 文節を含む従属項目は使用できません。

ASCENDING/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 ASCENDING または DESCENDING の順に並べておく必要があります。
- キーは、以下の Usage が可能です。BINARY、DISPLAY、DISPLAY-1、NATIONAL、 PACKED-DECIMAL、COMPUTATIONAL、COMPUTATIONAL-1、COMPUTATIONAL-2、COMPUTATIONAL-3、COMPUTATIONAL-4、または COMPUTATIONAL-5 です。
- 1 つのテーブル・エレメントに関連付けられた全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 文節のサブジェクトおよび OCCURS 文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、その記入項目に従属していますが、WEEK-RECORD のキーは、その従属記入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。60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。

- ロール・ストレージ・セクション
- クラス定義 (オブジェクト・インスタンス変数) のワーキング・ストレージ・セクション
- リンケージ・セクションは、以下のとおりです。
  - メソッド
  - RECURSIVE 属性を指定してコンパイルしたプログラム
  - THREAD オプションを指定してコンパイルしたプログラム内で

注: 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。

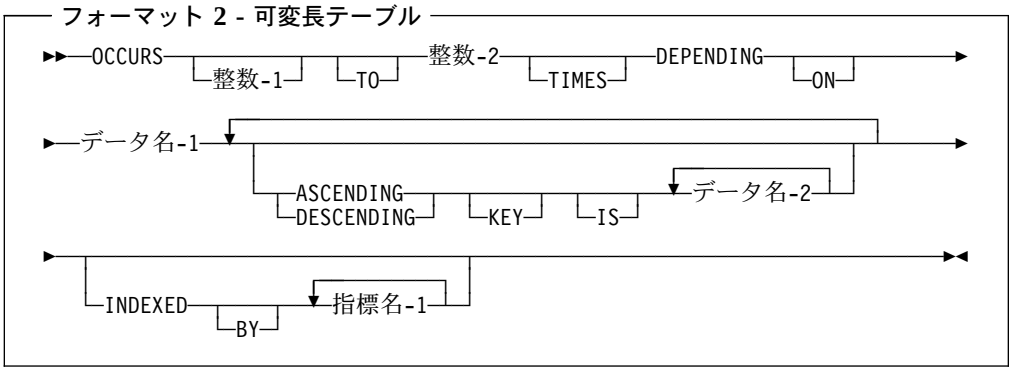
未参照の指標名を固有に定義する必要はありません。

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。

GLOBAL 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 GLOBAL 属性を持ちます。したがって、索引名の有効範囲は、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あり、命名規則の同じ有効範囲が適用されます。

可変長テーブル

可変長テーブルを指定するには、OCCURS DEPENDING ON 文節を使用します。



整数-1
最小のオカレンス回数。

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 とみなされます。

整数-2
最大のオカレンス回数。

整数-2 は、整数-1 より大きくなければなりません。

サブジェクト記入項目の**長さ**は固定長です。変数であるサブジェクト記入項目が**繰り返される回数**にすぎません。

OCCURS DEPENDING ON 文節

OCCURS DEPENDING ON 文節は、可変長テーブルを指定します。

データ名-1
OCCURS DEPENDING ON 文節の**オブジェクト**、つまり、現行の**サブジェクト**項目のオカレンス回数を表す現行値を持つデータ項目を指定してください。そのオカレンス回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 文節 (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。オブジェクトにウィンドウ化日付フィールドを指定することはできません。

OCCURS DEPENDING ON 文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 文節のオブジェクトの位置は自由にはなりません。OCCURS DEPENDING ON 文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 文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記入項目で OCCURS 文節を指定する場合、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、記入項目のサブジェクトと同じデータ部に記述しなければなりません。

GLOBAL 文節を含むデータ記述記入項目に従属するデータ記述記入項目の中で OCCURS 文節を指定する場合、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、記入項目のサブジェクトと同じデータ部に記述しなければなりません。

OCCURS 文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。

従属する OCCURS DEPENDING ON 項目を含んでいるデータ項目か、OCCURS DEPENDING ON 項目の後にあるがそれに従属しているわけではないデータ項目、またはグループ項目が参照されるとき、OCCURS DEPENDING ON 文節のオブジェクトの値は、整数-1 ～整数-2 の範囲内になるようにしなければなりません。

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
- INVOKE ... USING BY REFERENCE
- MOVE ... TO ID
- READ ... INTO ID
- RELEASE ID FROM ...
- RETURN ... INTO ID
- REWRITE ID FROM ...
- STRING ... INTO ID
- UNSTRING ... INTO ID DELIMITER IN ID
- WRITE ID FROM ...

CALL ... USING BY REFERENCE ID ステートメントの ID として可変長グループが指定されている場合は、必ず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 文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または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 文節のオブジェクトは、整数-1 ～整数-2 の範囲内になるようにしなければなりません。

結果として 1 つまたは複数の項目の位置が変化するような方法で、OCCURS DEPENDING ON IS 文節が使用されるとき、その文節は、**複合 OCCURS DEPENDING ON** 文節です。複合 OCCURS DEPENDING ON 文節は、以下のような場合に定義されます。

- 複合 OCCURS DEPENDING ON 文節は、OCCURS DEPENDING ON を使用して記述される、1 つまたは複数の従属項目を含む (すなわち、ネストされた OCCURS DEPENDING ON を*含む*) サブジェクトに対して指定されます。
- 複合 OCCURS DEPENDING ON 文節は、OCCURS DEPENDING ON を使用して記述される項目に、どのレベルにおいても従属する (すなわち、ネストされた OCCURS DEPENDING ON *である*) 項目に対して指定されます。
- 複合 OCCURS DEPENDING ON 文節は、1 つまたは複数の非従属項目が後に付く項目に対して指定されます。この非従属項目は、OCCURS DEPENDING ON を使用してもまたは使用しなくても記述することができます。

  注: これらの非従属項目は、OCCURS DEPENDING ON 文節のオブジェクトにはなれません。

複合 OCCURS DEPENDING ON 文節に記述される従属項目、または複合 OCCURS DEPENDING ON 文節を使用して記述される項目の後に続く非従属項目は、*位置が変化する項目* です。すなわち、その位置は、OCCURS DEPENDING ON オブジェクトの値によって変化します。

ファイル記述 (FD) 記入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 文節を含めることができます。

INDEXED BY 句を、OCCURS DEPENDING ON 文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。

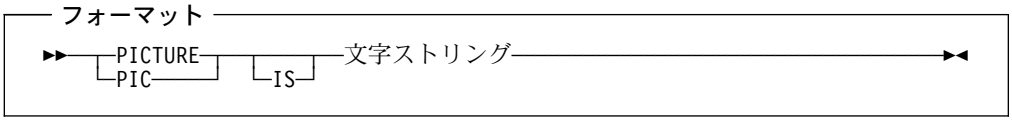
複合 OCCURS DEPENDING ON の詳細については、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

ASCENDING/DESCENDING KEY 句や INDEXED BY 文節については、172ページの『固定長テーブル』を参照してください。

PICTURE 文節

PICTURE 文節は、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 文節は、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
- RENAMES 文節のサブジェクト
- USAGE IS INDEX を使用して記述されている項目
- USAGE IS POINTER、USAGE IS FUNCTION-POINTER、USAGE IS PROCEDURE-POINTER、または USAGE IS OBJECT REFERENCE で定義されているデータ項目
- 内部浮動小数点データ項目

これらの除外例の場合には、文節は使用できません。

PICTURE 文節は、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

PICTURE 文字ストリングは、記号として使用される特定の COBOL 文字から構成されています。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。

PICTURE 文字ストリングの長さは、50 文字までです。

PICTURE 文節で使用される記号

PICTURE 文節で使用される各記号の意味は、179ページの表10 で定義されています。PICTURE 文節の各記号を指定する際に守らなければならないシーケンスは、181ページの図5 に示されています。PICTURE 文節で使用する記号のさらに詳しい説明が、その図の後にあります。

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取り替えることができます。

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, P, S, V, X, Z, CR, DB
E, G, N
```

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。

PICTURE 文節の以下の説明で、*cs* は任意の有効な通貨記号を表しています。**サイズ**という見出しは、データ項目の実際のサイズが分かるように、記号のバイト数を示しています。詳細については、182ページの『通貨記号』を参照してください。

表 10 (1/2). PICTURE 文節の記号の意味

記号	意味	サイズ	制約事項
A	英字またはスペースだけを入れることができる文字位置。	1 バイトを占めます。	
B	DBCS でないデータの場合 - スペース文字が挿入される文字位置。	1 バイトを占めます。	
	DBCS データの場合 - DBCS スペースが挿入される文字位置。1 つの DBCS スペースを含む単一の DBCS 文字位置を表します。	2 バイトを占めます。	
E	外部浮動小数点項目の指数の開始を示します。	1 バイトを占めます。	
G	DBCS 文字位置。	2 バイトを占めます。	DBCS でない項目の場合は指定できません。
N	Usage NATIONAL を使用して指定したときの、または Usage は指定せず、NSYMBOL(NATIONAL) コンパイラー・オプションが有効なときの国別文字の位置。 Usage DISPLAY-1 を使用して指定したときの、または Usage は指定せず、NSYMBOL(DBCS) コンパイラー・オプションが有効なときの DBCS 文字位置。	2 バイトを占めます。	DISPLAY-1 または NATIONAL 以外の Usage が指定されている場合は指定できません。
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。181ページの『P 記号』も参照してください。	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現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。	PICTURE 文字ストリング中の左端または右端の桁位置で、P が連続しているストリングのフォーマットに限り表示されます。
S	算術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)。算術演算符号は、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。	基本項目サイズを判別する際の計算に入れられません。ただし、関連した SIGN 文節に SEPARATE CHARACTER 句 (これは 1 バイトを占めます) が指定されている場合は計算に入れられます。	PICTURE ストリングの中で左端の文字として記述しなければなりません。
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。	文字ストリングの中で一度だけ使用することができます。
X	コンピューターの文字セットの任意の文字を入れることができる文字位置。	1 バイトを占めます。	

表 10 (2/2). PICTURE 文節の記号の意味

記号	意味	サイズ	制約事項
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、データ項目サイズのカウントに入れられます。	
9	数表示を含む文字位置。	「9」は、データ項目サイズのカウントに入れられます。	
0	数字の 0 が挿入される文字位置。	「0」は、データ項目サイズのカウントに入れられます。	
/	スラッシュ文字が挿入される文字位置。	「/」は、データ項目サイズのカウントに入れられます。	
,	コンマが挿入される文字位置。	「,」は、データ項目サイズのカウントに入れられます。	
.	位置合わせ用の小数点を表す編集記号。また、これはピリオドが挿入される文字位置を表します。	「.」は、データ項目サイズのカウントに入れられます。	
+ - 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。	記号として使用される各文字は、データ項目のサイズを決定する際にカウントに入れられます。	これらの記号は、1 つの文字ストリングの中では併用することはできません。
*	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、項目サイズのカウントに入れられます。	
cs	通貨記号。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・ページで値 X'5B' 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 ($) で表されます。詳細については、182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、通貨記号が出現するたびに 1 文字がデータ項目のサイズに加算されます。	

181ページの図5 は、ピクチャー記号を指定して、ピクチャー文字ストリングを形成することができるシーケンスを示します。

第1の記号 / 第2の記号		固定挿入記号										浮動挿入記号						その他の記号							
		B	0	/	,	.	{+ −}	{+ −}	{CR DB}	cs	E	{Z *}	{Z *}	{+ −}	{+ −}	cs	cs	9	A X	S	V	P	P	G	N
固定挿入記号	B ,	●	●	●	●	●	●			●		●	●	●	●	●	●	●	●		●		●	●	
	0	●	●	●	●	●	●			●		●	●	●	●	●	●	●	●		●		●		
	/	●	●	●	●	●	●			●		●	●	●	●	●	●	●	●		●		●		
	,	●	●	●	●	●	●			●		●	●	●	●	●	●	●			●		●		
	.	●	●	●	●		●			●		●		●		●		●							
	{+ −}																								
	{+ −}	●	●	●	●	●				●	●	●	●			●	●	●			●	●	●		
	{CR DB}	●	●	●	●	●				●		●	●			●	●	●			●	●	●		
	cs						●																		
	E				●	●												●			●				
浮動挿入記号	{Z *}	●	●	●	●		●			●		●													
	{Z *}	●	●	●	●	●	●			●		●	●								●		●		
	{+ −}	●	●	●	●					●				●											
	{+ −}	●	●	●	●	●				●					●	●					●		●		
	cs	●	●	●	●		●									●									
	cs	●	●	●	●	●	●								●	●					●		●		
その他の記号	9	●	●	●	●	●	●			●		●		●		●		●	●	●	●		●		
	A X	●	●	●														●	●						
	S																								
	V	●	●	●	●		●			●		●		●		●		●		●		●			
	P	●	●	●	●		●			●		●		●		●		●		●		●			
	P						●			●										●	●		●		
	G	●																						●	
	N																								●

図 5. PICTURE 文節の記号のシーケンス

図の凡例:

- 黒い丸は、文字ストリングにおいて、その列の最上段の記号が、その行の左側の記号より左であればどこでも指定できることを示します。

{ } 中括弧は、相互に排他的な項目を示しています。

2 回現れる記号 固定挿入記号 + と --、浮動挿入記号 Z、*、+、-、および *cs*、および記号 P は、2 回ずつ現れています。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。

P 記号

位取り位置の文字 P は、想定小数点を暗黙のうちに示すため (一連の P が左端の PICTURE 文字であればそれらに P の左側、右端の PICTURE 文字であればそれらの P の右側)、想定小数点を示す記号 V は、その PICTURE 記述の左端または右端の文字としては冗長です。

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。これらの演算には、次のものがあります。

- 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、または環境部の SPECIAL-NAMES 段落の CURRENCY SIGN 文節のいずれかで指定する単一文字によって表されます。

デフォルトの通貨記号は、本書では $ で表されますが、実際のデフォルトの通貨記号は、コンパイル時に有効である EBCDIC コード・ページの値 X'5B' 付きの文字です。

CURRENCY SIGN 文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 文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 文節については、104 ページの『CURRENCY SIGN 文節』を参照してください。CURRENCY および NOCURRENCY コンパイラー・オプションの詳細については、 *Enterprise COBOL プログラミング・ガイド* 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 文節記号とは異なり、通貨記号は大文字小文字が区別されます。たとえば、'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。

PICTURE 文節の以下の説明で、*cs* は任意の有効な通貨記号を表しています。

文字ストリングの表現

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 0 でない整数は、その記号が連続する回数を指定します。

例: 次の 2 つの PICTURE 文節は、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
```

PICTURE 記号の V を除いて、文字ストリングに上記のいずれかの記号があると、そのおのおのはデータ項目中にその 1 文字または有効な一連の文字があることを表します。

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、その項目の次のデータ・カテゴリーが決定されます。

- 英字項目
- 数字項目
- 数字編集項目
- 英数字項目
- 英数字編集項目
- DBCS 項目
- 外部浮動小数点項目
- 国別項目

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。

標準データ・フォーマットの項目の内容は、英字 26 文字とスペース文字のうち任意の文字で構成されていなければなりません。

その他の文節: 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 文節では、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。

数字項目

数字項目には次のような種類があります。

- 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)

数字日付フィールドの場合、PICTURE 文字ストリングに含めることができるのは、記号 9 および S だけです。すべての他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 だけを含めることができます。

2 進数項目の場合は、数字の桁数は 1 ～ 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・オプションが有効であるときは、1 ～ 18 桁で、ARITH(EXTEND) コンパイラー・オプションが

有効なときは、1 ～ 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 文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 ～ 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 through 9999
      S99    -99 through +99
   S999V9    -999.9 through +999.9
   PPP999     0 through .000999
  S999PPP    -1000 through -999000 and
             +1000 through +999000 or zero
```

その他の文節: 項目の USAGE は、DISPLAY、BINARY、COMPUTATIONAL、PACKED-DECIMAL、COMPUTATIONAL-3、COMPUTATIONAL-4、または COMPUTATIONAL-5 とすることができます。

VALUE 文節には、形象定数 ZERO を指定することができます。

基本数字項目と関連付けられた VALUE 文節には、数字リテラルまたは形象定数 ZERO を指定する必要があります。基本数字項目を含むグループ項目に関連付けられた VALUE 文節には、**英数字**リテラルまたは形象定数を指定しなければなりません。グループ項目は英数字項目とみなされるからです。いずれの場合にも、リテラルは指定されたとおりに扱われ、編集は実行されません。

このような場合、NUMPROC および TRUNC コンパイラー・オプションは、数字データ項目の使用に影響を与えることがあります。詳細については、 *Enterprise COBOL プログラミング・ガイド* 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、次の記号を含めることができます。

```
B  P  V  Z  9  0  /  ,  .  +  -  CR  DB  *  cs
```

許容される記号の組み合わせは、PICTURE 文節の許容された記号順序 (181ページの図5 を参照) および編集規則 (188ページの『PICTURE 文節の編集』を参照) によって決められます。

次の規則も適用されます。

- 項目には BLANK WHEN ZERO 文節を指定するか、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
  ```

- 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 ～ 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 ～ 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、249 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。

その他の文節: 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 文節では、英数字リテラルまたは形象定数を指定しなければなりません。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。

英数字項目

PICTURE 文字ストリングは、次のいずれかにより構成されていなければなりません。

- 1 つまたは複数の、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。

その他の文節: 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 文節では、英数字リテラルまたは形象定数を指定しなければなりません。

英数字編集項目

PICTURE 文字ストリングには、次の記号を含めることができます。

```
A  X  9  B  0  /
```

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。

その他の文節: 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 文節では、英数字リテラルまたは形象定数を指定しなければなりません。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。

DBCS リテラルの文字の全範囲が使用できます。

関連する VALUE 文節では、DBCS リテラルまたは形象定数 SPACE を含まなければなりません。

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 文節を省略した場合には、USAGE DISPLAY-1 が暗黙に指定されます。

国別項目

PICTURE 文字ストリングには、ピクチャー記号 N の、1 つまたは複数のオカレンスを含めることができます。

上記の規定は、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 文節が指定されているときに、適用されます。USAGE NATIONAL 文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 文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。データ項目の内容は、Unicode UTF-16、CCSID 01200 で表されます。

関連する VALUE 文節では、国別リテラル、16 進表記の国別リテラル、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
- ALL 国別リテラル

他の文節

国別データ項目を定義するには、USAGE 文節で、NATIONAL 文節のみを指定することができます。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 文節を省略した場合には、USAGE NATIONAL が暗黙に指定されます。

以下の文節は、使用できます。

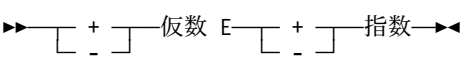
- JUSTIFIED
- EXTERNAL
- GLOBAL
- OCCURS
- REDEFINES
- RENAMES
- SYNCHRONIZED

以下の文節は、使用できません。

- BLANK WHEN ZERO
- SIGN
- DATE FORMAT

外部浮動小数点項目

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 ～ 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
指数は、記号 99 で構成されなければなりません。

その他の文節: OCCURS、REDEFINES、RENAMES、および USAGE 文節は、外部浮動小数点項目と関連付けることができます。

SIGN 文節は、説明文として受け付けられ、符号の表現に対して影響を与えません。

SYNCHRONIZED 文節は、説明文として扱われます。

次に示す文節では、外部浮動小数点項目を使用することはできません。

- BLANK WHEN ZERO
- JUSTIFIED
- VALUE

PICTURE 文節の編集

PICTURE 文節の中で編集を実行する一般的な方法には、次の 2 通りがあります。

- 挿入による編集
  - 単純挿入
  - 特別挿入
  - 固定挿入
  - 浮動挿入
- 抑止と置換による編集
  - 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、表11 に示します。

表 11. 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
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*
英数字	なし	なし
英数字編集	単純挿入	B 0 /
DBCS	単純挿入	B
外部浮動小数点	特別挿入	.

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 項目で有効です。

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。

以下に例を示します。

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,b234,b000
99,999	12345	12,345
GGBBGG	D1D2D3D4	D1D2bbbbD3D4

特別挿入による編集

このタイプの編集は、数字編集、または外部浮動小数点項目に対して有効です。

ピリオド (.) は特別な挿入記号です。それは整列を目的とした実小数点を表すこともあります。

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。

以下に例を示します。

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

このタイプの編集は、数字編集項目に対してのみ有効です。次の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。

編集記号 (PICTURE 文字ストリング内)	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)
+	+	-
-	space	-
CR	2 spaces	CR
DB	2 spaces	DB

以下に例を示します。

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
$9999.99DB	-123.45	$0123.45DB

浮動挿入による編集

このタイプの編集は、数字編集項目に対してのみ有効です。

次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、これらの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、実際にその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、実際にその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。0 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 (.) が特殊挿入記号として浮動ストリング内に組み込まれている場合、それは文字ストリングの一部とみなされます。

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

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 通りあります。したがって、次のように 2 通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 0 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 0 でなければ、その結果は規則 1 の場合と同様になります。

以下に例を示します。

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

このタイプの編集は、数字編集項目に対してのみ有効です。

ゼロ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 (.) が特殊挿入記号として浮動編集ストリング内に組み込まれている場合、それはストリングの一部とみなされます。

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 通りあり、それに応じて 2 通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。抑止は、次に示す文字のうち最も左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 0 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。

以下に例を示します。

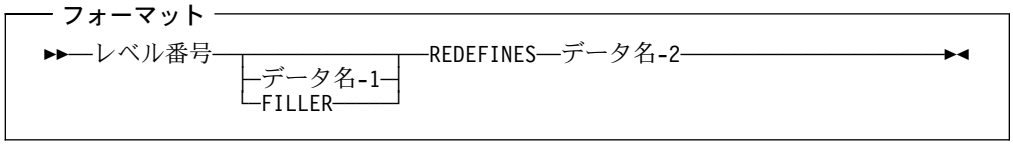
PICTURE	データの値	編集結果
****.**	0000.00	****.**
ZZZZ.ZZ	0000.00	
ZZZZ.99	0000.00	.00
****.99	0000.00	****.00
ZZ99.99	0000.00	00.00
Z,ZZZ.ZZ+	+123.456	123.45+
*,***.**+	-123.45	**123.45-
,*,***.**+	+12345678.9	12,345,678.90+
$Z,ZZZ,ZZZ.ZZCR	+12345.67	$ 12,345.67
$B*,***,***.**BBDB	-12345.67	$ ***12,345.67 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 文節の両方を指定することはできません。

REDEFINES 文節

REDEFINES 文節によって、異なるデータ記述記入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



注: レベル番号、データ名-1、および FILLER は、REDEFINES 文節自体の一部ではありません。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 文節は、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 文節は、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 とデータ名-2 のレベル番号は同一でなければならず、レベル 66 またはレベル 88 以外でなければなりません。

データ名-1、FILLER
同じ領域に関する記述でどちらか一方を選択します。これは、**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-2
これは、**再定義される**項目、すなわち **REDEFINES のオブジェクト**です。

FD 記入項目に従属するレベル-01 記入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 の記入項目が、1 番目の記入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 の記入項目には、REDEFINES 文節を指定することはできません。

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記入項目が、これら記入項目の間にあってはなりません。以下に例を示します。

```
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.
```

この例では、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。

データ名-2、すなわち再定義される項目のデータ記述記入項目には、REDEFINES 文節を含めることができます。

再定義される項目のデータ記述記入項目に OCCURS 文節を含めることはできません。ただし、再定義される項目は、データ記述記入項目に OCCURS 文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 文節の中で再定義されている項目を参照するとき、この参照には添え字付けをすることはできません。再定義される項目も再定義している項目も、またはそれらに従属している項目も、OCCURS DEPENDING ON 文節を含むことはできません。

REDEFINES 文節を含むデータ記述記入項目の中で GLOBAL 文節を使用する場合、REDEFINES 文節のサブジェクトだけが、グローバル属性を持ちます。

EXTERNAL 文節は、REDEFINES 文節と同じデータ記述記入項目上では指定することができません。

データ名-2 によって参照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るか、または 01 以外のレベル番号を使用して指定される場合は、それに含まれる文字位置の数は、この記入項目のサブジェクトによって参照されるデータ項目内の文字位置の数以上でなけれ

ばなりません。データ名-2 によって参照されるデータ名が、01 のレベル番号で指定され、かつ外部データ・レコードとして宣言されていない場合は、このような制約事項はありません。

データ項目が、ファイル記述 (FD) 記入項目の複数の 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 文節を含めることができます。

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記入項目は、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記入項目を間に介在させることはできません。複数回にわたって行われる再定義は、すべてそのストレージ域を定義した元の記入項目のデータ名を使用しなければなりません。以下に例を示します。

```
05    A                       PICTURE 9999.
05    B REDEFINES A           PICTURE 9V999.
05    C REDEFINES A           PICTURE 99V99.
```

再定義している記入項目 (データ名-1 という名前で識別されている項目)、および従属記入項目に VALUE 文節を含めることはできません。

REDEFINES 文節の考慮事項

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。以下に例を示します。

```
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。以下に例を示します。

```
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.
```

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、再定義してもどのデータも消去されず、以前の記述と取り替え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 および 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。

最初のステートメントでは、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想定します。2 番目のステートメントでは、同じ物理領域 (C として記述されている) は、Y の値

とフォーマットを想定します。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 の値に置換さ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、既存のデータが変更されることはありません。以下に例を示します。

```
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。

上記の例は、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 文節の使用例

REDEFINES 文節は、再定義されるストレージ域の範囲内にある項目 (すなわち、再定義される項目に従属する項目) に関して指定することができます。以下に例を示します。

```
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.
```

また、再定義する項目に従属している項目に関しても、REDEFINES 文節を指定することができます。以下に例を示します。

```
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.
```

予想外の結果

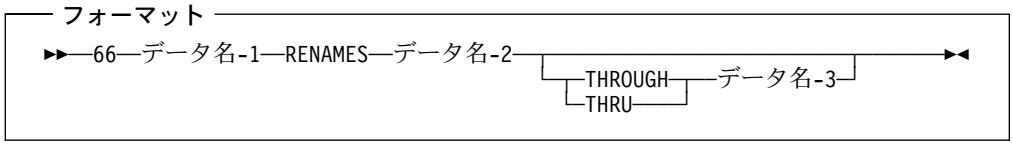
次の場合には、予想外の結果が起こることがあります。

- 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 であるときに、ステートメント MOVE B TO C が実行されたとき)。
- 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 であるときに、ステートメント MOVE C TO B が実行されたとき)。

RENAMES 文節

RENAMES 文節は、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 文節を含むデータ記述記入項目では、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 文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 文節を記述できます。1 つの論理レコードに関連付けられた RENAMES 文節は、すべてそのレコードの最後のデータ記述記入項目の直後に記述する必要があります。

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 の記入項目で、レベル-01、レベル-77、レベル-88、または別のレベル-66 の記入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。レベル標識記入項目またはレベル-01 記入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 記入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。これらの名前は両方とも修飾できます。

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記入項目についてのデータ記入項目で、OCCURS 文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 ON 文節を指定することはできません。

データ名-3 を指定した場合、データ名-1 は次に示す範囲ですべての基本項目を含むグループ項目として扱われます。

- データ名-2 (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。

キーワードの THROUGH と THRU は同じ意味です。

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。

データ名-3 を指定しない場合には、データ名-2 の全データ属性は、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 がグループ項目である場合は、データ名-1 はグループ項目として扱われます。
- データ名-2 が基本項目である場合は、データ名-1 は基本項目として扱われます。

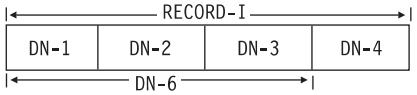
図6 に、有効な RENAMES 文節の指定と無効な指定を示します。

COBOL 仕様

例 1（有効）

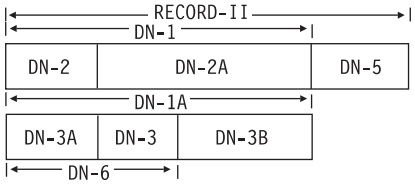
```
01   RECORD-I.
     05 DN-1...  .
     05 DN-2...  .
     05 DN-3...  .
     05 DN-4...  .
66   DN-6 RENAMES DN-1 THROUGH DN-3.
```

ストレージのレイアウト



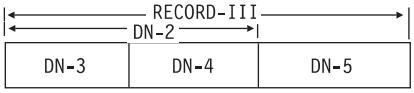
例 2（有効）

```
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 が不確定
```



例 4（無効）

```
01   RECORD-IV.
     05 DN-1.
        10 DN-2A...  .
        10 DN-2B...  .
        10 DN-2C REDEFINES DN-2B.
           15 DN-2...  .
           15 DN-2D... .
     05 DN-3...  .
66   DN-4 RENAMES DN-1 THROUGH DN-2.  DN-4 が不確定
```

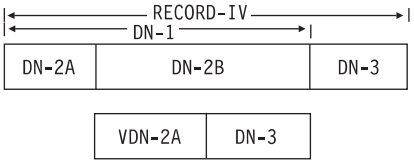
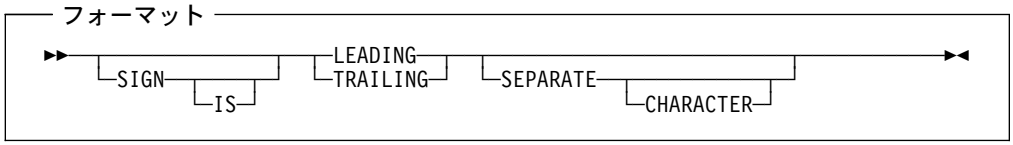


図 6. RENAMES 文節 - 有効な指定と無効な指定

SIGN 文節

SIGN 文節は、数字項目に関して、算術演算符号の表示位置やモードを指定します。

フォーマット



SIGN 文節は、数字データ記述記入項目 (すなわち、PICTURE 文字ストリングが S を含むようなデータ記述記入項目)、またはそのような基本記入項目を少なくとも 1 つは含むグループ項目に対してのみ指定できます。明示的にあるいは暗黙的に、USAGE IS DISPLAY の指定が必要です。

基本記入項目か、またはグループ記入項目 (SIGN 文節が指定されているグループ項目に従属している) のどちらかに SIGN 文節が指定されている場合は、従属記入項目の SIGN 文節がその従属記入項目よりも先にきます。

CODE-SET 文節を FD 記入項目の中で指定する場合、そのファイル記述記入項目と関連付けられた符号付き数字データ記述記入項目は、SIGN IS SEPARATE 文節と共に記述する必要があります。

SIGN 文節が必要であるのは、算術演算符号の性質や位置を明示的に記述する必要がある場合だけです。

SIGN 文節を指定すると、算術演算符号が適用される数字データ記述記入項目、またはそれが適用されるグループに従属するそれぞれの符号付き数字データ記述記入項目に関して、算術演算符号の表示位置とそのモードを定義します。

SEPARATE CHARACTER 句を指定しない場合、次のようになります。

- 算術演算符号は、基本数字データ項目の LEADING または TRAILING 桁位置 (それらのうちどちらを指定するにしても) 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 句を指定する場合は、次のようになります。

- 算術演算符号は、基本数字データ項目の LEADING または TRAILING 文字位置 (それらのうちどちらを指定するにしても) 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、正の算術演算符号として使用される文字です。
- - は、負の算術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。

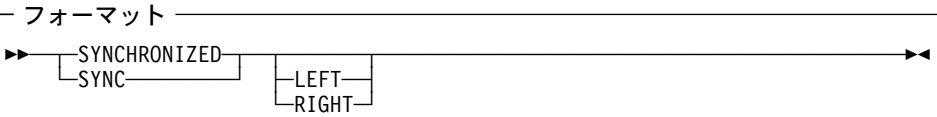
記号 S を含む PICTURE 文字列を持つ数字データ記述記入項目はいずれも、符号付きの数字データ記述記入項目です。そのような記入項目に対して SIGN 文節も指定されており、しかも計算や比較の際に変換が必要であれば、変換は自動的に行われます。

外部浮動小数点項目に対する SIGN 文節は、説明文として扱われます。内部浮動小数点項目の場合は、SIGN 文節を指定してはなりません。

SYNCHRONIZED 文節

SYNCHRONIZED 文節は、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 文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 文節は、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 文節を指定しても変化することはありません。

表12 に、SYNCHRONIZE 文節が他の言語エレメントに与える影響を示します。

表 12 (1/2). SYNCHRONIZE 文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 文節	OCCURS 文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。
DISPLAY または PACKED-DECIMAL	各項目が構文チェックされますが、 SYNCHRONIZED 文節の実行には何も影響しません。
NATIONAL	各項目が構文チェックされますが、SYNCHRONIZED 文節の実行には何も影響しません。

表 12 (2/2). SYNCHRONIZE 文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
BINARY または COMPUTATIONAL	REDEFINES 文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 文節が、従属データ項目 (レベル番号が 02 ～ 49 のデータ項目) に対して指定されたときは、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 ～ 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) ～ 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 文節が 2 進数項目に対して指定されるとき、遊びバイト用のスペースは確保されません。
POINTER、 PROCEDURE-POINTER、 FUNCTION-POINTER、 OBJECT REFERENCE	データはフルワード境界に位置合わせされます。
COMPUTATIONAL-1	データはフルワード境界に位置合わせされます。
COMPUTATIONAL-2	データはダブルワード境界に位置合わせされます。
COMPUTATIONAL-3	データは、PACKED-DECIMAL 項目の SYNCHRONIZED 文節と同様に扱われます。
COMPUTATIONAL-4	データは、COMPUTATIONAL 項目の SYNCHRONIZED 文節と同様に扱われます。
COMPUTATIONAL-5	データは、COMPUTATIONAL 項目の SYNCHRONIZED 文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 文節の実行には何も影響しません。

REDEFINES 文節	REDEFINES 文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合わせしなければなりません。たとえば、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.`

ファイル・セクションでは、コンパイラーは、SYNCHRONIZED 文節を含むレベル-01 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、正しい境界に合わせるために、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。

ワーキング・ストレージ・セクションでは、コンパイラーはすべてのレベル-01 項目をダブルワード境界に位置合わせします。

リンケージ・セクションでは、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。

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
レコード内で同期された各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それをいくつ追加すればよいか判別しなければなりません。これを計算するための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 進数項目の場合。つまり USAGE IS INDEX、USAGE IS POINTER、USAGE IS PROCEDURE-POINTER、USAGE IS OBJECT REFERENCE、USAGE IS FUNCTION-POINTER、および COMPUTATIONAL-1 データ項目)。

  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。

以下に例を示します。

```
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 文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 文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。たとえば、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図7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-HRS              PICTURE S9(3)  COMP SYNC.
    10  COMP-NAME             PICTURE X(5).
```

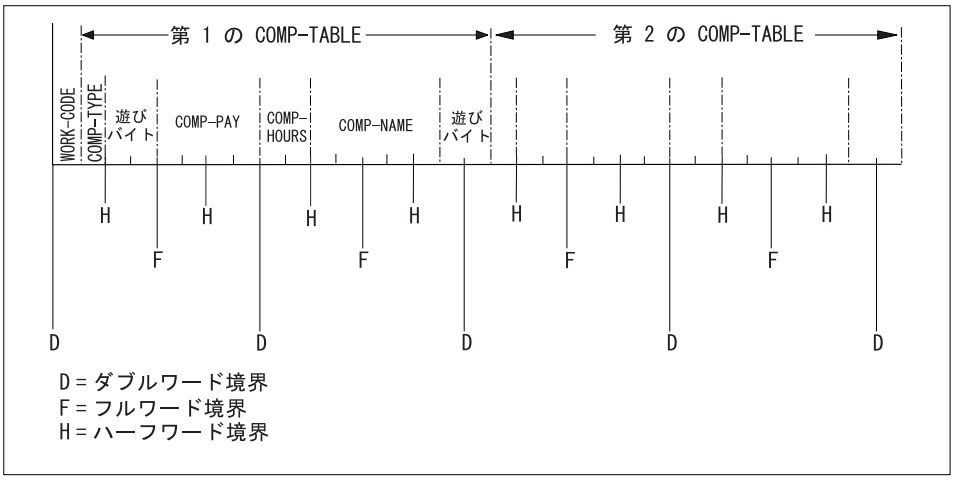


図 7. レコード内の遊びバイトの挿入

COMP-PAY と COMP-HRS 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-HRS 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-HRS               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、今度は図8 に示したように見えることになります。

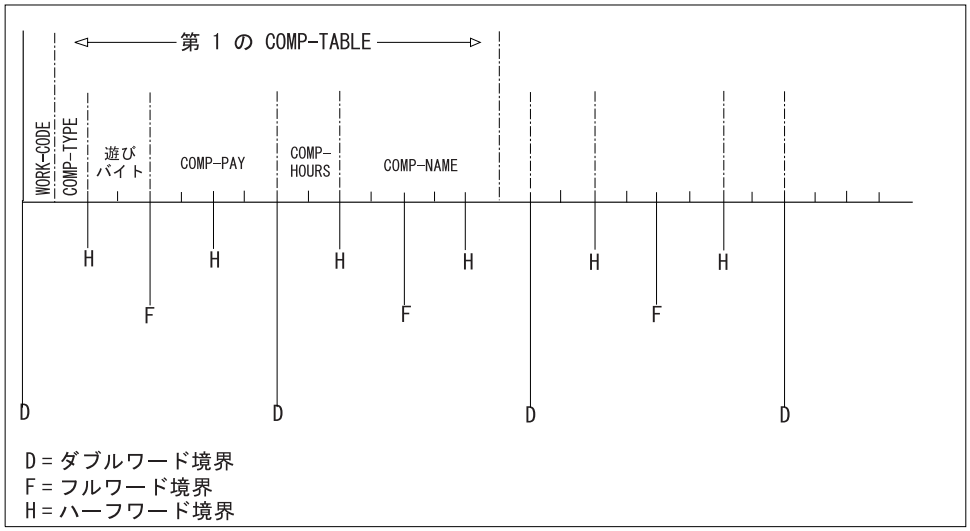


図 8. レコード間への遊びバイトの挿入

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

バッファーで処理されるブロック化論理レコードがファイルに含まれており、そのいずれかのレコードが SYNCHRONIZED 文節を指定された 2 進項目を含んでいる場合は、レコード間に遊びバイトを必要なだけ追加して適切に位置合わせすることにより、パフォーマンスを向上させることができます。

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。(可変長レコードの場合には、カウント・フィールド用にさらに 4 バイトを追加する必要があります。) 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。

レコード内およびレコード間の両方について、遊びバイトを算出する方法を明らかにするために、次のようなレコード記述について考えてみます。

```
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、これまでの説明で示した規則により、m = 4、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。

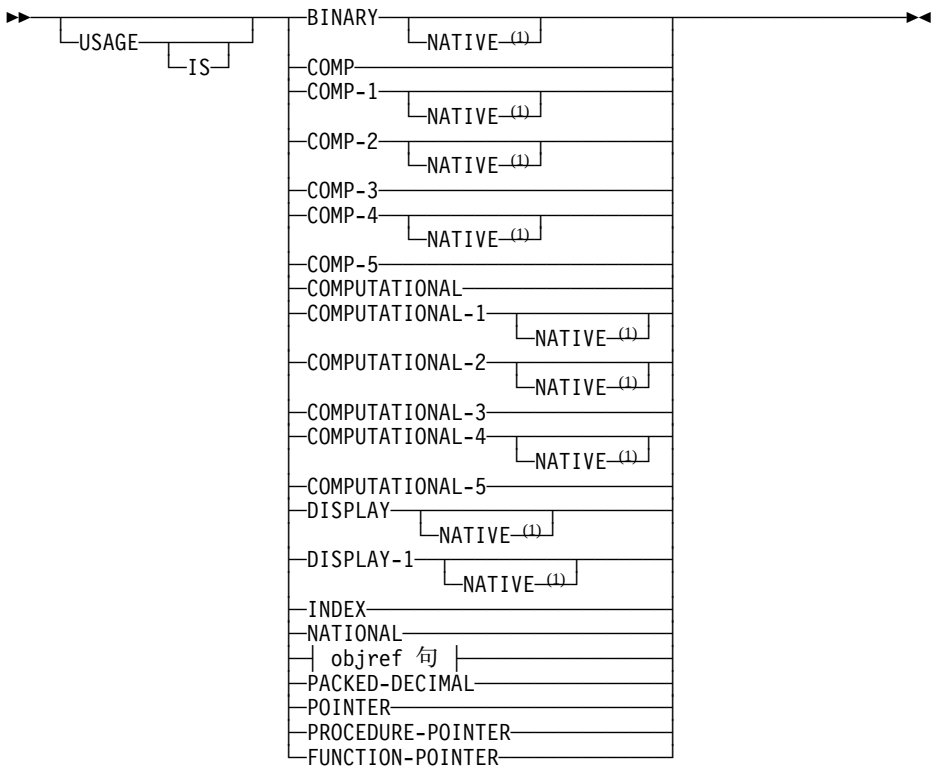
したがって、最終的なレコード記述項目は、次のようになります。

```
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]
```

USAGE 文節

USAGE 文節は、コンピューター・ストレージにおけるデータ項目のフォーマットを指定します。

フォーマット 1



注:
[1] NATIVE はコメントとして扱われます。

USAGE 文節は、66 と 88 以外のレベル番号を持つデータ記述記入項目に対して指定することができます。ただし、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この文節は適用されます。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 文節は、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。このフォーマットは、特定の手続き部ステートメントが使用されると、制約を受けることがあります。

USAGE 文節がグループ・レベルまたは基本レベルのいずれかで指定されないと、USAGE 文節には、以下の含みがあります。

- PICTURE 文節に G または N 以外の記号が含まれるときは Usage DISPLAY。
- PICTURE 文節に 1 つまたは複数の記号 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 文節に 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 文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。詳細については、167ページの『DATE FORMAT 文節と他の文節との結合』を参照してください。

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。グループ項目の USAGE がこれら計算用項目のうちいずれかを使用して記述されている場合、そのグループの中の基本項目はこの USAGE を持ちます。

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 つの算術演算符号
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 ～ 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 文節で定義された 10 進数の桁数によって異なります。

PICTURE 文節の示す桁	占有するストレージ
1 ～ 4	2 バイト (ハーフワード)
5 ～ 9	4 バイト (フルワード)
10 ～ 18	8 バイト (ダブルワード)

バイナリー・データは、「ビッグ・エンディアン」です。演算符号は、左端ビットに含まれます。

注: BINARY および TRUNC コンパイラー・オプションは、BINARY、COMPUTATIONAL、および COMPUTATIONAL-4 データ項目に影響します。これらのコンパイラー・オプションの影響については、 *Enterprise COBOL プログラミング・ガイド* 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。この種の項目は、0 ～ 9 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。 *Enterprise COBOL プログラミング・ガイド* を参照してください。

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 項目の長さは 4 バイトです。

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。

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、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。

注: 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。

Picture	ストレージ表記	数値
S9(1) ～ S9(4)	2 進数ハーフワード (2 バイト)	-32768 ～ +32767
S9(5) ～ S9(9)	2 進数フルワード (4 バイト)	-2,147,483,648 ～ +2,147,483,647
S9(10) ～ S9(18)	2 進数ダブルワード (8 バイト)	-9,223,372,036,854,775,808 ～ +9.223,372,036,854,775,807
9(1) ～ 9(4)	2 進数ハーフワード (2 バイト)	0 ～ 65535
9(5) ～ 9(9)	2 進数フルワード (4 バイト)	0 ～ 4,294,967,295
9(10) ～ 9(18)	2 進数ダブルワード (8 バイト)	0 ～ 18,446,744,073,709,551,615

COMP-5 データ項目の 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。たとえば、PICTURE S99V99 COMP-5 の記述があるデータ項目は、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 ～ +327.67 の範囲の値をサポートします。

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
- 数字編集
- 外部浮動小数点
- 外部 10 進数 (数字)

英字、**英数字**、**英数字編集**、および**数字編集**の各項目については、183ページの『データ・カテゴリーと PICTURE の規則』で説明しています。

外部 10 進数項目は、**ゾーン 10 進数**項目とも呼ばれます。数字の各桁は、1 バイトで表されます。各バイトの高位 4 ビットはゾーン・ビットです。最低位バイトの高位 4 ビットは項目の符号を表します。各バイトの低位 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 文字ストリングで使用できる値は、9、演算符号 S、想定小数点 V、および 1 つまたはそれ以上の P だけです。

DISPLAY-1 句

DISPLAY-1 句は、項目を DBCS として定義します。

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、プロシージャーの入り口点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 バイトの基本項目です。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が、長さは、8 バイトではなく、4 バイトです。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。

関数ポインターには、以下のアドレスを含めることができます。

- 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

または、NULL を含めることができます。

関数ポインター・データ項目の VALUE 文節には、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、213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 (必ずしもテーブルと結び付いている必要はありません) で、今後の参照に備えて指標名の値を保持しておくために使います。SET ステートメントを使用して、指標データ項目に指標名の値 (テーブル中のオカレンス項目数の値) を割り当てることができます。

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、手続き部のヘッダーの 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グループ項目の一部であることができます。

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持しますが、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。したがって、SEARCH ステートメントまたは SET ステートメントの中で直接に参照されたり、MOVE ステートメントや入出力ステートメントの中で間接的に参照されても、それらステートメントの実行時に値の変換は起こりません。

USAGE IS INDEX 文節は、どのようなレベルででも記述できます。USAGE IS INDEX 文節を使用してグループ項目を記述すると、そのグループ中の基本項目は指標データ項目になります。ただし、グループ自体は指標データ項目ではないので、SEARCH ステートメントや SET ステートメントまたは比較条件にそのグループ名を使用することはできません。基本項目の USAGE 文節は、その基本項目が属するグループの USAGE 文節と矛盾することはできません。

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、または VALUE の文節は、USAGE IS INDEX 文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の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、項目を**国別データ項目**として定義します。データ項目には、国別のクラスとカテゴリーがあります。

Usage NATIONAL のデータ項目は、ストレージにおいて、ビッグ・エンディアン・フォーマットの Unicode UTF-16 (CCSID 01200) で表されます。

国別データ項目に関連付けられている PICTURE 文節には、1 つまたは複数のピクチャー記号 N のみを含めることができます。

OBJECT REFERENCE 句

OBJECT REFERENCE 句を使用して定義されたデータ項目を**オブジェクト・リファレンス**といいます。

クラス名-1
オプションのクラス名。

クラス名-1 は、そのクラスまたは最外部プログラムの構成セクション中の REPOSITORY 段落で宣言しなければなりません。

クラス名-1 を指定すると、データ名が常にクラス名-1 のクラスまたはクラス名-1 から導出されたクラスのオブジェクト・インスタンスを参照するよう指示されます。

注: プログラマーは、参照されるオブジェクトがこの要件に適合することを確認する必要があります。違反は診断されません。

クラス名-1 を指定しないと、データ名はどのクラスのオブジェクトでも参照できます。この場合、データ名-1 は「汎用」オブジェクト・リファレンスです。

グループ項目のセマンティクスに影響を与えずに、そのグループ項目中のデータ名-1 を指定できます。グループに影響を与えるステートメントが実行される際に、値または他のオブジェクト・リファレンスの特殊処理は変換されません。グループは引き続き英数字データ項目として使用されます。

USAGE IS OBJECT REFERENCE 文節は、レベル 66 または 88 以外のどのレベルでも使用できます。グループ項目が、USAGE IS OBJECT REFERENCE 文節を使用して記述されている場合、そのグループ内の基本項目は、オブジェクト・リファレンス・データ項目です。グループ自体はオブジェクト・リファレンスではありません。基本項目の USAGE 文節は、その基本項目を含むグループの USAGE 文節と矛盾していてはなりません。

オブジェクト・リファレンスは、ファクトリー定義、オブジェクト定義、メソッド、またはプログラムのデータ部のどのセクションでも定義できます。オブジェクト・リファレンス・データ項目を使用できるのは、次のものだけです。

- SET ステートメントの中 (フォーマット 7 の場合のみ)
- 比較条件
- INVOKE ステートメント
- INVOKE ステートメント中の USING または RETURNING 句
- CALL ステートメント中の USING または RETURNING 句
- プログラム手続き部または ENTRY ステートメントの USING または RETURNING 句
- メソッド手続き部の USING または RETURNING 句

オブジェクト・リファレンス・データ項目には次のことが当てはまります。

- CORRESPONDING 演算中は無視されます。
- INITIALIZE ステートメントの影響を受けません。
- REDEFINES 文節のサブジェクトまたはオブジェクトになることができます。
- 条件変数にはなれません。
- ファイルに書き込めます (しかし、その後でレコードを読み取る際にはオブジェクト・リファレンスの内容は未定義になります)。

オブジェクト・リファレンス・データ項目の VALUE 文節には、NULL または NULLS だけを指定できます。

SYNCHRONIZED 文節と USAGE IS OBJECT REFERENCE を一緒に使用すると、オブジェクト・リファレンス・データ項目の有効な位置合わせを行うことができます。

DATE FORMAT、JUSTIFIED、PICTURE、および BLANK WHEN ZERO の文節は、USAGE IS OBJECT REFERENCE 文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のに使用することはできません。

POINTER 句

USAGE IS POINTER を使用して定義されたデータ項目を**ポインター・データ項目**といいます。ポインター・データ項目は 4 バイトの基本項目です。

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したり、他のポインター項目に移動したりすることが可能です。

ポインター・データ項目は、次の場合にのみ使用できます。

- SET ステートメントの中 (フォーマット 5 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、または手続き部のヘッダーの中

USAGE IS POINTER 文節は、レベル 88 以外であればどのレベルででも記述することができます。USAGE IS POINTER 文節を使用してグループ項目を記述すると、そのグループ中の基本項目はポインター・データ項目になります。ただしグループ自体はポインター・データ項目ではないので、ポインター・データ項目を使用できる構文でそのグループ名を使用することはできません。基本項目の USAGE 文節は、その基本項目が属するグループの USAGE 文節と矛盾することはできません。

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であることができます。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 文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 文節には、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。

DATE FORMAT、JUSTIFIED、PICTURE、および BLANK WHEN ZERO の文節は、USAGE IS POINTER 文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のに使用することはできません。

ポインター・データ項目は、CORRESPONDING 演算では無視されます。

ポインター・データ項目を、データ・セット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。

注: 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。詳細については、 *Enterprise COBOL プログラミング・ガイド* 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・データ項目は、8 バイト基本項目です。

プロシージャー・ポインターには、以下のアドレスを含めることができます。

- 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。

または、NULL を含めることができます。

プロシージャー・ポインター・データ項目は、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 句または手続き部のヘッダーの中

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。

USAGE IS PROCEDURE-POINTER 文節は、レベル 88 以外であればどのレベルででも記述することができます。USAGE IS PROCEDURE-POINTER 文節を使用してグループ項目を記述すると、そのグループ中の基本項目はプロシージャー・ポインター・データ項目になります。ただしグループ自体はプロシージャー・ポインターではないので、プロシージャー・ポインター・データ項目を使用できる構文でそのグループ名を使用することはできません。基本項目の USAGE 文節は、その基本項目が属するグループの USAGE 文節と矛盾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、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・データ項目をデータ・セット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 文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。

GLOBAL、EXTERNAL、および OCCURS の各文節は、USAGE IS PROCEDURE-POINTER と共に使用することができます。

プロシージャー・ポインター・データ項目に対する VALUE 文節は、NULL または NULLS のみ指定できます。

DATE FORMAT、JUSTIFIED、PICTURE、および BLANK WHEN ZERO の文節は、USAGE IS PROCEDURE-POINTER 文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の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 演算の中では無視されます。

NATIVE 句

NATIVE 句はコメントとして扱われます。

VALUE 文節

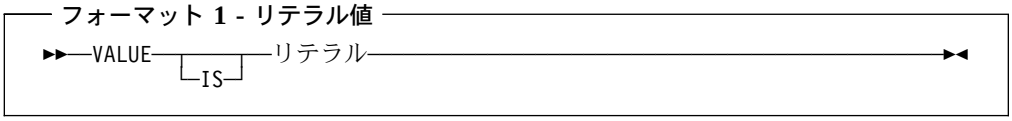
VALUE 文節は、データ項目の初期内容または条件名と関連付けられる値を指定します。VALUE 文節の用途は、それがデータ部のどのセクションで指定されているかに応じて異なります。

クラスのワーキング・ストレージ・セクションでは、VALUE 文節は条件名項目だけで使用できます。

ファイル・セクションおよびリンケージ・セクションにおいて、条件名項目以外の項目で VALUE 文節を使用すると、その VALUE 文節はコメントとして扱われます。

ワーキング・ストレージ・セクションでは、VALUE 文節は、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、その内容は未確定です。

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 文節や JUSTIFIED 文節が指定されていても、それらとは無関係です。

OCCURS 文節を含むか、または OCCURS 文節に従属しているデータ記述記入項目の中に指定されているフォーマット 1 の VALUE 文節は、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 文節の 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数を持っているものとみなされます。

VALUE 文節は、EXTERNAL 文節かまたは REDEFINES 文節のいずれかを持つ項目を含むデータ記述記入項目か、またはそれに従属しているデータ記述記入項目に対しては、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。

VALUE 文節をグループ・レベルで指定する場合には、そのリテラルは、英数字リテラルまたは形象定数でなければなりません。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設定されます。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、項目に次のいずれかの文節があると、VALUE 文節を指定できません。該当する文節は JUSTIFIED、SYNCHRONIZED、または USAGE (USAGE DISPLAY 以外) です。

VALUE 文節は、データ記述記入項目の中にある他の文節や、この項目の階層のデータ記述の中にある他の文節と矛盾するものであってはなりません。

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 文節では、すべて浮動小数点リテラルを指定する必要があります。条件名 VALUE 句も、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 文節、または条件名 VALUE 句の中で指定できます。

浮動小数点リテラル値については、29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。

VALUE 文節を、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。

DBCS 項目と関連付けられた VALUE 文節は、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。

国別リテラルを指定する VALUE 文節は、国別データ項目にのみ関連付けることができます。

国別データ項目に関連付けられている VALUE 文節は、国別リテラル、または形象定数 ZERO、SPACE、QUOTES、または ALL 国別リテラルのいずれか 1 つを指定しなければなりません。

先行するデータ項目が DEPENDING ON 句を持つ OCCURS 文節を含む場合には、それに続くデータ項目は VALUE 文節を含むことはできません。

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、9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。
- 項目が数字である場合には、VALUE 文節のリテラルは、すべて数字でなければなりません。リテラルがワーキング・ストレージの項目の値を定義している場合は、そのリテラルは数字の移動の規則に従って位置合わせされますが、1 つだけ制約があります。それは、0 以外の数字を切り捨てる必要のある値をリテラルに定義できないことです。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。
- いくつかの例外はありますが、VALUE 文節内の数値リテラルはその項目の PICTURE 文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。たとえば、PICTURE 99PPP である場合、リテラルは 1000 ～ 99000 の範囲内であるか、または 0 でなければなりません。PICTURE PPP99 である場合、リテラルは 0.00000 ～ 0.00099 の範囲内でなければなりません。

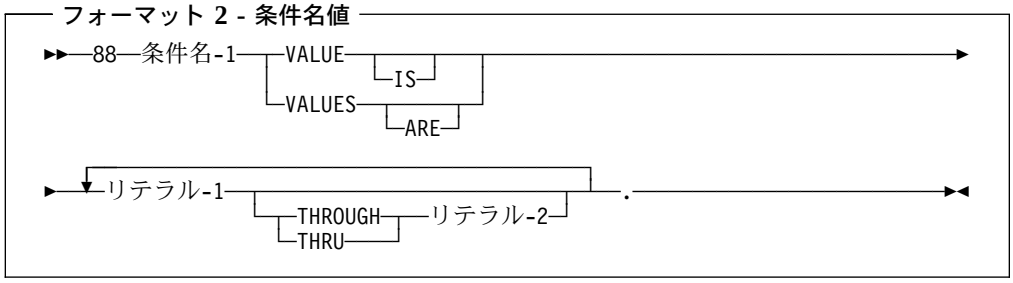
  例外を以下に示します。

  - Usage COMP-5 を使用して記述したデータ項目には、その PICTURE 文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 文節にピクチャー記号 P はありません。

    上記の項目の VALUE 文節は、本来のバイナリー表現の容量と同じ大きさまでの値を持つことができます。

- 項目が、基本項目またはグループ項目のうちで、英字、英数字、英数字編集、または数字編集のいずれかの項目である場合、VALUE 文節のリテラルはすべて、英数字のリテラル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。
- PICTURE 文節内の編集文字の機能は、記述された項目が最初に現れた時の判別では、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。たとえば、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 文節は、VALUE "+012.34" と指定する必要があります。

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 文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。

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 (235ページの『条件式』を参照)。

リテラル-1
リテラル-1 だけを指定する場合、条件名は、単一の値と関係付けられます。

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
この場合、条件名は少なくとも値の 1 つの範囲と関係付けられます。THROUGH 句を使用する場合、関連データ項目が年末尾型ウィンドウ化日付フィールドである場合を除いて、リテラル-1 は常にリテラル-2 よりも小さくなければなりません。詳細については、『条件名記入項目の規則』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。

リテラルが DBCS である場合、THROUGH 句によって指定される DBCS 値の範囲は、DBCS 文字の 16 進値の 2 進照合シーケンスに基づいています。

| リテラルが国別である場合、THROUGH 句によって指定される国別文字値の範囲は、そのリテラルによって表される国別文字の 16 進値の 2 進照合シーケンスに基づいています。

関連付けられている条件変数が DBCS データ項目である場合には、フォーマット 2 VALUE 文節で指定されている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 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国別データ項目である場合には、フォーマット 2 VALUE 文節で指定されている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。形象定数 ZERO、SPACE、QUOTE、および ALL 国別リテラルを指定することができます。

条件名記入項目の規則

- VALUE 文節は、条件名記入項目内で必須であると同時に、その記入項目内の唯一の文節でなければなりません。各条件名記入項目は、先行の条件変数と関連付けられます。したがって、レベル-88 記入項目の前には必ず、条件変数に関する記入項目、または別のレベル-88 記入項目 (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 記入項目は、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、区切り文字のコンマ、または区切り文字のセミコロンが必要です。

各記入項目は、区切り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、その条件変数の記入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記入項目にすることができます。
  - 別の条件名。
  - RENAMES 文節 (レベル-66 項目)。
  - USAGE IS INDEX で記述された項目。
  - USAGE IS POINTER、USAGE IS PROCEDURE-POINTER、USAGE IS FUNCTION-POINTER、または USAGE IS OBJECT REFERENCE で記述された項目。
- 条件名は、グループ項目のデータ記述記入項目と関連付けることができます。この場合、次のような条件があります。
  - 条件名の値は、英数字リテラルまたは形象定数として指定しなければなりません。
  - 条件名の値のサイズは、グループ内のすべての基本項目のサイズの合計を超えることはできません。
  - グループ内のどのエレメントにも、JUSTIFIED 文節または SYNCHRONIZED 文節を含めることはできません。

  グループには、任意の Usage の項目を含めることができます。

  条件名は、グループ・レベルとグループ内の従属レベルの両方で指定することができます。

  グループ・レベルで条件名を定義した場合、それが必然的に意味する比較テストは、そのグループ内の基本項目の性質とは無関係に、英数字オペランドの比較の規則に従って行われます。
- DBCS データ項目に対する比較テストは、DBCS 項目の比較の規則に従って実行されます。これらの規則に関する説明は、249ページの『DBCS オペランドの比較』を参照してください。
- 国別データ項目に対する比較テストは、国別オペランドの比較の規則に従って実行されます。これらの規則に関する説明は、249ページの『国別オペランドの比較』を参照してください。
- 国別リテラルを指定する VALUE 文節は、国別データ項目に対してのみ定義され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 文節は、DBCS データ項目に対してのみ定義される条件名に関連付けることができます。
- 条件名記入項目内のリテラルのタイプは、その条件変数のデータ型と矛盾しないようにし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。

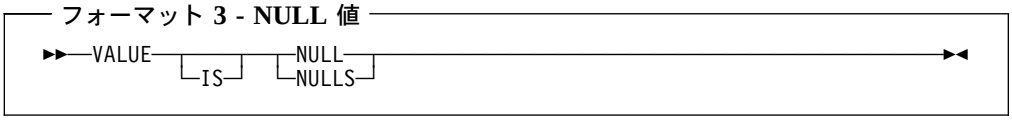
```
05  CITY-COUNTY-INFO.
    88  BRONX                  VALUE "03NYC".
    88  BROOKLYN               VALUE "24NYC".
    88  MANHATTAN              VALUE "31NYC".
    88  QUEENS                 VALUE "41NYC".
    88  STATEN-ISLAND          VALUE "43NYC".
  10  COUNTY-NO                PICTURE 99.
      88  DUTCHESS             VALUE 14.
      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...
```

- 項目がウィンドウ化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 とリテラル-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。

ウィンドウ化日付フィールドで条件名を使用する方法については、239ページの『条件名条件とウィンドウ化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

フォーマット 3 - NULL 値



このフォーマットは、USAGE IS POINTER、USAGE IS PROCEDURE POINTER、またはUSAGE IS 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を割り当てます。また、無効なオブジェクト・リファレンスを、USAGE IS OBJECT REFERENCE と定義されている項目の初期値として割り当てます。

VALUE IS NULL を指定できるのは、暗黙的または明示的に USAGE IS POINTER、USAGE IS PROCEDURE-POINTER、USAGE IS FUNCTION-POINTER、または USAGE IS OBJECT REFERENCE として定義されている基本項目だけです。

手続き部

手続き部の構造 223
メソッド手続き部の要件 223
手続き部のヘッダー 224
宣言部分 227
プロシージャー 228
算術式 230
条件式 235
ステートメントのカテゴリー 257
ステートメントによる操作 261

手続き部のステートメント 274
ACCEPT ステートメント 274
ADD ステートメント 278
ALTER ステートメント 281
CALL ステートメント 283
CANCEL ステートメント 290
CLOSE ステートメント 292
COMPUTE ステートメント 296
CONTINUE ステートメント 298
DELETE ステートメント 299
DISPLAY ステートメント 301
DIVIDE ステートメント 304
ENTRY ステートメント 307
EVALUATE ステートメント 308
EXIT ステートメント 312
EXIT METHOD ステートメント 313
EXIT PROGRAM ステートメント 314
GOBACK ステートメント 315
GO TO ステートメント 316
IF ステートメント 318
INITIALIZE ステートメント 320
INSPECT ステートメント 323
INVOKE ステートメント 332
MERGE ステートメント 340
MOVE ステートメント 346
MULTIPLY ステートメント 352
OPEN ステートメント 354
PERFORM ステートメント 360
READ ステートメント 370
RELEASE ステートメント 377
RETURN ステートメント 379
REWRITE ステートメント 381
SEARCH ステートメント 384
SET ステートメント 391
SORT ステートメント 398

© Copyright IBM Corp. 1991, 2001

START ステートメント 405
STOP ステートメント 408
STRING ステートメント 409
SUBTRACT ステートメント 414
UNSTRING ステートメント 417
WRITE ステートメント 425
XML PARSE ステートメント 433

手続き部の構造

手続き部はオプションの部です。

プログラム手続き部
手続き部は、オプションの宣言部分と、セクション、段落、文、ステートメントのいずれかまたはすべてを含むプロシージャーで構成されています。

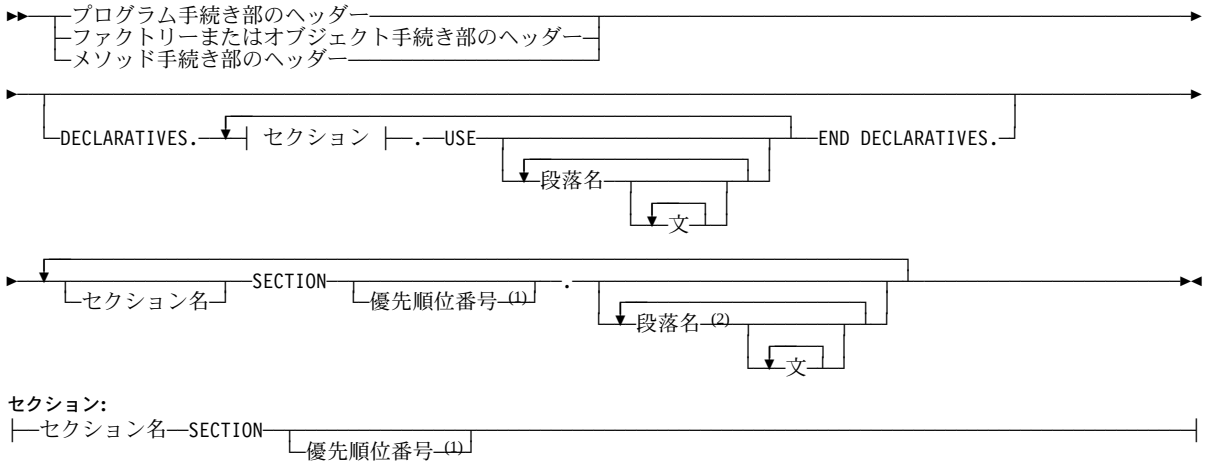
ファクトリー手続き部
ファクトリー手続き部には、ファクトリー・メソッド定義だけが含まれます。

オブジェクト手続き部
オブジェクト手続き部には、オブジェクト・メソッド定義だけが含まれます。

メソッド手続き部
メソッド手続き部は、オプションの宣言部分と、セクション、段落、文、ステートメントのいずれかまたはすべてを含むプロシージャーで構成されています。メソッドは他のメソッドの INVOKE を実行したり、再帰的に INVOKE を実行したり、プログラムに CALL を発行したりできます。メソッド手続き部に、ネストされたプログラムかメソッドを入れることはできません。

メソッド手続き部に関する詳細は、『メソッド手続き部の要件』を参照してください。

フォーマット - 手続き部



セクション:

注:

1 優先順位番号は、メソッド、再帰的プログラム、または THREAD オプションを使用してコンパイルされたプログラムにおいては無効です。

2 セクション名を省略する場合は、段落名も省略できます。

メソッド手続き部の要件

メソッド手続き部を使う場合は、次の点を知っている必要があります。

- EXIT METHOD ステートメントまたは GOBACK ステートメントを使用して、呼び出しメソッドまたはプログラムに制御権を戻すことができます。個々のメソッド手続き部の最後のステートメントとして、暗黙 EXIT METHOD ステートメントが生成されます。

© Copyright IBM Corp. 1991, 2001

EXIT METHOD ステートメントに関する詳細は、313ページの『EXIT METHOD ステートメント』を参照してください。

- メソッド中で STOP RUN ステートメント (実行単位を終了する) を使用できます。
- メソッド手続き部で RETURN-CODE 特殊レジスターを使用して、呼び出されたサブプログラムからの戻りコードにアクセスできます。しかし、現行メソッドの呼び出し側に RETURN-CODE 値は戻されません。現行メソッドの呼び出し側に値を戻すには、手続き部の RETURNING データ名を使用してください。詳細については、『手続き部のヘッダー』 の RETURNING データ名-2 に関する説明を参照してください。

メソッド手続き部に以下のステートメント、または文節を指定することはできません。

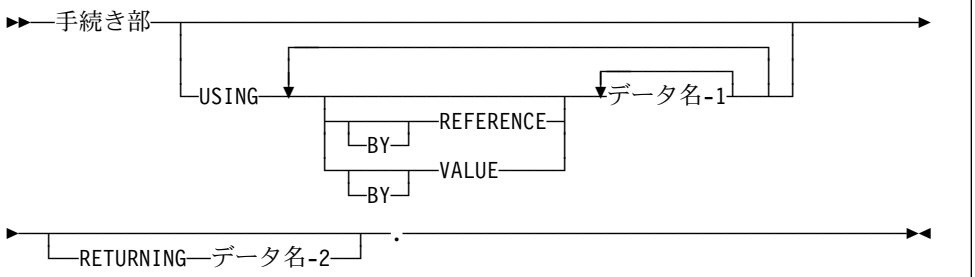
- ALTER
- ENTRY
- EXIT PROGRAM
- 指定されたプロシージャー名のない GO TO
- SEGMENT-LIMIT
- USE FOR DEBUGGING

手続き部のヘッダー

手続き部を指定すると、その手続き部は、以下のいずれかのヘッダーで識別されます。どのヘッダーで識別されるかは、プログラム、ファクトリー定義、オブジェクト定義、またはメソッド定義のいずれが指定されているかによって決まります。

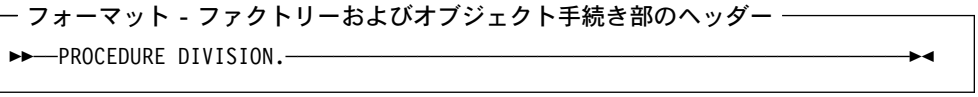
プログラムの手続き部のヘッダーには、以下のフォーマットが使用されます。

フォーマット - プログラム手続き部のヘッダー



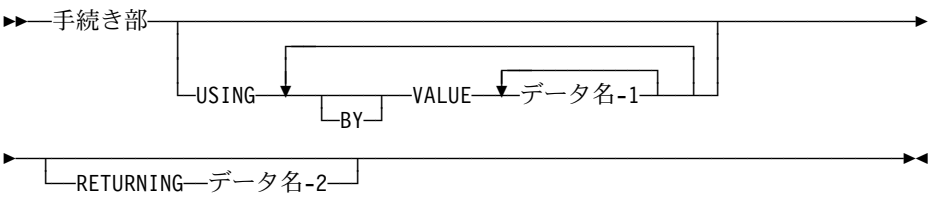
FACTORY 段落または OBJECT 段落の手続き部のヘッダーには、以下のフォーマットが使用されます。

フォーマット - ファクトリーおよびオブジェクト手続き部のヘッダー



メソッドの手続き部のヘッダーには、以下のフォーマットが使用されます。

フォーマット — メソッド手続き部のヘッダー



USING

USING 句は、プログラムまたはメソッドの呼び出し時にプログラムまたはメソッドが受け取るパラメーターを指定します。

USING 句は、非宣言部分の先頭に入力された呼び出されるサブプログラムの手続き部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またはメソッドのリンケージ・セクションで、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 プログラムまたはメソッドのデータ部では、任意のレベルのデータ項目にすることができます。INVOKE ステートメントの USING 句で指定されたデータ項目は、呼び出し側の COBOL プログラムまたはメソッドのデータ部では、任意のレベルのデータ項目にすることができます。

注: 手続き部ヘッダーの USING 句の中のデータ項目のデータ記述記入項目の中には、REDEFINES 文節を指定できます。

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、次の条件を満たせば ENTRY ステートメントの中で有効です。それは、呼び出されるサブプログラムまたは呼び出されるメソッドのリンケージ・セクションで、個々の USING ID がレベル 01 またはレベル 77 の項目として定義されなければならないことです。

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。 COBOL メソッドは、Java または COBOL からしか呼び出すことができません。

呼び出す側と呼び出される側のサブプログラム、または、呼び出す側のメソッド / プログラムと呼び出される側のメソッド内で 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。したがって、呼び出しプログラムと呼び出されるプログラム、または呼び出しメソッド / プログラムと呼び出されるメソッドの中の指標名は、常にそれぞれ個別の指標を参照します。

CALL USING または INVOKE USING ステートメントで指定される ID は、呼び出しプログラム、または呼び出しメソッド / プログラムで使用可能で、かつ呼び出されるプログラムまたは呼び出されるメソッドの論理的終わりを指定します。これらの項目は、どのデータ部セクションにも定義できます。

手続き部の USING 句には、特定の ID を複数回指定できます。CALLまたは INVOKE 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 *(プログラムのみ)*
BY CONTENT または BY REFERENCE によって引き数を渡す場合は、PROCEDURE/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または呼び出されるメソッドの手続き部の 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 のリンケージ・セクション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、手続き部ヘッダーの 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
メソッド手続き部ヘッダーの USING 句に指定するパラメーターは、メソッド BY VALUE に渡す必要があります。

BY VALUE により引き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、引き数の値が渡されます。受け取り側のサブプログラムまたはメソッドは、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 により渡された引き数に対応する仮パラメーターを変更しても、その引き数には影響がありません。

Enterprise COBOL プログラミング・ガイド では、こうした概念の例を具体的に説明しています。

データ名-1
リンケージ・セクションでは、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。

メソッド手続き部のヘッダーでデータ名-1 がオブジェクト・リファレンスである場合、そのオブジェクト・リファレンスのデータ記述記入項目には、クラス名を明示的に指定する必要があります。つまり、データ名-1 は、ユニバーサル・オブジェクト・リファレンスであってはなりません。

メソッドの場合、パラメーター・データ型は、COBOL と Java との間で相互に運用が可能なデータ型しか使用できません。詳しくは、336ページの『COBOL と Java の相互運用可能なデータ型』を参照してください。

RETURNING 句
RETURNING 句は、プログラムまたはメソッドを実行した結果戻されるデータ項目を指定します。

データ名-2

データ名-2 は、RETURNING データ項目です。リンケージ・セクションでは、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。

メソッド手続き部のヘッダーでは、データ名-2 のデータ型を Java 相互協調処理に対応したデータ型にする必要があります。詳しくは、336ページの『COBOL と Java の相互運用可能なデータ型』を参照してください。

RETURNING データ項目は、出力専用のパラメーターです。メソッドへの入力時、RETURNING データ項目の初期状態の値は定まっておらず、予測できません。RETURNING データ項目の値を参照するには、PROCEDURE DIVISION RETURNING データ項目を初期化する必要があります。メソッドが戻されるときに、呼び出し側に渡される値は、PROCEDURE DIVISION RETURNING データ項目の最終的な値です。INVOKE RETURNING ID およびメソッド RETURNING のデータ項目へ準拠するための要件については、334ページの『RETURNING 句』を参照してください。

次のプログラムには、手続き部 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手続き部 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だけに手続き部 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。

リンケージ・セクションの項目への参照

呼び出されるプログラムまたはメソッドのリンケージ・セクションで定義されたデータ項目は、これらが次の条件のいずれかを満たす場合にのみ、そのプログラムの手続き部の中で参照することができます。

- それらのデータ項目が ENTRY ステートメント、または手続き部のヘッダーの USING 句のオペランドである場合
- それらのデータ項目が SET ADDRESS OF、CALL...BY REFERENCE ADDRESS OF、または INVOKE...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 文節または RENAMES 文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

宣言部分には、例外条件が起こったときに実行される 1 つまたは複数の特定の目的を持った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手続き部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す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。USE ステートメントについての詳細は、529ページの『USE ステートメント』 を参照してください。

宣言セクション・グループの前には、キーワード DECLARATIVES を指定します。DECLARATIVES は、手続き部のヘッダーの後の行に指定します。また、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、後に区切り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。

手続き部の宣言部分では、各セクション・ヘッダーは後に区切りピリオドを付け、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区切り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 ステートメントには以下の 3 つのフォーマットがあります。

1. EXCEPTION 宣言 (529ページの『USE ステートメント』を参照)
2. DEBUGGING 宣言 (529ページの『USE ステートメント』を参照)
3. LABEL 宣言 (529ページの『USE ステートメント』を参照)

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は戻されます。

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。

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

手続き部では、1 つの**プロシージャー**は次のものから構成されています。

- 1 つの**セクション**またはセクションのグループ
- 1 つの**段落**または段落のグループ

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

セクションのヘッダーの後ろには、必要に応じて、1 つまたは複数の段落を続けます。

セクションのヘッダー

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)、および区切り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 ～ 99 の範囲の値。

宣言部分のセクションの優先順位番号は、0 ～ 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。

- メソッド定義内
- RECURSIVE 属性を指定して宣言されているプログラム内
- THREAD コンパイラー・オプションを指定してコンパイルしたプログラム内

1 つのセクションは、次のセクションのヘッダーの直前で終了するか、手続き部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。

段落

段落名の後ろには、区切りピリオドを続けます。必要に応じて、1 つまたは複数の文をさらに続ける場合もあります。

注: 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 ID 部のヘッダー、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)、手続き部の中に段落名は不要です。

段落は次の段落名またはセクションのヘッダーの直前で終了するか、手続き部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 つまたは複数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 つまたは複数の**ステートメント**からなり、区切りピリオドで終わります。

ステートメント
COBOL 動詞で始まる記号 (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。手続き部の参照では (クラス・テストの場合を除き)、ID の内容は、PICTURE 文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。さもなければ、結果はどのようになるか分かりません。

実行は、宣言部分を除く手続き部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。

手続き部の終わりは、次のいずれかによって指示されます。

- 見出し部のヘッダー。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 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、次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、および (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、および (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。たとえば、4 の平方根の場合を考えてみましょう。

```
4 ** 0.5 (the square root of 4)
```

この場合、+2 と -2 が計算結果となります。Enterprise COBOL は必ず +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、2 つの単項算術演算子 (表13) を算術式で使用することができます。これらの算術演算子は、その前後にスペースを伴う特定の文字で表されます。

表 13. 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		

注: 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。この場合、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。

算術式で括弧を使用すると、エレメントを評価する順序を指定できます。

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を使用した式の場合には、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順位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。

表14 は、対にできる算術記号を示したものです。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 14. 算術記号の有効な対

	2 番目の記号				
最初の記号	ID または リテラル	* / ** + -	単項 + または 単項 -	(	)
ID またはリテラル	いいえ	はい	いいえ	いいえ	はい
* / ** + -	はい	いいえ	はい	はい	いいえ
単項 + または単項 -	はい	いいえ	いいえ	はい	いいえ
(	はい	いいえ	はい	はい	いいえ
)	いいえ	はい	いいえ	いいえ	はい

日付フィールドでの算術演算

日付フィールドを含む算術演算は、以下のものに限定されています。

- 非日付データと日付フィールドの加算
- 日付フィールドからの非日付データの減算
- 互換日付フィールドからの日付フィールドの減算

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化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 -
- 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、以下のページに示します。

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。

- 278ページの『ADD ステートメント』

- 296ページの『COMPUTE ステートメント』
- 414ページの『SUBTRACT ステートメント』

注:

1. 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。たとえば、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化日付フィールドに 1 を加算すると、値は 000101 ではなく 991232 になってしまいます。
2. ウィンドウ化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。DATEPROC(TRIG) コンパイラー・オプションが有効である場合、この拡張ではウィンドウ化日付フィールドのトリガー値が考慮されます。正規ウィンドウ化とトリガーを考慮したウィンドウ化の拡張については、165ページの『ウィンドウ化日付フィールドのセマンティクス』を参照してください。

日付フィールドの関連する加算

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 15. 加算で日付フィールドを使用した場合の結果

	第 2 オペランド	
第 1 オペランド	**非日付データ**	**日付フィールド**
非日付データ	非日付データ	日付フィールド
日付フィールド	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、234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。

日付フィールドの関連する減算

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 16. 減算で日付フィールドを使用した場合の結果

	第 2 オペランド	
第 1 オペランド	**非日付データ**	**日付フィールド**
非日付データ	非日付データ	許可されない
日付フィールド	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

以下のステートメントは、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 つまたは複数の受け取りフィールドに保管します。

ADD
COMPUTE
DIVIDE
MULTIPLY
SUBTRACT

注: MULTIPLY ステートメントでは、GIVING ID だけに日付フィールドを使用することができます。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化日付フィールドは、165ページの『ウィンドウ化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、ウィンドウ化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 文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 (ウィンドウ化日付フィールド・オペランド、または結果の自動拡張後) の両方が非日付データであるかのように進められます。

ON SIZE ERROR 文節が指定された場合は、235ページの表17 に、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)。

235ページの表17 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。

ウィンドウ化なし
ステートメントは、263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。

ウィンドウ化...

...非日付データ送り出しフィールドの場合
非日付データ送り出しフィールドは、ウィンドウ化日付受け取りフィールドと互換性のあるウィンドウ化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 であるウィンドウ化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。

...日付送り出しフィールドの場合
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化日付フィールド・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、高位または左端の 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。

たとえば、

```
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)。

サイズ・エラー処理とトリガー値: DATEPROC(TRIG) コンパイラー・オプションが有効であり、送り出しフィールドの内容がトリガー値 (0 または全桁 9) である場合、サイズ・エラー命令ステートメントが実行され、結果は受け取りフィールドに保管されません。

非日付データは、推定日付フォーマットのすべての桁位置が 9 の場合、全桁 9 のトリガー値であると見なされます。したがって、受け取り側の日付フォーマットが YYXXX の場合、非日付データ値 99,999 はトリガーですが、値 9,999 と 999,999 はトリガーではありません。もちろん、大きい方の値 999,999 は常にサイズ・エラーになります。

表 17. 日付フィールドの関係する算術演算結果の保管 - ON SIZE ERROR を指定した場合

	送り出しフィールド	
受け取りフィールド	**非日付データ**	**日付フィールド**
非日付データ	ウィンドウ化なし	許可されない
ウィンドウ化日付フィールド	ウィンドウ化	ウィンドウ化
拡張日付フィールド	ウィンドウ化なし	ウィンドウ化なし

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。

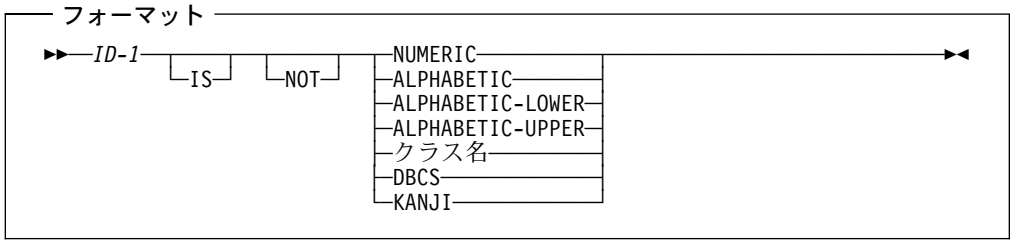
- クラス条件
- 条件名条件
- 比較条件
- 符号条件
- スイッチ状況条件

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、または環境部の SPECIAL-NAMES 段落で定義されている CLASS 文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 を使用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。

表18 には、各種の ID のタイプで有効なクラス条件の形式が示されています。

ID-1 が関数 ID である場合、この ID-1 は、英字関数または国別関数を参照する必要があります。

グループ項目は、基本英数字項目が使用できるクラス条件で使用することができますが、符号の付いた基本項目がグループに 1 つまたは複数個含まれる場合は、NUMERIC クラス条件を*使用できません。*

ID-1 が国別データ項目の場合、クラス条件は、指定された文字クラスに関連付けられている文字の国別文字表示についてテストします。たとえば、

```
IF national-item IS ALPHABETIC
```

という形式のクラス条件を指定すると、国別文字で表示されている、小文字および大文字のラテン頭文字 A ～ Z、およびスペースがテストされます。

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。たとえば、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 は、算術演算符号の付いた 0 ～ 9 の文字、または算術演算符号の付かない 0 ～ 9 の文字だけで構成されます。

PICTURE 文節が算術演算符号を含まない場合は、内容が数字であり、かつ算術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 文節が算術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算術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 は、全体が A ～ Z の小文字または大文字の英字とスペースの任意の組み合わせで構成されます。

ALPHABETIC-LOWER
ID は、その全体が小文字の英字 a ～ z とスペースとの任意の組み合わせで構成されます。

ALPHABETIC-UPPER
ID は、その全体が大文字の英字 A ～ Z とスペースとの任意の組み合わせで構成されます。

クラス名
ID 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 は、DBCS 文字だけで構成されます。

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' ～ X'FE' の範囲で、DBCS のブランクは X'4040' です。

KANJI
ID は、DBCS 文字だけで構成されます。

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'～X'7F'、第 2 バイト目が X'41'～X'FE'、DBCS のブランクは X'4040' です。

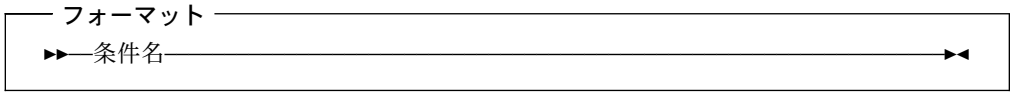
表 18. 各種の ID のタイプに対するクラス条件の有効な形式

ID のタイプ	クラス条件の有効な形式	
英字	ALPHABETIC ALPHABETIC-LOWER ALPHABETIC-UPPER クラス名	NOT ALPHABETIC NOT ALPHABETIC-LOWER NOT ALPHABETIC-UPPER NOT クラス名
英数字、英数字編集、または数字編集	ALPHABETIC ALPHABETIC-LOWER ALPHABETIC-UPPER NUMERIC クラス名	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	NOT NUMERIC NOT ALPHABETIC NOT ALPHABETIC-LOWER NOT ALPHABETIC-UPPER
数字	NUMERIC クラス名	NOT NUMERIC NOT クラス名

条件名条件

条件名条件は条件変数をテストし、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 文節の条件名フォーマットに定義するように、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  TEEN-AGER        VALUE 13 THRU 19.
```

AGE-GROUP は条件変数で、INFANT、BABY、CHILD、および TEEN-AGER 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 ～ 12 のテスト)
IF TEEN-AGER...         (値 13 ～ 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化日付フィールドの比較

条件変数がウィンドウ化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、ウィンドウ化日付フィールドの値と同じように扱われます。すなわち、165ページの『ウィンドウ化日付フィールドのセマンティクス』で説明しているように、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。

たとえば、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...
```

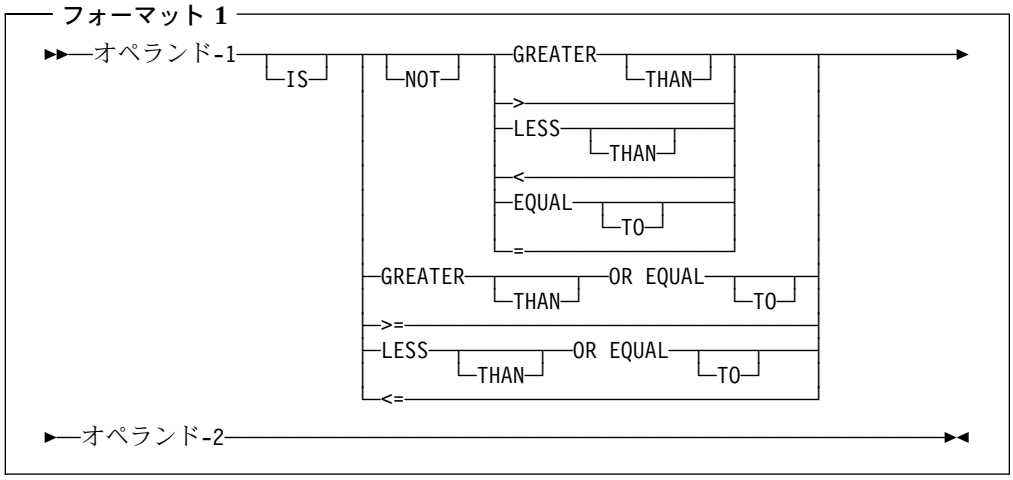
ウィンドウ化日付フィールドを使う比較では、世紀ウィンドウに関係なくリテラルは接頭部として “19” が付けられたかのように扱われるからです。したがって、上記の条件は実際には次のようになります。

```
IF 20051220 = 19051220...
```

条件式でウィンドウ化日付フィールドを使用する方法について詳しくは、241ページの『日付フィールド』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。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。



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。

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。表19 に、比較演算子とそれらの意味を示しています。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 <= との間にはスペースを入れないでください。

表 19. 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 <=	より小さいかまたは等しい

日付フィールド

日付フィールドは、英数字、外部 10 進数、または内部 10 進数にすることができます。このような項目の比較については、妥当性およびモードに関する既存の規則 (数字または英数字) がそのまま適用されます。たとえば、英数字の日付フィールドは、内部 10 進数の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、ウィンドウ化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。

242ページの表20 は、年末尾型日付フィールド以外で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。

ウィンドウ化なし
: オペランドが両方とも非日付データであるかのように、ウィンドウ化なしで比較が実行されます。

ウィンドウ化
: 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化日付フィールドは、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 (165ページの『ウィンドウ化日付フィールドのセマンティクス』を参照してください)。

   DATEPROC(TRIG) コンパイラー・オプションが有効である場合、この拡張では日付フィールドの比較オペランドのトリガー値が考慮されます。

2. 反復する英数字の形象定数は、比較されるウィンドウ化日付フィールド (英数字の非日付データの比較オペランド数) 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。

   DATEPROC(NOTRIG) コンパイラー・オプションが有効である場合、比較は拡張西暦年の世紀部分を 19 とみなすことによって非日付データが拡張されたかのようにして実行されます。

   DATEPROC(TRIG) コンパイラー・オプションが有効である場合、比較では非日付データ・オペランドの日付トリガー値が考慮されます。英数字のオペランドの場合のトリガー値は LOW-VALUE、HIGH-VALUE、および SPACE です。DATE FORMAT 文節に最低 1 つの X が含まれるウィンドウ化日付フィールド (つまりウィンドウ化西暦年だけではないウィンドウ化日付フィールド) と比較する英数字および数字の場合、全桁 0 または全桁 9 の値もトリガーとして扱われます。非日付データ・オペランドの内容がトリガー値の場合、比較は、トリガー値を拡張西暦年の仮定世紀部分に非日付データ・オペランドをコピーすることによって拡張されたかのようにして実行されます。非日付データ・オペランドの内容がトリガー値でないなら、拡張西暦年の世紀部分は 19 とみなされます。

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 20. 日付フィールドとの比較

	第 2 オペランド		
第 1 オペランド	**非日付データ**	**ウィンドウ化日付フィールド**	**拡張日付フィールド**
非日付データ	ウィンドウ化なし	ウィンドウ化 [1]	ウィンドウ化なし
ウィンドウ化日付フィールド	ウィンドウ化 [1]	ウィンドウ化	ウィンドウ化
拡張日付フィールド	ウィンドウ化なし	ウィンドウ化	ウィンドウ化なし

注:

1. ウィンドウ化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化西暦年を含むものと想定されます。詳細については、「ウィンドウ化」比較の定義の項目 3 と 4 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、232ページの『日付フィールドでの算術演算』を参照してください。

DBCS 項目

DBCS データ項目およびリテラルはすべての比較演算子と一緒に使用することができます。比較は、DBCS 文字の 16 進値の 2 進照合シーケンスに基づいて行われます。DBCS 項目が同じ長さでない場合、短い方の項目の右側に DBCS のスペースが埋め込まれます。

注: PROGRAM COLLATING SEQUENCE 文節は、DBCS データ項目と DBCS リテラルの比較には適用されません。

DBCS 項目は、国別項目と DBCS 項目とだけ比較できます。

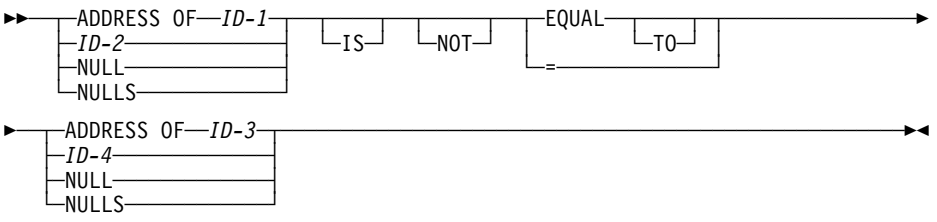
ポインター・データ項目

ポインター・データ項目を指定したときには、比較演算子として EQUAL および NOT EQUAL だけが使えます。ポインター・データ項目は、USAGE IS POINTER として明示的に定義されている項目、または USAGE IS 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



ID-1
ID-3

レベル 66 とレベル 88 を除き、リンケージ・セクション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 IS POINTER として記述されている必要があります。

NULL(S)

これは、この構文図からもわかるとおり、もう一方のオペランドが、USAGE IS 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。

表21 は、USAGE IS 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 21. USAGE IS POINTER、NULL、および ADDRESS OF に対して可能な比較

	第 2 オペランド		
第 1 オペランド	USAGE IS POINTER	ADDRESS OF	NULL/NULLS
USAGE IS POINTER	はい	はい	はい
ADDRESS OF	はい	はい	はい
NULL/NULLS	はい	はい	いいえ

注:

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・データ項目と関数ポインター・データ項目

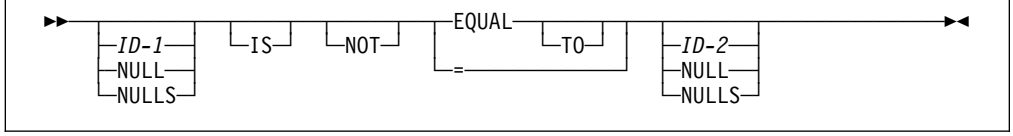
比較演算子として EQUAL と NOT EQUAL しか使用できない場合がありますが、それは、比較条件でプロシージャー・ポインター・データ項目または関数ポインター・データ項目を指定しているときです。

プロシージャー・ポインター・データ項目は、USAGE IS PROCEDURE-POINTER として明示的に定義されます。関数ポインター・データ項目は、USAGE IS 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・ポインターまたは関数ポインター



ID-1
ID-2

USAGE IS PROCEDURE-POINTER または USAGE IS FUNCTION-POINTER として記述します。ID-1 と ID-2 を同じ内容にする必要はありません。

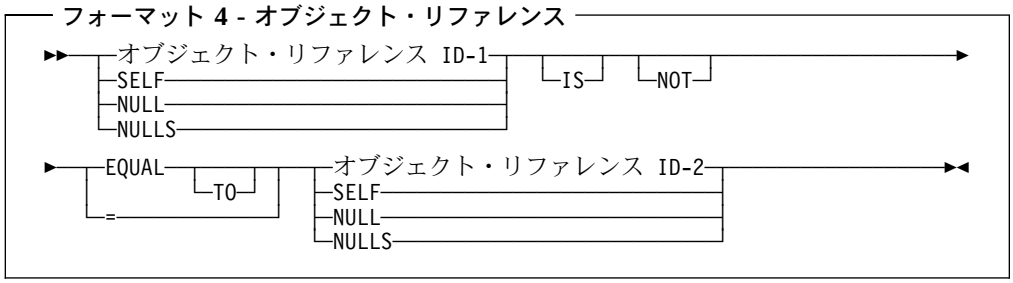
NULL(S)

この構文図に示すとおり、NULL(S) を使用できるのは、他のオペランドが USAGE IS FUNCTION-POINTER または USAGE IS PROCEDURE-POINTER として定義されているときだけです。つまり、NULL=NULL になることはありえません。

オブジェクト・リファレンス・データ項目

USAGE OBJECT REFERENCE のあるデータ項目を、USAGE OBJECT REFERENCE の別のデータ項目か、NULL、NULLS、または SELF と比較して、等しいかどうかを調べることができます (SELF との比較ができるのは、メソッドの中だけです)。データ項目が同一のオブジェクトを識別している場合に限り、2 つのオブジェクト・リファレンスは等価比較を行います。

フォーマット 4 - オブジェクト・リファレンス



数字オペランドと英数字オペランドの比較

数字オペランドの比較

数字オペランドの代数値が比較されます。

- オペランドの長さ (桁数) は意味を持ちません。
- 無符号の数字オペランドは正とみなされます。
- 0 は符号に関係なく、固有の値とみなされます。
- 数字オペランドは、それぞれに指定されている USAGE の種類に関係なく、比較することができます。

数字オペランドと国別オペランドの比較については、249ページの『国別オペランドの比較』を参照してください。

表22 では、**数字**オペランドと比較できるその他の内容がすべて要約されています。

以下は、表22 および 表23 で使用されている記号です。

AN = 英数字オペランドの比較
NU = 数字オペランドの比較
ブランク = 比較はできない

表 22. 数字の第 2 オペランドと比較できる項目

	第 2 オペランド								
第 1 オペランド	**ZR**	**NL**	**ED**	**BI**	**AE**	**ID**	**IFP**	**EFP**	**FPL**
英数字オペランド									
グループ (GR)	AN	AN [1]	AN [1]					AN	
英字 (AL)	AN	AN [1]	AN [1]					AN	
英数字 (AN)	AN	AN [1]	AN [1]					AN	
英数字編集 (ANE)	AN	AN [1]	AN [1]					AN	
数字編集 (NE)	AN	AN [1]	AN [1]					AN	
形象定数 (FC [2])			AN [1]					AN	
英数字リテラル (ANL)			AN [1]					AN	
数字オペランド									
形象定数 ZERO (ZR)			NU	NU	NU	NU	NU	NU	
数字リテラル (NL)			NU	NU	NU	NU	NU	NU	
外部 10 進数 (ED)	NU	NU	NU	NU	NU	NU	NU	NU	NU
2 進数 (BI)	NU	NU	NU	NU	NU	NU	NU	NU	NU
算術式 (AE)	NU	NU	NU	NU	NU	NU	NU	NU	NU
内部 10 進数 (ID)	NU	NU	NU	NU	NU	NU	NU	NU	NU
内部浮動小数点 (IFP)	NU	NU	NU	NU	NU	NU	NU	NU	NU
外部浮動小数点 (EFP)	NU	NU	NU	NU	NU	NU	NU	NU	NU
浮動小数点リテラル (FPL)			NU	NU	NU	NU	NU	NU	

注:

1 整数項目のみ
2 ZERO 以外のすべての形象定数を含む

英数字オペランドの比較

英数字オペランドの比較は、使用中の文字セットの照合シーケンスに基づいて行われます。

- EBCDIC 文字セットに関しては、EBCDIC 照合シーケンスを使用します。
- ASCII 文字セットに関しては、ASCII 照合シーケンスを使用します (556ページの付録 C、『EBCDIC および ASCII の照合シーケンス』を参照)。
- PROGRAM COLLATING SEQUENCE 文節を OBJECT-COMPUTER 段落の中で指定した場合、SPECIAL-NAMES 段落の英字名文節に関係付けられている照合シーケンスが使用されます。

各オペランドのサイズは、そのオペランドの合計文字数です。サイズは比較の結果に影響します。次の 2 つの考慮すべき場合があります。

オペランドのサイズが等しい場合

2 つのオペランドの対応する位置にある文字が、左端の文字から始めて右端の文字まで比較されます。

対応する文字が最後まですべて等しければ、両オペランドは等しいと見なされます。

等しくない文字の組が検出されると、それら 2 つの文字の照合シーケンスでの相対位置が判定されます。照合シーケンスでの位置が高い方の文字を含むオペランドが、より大きいオペランドであると見なされます。

オペランドのサイズが等しくない場合

オペランドのサイズが等しくないときは、短い方のオペランドの右側がスペースで拡張されて、両方のオペランドのサイズ等しくなるように、比較が行われます。

英数字オペランドと国別オペランドの比較については、249ページの『国別オペランドの比較』を参照してください。

247ページの表23 は、**英数字**オペランドと比較できる、その他の比較内容について要約しています。

表 23. 英数字の第 2 オペランドと比較できる項目

	第 2 オペランド						
第 1 オペランド	**GR**	**AL**	**AN**	**ANE**	**NE**	**FC** [2]	**ANL**
英数字オペランド							
グループ (GR)	AN	AN	AN	AN	AN	AN	AN
英字 (AL)	AN	AN	AN	AN	AN	AN	AN
英数字 (AN)	AN	AN	AN	AN	AN	AN	AN
英数字編集 (ANE)	AN	AN	AN	AN	AN	AN	AN
数字編集 (NE)	AN	AN	AN	AN	AN	AN	AN
形象定数 (FC [2]**)**	AN	AN	AN	AN	AN		
英数字リテラル (ANL)	AN	AN	AN	AN	AN		
数字オペランド							
形象定数 ZERO (ZR)	AN	AN	AN	AN	AN		
数字リテラル (NL)	AN [1]	AN [1]	AN [1]	AN [1]	AN [1]		
外部 10 進数 (ED)	AN [1]	AN [1]	AN [1]	AN [1]	AN [1]	AN [1]	AN [1]
2 進数 (BI)							
算術式 (AE)							
内部 10 進数 (ID)							
内部浮動小数点 (IFP)							
外部浮動小数点 (EFP)	AN	AN	AN	AN	AN	AN	AN
浮動小数点リテラル (FPL)							

注:

1 整数項目のみ
2 ZERO 以外のすべての形象定数を含む

数字オペランドと英数字オペランドの比較

上述の英数字オペランドの比較規則が適用されます。また、数字オペランドと英数字オペランドが比較されるときは、それらの USAGE は同じでなければなりません。そのような比較においては、次の規則があります。

- 数字オペランドは、整数リテラルまたは整数データ項目として記述されていなければなりません。
- 非整数リテラルと非整数データ項目は、英数字オペランドと比較することはできません。
- 外部浮動小数点項目は、英数字オペランドと比較できます。

どちらかのオペランドがグループ項目であれば、上述の英数字オペランド比較の規則が適用されます。さらに次のような規則があります。

- 英数字オペランドが**リテラルまたは基本データ項目**の場合、数字オペランドは、同じサイズの英数字基本データ項目に移動され、次にこの英数字データ項目の内容が英数字オペランドと比較されるかのように処理されます。
- 英数字オペランドが**グループ項目**の場合、数字オペランドは、同じサイズのグループ項目に移動され、次にこのグループ項目の内容が英数字オペランドと比較されるかのように処理されます。

  346ページの『MOVE ステートメント』を参照してください。

指標名と指標データ項目の比較

指標名と指標データ項目に関連する比較は、次の規則に従います。

- 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、これにより、指標名を整数関数や数字関数と比較することが可能になります。
- 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。

表24 は、指標名と指標データ項目に関して、有効な比較を示したものです。

表 24. 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	オカレンス番号をデータ名と比較	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

DBCS オペランドの比較

DBCS オペランドを比較する場合の規則は、英数字オペランドを比較する場合の規則と同じです。

比較は、DBCS 文字の 16 進値の 2 進照合シーケンスに基づいて行われます。

注: PROGRAM COLLATING SEQUENCE 文節は、DBCS オペランドの比較には適用されません。

国別オペランドの比較

クラス国別のオペランドは、以下のものと比較できます。

- 国別オペランド
- USAGE DISPLAY を伴う数字整数データ項目
- 数字整数リテラル
- 形象定数 ZERO、SPACE、QUOTE、およびすべてのリテラル
- 英字オペランド
- 英数字オペランド
- 英数字編集オペランド
- 数字編集オペランド
- DBCS オペランド
- グループ項目

オペランドには、データ項目、リテラル、または関数を指定できます。

グループ項目を除き、クラス国別ではないオペランドは、比較を行う前に国別データ項目に変換されます。

国別オペランドとグループ項目を比較する場合、国別オペランドは、同じサイズのグループ項目に移動されるかのように処理されます。比較は、2 つのグループ項目に対する英数字比較の規則に従って進行されます。

以下に、オペランドのクラス国別への変換について説明します。

DBCS
DBCS オペランドは、同じ長さの一時国別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文字コード・ページは、ソース・コードのコンパイル時に CODEPAGE コンパイラー・オプションに対して有効なコード・ページです。

英数字
英数字オペランドは、オペランドの文字を表現するのに必要なサイズを持つ一時国別データ項目に移動されるかのように処理されます。対応する国別文字に英数字文字を変換する際は、ソース・コードのコンパイル時に CODEPAGE コンパイラー・オプションに対して有効な ECBDIC ソース・コードが使用されます。

英字
英数字編集
数字編集
オペランドは、英数字項目であるかのように変換されます。

数字
数字オペランドは、同じサイズを持つ基本英数字項目に移動され、次に英数字オペランドとして変換されるかのように処理されます。

このような暗黙の移動は、MOVE ステートメントの規則に従って実行されます。

生成された国別データ項目は、2 つの国別オペランド用に記述された比較演算で使用されます。

比較は、文字の 16 進値を使用して実行されます。PROGRAM COLLATING SEQUENCE 文節は、国別オペランドの比較には影響を及ぼしません。

2 つの国別オペランドの比較

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'0020') 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を適用して実行されます。

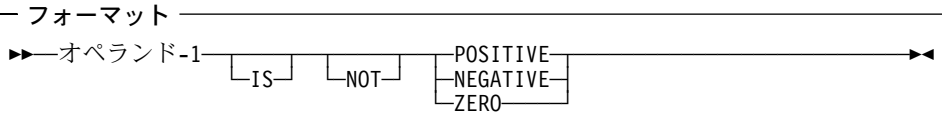
2 つのオペランドの長さが同じ場合は、オペランド内の同じ位置にある国別文字同士が比較されます。比較は左端から開始され、途中で異なる国別文字が検出されるか、右端に到達するまで繰り返されます。対応する国別文字がすべて同じであった場合は、2 つのオペランドが等しいと判別されます。

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 2 進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。

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



オペランド-1
数字 ID として定義されていなければなりません。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

無符号のオペランドは、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。たとえば、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。

NUMPROC コンパイラー・オプションを使用している場合は、符号条件テストの結果に影響が及ぶことがあります。詳しくは、*Enterprise COBOL プログラミング・ガイド* を参照してください。

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化日付フィールドの ID である場合、日付ウィンドウ化は行われないため、符号条件を使用してウィンドウ化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、式中のウィンドウ化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。

たとえば、次の場合、

- ID WIN-DATE がウィンドウ化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
```

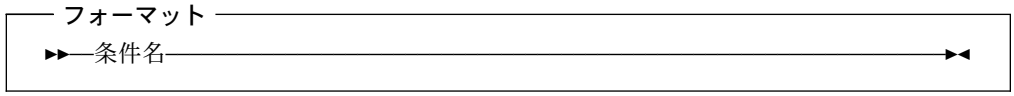
逆に、以下の符号条件は偽であると評価されます。

```
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 (97ページの『SPECIAL-NAMES 段落』を参照)。

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 (条件名に関係付けられた値は、英数字とみなされます)。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、*Enterprise COBOL プログラミング・ガイド* 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、またはこ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 25. 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、次のような優先順位の規則 (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

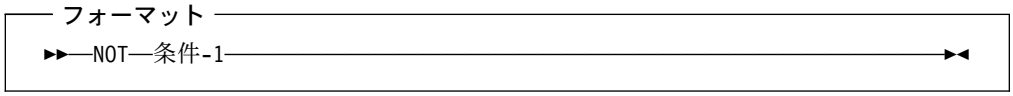
複合条件は、次のどちらかにすることができます。

- 単純否定条件
- 複合条件 (その否定形も可)

単純否定条件

単純条件を否定するには、論理演算子の NOT を使います。

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。

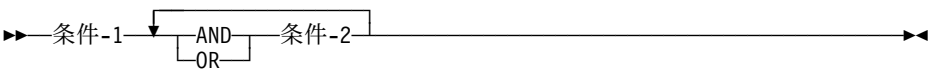
```
NOT A IS EQUAL TO B.

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 26. 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。

254ページの表27 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 27. 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。以下に例を示します。

```
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

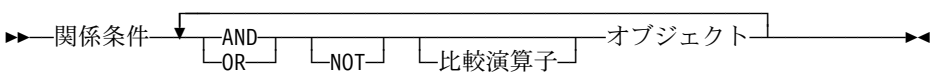
1. (NOT (A IS GREATER THAN B)) が評価され、ある中間的な真の値として、**t1** が得られます。**t1** が真であれば複合条件が真となり、それ以上の評価は行われません。**t1** が偽であれば、評価はさらに次のように継続されます。
2. (A + B) が評価され、ある中間的な結果として **x** が得られます。
3. (**x** IS EQUAL TO C) が評価され、ある中間的な真の値として **t2** が得られます。**t2** が偽であれば、複合条件自体が偽となり、それ以上の評価は行われません。**t2** が真ならば、評価はさらに次のように継続されます。
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、次のどちらかの省略形で記述できます。

- サブジェクトの省略
- サブジェクトと比較演算子の省略

フォーマット



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。253ページの表26 を参照してください。

NOT の直後に置かれた語が GREATER THAN、>、LESS THAN、<、EQUAL TO、および = であるときは、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。

省略形で結合した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、および = 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、したがって、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ることに注意してください。
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や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。たとえば、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、簡略結合比較条件を記述するときの規則を要約しています。

表 28. 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、比較条件を簡略的に結合した例 (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 29. 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、次の 4 つのカテゴリーがあります。

- 命令ステートメント
- 条件付きステートメント
- 範囲区切りステートメント
- コンパイラー指示ステートメント

命令ステートメント

命令ステートメントは、プログラムが無条件に実行する処置を指定するステートメントであり、また、明示的範囲終了符号で終了する条件ステートメントでもあります (260ページの『範囲区切りステートメント』を参照)。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範囲を限定され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 (260ページの『範囲区切りステートメント』を参照)。表30 は、COBOL の命令ステートメントを示したものです。

表 30 (1/2). 命令ステートメント

算術演算
ADD [1]
COMPUTE [1]
DIVIDE [1]
MULTIPLY [1]
SUBTRACT [1]
データの移動
ACCEPT (DATE,DAY,DAY-OF-WEEK,TIME)
INITIALIZE
INSPECT
MOVE
SET
STRING [2]
UNSTRING [2]
XML PARSE [8]

表 30 (2/2). 命令ステートメント

終了

STOP RUN
EXIT PROGRAM
EXIT METHOD
GOBACK

入出力

ACCEPT ID
CLOSE
DELETE [3]
DISPLAY
OPEN
READ [4]
REWRITE [3]
START [3]
STOP リテラル
WRITE [5]

順序付け

MERGE
RELEASE
RETURN [6]
SORT

プロシージャーのブランチ

ALTER
EXIT
GO TO
PERFORM

プログラムまたはメソッドのリンケージ

CALL [7]
CANCEL
INVOKE

テーブル操作

SET

注:

1 ON SIZE ERROR 句または NOT ON SIZE ERROR 句 (またはその両方) が指定されていないもの。
2 ON OVERFLOW 句または NOT ON OVERFLOW 句 (またはその両方) が指定されていないもの。
3 INVALID KEY 句または NOT INVALID KEY 句 (またはその両方) が指定されていないもの。
4 AT END 句、NOT AT END 句、INVALID KEY 句、および NOT INVALID KEY 句 (またはこれらのいずれか) が指定されていないもの。
5 INVALID KEY 句、NOT INVALID KEY 句、END-OF-PAGE 句、および NOT END-OF-PAGE 句 (またはこれらのいずれか) が指定されていないもの。
6 AT END 句または NOT AT END 句 (またはその両方) が指定されていないもの。
7 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 (またはその両方) が指定されていないもの。
8 ON EXCEPTION 句または NOT ON EXCEPTION 句 (またはその両方) が指定されていないもの。

条件ステートメント

条件ステートメントは、ある条件の真の値を判別すること、そしてオブジェクト・プログラムの次にとる処置がこの真の値によって決まることを指定します (235ページの『条件式』を参照)。表31 は、**条件** (たとえば、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 ステートメントをリストしています。

表 31 (1/2). 条件ステートメント

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 PARSE ... ON OVERFLOW
	XML PARSE ... NOT ON OVERFLOW
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

表 31 (2/2). 条件ステートメント

プログラムまたはメソッドのリンケージ	CALL ... ON OVERFLOW CALL ... ON EXCEPTION CALL...NOT ON EXCEPTION INVOKE ... ON EXCEPTION INVOKE ... 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。ただし、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。

明示的範囲終了符号

EXPLICIT SCOPE TERMINATOR は、特定の手続き部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-READ
END-CALL                END-RETURN
END-COMPUTE             END-REWRITE
END-DELETE              END-SEARCH
END-DIVIDE              END-START
END-EVALUATE            END-STRING
END-IF                  END-SUBTRACT
END-INVOKE              END-UNSTRING
END-MULTIPLY            END-WRITE
END-PERFORM             END-XML
```

暗黙の範囲終了符号

IMPLICIT SCOPE TERMINATOR とは、文の終わりにある区切りピリオドのことで、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。

注: 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。

コンパイラー指示ステートメント

指定した処置を行うようにコンパイラーに指示するステートメントについては、506ページの『コンパイラー指示ステートメント』で説明します。

ステートメントによる操作

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 句 (CORR) を使用することによって、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グループ項目が指定されている必要があります。

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 として言及します。

1 対のデータ項目 (従属項目) は一方は ID-1 から、もう一方は ID-2 からのもので、次に示す条件が真ならばその 1 対のデータ項目は対応しています。

- 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 と 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 USAGE IS INDEX 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。
- 従属項目でその記述に含まれない文節は、REDEFINES、RENAMES、OCCURS、USAGE IS INDEX、USAGE IS POINTER、USAGE IS PROCEDURE-POINTER、USAGE IS FUNCTION-POINTER、または USAGE IS OBJECT REFERENCE 文節です。

  ただし、ID-1 および ID-2 は、REDEFINES 文節や OCCURS 文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。

- ID-1 も ID-2 も、USAGE IS POINTER、USAGE IS FUNCTION-POINTER、USAGE IS PROCEDURE-POINTER、または USAGE IS OBJECT REFERENCE として記述されません。

ID-1 と ID-2 (またはその一方) は、FILLER 項目に従属することができます。

たとえば、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.
```

ここで 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 と 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。また、ITEM-D と ITEM-D (X) は含まれません。これは ITEM-D (X) が REDEFINES 文節をそのデータ記述中に含むからです。さらに、ITEM-F と ITEM-F (X) は含まれません。これはそれらが USAGE IS INDEX として定義されているからです。ITEM-1 は、ID-1 または ID-2 のどちらか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・エラー状態が発生した場合、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

GIVING という語の後に置かれる ID の値は、算術演算の計算結果に等しく設定されます。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

小数点の位置合わせが行われた後、算術演算の結果の小数部の桁数が、演算結果用の ID の小数部に指定されている桁数と比較されます。

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、結果用の ID の最下位の桁は、超過分の最上位桁が 5 以上であるときはいつでも 1 だけ増加されます。

結果用の ID が右端に P (複数個) を持つ PICTURE 文節によって記述されており、また計算結果が指定された整数桁数を超える場合、ストレージが割り振られている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。浮動小数点の算術式については、 *Enterprise COBOL プログラミング・ガイド* 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。たとえば、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、次の 4 つの場合に起こります。

- 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化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。たとえば、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に、次の SUBTRACT ステートメントを使用するとサイズ・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 (ウィンドウ化日付フィールド) が 1930 という年の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。ウィンドウ化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、233ページの『日付フィールドの関連する減算』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・エラーの発生について、詳しくは 234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 32 (1/2). 指数計算のサイズ・エラー条件

サイズ・エラー	SIZE ERROR 文節が指定されているときに取られる処置	SIZE ERROR 文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。

表 32 (2/2). 指数計算のサイズ・エラー条件

サイズ・エラー	SIZE ERROR 文節が指定されているときに取られる処置	SIZE ERROR 文節が指定されていないときに取られる処置
0 が負数乗された	SIZE ERROR 命令が実行されます。	プログラムが異常終了します。
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。

結果の ID が USAGE IS BINARY、COMPUTATIONAL、または COMPUTATIONAL-4 を使用して定義されている場合、そこに入れることができる最大値は、それに関連した 10 進数 PICTURE 文字ストリングによって暗黙に指定された最大値です。

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

算術ステートメントは計算を行うために使用されます。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。COMPUTE ステートメントを使用すれば、これらの演算を 1 つの式の中で記号によって組み合わせることができます。

算術ステートメント・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。計算の途中で、コンパイラーが必要なデータ変換や小数点の位置合わせをします。

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 33. 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	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 (ただし、GIVING という語の後のオペランドは除く)
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

たとえば、各項目がデータ部で次のように定義されているとします。

```
A  PICTURE 9(7)V9(5).
B  PICTURE 9(11)V99.
C  PICTURE 9(12)V9(3).
```

次の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を持ちます。

```
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。

算術ステートメントにおいて、希望どおりの精度を最終的な結果で得るには、十分な桁数と小数点位置を持つデータを定義することが重要です。詳しくは、 *Enterprise COBOL プログラミング・ガイド* の中間結果に関する項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内のオペラン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。

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。

- 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。
- 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。

たとえば、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、および XML PARSE。

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、外部メディアに収められているファイルとの間で、データを転送し、また入出力装置間で得られる (出力装置へ送られる) 少量のデータを制御します。

COBOL では、プログラムで使用できるファイル・データの単位はレコードであるため、プログラマーはレコードに注意を払うだけで済みます。バッファーや内部記憶域へのデータの移動、妥当性検査、エラー訂正 (可能な場合)、ブロック化と非ブロック化、ボリューム切り替えプロシージャーといった処理は、自動的に行われます。

環境部とデータ部でファイルがどのように記述されているかによって、手続き部で使用できる入出力ステートメントが決まります。各種のファイル編成で使用できるステートメントは、358ページの表47 および 358ページの表48 に示されています。

次のセクションでは、**ボリューム**と**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。

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。そのような共通の処理機能としては、次のようなものが挙げられます。

- 状況キー
- 無効キー条件
- INTO/FROM ID句
- ファイル位置標識

状況キー

FILE-CONTROL 記入項目で FILE STATUS 文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、指定した状況キー (FILE STATUS 文節で指定した 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。値が状況キーに入れられた後で、その要求に関連のある EXCEPTION/ERROR 宣言または INVALID KEY/AT END 句が実行されます。

2 種類の状況キー・データ名があります。1 つは、FILE-CONTROL 記入項目の FILE STATUS 文節の中でデータ名-1 によって記述されます。これは、状況キー 1 として認識される最初の 1 文字と、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、268ページの表34 に示しています。

もう 1 つの状況キーは、FILE-CONTROL 記入項目の FILE STATUS 文節の中にデータ名-8 として記述されます。データ名-8 は、QSAM ファイルには適用されません。データ名-8 については、125ページの『FILE STATUS 文節』を参照してください。

表 34 (1/4). 状況キーの値と意味

高位数字	意味	低位数字	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、重複できる代替キーのある指標ファイルだけに適用されます。 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存在しませんでした。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。これは、VSAM 順次ファイルには適用されません。
		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、そのファイルが存在しませんでした。
		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。
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ていました。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、存在し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 34 (2/4). 状況キーの値と意味

高位数字	意味	低位数字	意味
3	永続エラー条件	0	それ以上の情報はありません。
		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 句を持つ OPEN ステートメントがオプショナル・ファイル以外のファイル上で試みられましたが、そのファイルは存在しません。
		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。 1. 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。 2. 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。 3. 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 34 (3/4). 状況キーの値と意味

高位数字	意味	低位数字	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 文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。 1. 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。 2. 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 34 (4/4). 状況キーの値と意味

高位数字	意味	低位数字	意味
9	インプリメンターで定義した条件	0	1. マルチスレッド化の場合のみ。ファイルをオープンしなかったスレッドに対して、VSAM ファイルまたは QSAM ファイルの CLOSE が試行されました。 2. マルチスレッド化なし。VSAM のみの場合。詳しくは、*Enterprise COBOL プログラミング・ガイド* の VSAM 戻りコードの説明を参照してください。
		1	VSAM の場合のみ: パスワードの失敗。
		2	論理エラー
		3	QSAM 以外のすべてのファイルで、リソースが利用できません。
		5	QSAM 以外のすべてのファイルで、ファイル情報が無効または不完全です。
		6	VSAM ファイルの場合: OUTPUT 句が指定された OPEN ステートメント、または I-O 句か EXTEND 句が指定された OPEN ステートメントがオプション・ファイルに対して試みられましたが、そのファイルに対する DD ステートメントが指定されていませんでした。 QSAM ファイルの場合: OUTPUT 句が指定された OPEN ステートメント、または I-O 句か EXTEND 句が指定された OPEN ステートメントがオプション・ファイルに対して試みられましたが、そのファイルに対して DD ステートメントではなく、CBLQDA(OFF) ランタイム・オプションが指定されていました。
		7	VSAM の場合のみ: OPEN ステートメントが正常に実行されました。ファイルの整合性が確認されました。
		8	SELECT ... ASSIGN 文節で指定された環境変数の内容が無効であるか、動的割り振りに失敗したことが原因で、ファイルをオープンできません。この状況が起こる条件についての詳細は、112ページの『ASSIGN 文節』を参照してください。

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 (この条件の発生原因については、93ページの『環境部』の該当するステートメントを参照してください)。無効キー条件が発生した場合、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. FILE-CONTROL 項目の中に FILE STATUS 文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状況キーに入れられます (268ページの表34 を参照)。
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。

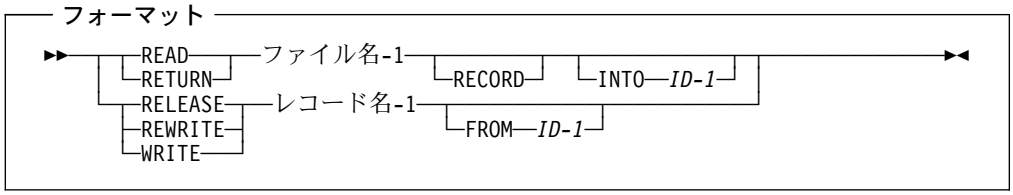
入出力操作の実行後に無効キー条件が存在しなければ、INVALID KEY 句は指定されていても無視され、次の処置が取られます。

1. 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。
2. 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/FROM ID 句

この句は 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。ID には、ワーキング・ストレージ・セクションまたはリンケージ・セクション内の記入項目の名前を指定するか、すでにオープンされた別ファイルのレコード記述の名前を指定する必要があります。レコード名 / ファイル名および ID は、同じストレージ域を参照することはできません。

フォーマット



- 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。現行レコードのサイズは、RECORD 文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 文節を含む場合には、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。

  ```
  MOVE identifier-1 TO record-name-1
  ```

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 文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

本書で説明するファイル位置標識は、入出力操作がある順序で行われているときに、指定されたファイル内で次にアクセスされるレコードを正確に指定するために使用される概念上の実体です。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。

手続き部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。

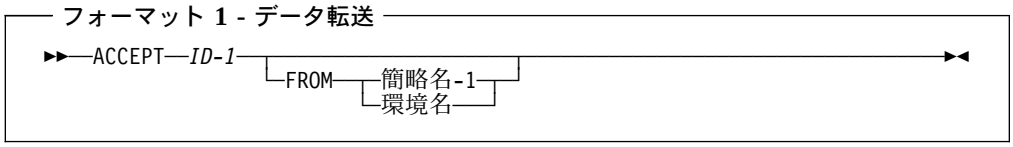
手続き部のステートメント

ACCEPT ステートメント

ACCEPT ステートメントは、指定された ID にデータを転送します。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 - データ転送



フォーマット 1 では、データは入出力装置から ID-1 に転送されます。FROM 句を省略する場合は、システム入力装置が前提とされます。

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。オペレーターは、該当のメッセージが与えられ、それに応答する必要があります。

ID-1

グループ項目、または基本英字、英数字、英数字編集、数字編集、外部 10 進数、DBCS、国別、外部浮動小数点データ項目を指定できます。

簡略名

SPECIAL-NAMES 段落内で入出力装置 (システム入力装置またはコンソール) と関連付ける必要があります。簡略名で使用可能な値の詳細については、97ページの『SPECIAL-NAMES 段落』を参照してください。

- システム入力装置

  データ転送の長さは入力装置上のレコードの長さと同じであり、その最大値は 32,760 です。

  システム入力装置は、ID-1 が満たされるか、または EOF になるまで読み取られます。ID-1 の長さが、システム入力装置のレコード長の偶数倍ではない場合には、最後のレコードは必要に応じて切り捨てられます。データが移動された後、または ID-1 が満たされる前に EOF に達すると、ID-1 の残り部分はブランクで埋められます。何らかのデータが ID-1 に移動される前に EOF に達すると、埋め込みは起こらずに ID-1 の内容は変更されません。各入力レコードは、直前の入力レコードと連結されます。

  入力レコードが、固定長フォーマットである場合は、入力レコード全体が使用されます。後尾につくブランクと先行のブランクを取り除く編集は行われません。

  入力レコードが可変調フォーマットの場合は、実際のレコード長を使用して、受信するデータの量が決められます。可変長フォーマットのレコードを使用している場合は、レコード定義語 (RDW) が、入力レコードの先頭から除去されます。実際の入力データだけが ID-1 に転送されます。

© Copyright IBM Corp. 1991, 2001

ID-1 が国別データ項目の場合、データは、変換と妥当性検査を行わずに ID-1 に転送されます。

- コンソール

  1. システム生成メッセージ・コードとその後に続く AWAITING REPLY リテラルが、自動的に表示されます。

     入力メッセージの最大長は、114 文字です。

  2. 実行は一時中断されます。

  3. メッセージ・コード (項目 1 のコードと同じもの) がコンソールから入力されてシステムにより確認された後、ACCEPT ステートメントの実行が再開されます。メッセージは ID-1 に移動されて PICTURE 文節に関係なく、左そろえにされます。 ID-1 が国別データ項目の場合は、EBCDIC から国別文字表示にメッセージが変換されます。変換では、ソース・コードのコンパイル時に CODEPAGE コンパイラー・オプションで指定された EBCDIC CCSID が使用されます。

     ACCEPT ステートメントは、次のいずれかが起こると終了します。

     – コンソールからデータを受信しなかった場合。たとえば、オペレーターが ENTER キーを押した場合。

     – ID がデータで満たされたとき。

     – 114 文字未満のデータが入力された場合。

     データを 114 バイト入力しても、ID がまだデータでいっぱいにならない場合は、さらにデータの要求がコンソールに出されます。

     114 文字以上のデータが入力された場合、最初の 114 文字だけがシステムによって認識されます。

     ID が入力メッセージよりも長いと、右端の文字位置には、スペースが埋め込まれます。

     入力メッセージが ID よりも長いと、ID の長さを超える部分の文字は、切り捨てられます。

  HFS ファイルまたは stdin から ACCEPT 入力データを入手する方法については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

環境名

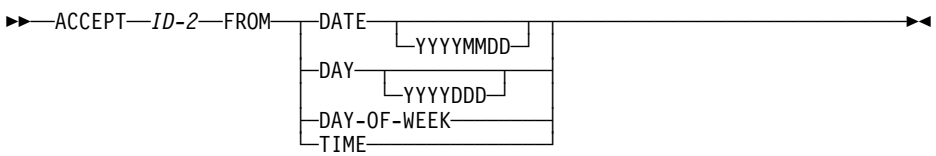
有効な**環境名**を指定することができます。有効な環境名のリストについては、99ページの表5 を参照してください。

注: 装置が、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 に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。346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 - システム情報の転送



ID-2
グループ、または基本英数字、英数字編集、数字編集、外部 10 進数、内部 10 進数、2 進数、内部浮動小数点、外部浮動小数点項目を指定できます。

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。

注: 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 (441ページの『組み込み関数』を参照してください)。

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。

DATE
暗黙の指定として PICTURE 9(6) を持ちます。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 YYXXXX 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。

```
2 digits for the year
2 digits for the month
2 digits for the day
```

したがって、1995 年 4 月 27 日は `950427` となります。

DATE YYYYMMDD
これは暗黙の指定として PICTURE 9(8) を持ちます。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。

```
4 digits for the year
2 digits for the month
2 digits for the day
```

したがって、1995 年 4 月 27 日は `19950427` となります。

DAY

これは暗黙の指定として PICTURE 9(5) を持ちます。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 YYXXX 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。

```
2 digits for the year
3 digits for the day
```

したがって、1995 年 4 月 27 日は 95117 となります。

DAY YYYYDDD

これは暗黙の指定として PICTURE 9(7) を持ちます。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。

```
4 digits for the year
3 digits for the day
```

したがって、1995 年 4 月 27 日は 1995117 となります。

DAY-OF-WEEK

これは暗黙の指定として PICTURE 9(1) を持ちます。

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
2 represents Tuesday
3 represents Wednesday
4 represents Thursday
5 represents Friday
6 represents Saturday
7 represents Sunday
```

したがって、水曜日は 3 と表されます。

TIME

これは暗黙の指定として PICTURE 9(8) を持ちます。

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。

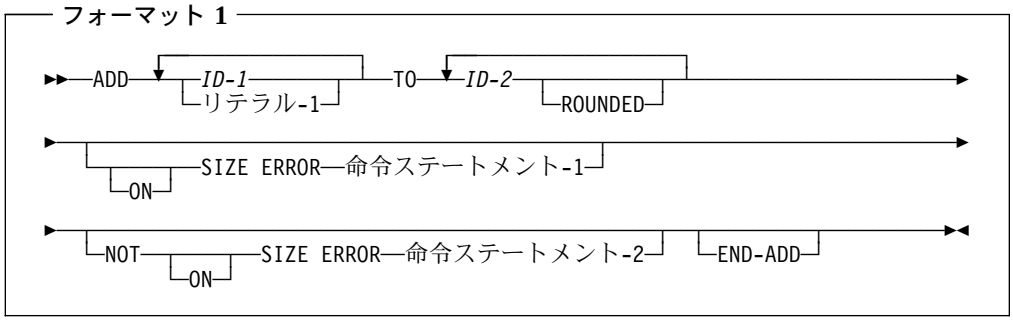
```
2 digits for hour of day
2 digits for minute of hour
2 digits for second of minute
2 digits for hundredths of second
```

したがって、2:41 PM は、14410000 となります。

ADD ステートメント

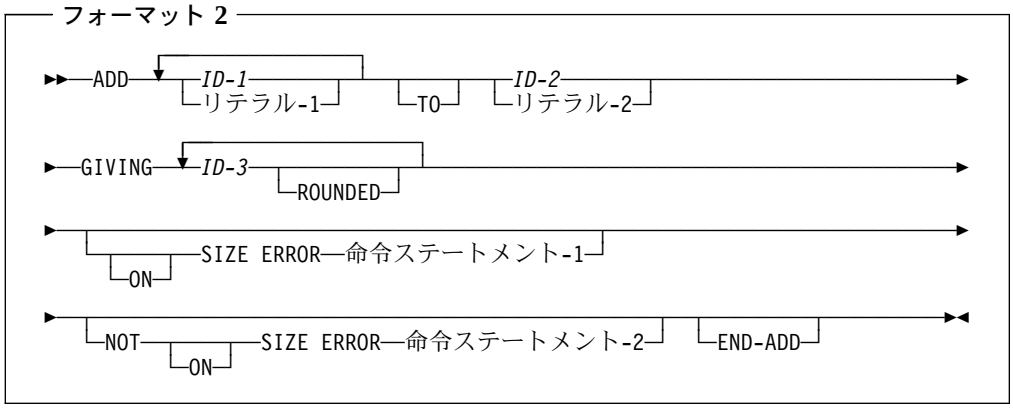
ADD ステートメントは、2 つ以上の数字オペランドを合計し、その結果を保管します。

フォーマット 1



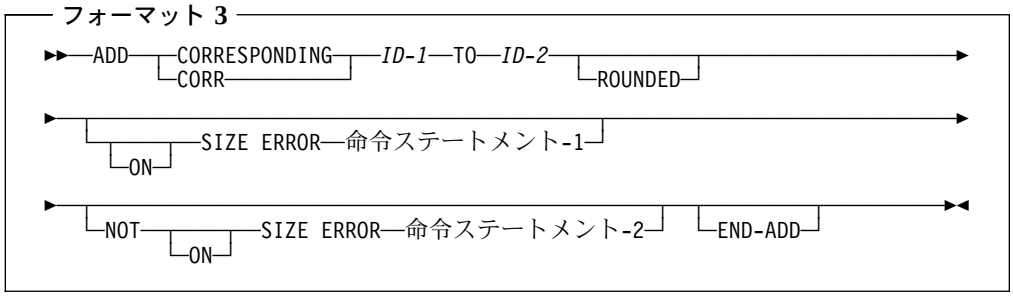
キーワード TO の前にあるすべての ID やリテラルが加算され、この合計が、ID-2 に加算され保管されます。この処理は、ID-2 が連続する場合、それぞれの ID-2 ごとに、ID-2 が指定されている順序で左から右へと繰り返されます。

フォーマット 2



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



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。

3 つのフォーマット全部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、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 つまたは複数の日付フィールドを指定することができます。ID-1 は日付フィールドを指定してはなりません。
- フォーマット 2 では、ID-1 と 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、ウィンドウ化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 つまたは複数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 つまたは複数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: 233ページの『日付フィールドの関連する加算』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。) 詳細については、234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、265ページの『算術ステートメント・オペランド』を参照してください。また、算術の中間結果については、 *Enterprise COBOL プログラミング・ガイド* 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、262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、263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)

261ページの『CORRESPONDING 句』を参照してください。

END-ADD 句

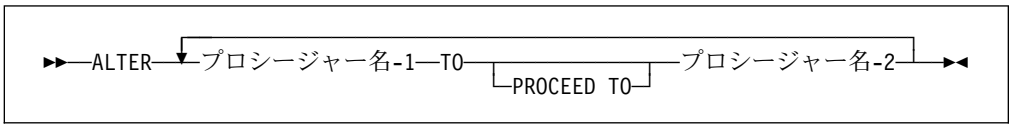
この明示的範囲終了符号は、ADD ステートメントの範囲を区切るために使用されます。END-ADD を使用すると、条件付き ADD ステートメントを別の条件ステートメントにネストさせることができます。END-ADD は、命令の ADD ステートメントと共に使用することもできます。

詳細については、260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に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が、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、プログラムの構造化に役立ちます。



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 (場合によっては複数) が実行されると、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手続き部でなければなりません。

プロシージャー名-2
手続き部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。たとえば、初期設定時にはある一連のステートメントを実行し、多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、メソッド、または THREAD オプションを指定してコンパイル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

優先順位が 50 以上であるセクション中の GO TO ステートメントを、別の優先順位を持つセクション中の ALTER ステートメントによって参照することはできません。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、オーバーレイ可能な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。

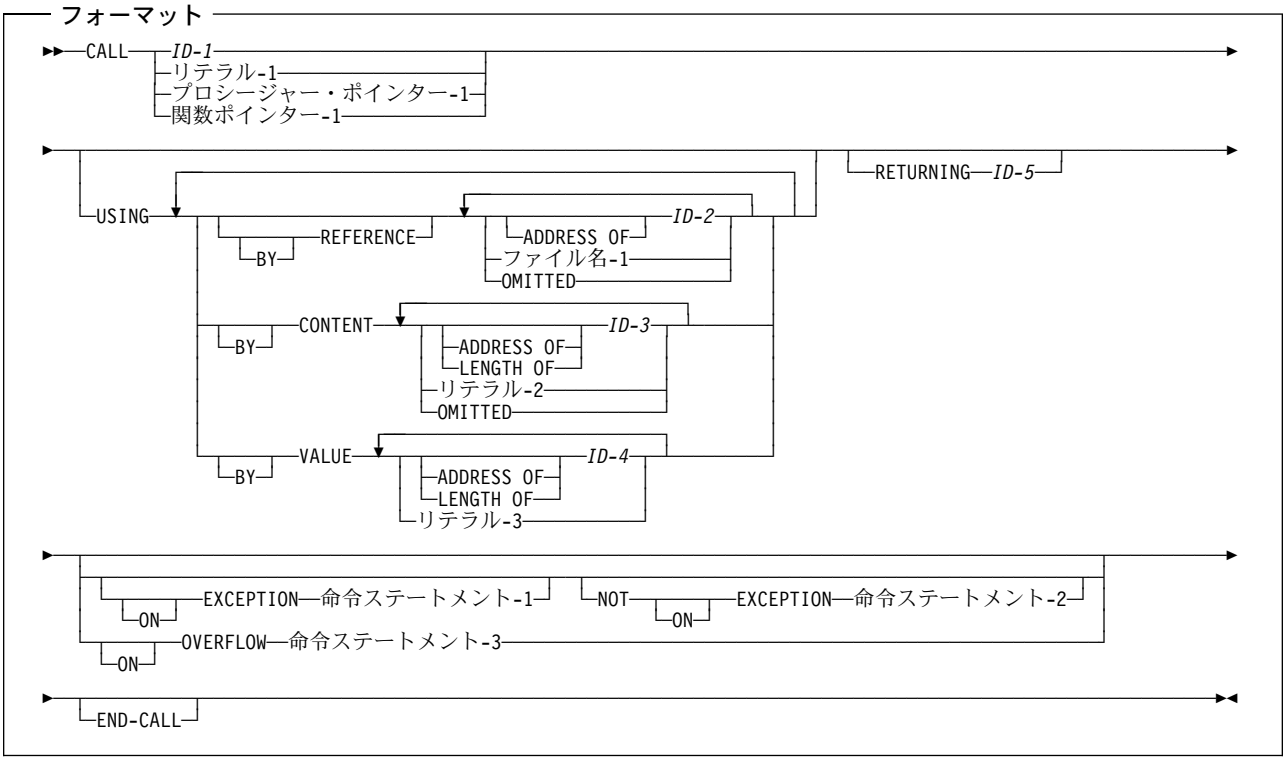
CALL ステートメント

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 属性で定義されたプログラムだけです。

CALL 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。

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 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、86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、 *Enterprise COBOL プログラミング・ガイド* の PGMNAME コンパイラー・オプションの説明も参照してください。

ID-1 は、ウィンドウ化日付フィールドにはできません。

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、手続き部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。

コンパイラーが、複数プログラム内で検出されるプログラム名への CALL をどのように解決するかについては、75ページの『プログラム名の命名規則』を参照してください。

USING 句

USING 句は、ターゲット・プログラムに渡される引き数を指定します。

USING 句を CALL ステートメントに入れるのは、手続き部のヘッダー、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、224ページの『手続き部のヘッダー』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手続き部のヘッダーまたは 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。

この場合、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。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
データ部内の任意のレベルの任意のデータ項目を指定できます。ID-2 は関数 ID にはできません。

注: リンケージ・セクションまたはファイル・セクションで定義する場合は、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。これを行うには、SET ADDRESS OF ID-2 TO ポインターまたは PROCEDURE/ENTRY USING のいずれかをコーディングします。

ファイル名-1
QSAM ファイルのファイル名。CALL ステートメントでファイル名を使用する方法については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

ADDRESS OF 特殊レジスター
ADDRESS OF 特殊レジスターの詳細については、13ページの『ADDRESS OF』を参照してください。

OMITTED
引き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、その手続き部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、呼び出しプログラム内の対応する引き数には影響ありません。

ID-3
データ部内の任意のレベルの任意のデータ項目を指定できます。ID-3 は関数 ID にはできません。

注: リンケージ・セクションまたはファイル・セクション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。これを行うには、SET ADDRESS OF ID-3 TO ポインターまたは PROCEDURE/ENTRY USING のいずれかをコーディングします。

リテラル-2
以下のようになります。

- 英数字リテラル
- 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、15ページの『LENGTH OF』を参照してください。

ADDRESS OF ID-3
ID-3 には、リンケージ・セクション、ワーキング・ストレージ・セクション、またはローカル・ストレージ・セクションの任意の場所で定義されたデータ項目を指定できます。ID-3 には、レベル 66 またはレベル 88 を除いた任意のレベルを指定できます。

OMITTED
引き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、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、後続のすべての引き数に適用されます。

ある引き数に 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、その引き数の値が渡されます。呼び出されるプログラムは BY VALUE 引き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、その主の変更は、この引き数には適用されません。

BY VALUE 引き数は主として非 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、手続き部 USING 句内の対応する仮パラメーター内の両方の引き数に、BY VALUE を指定または暗黙指定しなければなりません。

ID-4
データ部内の基本データ項目にしなければなりません。これは、次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)
- オブジェクト・リファレンス (USAGE OBJECT REFERENCE)
- 1 つの単一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)

以下のものも、BY VALUE によって渡されます。

- 長さが 1 の usage display の参照変更項目
- 長さが 1 の usage national の参照変更項目
- SHIFT-IN および SHIFT-OUT 特殊レジスター
- LINAGE-COUNTER 特殊レジスター (バイナリー使用の場合)

ADDRESS OF ID-4

ID-4 には、リンケージ・セクション、ワーキング・ストレージ・セクション、またはローカル・ストレージ・セクションで定義されたデータ項目を指定できます。ID-4 には、レベル 66 またはレベル 88 を除いた任意のレベルを指定できます。

LENGTH OF 特殊レジスター

BY VALUE によって渡される 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、15ページの『LENGTH OF』を参照してください。

リテラル-3

以下のいずれかにする必要があります。

- 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
- 単一バイトの形象定数
  - 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。

リテラル-3 は、固定点数字リテラルである場合、9 またはそれ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 リテラルでなければなりません。

RETURNING 句

ID-5

データ部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 のリンケージ規則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。

- 呼び出されるプログラムでは、その手続き部のヘッダーに 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手続き部 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 文節により、PICTURE 文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)。

ID-6 が usage INDEX、POINTER、 FUNCTION-POINTER、PROCEDURE-POINTER、または OBJECT REFERENCE の場合、ターゲットが戻されるときは、SET ステートメントのどちらかの規則を使用して、その戻り値が ID-5 に割り当てられます。

注: 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。

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・ステートメントまたは条件ステートメントが実行される場合、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が使用可能であれば)、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。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ただちに、制御が CALL ステートメントの終了へと移されます。

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

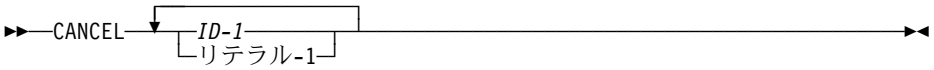
この明示的範囲終了符号は、CALL ステートメントの範囲を区切るために使用されます。END-CALL を使用すると、条件付き CALL ステートメントを別の条件付きステートメントにネストすることができます。END-CALL は、命令 CALL ステートメントとも使用できます。

詳細については、260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、86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、 *Enterprise COBOL プログラミング・ガイド* の PGMNAME コンパイラー・オプションの説明も参照してください。

ID-1 は、ウィンドウ化日付フィールドにはできません。

CANCEL ステートメントに指定する ID の各リテラルまたは内容は、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 ステートメントの中で参照されるプログラム名は、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、*Enterprise COBOL プログラミング・ガイド* を参照してください。

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 ステートメントによって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。これらのファイルのいずれかと関連する USE プロシージャーがあったとしても、それは実行されません。

呼び出されたサブプログラムを取り消すことができるのは、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が参照されたとき、そのサブプログラムがメンバーである実行単位が終了したとき、またはサブプログラムが INITIAL 属性を持つ場合にそのサブプロ

グラム内で EXIT PROGRAM ステートメントまたは GOBACK ステートメントが実行されたときです。

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 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、呼び出されるプログラムは、呼び出し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、呼び出されてもまだ EXIT PROGRAMまたは GOBACK ステートメントを実行していないプログラムを参照することはできません。

ただし、取り消し側プログラムは、呼び出し階層において取り消すプログラムより高位または等しいレベルであれば、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。以下に例を示します。

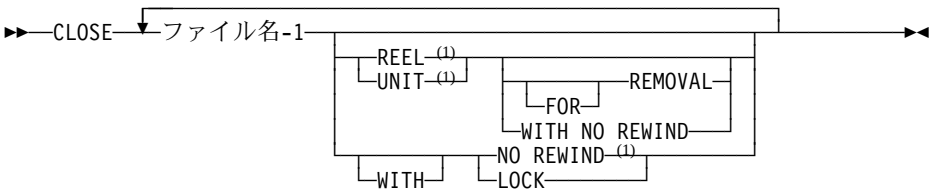
```
A calls B and B calls C        (A が制御を持つ場合、
                                C を取り消すことができる。)

A calls B and A calls C        (C が制御を持つ場合、
                                B を取り消すことができる。)
```

CLOSE ステートメント

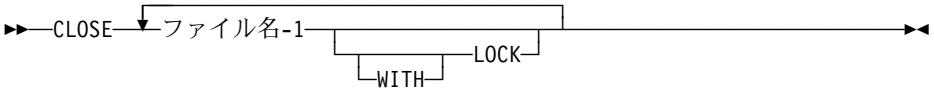
CLOSE ステートメントは、ボリュームおよびファイルに対しては意味を持ちません。

フォーマット 1 - 順次

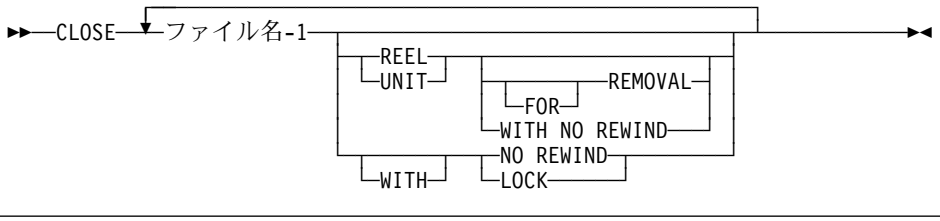


注:
[1] VSAM ファイルに関して REEL、UNIT、および NO REWIND 句は無効です。

フォーマット 2 - 索引付きファイルおよび相対ファイル



フォーマット 3 - 行順次ファイル



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/UNIT

これらの句は、QSAM のマルチボリューム・ファイルまたは単一ボリューム・ファイルに対してだけ指定できます。用語 REEL と UNIT は、どちらを使用しても同じになります。

WITH NO REWIND および FOR REMOVAL

これらの句は、QSAM テープ・ファイルに対してだけ適用されます。これらの句は、適用されない記憶装置に指定されている場合、無視されます。

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。

- 他の入出力ステートメントを実行する前に、そのファイルに対して OPEN ステートメントを実行しなければなりません。

FILE-CONTROL 記入項目内に FILE STATUS 文節が指定されている場合は、関連する状況キーが、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

あるファイルに対して FILE-CONTROL 記入項目内に SELECT OPTIONAL 文節が指定されている場合、そのファイルが実行時に存在しないと、標準のファイル終了処理は行われません。QSAM ファイルの場合は、ファイル位置標識および現行ボリューム・ポインターは変更されません。

ファイルは、以下のタイプに分けられます。

リール / 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。VSAM のファイルは、すべてリール / ユニット・タイプ以外のファイルです。QSAM ファイルは、リール / ユニット・タイプ以外のファイルにすることができます。

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。VSAM のファイルはすべて単一ボリュームに置かれます。QSAM ファイルは単一ボリュームにできます。

順次マルチボリューム
複数のボリュームに入っている順次ファイル。QSAM ファイルは、マルチボリュームにできる唯一のファイルです。ボリュームの概念は、VSAM のファイルに対しては意味を持ちません。

CLOSE ステートメントの許可される組み合わせについては、次の表を参照してください。

- 表35 - 順次ファイル
- 294ページの表36 - 索引ファイルおよび相対ファイル
- 294ページの表37 - 行順次ファイル

各英字の意味は、294ページの表38 に示されています。

表 35 (1/2). 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 REEL/UNIT	F	F、G	F、G
CLOSE REEL/UNIT WITH NO REWIND	F	B, F	B, F

表 35 (2/2). 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 REEL/UNIT FOR REMOVAL	D	D	D
CLOSE WITH NO REWIND	C、H	B、C	A、B、C
CLOSE WITH LOCK	C、E	C、E、G	A、C、E、G

表 36. 索引ファイル・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 37. 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 38 (1/2). 順次ファイル・タイプのキーの意味

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)。
B	**現在のリールは巻き戻されない -** 現在のボリュームは現在位置のままになる。

表 38 (2/2). 順次ファイル・タイプのキーの意味

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。
D	**ボリュームを取り外す** - コメントとして扱われる。
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・アンロードされる。
F	**ボリュームをクローズする** **入力ファイルおよび入出力ファイル** - 現在のリール / ユニットがファイルに対して最後であるか、または唯一のリール / ユニットであるか、あるいはリールが非リール / ユニット・メディア上にある場合は、ボリューム切り替え処理は行われない。ファイルに関して別のリール / ユニットが存在している場合は、次の操作が実行される。ボリューム切り替え、最初のボリューム・ラベル・プロシージャー、さらに新規ボリューム上の最初のレコードが読み取り可能になる。現行ボリュームにデータ・レコードが存在しない場合は、別のボリュームへの切り替えが行われる。 **出力 (リール / ユニット・メディア) ファイル** - 終了ボリューム・ラベル・プロシージャー、ボリューム切り替え、および開始ボリューム・ラベル・プロシージャーが実行される。次の WRITE ステートメントが実行されると、次の論理レコードが使用可能な次の直接アクセス・ボリューム上に置かれる。REEL 句でクローズ・ステートメントを使用しても、出力ファイルはクローズされない。ボリューム終了条件が起きるだけである。 **出力 (非リール / ユニット・メディア) ファイル** - CLOSE ステートメントの実行は正しく行われたとみなされる。ファイルはオープン・モードのままであり、このファイルに関連付けられた I-O 状況キーの値が更新される以外、何も処置は行われない。
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。
H	**オプションの句を無視する** - CLOSE ステートメントは、オプションの句がなにも指定されていないものとして、実行される。

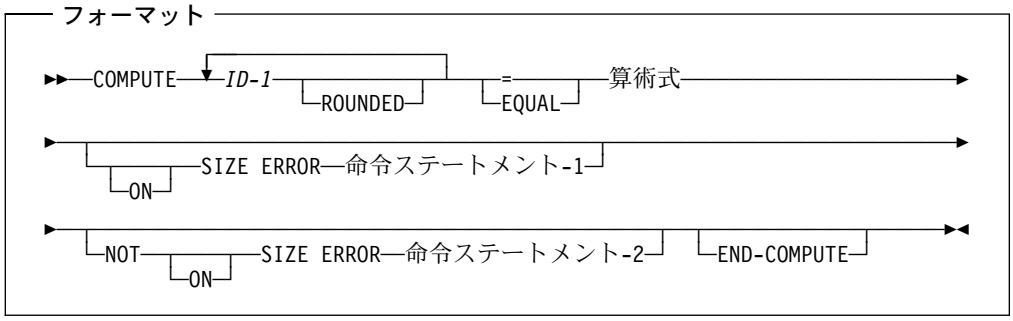
COMPUTE ステートメント

COMPUTE ステートメントは、算術式の値を 1 つまたは複数のデータ項目に割り当てます。

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 (単数または複数)、または基本数字編集項目 (単数または複数) を指定します。

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、234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

230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、その値が ID-1 によって参照される各データ項目の新しい値として保管されます。

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 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、262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、263ページの『SIZE ERROR 句』を参照してください。

END-COMPUTE 句

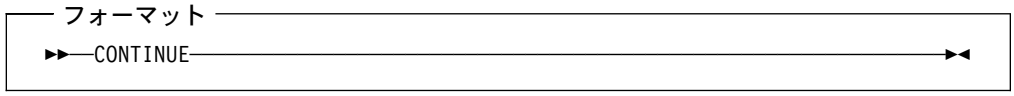
この明示的範囲終了符号は、COMPUTE ステートメントの範囲を区切るために使用されます。END-COMPUTE 句を使用することによって、条件付き COMPUTE ステートメントを別の条件ステートメントにネストすることができます。END-COMPUTE 句は、命令 COMPUTE ステートメントと共に使用することもできます。

詳細については、260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 ステートメントを使用することによって、ノーオペレーション・ステートメントを指定することができます。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。

フォーマット

ファイル名-1
データ部の FD 記入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 文節の 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。

FILE-CONTROL 記入項目に FILE STATUS 文節が指定されている場合は、DELETE ステートメントの実行時に、関連する状況キーが更新されます。

ファイル位置標識は、DELETE ステートメントの実行によって影響を受けることはありません。

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、最後に行われた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 句は、指定することは**できません。**ただし、EXCEPTION/ERROR プロシージャーは指定することができます。

ランダム・アクセス・モードまたは動的アクセス・モード

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 (267ページの『共通の処理機能』にある「無効キー条件」を参照)。

INVALID KEY 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

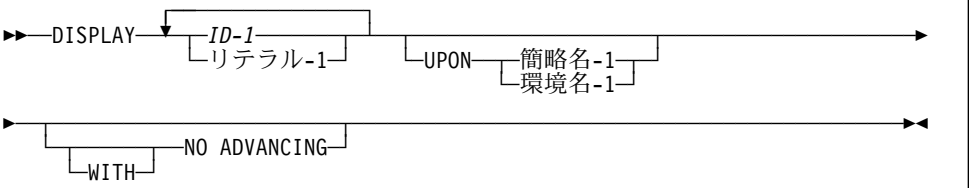
この明示的範囲終了符号は、DELETE ステートメントの範囲を区切るために使用されます。END-DELETE 句を使用することによって条件的な DELETE ステートメントを他の条件的なステートメントの中にネストすることができます。END-DELETE 句は、命令の DELETE ステートメントと共に使用することもできます。

詳細については、260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。

フォーマット



ID-1

ID-1 が数字で、外部 10 進数として記述されていない場合は、以下のようにして外部フォーマットに自動的に変換されます。

- 2 進数項目または内部 10 進数項目は、外部 10 進数項目に変換されます。負符号の付いた値では、最低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 文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 文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、外部 10 進数に変換されて、PIC 9(10) という PICTURE 文節を持ちます。

出力が CONSOLE に送信される場合、USAGE NATIONAL で定義されたデータ項目は、国別文字表示から EBCDIC に変換されます。変換では、ソース・コードのコンパイル時に CODEPAGE コンパイラー・オプションで指定された EBCDIC コード・ページが使用されます。EBCDIC 文字以外の国別文字はデフォルトの置換文字に変換されますが、例外条件が示されたり、発生したりすることはありません。

出力が CONSOLE に送信されない場合、USAGE NATIONAL で定義されたデータ項目は、変換やデータ妥当性検査を行わずに書き込まれます。

その他のデータ・カテゴリーは変換が不要です。

USAGE IS PROCEDURE-POINTER、 USAGE IS FUNCTION-POINTER、または USAGE IS OBJECT REFERENCE で定義されたデータ項目は、DISPLAY ステートメントに指定できません。

USAGE IS INDEX と共に定義された指標名またはデータ項目は、DISPLAY ステートメントでは指定できません。

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。正しい結果を得るには、DBCS のシフトアウトおよびシフトイン制御文字を認識する機能が出力装置に必要です。

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形象定数にできます。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。

すべての形象定数を使用できます。

UPON
簡略名または**環境名**は、SPECIAL-NAMES 段落内で出力装置と関連付けなければなりません。

各装置のデフォルトの論理レコード・サイズは、次のように想定されます。

システム論理出力装置 = 120 文字
システムせん孔装置 = 80 文字
コンソール = 100 文字

それぞれの装置に対して、最大の論理レコード・サイズは次のとおりです。

システム論理出力装置 = 255 文字
システムせん孔装置 = 255 文字
コンソール = 100 文字

注: システムせん孔装置では、最後の 8 文字が PROGRAM-ID 名として使用されます。

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、99ページの表5 にあります。

DISPLAY 出力を stdout にルーティングする方法については、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

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付ける機能を持っていれば、表示された最後のオペランドの最後の文字のすぐ後に文字位置が置かれたままになります。出力装置が特定の文字位置に位置付ける機能を持たなければ、該当する場合には、縦方向の位置だけが影響を受けます。この結果、上重ね印刷が行われる可能性があります。

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。

Enterprise COBOL は、特定の文字位置に位置決めできる出力装置をサポートしていません。DISPLAY ステートメントの詳細については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

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。

DBCS 項目を分割するには、シフト・コードを挿入する必要があります。つまり、DBCS オペランドを複数のレコードに分割するときは、現在のレコードの終わりにシフトイン文字を挿入し、次のレコードの始めにシフトアウト文字を挿入します。必要があれば、シフトイン文字の後にスペースを埋め込みます。挿入された追加のシフト・コードとスペースは、コンパイラーが必要とするレコード数を計算するとき、そのカウントに含められます。

最後のオペランドが出力装置に転送されると、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。

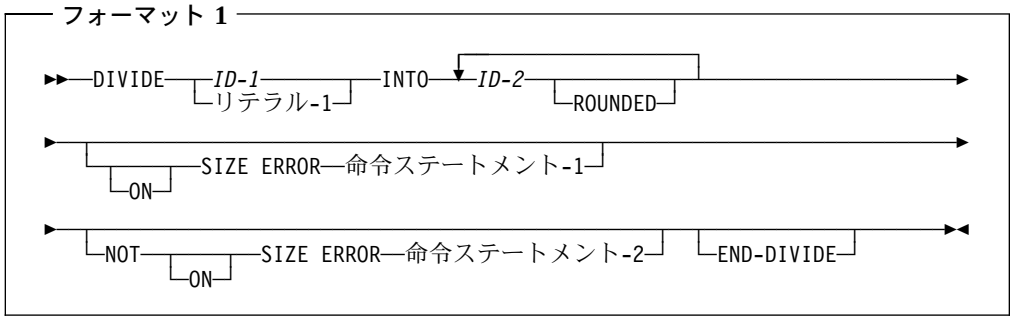
DBCS データ項目または DBCS リテラルが DISPLAY ステートメントに指定されている場合、送り出しフィールドのサイズは、リストされたすべてのオペランドのバイトの総数ですが、DBCS 文字の 1 文字を 2 バイトとしてカウントし、それに DBCS に必要なシフト・コードを加えます。

注: DISPLAY ステートメントを実行すると、プリンターは印刷**前**に行送りを行います。

DIVIDE ステートメント

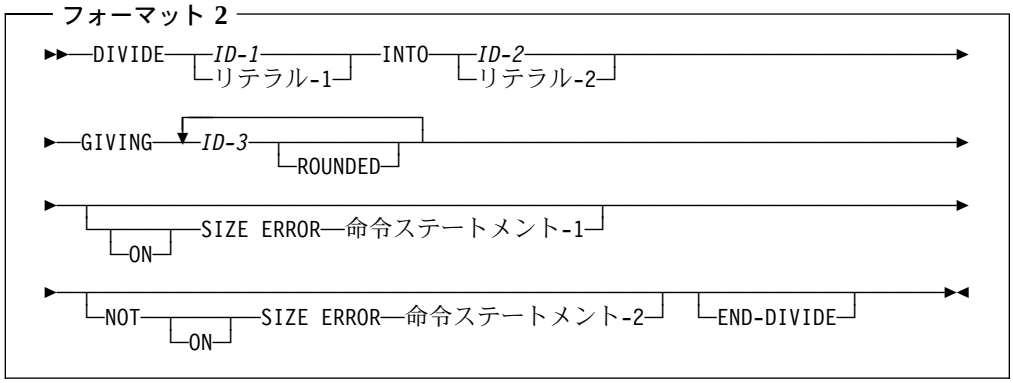
DIVIDE ステートメントは、ある数字データ項目を他の数字データ項目 (複数も可) で割り、データ項目の値を商と剰余に設定します。

フォーマット 1



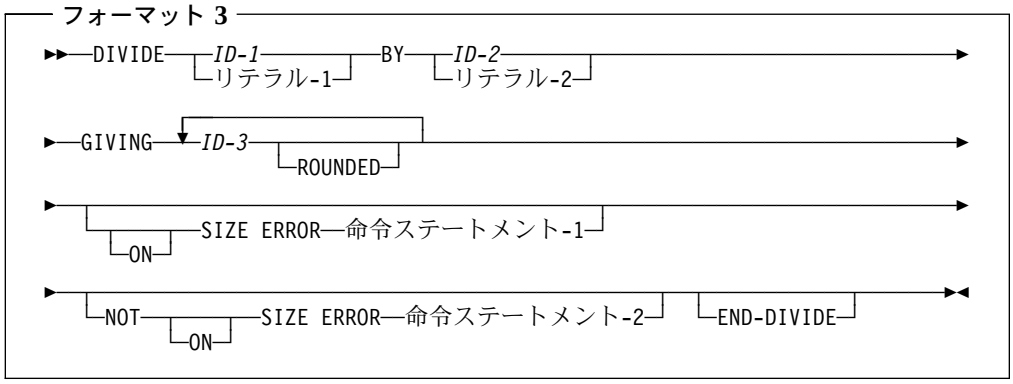
フォーマット 1 では、ID-1 またはリテラル-1 の値は、ID-2 の値に割り算されて、その商が ID-2 に保管されます。ID-2 が連続して現れるたびに、ID-2 が指定されている順に左から右へと割り算が行われます。

フォーマット 2



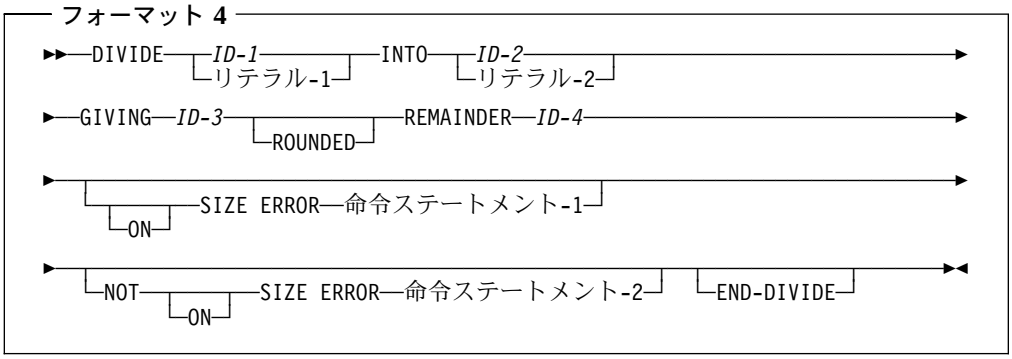
フォーマット 2 では、ID-1 またはリテラル-1 の値が割り算されて、ID-2 またはリテラル-2 の値になります。その商の値は、ID-3 によって参照される各データ項目に保管されます。

フォーマット 3



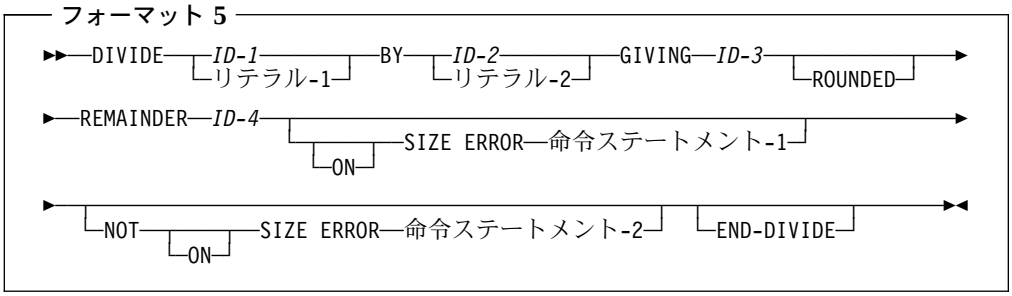
フォーマット 3 では、ID-1 またはリテラル-1 の値が、ID-2 またはリテラル-2 の値によって割り算されます。その商の値は、ID-3 によって参照される各データ項目に保管されます。

フォーマット 4



フォーマット 4 では、ID-1 またはリテラル-1 の値が割り算されて、ID-2 またはリテラル-2 の値になります。その商の値は ID-3 に、その剰余の値は ID-4 に保管されます。

フォーマット 5



フォーマット 5 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 によって割り算されます。その商の値は ID-3 に、その剰余の値は 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ようになります。

ID-1、ID-2
基本数字項目である必要があります。ID-1 と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、234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、262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 (商) 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、263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、263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対して使用し、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。

詳細については、260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

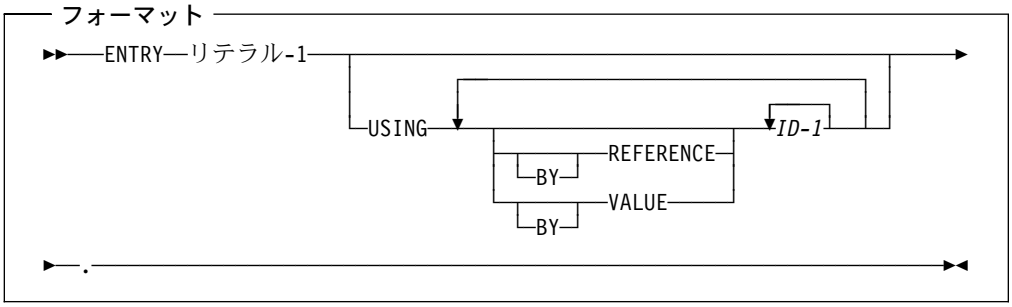
ENTRY ステートメントは、呼び出されたサブプログラムの代替入り口点を設定します。

ENTRY ステートメントは、次のプログラム中では使用**できません**。

- 手続き部 RETURNING 句を使用する戻り値を指定するプログラムでは使用できません。詳細については、224ページの『手続き部のヘッダー』の RETURNING の項を参照してください。
- ネストされたプログラムでは使用できません。ネストされたプログラムの詳細については、74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

英数字リテラルでなければならず、最も外側の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 (86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、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

USING 句

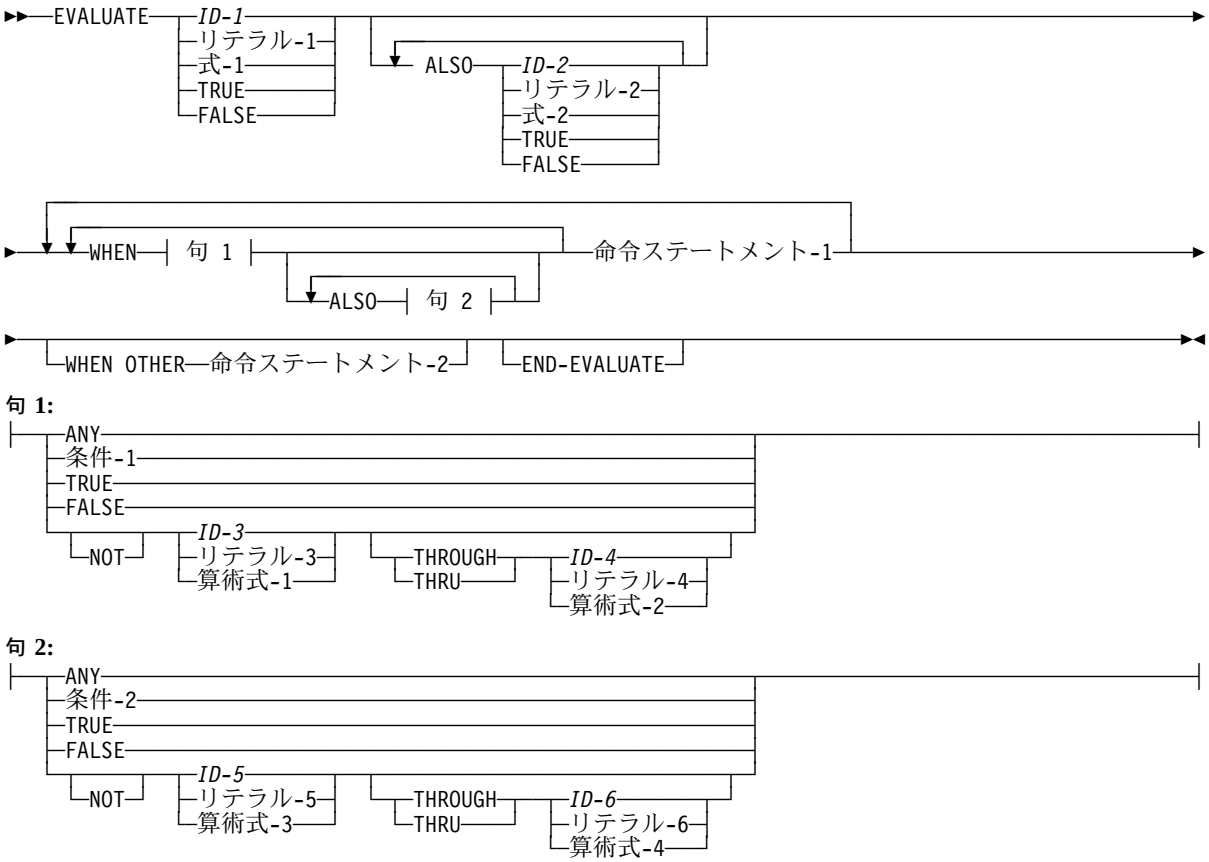
手続き部 ...RETURNING 句を含むプログラムでは、ENTRY ステートメントを指定しないでください。

USING 句の説明については、224ページの『手続き部のヘッダー』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。このステートメントは、複数の条件を評価することができます。つまり、IF ステートメントは複合条件で構成することが可能です。オブジェクト・プログラムの以降の処置は、これらの評価の結果次第です。

フォーマット



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連する比較については、241ページの『日付フィールド』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。

詳細については、260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、または国別値、数字、英数字 DBCS、または国別値の範囲、または真の値が代入さ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 (あるいはその両方) 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 (あるいはその両方) によって指定される選択オブジェクトには、指定し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 とリテラル-5 (またはその一方) が形象定数の ZERO、QUOTE、または SPACE である場合は、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 と算術式-3 (またはその一方) 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 (230ページの『算術式』を参照)。

- **条件**式として条件-1、条件-2、... が指定された選択サブジェクトと、条件-1 と条件-2 (またはその一方) 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 (235ページの『条件式』を参照)。
- キーワード TRUE または FALSE によって指定されたどの選択サブジェクトまたは選択オブジェクトにも、真の値が割り当てられます。真の値「真」は、キーワード TRUE を付けて指定された項目に割り当てられ、真の値「偽」は、キーワード FALSE を付けて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になります。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対して指定されている場合、その項目に割り当てられる値は、その項目に対して NOT 句が指定されていないとすれば割り当てられたはずの値、または範囲内の値を除く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、選択サブジェクトに割り当てられた値と選択オブジェクトに割り当てられた値を比較し、いずれかの WHEN 句が選択サブジェクトの集合を満たすかどうかを判別するように処理されます。この比較は、次のようにして行われます。

1. 選択オブジェクトの集合内にある各選択オブジェクトは、それぞれにとって最初の WHEN 句に出会うと、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、または国別値が割り当てられている場合、あるいは、数字、英数字、DBCS、または国別値が範囲で割り当てられている場合は、選択オブジェクトに割り当てられたその値、または範囲内の 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、比較は条件を満たしたことになります。

   b. 比較される項目が、真の値を割り当てられている場合には、両方の項目が同じ真の値を割り当てられていれば、比較は条件を満たしたことになります。

   c. 比較される選択オブジェクトが、キーワード ANY を付けて指定されている場合には、選択サブジェクトの持つ値に関係なく、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集合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には、その選択オブジェクトの集合を含む WHEN 句が、選択サブジェクトの集合を満たすものとして選ばれます。

3. 比較される選択オブジェクトの集合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かった場合には、その選択オブジェクトの集合は、選択サブジェクトの集合を満たさなかったことになります。

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・プログラム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

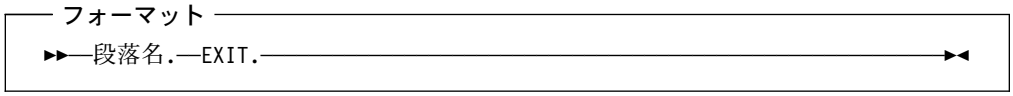
比較操作が完了すると、EVALUATE ステートメントの実行は、以下の順序で進められます。

- 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 句がまったく選択されず、しかも WHEN OTHER 句を指定していない場合には、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範囲は、選ばれた WHEN 句の範囲の終わりに達するか、または WHEN OTHER 句の範囲の終わりに達したときに終了します。あるいは、選ばれた WHEN 句が存在せず、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対して共通の終了点を与えます。

フォーマット



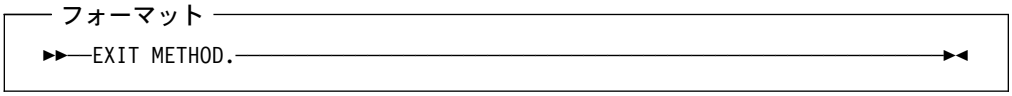
EXIT ステートメントを使用すると、プログラム内の所定の点にプロシージャー名を割り当てることができます。

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。

EXIT METHOD ステートメント

EXIT METHOD ステートメントは、呼び出されるメソッドの終わりを指定します。

フォーマット



EXIT METHOD はメソッドの手続き部にしか指定できません。EXIT METHOD を使用すると、メソッドの実行が終了され、制御は呼び出しステートメントに戻ります。内包メソッドで手続き部 RETURNING 句を指定している場合、その RETURNING 句によって参照されるデータ項目が、メソッド呼び出しの結果となります。

呼び出しごとに、メソッド特定のデータを**最後に使われた状態**にする必要がある場合は、それをメソッドワーキング・ストレージ内で宣言します。呼び出しごとに、メソッド特定のデータを**初期**状態にする必要がある場合は、それをメソッド・ローカル・ストレージ内で宣言します。

制御がメソッド定義内の EXIT METHOD ステートメントに達した場合、制御は呼び出しプログラム内の点に戻るか、または INVOKE ステートメントのすぐ後のメソッドに戻ります。呼び出しプログラムまたはメソッドの状態は、それが INVOKE ステートメントを実行した時点で存在していた状態と同じです。

ただしデータ項目の内容、および呼び出しプログラム (またはメソッド) と呼び出されるメソッド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。呼び出されるメソッドの状態は変更されませんが、メソッドによって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。

EXIT METHOD ステートメントを一連の命令ステートメントの最後のステートメントにする必要はありませんが、EXIT METHOD 以降のステートメントは実行されなくなります。

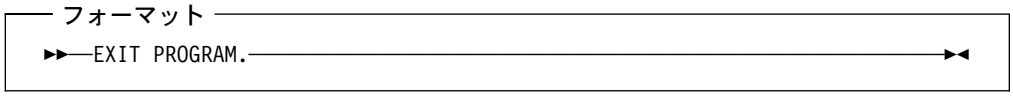
呼び出されるメソッド内に次に実行可能なステートメントがない場合は、暗黙の EXIT METHOD ステートメントが実行されます。

EXIT PROGRAM ステートメント

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、制御を呼び出しプログラムに戻します。

EXIT PROGRAM はプログラムの手続き部にしか指定できません。このステートメントは、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。

フォーマット



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、呼び出しプログラムのプログラム状態は、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、および呼び出しプログラム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プログラム状態は無変更ですが、ただし、その呼び出されたプログラムによって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。

サブプログラムに手続き部 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。

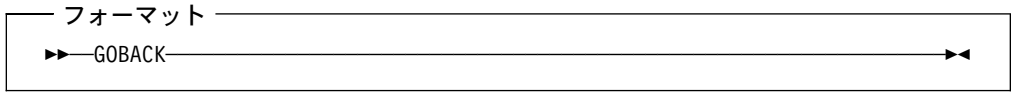
呼び出されたプログラムに次の実行可能ステートメントがない場合は、暗黙の EXIT PROGRAM ステートメントが実行されます。

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 (または EXIT METHOD ステートメントが呼び出されるメソッドの一部としてコーディングされている場合と同様)、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または呼び出されるメソッド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。このステートメントは、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。

INVOKE 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 METHOD ステートメントの場合と同じように、呼び出しプログラム中の INVOKE 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、メインプログラムとサブプログラムで GOBACK ステートメントに対して取られる処置を示したものです。

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させる)。	呼び出しプログラムへ戻る。

GO TO ステートメント

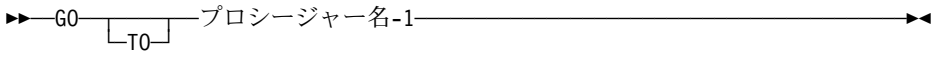
GO TO ステートメントは、手続き部のある部分から別の部分へ制御を移します。GO TO ステートメントには、次のような種類があります。

- 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指定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を移します。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ていた場合はこの限りではありません (281ページの『ALTER ステートメント』を参照)。

フォーマット 1 - 無条件



プロシージャー名-1
GO TO ステートメントと同じ手続き部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。

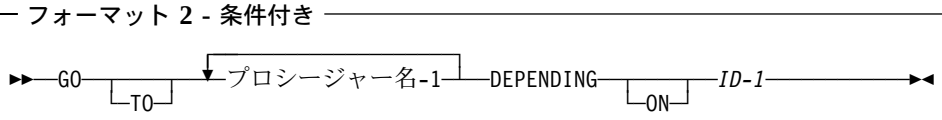
無条件 GO TO ステートメントが一連の命令ステートメントの先頭または途中に指定されていると、その後のステートメントが実行されません。

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 ステートメント

条件付き GO TO ステートメントは、ID 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



プロシージャー名-1
その GO TO ステートメントと同じ手続き部の中にあるプロシージャーまたはセクションである必要があります。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 は、ウィンドウ化日付フィールドにはできません。

1 を指定すると、最初のプロシージャー名-1 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、2 番目のプロシージャー名-1 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 ～ n (n はこの GO TO ステートメント中で指定されているプロシージャー名の個数) の範囲以外の値である場合、制御の移動は起きません。その代わり、制御は、正規の実行シーケンスの次のステートメントに渡されます。

変更される GO TO

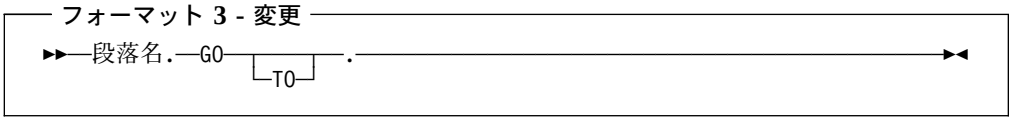
変更される GO TO ステートメントは、ALTER ステートメント中に指定された段落の最初のステートメントに制御を移します。

変更される GO TO ステートメントは、以下のものには指定できません。

- RECURSIVE 属性を持つプログラムまたはメソッド。
- THREAD コンパイラー・オプションを指定してコンパイルしたプログラム

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

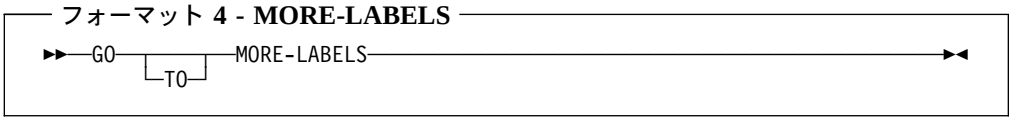


ALTER ステートメントがある段落を参照している場合、その段落は無条件 GO TO ステートメントまたは変更される GO TO ステートメントを後に付けた段落名のみで構成することもできます。

MORE-LABELS GO TO

GO TO MORE-LABELS ステートメントは、LABEL 宣言部分の中でのみ指定することができます。

フォーマット 4 - MORE-LABELS

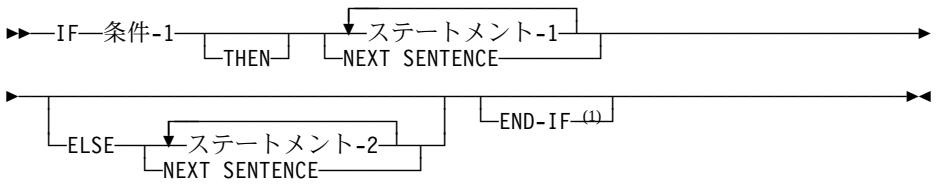


詳しくは、*Enterprise COBOL プログラミング・ガイド* を参照してください。

IF ステートメント

IF ステートメントは条件を評価し、その評価結果に応じてオブジェクト・プログラム中で処置を選択できるようにします。

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

これは、単純条件にも複合条件にもすることができます (235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

次のいずれかにできます。

- 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**区切りピリオド**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。

注: 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についての詳細は、260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、次のいずれかにより終わらせることができます。

- ネスト構造で同じレベルにある END-IF 句。
- 分離記号のピリオド。
- 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

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、それが実行されます。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・ステートメントが入っていない場合は、ELSE 句が指定されていても無視され、対応する END-IF 句の後、または分離記号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、次の分離記号の直前にある暗黙の 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、次の処置の 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、それが実行されます。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-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 句の後または区切り文字の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は、制御は、次の分離記号の直前にある暗黙の CONTINUE STATEMENT に渡されます。
- ELSE ステートメント-2 も ELSE NEXT SENTENCE も指定していない場合、制御は、対応する END-IF または分離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。

注: 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、文の分離記号である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 やステートメント-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 (IF ステートメントに IF ステートメントが入っている場合) 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、最も近い場所にある先行の IF (まだ ELSE と一致していないか、または暗黙的あるいは明示的に終了されていないもの) と一致します。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。

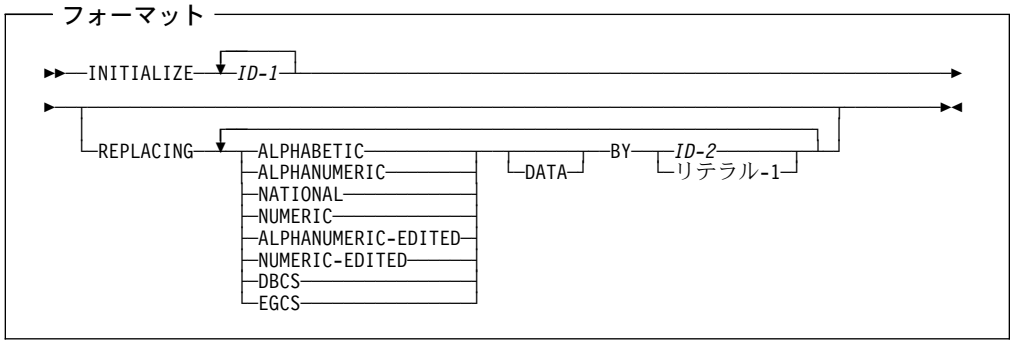
INITIALIZE ステートメント

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。このステートメントは、機能的には 1 つまたは複数の MOVE ステートメントを実行するのと同じことになります。

REPLACING 句を使用しないときは、

- SPACE が、英字、英数字、国別、英数字編集、および DBCS 項目の暗黙の送り出しフィールドとなります。
- ZERO が、数字項目、数字編集項目の暗黙の受け取りフィールドとなります。

フォーマット



ID-1

受け取り領域 (単数または複数)。

ID-2、リテラル-1

送り出し領域 (単数または複数)。

ID-2 またはリテラル-1 のクラスが国別の場合は、REPLACING 句の NATIONAL オプションを指定する必要があります。

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。

注: ID-1 のデータ記述記入項目には、OCCURS 文節の DEPENDING ON 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、同じ 01 レベル項目内における、OCCURS 文節の DEPENDING ON 句に続く随所の項目またはグループを初期化することはできません。

ID-1 のデータ記述記入項目に、RENAMES 文節を入れることはできません。指標データ項目は、INITIALIZE ステートメントのオペランドにす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。

REPLACING 句

REPLACING 句を使用するときは、

- ID-2 またはリテラル-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 ステートメント内の送り出しフィールドとして有効なカテゴリーにする必要があります。浮動小数点データ項目や浮動小数点リテラルは、NUMERIC のカテゴリーに属しているかのように処理されます。
- 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。
- REPLACING という語に続くキーワードは、142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。

DBCS
EGCS
DBCS リテラルで許される文字を指します。

INITIALIZE ステートメントの規則

1. ID-1 が基本項目として記述されているか、またはグループ項目として記述されているかに関係なく、処理はすべて、それぞれが受け取りフィールドとして基本項目を持つ一連の MOVE ステートメントが書かれているかのように行われます。

   REPLACING 句が指定されている場合、

   - ID-1 がグループ項目を参照している場合、そのデータ項目内にある ID-1 カテゴリーによって参照されるいずれの基本項目も、それが REPLACING 句の中で指定されたカテゴリーに属している場合に限り、初期設定されます。
   - ID-1 が基本項目を参照している場合、その項目は、REPLACING 句で指定されたカテゴリーに属している場合に限り初期設定されます。

   この初期設定は、ID-2 またはリテラル-1 によって参照されるデータ項目が、暗黙の MOVE ステートメントで識別された項目に対して送り出しオペランドとして動作するかのようにして起こります。

   次の例外を除き、そのような基本受け取り項目は、グループ中のテーブル項目のオカレンスも含めて、すべて影響を受けます。

   - 指標データ項目
   - オブジェクト・リファレンス
   - USAGE IS POINTER、USAGE IS FUNCTION-POINTER、または USAGE IS PROCEDURE-POINTER で定義されているデータ項目
   - 基本 FILLER データ項目
   - ID-1 に従属し REDEFINES 文節を含む項目、またはそのような項目に従属するすべての項目 (ただし、ID-1 は REDEFINES 文節を含むことも、または再定義する項目に従属することもできます)。

2. ID-1 によって参照される領域は、ID-1 がステートメントに現れる順 (左から右) に初期設定されます。グループ受け取りフィールド内では、影響を受ける基本項目は、それらがグループ内で定義されている順に初期設定されます。

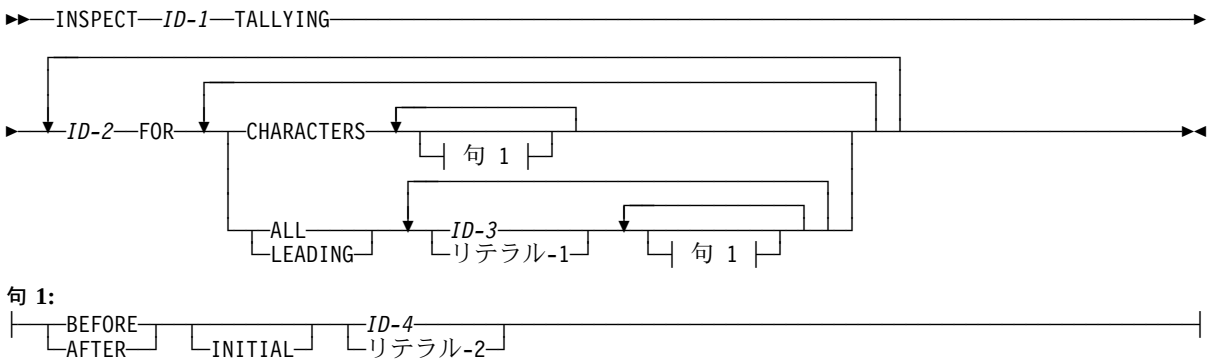
3. ID-1 が ID-2 と同じストレージ域に置かれる場合は、このステートメントの実行結果は、これらのオペランドが同じデータ記述記入項目によって定義されている場合であっても、未定義のままです。

INSPECT ステートメント

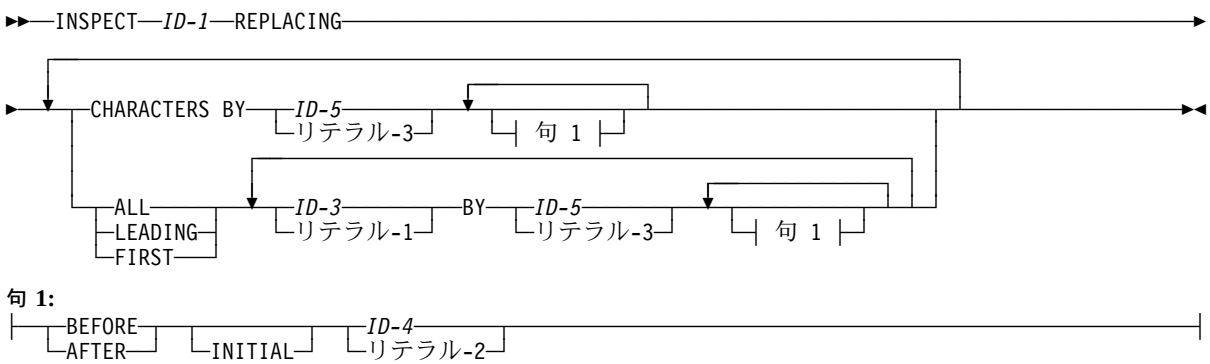
INSPECT ステートメントは、データ項目内の文字または文字のグループを数えたり、置き換えたり、または数えて置き換えたりすることを指定します。

- データ項目内で特定の文字 (英数字、DBCS、または国別) のオカレンスの回数を数えます (フォーマット 1 および 3)。
- 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)。

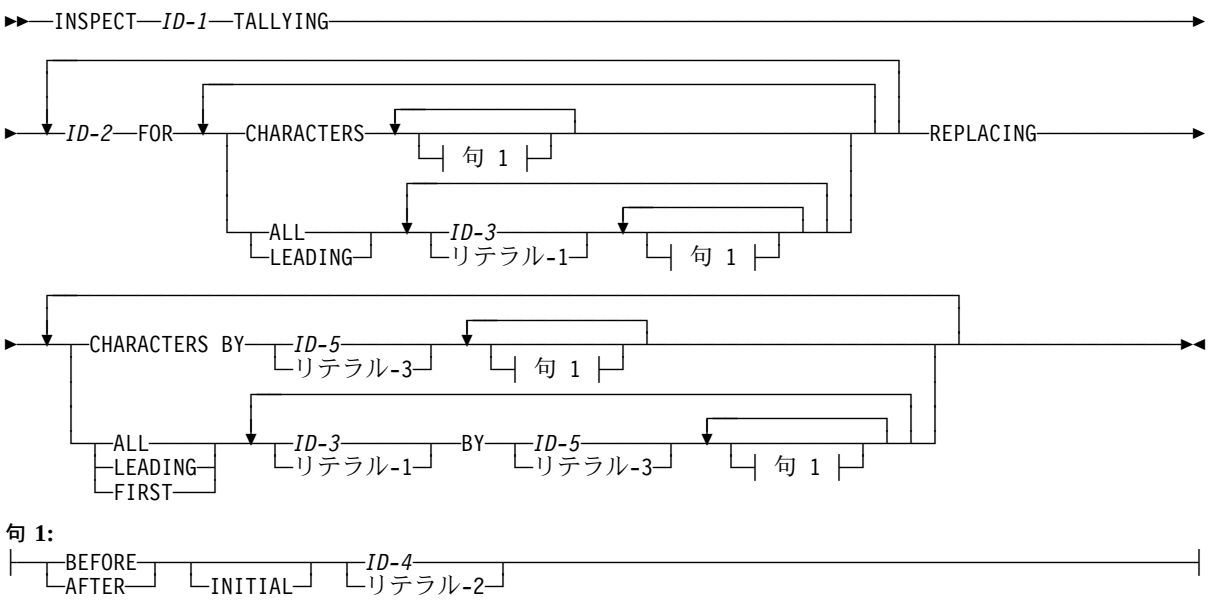
フォーマット 1



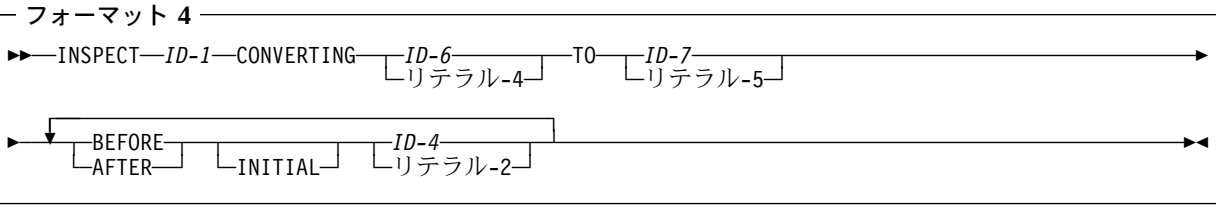
フォーマット 2



フォーマット 3



フォーマット 4



INSPECT ステートメントのどの ID もウィンドウ化日付フィールドにはできません。

ID-1
これは**検査項目**であり、次のいずれかにできます。

- 英数字データ項目
- USAGE DISPLAY を指定した数字データ項目
- DBCS データ項目
- 国別データ項目
- 外部浮動小数点項目

ID およびリテラル (ID-2 を除く) は、どちらかが DBCS 項目であれば、すべてが DBCS 項目、DBCS リテラルまたは DBCS データ項目のいずれかでなければなりません。ID-2 は、DBCS 項目にできません。データのバイト数ではなく DBCS 文字数が、累計されて ID-2 に入れられます。

ID およびリテラル (ID-2 は除く) は、どちらかのクラスが国別であれば、すべてが国別データ項目、または国別リテラルでなければなりません。国別文字の位置は、累計されて ID-2 に入れられます。

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がデータ項目内のオカレンスの回数を数えます。

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 が国別データ項目の場合は国別文字が数えられます。それ以外の場合は、英数字文字 (バイト) が数えられます。

ID-2

これは**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、外部浮動小数点項目にはできません。

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。

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。

ID-3 は、次のいずれかにできます。

- 基本英数字データ項目
- DBCS データ項目
- 国別データ項目
- USAGE DISPLAY を指定した数字データ項目
- 外部浮動小数点項目

リテラル-1 は、ALL という語で始まらない形象定数にすることができます。 ID-1 が国別の場合、形象定数は、1 文字の国別リテラルであるとみなされます。 ID-1 が DBCS の場合は、形象定数を指定できません。それ以外の場合、形象定数は、1 文字の英数字リテラルであるとみなされます。

CHARACTERS

CHARACTERS が指定されていても、BEFORE も AFTER も指定されていない場合は、検査される項目 (ID-1) 内で、カウント・フィールド (ID-2) が 1 文字ごとに 1 ずつ増加されます。したがって、INSPECT TALLYING 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、被検査項目の中にある文字位置数だけ増えることになります。

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、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、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ランスは、この累計被比較数が関与する最初の比較において、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 または AFTER 句が指定されていない場合、置換フィールドは指定の項目 (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-3 は、以下が可能です。

- 基本英数字データ項目
- DBCS データ項目
- 国別データ項目
- USAGE DISPLAY を指定した数字データ項目
- 外部浮動小数点項目

リテラル-1 は、英数字、DBCS、または国別でなければなりません。ID-1 が国別の場合は、リテラル-1 も国別でなければなりません。 ID-1 が DBCS の場合は、リテラル-1 も DBCS でなければなりません。

リテラル-1 は、ALL という語で始まらない形象定数にすることができます。 ID-1 が国別の場合、形象定数は、1 文字の国別リテラルであるとみなされます。 ID-1 が DBCS の場合は、形象定数を指定できません。それ以外の場合、形象定数は、1 文字の英数字リテラルであるとみなされます。

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。

ID-5 は、以下が可能です。

- 基本英数字データ項目
- DBCS データ項目
- 国別データ項目
- USAGE DISPLAY を指定した数字データ項目
- 外部浮動小数点項目

リテラル-3 は、任意の形象定数にすることができます (ただし、ALL では始まらないこと)。形象定数の長さは、サブジェクト・フィールドの長さと同じです。 ID-1 が国別の場合、形象定数は、国別リテラルであるとみなされます。 ID-1 が DBCS の場合は、形象定数を指定できません。それ以外の場合、形象定数は、英数字リテラルであるとみなされます。

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、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すべて置換します。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、置換フィールドは指定の項目 (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。

置換規則
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、単一文字置換フィールド (1 文字の長さでなければならない) 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、この 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 またはリテラル-2
これは、**区切り文字**です。

ID-4 は、以下が可能です。

- 基本英数字データ項目
- DBCS データ項目
- 国別データ項目
- USAGE DISPLAY を指定した数字データ項目
- 外部浮動小数点項目

リテラル-2 は、任意の形象定数にすることができます (ただし、ALL では始まらないこと)。 ID-1 が国別の場合、形象定数は、1 文字の国別リテラルであるとみなされます。ID-1 が DBCS の場合は、形象定数を指定できません。それ以外の場合、形象定数は、1 文字の英数字リテラルであるとみなされます。

区切り文字はカウントまたは置換は行われません。ただし、被検査項目のカウントや置換文字がこれらの ID やリテラルによって区切られます。

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 (またはその両方) 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、カウントや置換は、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 (またはその両方) 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。

ID-6 は、以下が可能です。

- 基本英数字データ項目
- DBCS データ項目
- 国別データ項目
- USAGE DISPLAY を指定した数字データ項目
- 外部浮動小数点項目

リテラル-4 は、任意の形象定数にすることができます (ただし、ALL では始まらないこと)。 ID-1 が国別の場合、形象定数は、1 文字の国別リテラルであるとみなされます。ID-1 が DBCS の場合は、形象定数を指定できません。それ以外の場合、形象定数は、1 文字の英数字リテラルであるとみなされます。

リテラル-4 と ID-6 のいずれにも、同じ文字が 2 回以上現れることはできません。

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。

ID-7 は、以下が可能です。

- 基本英数字データ項目
- DBCS データ項目
- 国別データ項目

- USAGE DISPLAY を指定した数字データ項目
- 外部浮動小数点項目

リテラル-5 は、任意の形象定数にすることができます (ただし、ALL では始まらないこと)。形象定数の長さは、ID-6 またはリテラル-4 の長さと同じです。 ID-1 が国別の場合、形象定数は、国別リテラルであるとみなされます。 ID-1 が DBCS の場合は、形象定数を指定できません。それ以外の場合、形象定数は、英数字リテラルであるとみなされます。

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、このステートメントの実行結果は、たとえそれらが同じデータ記述記入項目によって定義されている場合であっても、未定義のままです。

ID およびリテラルのデータ型

表 39. 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内容	処理
英数字または英字項目	英数字文字ストリングとして処理される。
英数字編集項目、数字編集項目、または符号なし数字 (外部 10 進数) 項目	英数字項目を参照する INSPECT ステートメントで、英数字として再定義される。
DBCS 項目	DBCS 文字ストリングとして処理される。
国別項目	国別文字ストリングとして処理される。
符号付き数字 (外部 10 進数) 項目	同じ長さの符号なし外部 10 進数項目に移動され、英数字項目を参照する INSPECT ステートメントで、英数字として再定義される。 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。
外部浮動小数点データ項目	英数字項目を参照する INSPECT ステートメントで、英数字として再定義される。

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。

以降の句の比較は、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 と REPLACING の例については、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

比較の周期

比較の周期は、以下のアクションで構成されます。

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ます。

   CHARACTERS 句を指定する場合、暗黙の 1 文字の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. 最初の被比較項目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項目が処理されるまで、後続のそれぞれの被比較項目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 ～ 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、327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

331ページの図9 は、INSPECT ステートメントの結果の例です。

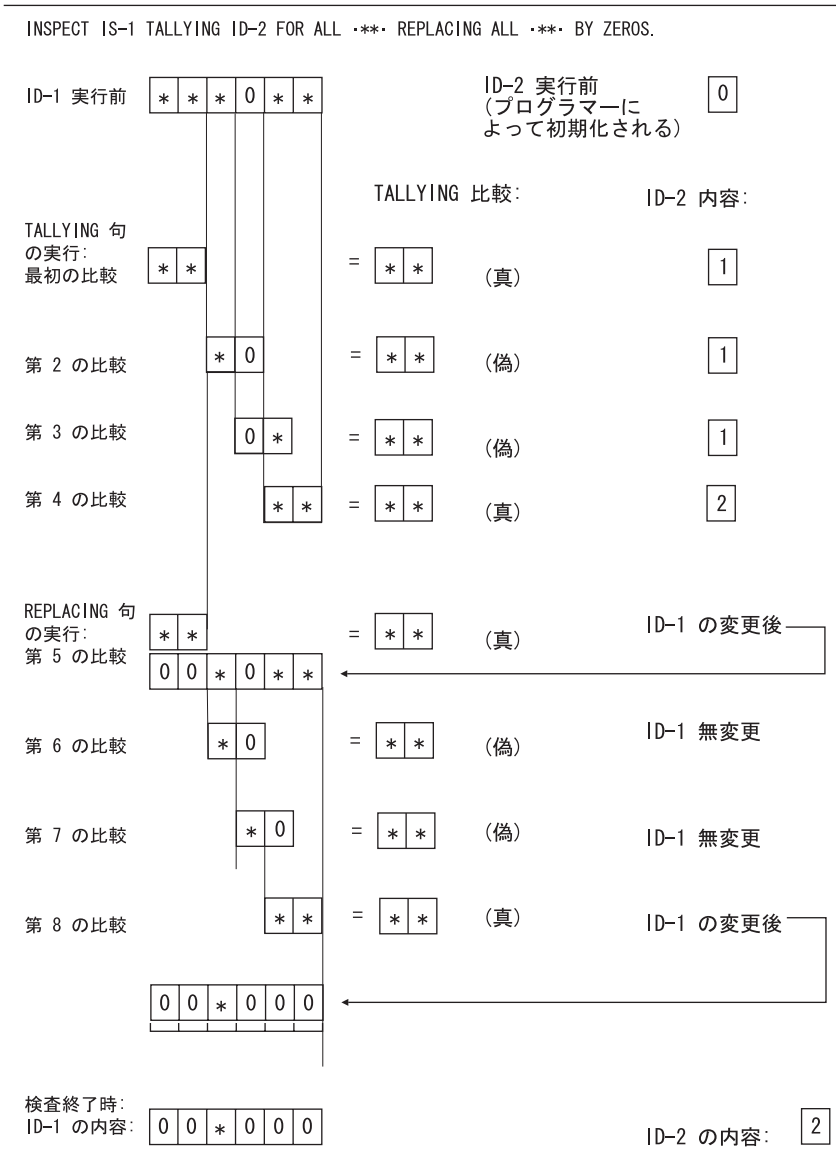
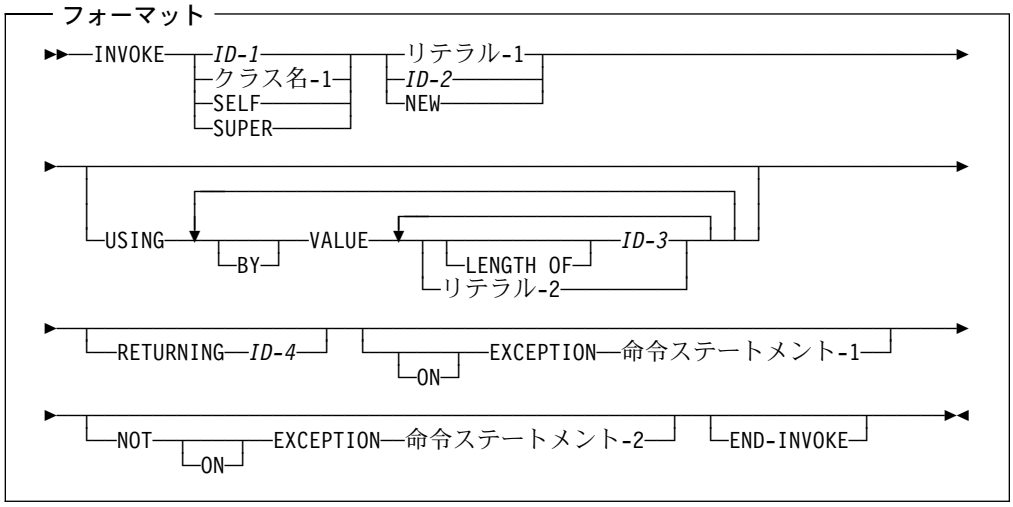


図 9. INSPECT ステートメントの実行結果の例

INVOKE ステートメント

INVOKE ステートメントは、COBOL または Java クラスのオブジェクト・インスタンスを生成したり、COBOL または Java クラスで定義されたメソッドを呼び出したりします。

フォーマット



ID-1

USAGE OBJECT REFERENCE として定義しなければなりません。ID-1 の内容は、メソッドが呼び出されるオブジェクトを指定します。

ID-1 を指定するときは、リテラル-1 か ID-2 を指定し、呼び出すメソッドの名前を識別する必要があります。

以下の場合、INVOKE ステートメントの結果は未定義です。

- ID-1 にオブジェクトへの有効な参照が入っていない場合
- ID-1 に NULL が入っている場合

クラス名-1

クラス名-1 をリテラル-1 か ID-2 と共に指定すると、INVOKE ステートメントにより、クラス名-1 の静的メソッドまたはファクトリー・メソッドが呼び出されます。リテラル-1 または ID-2 は、呼び出されるメソッドの名前を指定します。クラス名-1 が Java クラスの場合は、メソッドを静的メソッドにする必要があります。クラス名-1 が COBOL クラスの場合は、メソッドをファクトリー・メソッドにする必要があります。

クラス名-1 を NEW と共に指定すると、INVOKE ステートメントにより、クラス名-1 クラスのインスタンスである新しいオブジェクトが生成されます。

クラス名-1 は、INVOKE ステートメントが入っているクラスまたプログラムの構成セクションの REPOSITORY 段落に指定する必要があります。

SELF

現在実行中のメソッドを呼び出す際に使用されたオブジェクトへの暗黙の参照です。SELF が指定されている場合、INVOKE ステートメントはメソッドの手続き部内に現れなければなりません。

SUPER
現在実行中のメソッドを呼び出す際に使用されたオブジェクトへの暗黙の参照です。呼び出されるメソッドを解決する際は、現在実行中のメソッドのクラス定義内で宣言されているメソッドと、そのクラスから派生されたクラス内で定義されているメソッドは無視されます。したがって、呼び出されるメソッドは、親クラスから継承されたものになります。

リテラル-1
リテラル-1 の値は、呼び出すメソッドの名前です。参照されるメソッドは、リテラル-1 によって識別されるメソッドをサポートしなければなりません。

リテラル-1 は、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。

リテラル-1 は、大文字小文字を区別して解釈されます。INVOKE ステートメントの USING 句に指定したメソッド名、引き数の数、および引き数のデータ型は、オブジェクトでサポートされた一致するシグニチャーを持つメソッドを選択する際に使用されます。メソッドは、オーバーロードすることができます。

ID-2
呼び出されるメソッドの名前が実行時に入れられる国別データ項目または英数字データ項目です。参照されるメソッドは、ID-2 によって識別されるメソッドをサポートしなければなりません。

ID-2 を指定する場合は、オプションの句を指定せずに、ID-1 を USAGE OBJECT REFERENCE として定義する必要があります。つまり、ID-1 はユニバーサル・オブジェクト・リファレンスでなければなりません。

リテラル-2 の内容は、大文字小文字を区別して解釈されます。INVOKE ステートメントの USING 句に指定したメソッド名、引き数の数、および引き数のデータ型は、オブジェクトでサポートされた一致するシグニチャーを持つメソッドを選択する際に使用されます。メソッドは、オーバーロードすることができます。

ID-2 は、ウィンドウ化日付フィールドにはできません。

NEW
NEW オペランドは、INVOKE ステートメントでクラス名-1 クラスの新しいオブジェクト・インスタンスを生成することを指定します。クラス名-1 は指定する必要があります。

クラス名-1 を Java でインプリメントした場合は、INVOKE ステートメントの USING 句を指定できます。INVOKE ステートメントの USING 句に指定した引き数の数と引き数のデータ型は、クラスでサポートされた一致するシグニチャーを持つ Java コンストラクターを選択する際に使用されます。クラス名-1 クラスのオブジェクト・インスタンスは割り振られ、選択されたコンストラクター (または、デフォルトのコンストラクター) は実行され、生成されたオブジェクトへの参照は戻されます。

クラス名-1 を COBOL でインプリメントした場合は、INVOKE ステートメントの USING 句を指定することはできません。クラス名-1 クラスのオブジェクト・インスタンスは割り振られ、インスタンス・データ項目は関連する VALUE 文節で指定された値に初期化され、生成されたオブジェクトへの参照は戻されます。

NEW を指定する場合は、RETURNING 句も指定する必要があります。詳しくは、334 ページの『RETURNING 句』を参照してください。

USING 句

USING 句は、ターゲット・メソッドに渡される引き数を指定します。引き数のデータ型および引き数のリンケージ規約は、Java でサポートされるものに制限されます。

BY VALUE 句

INVOKE ステートメントに指定する引き数は、BY VALUE によって渡す必要があります。

BY VALUE 句は、送り出すデータ項目への参照ではなく、引き数の値を渡すことを指定します。呼び出されるメソッドは、BY VALUE 引き数に対応する仮パラメーターを修正できますが、呼び出されるメソッドは送信する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、変更は引き数には適用されません。

ID-3
データ部内の基本データ項目にしなければなりません。ID-3 のデータ型は、Java 相互協調処理に対応したデータ型にする必要があります。詳しくは、336ページの『COBOL と Java の相互運用可能なデータ型』を参照してください。

ID-3 がオブジェクト・リファレンスである場合は、そのオブジェクト・リファレンスのデータ記述記入項目に、クラス名を明示的に指定する必要があります。つまり、ID-3 は、ユニバーサル・オブジェクト・リファレンスにはしないでください。

339ページの『COBOL と Java における各種の引き数の型』には、ID-3 としてもサポートされる各種の例と、それらに対応する Java 型が示されています。

リテラル-2
これは、Java 相互協調処理に適したタイプにする必要があります。サポートされるリテラルの形式とそれらに対応する Java 型については、339ページの『COBOL と Java における各種の引き数の型』を参照してください。

リテラル-2 は DBCS リテラルでなければなりません。

LENGTH OF *ID-3*
ID-3 の長さを LENGTH OF 特殊レジスターの引き数として渡すことを指定します。BY VALUE によって渡される LENGTH OF 特殊レジスターは、PIC 9(9) バイナリー値として扱われます。LENGTH OF 特殊レジスターの詳細については、15ページの『LENGTH OF』を参照してください。

RETURNING 句

RETURNING 句は、呼び出したメソッドから戻された値を入れるデータ項目を指定します。INVOKE ステートメントに RETURNING 句を指定できるのは、COBOL または Java で作成されたメソッドを呼び出すときです。

ID-4
RETURNING データ項目。ID-4 は、以下のように扱われます。

- DATA DIVISION 内に定義しなければなりません。
- 参照変更にすることはできません。
- 例外が起きる場合は、変更されません。

ID-4 のデータ型は、Java 相互協調処理に対応したデータ型にする必要があります。詳しくは、336ページの『COBOL と Java の相互運用可能なデータ型』を参照してください。

一般に、ID-4 がオブジェクト・リファレンスの場合は、特定のクラスに合わせて入力されたオブジェクト・リファレンスでなければなりません。ただし、例外として、INVOKE ステートメントに NEW オペランドが指定されている場合は、戻される項目としてユニバーサル・オブジェクト・リファレンスもサポートされます。

ID-4 が指定され、ターゲット・メソッドが COBOL で作成されている場合、そのターゲット・メソッドの手続き部ヘッダーには RETURNING 句が必要です。ターゲット・メソッドが戻されるとき、ID-4 が USAGE OBJECT REFERENCE で記述されている場合は、SET ステートメントの規則を使用して、ID-4 に戻り値が割り当てられます。それ以外の場合は、MOVE ステートメントの規則が使用されます。

INVOKE... RETURNING データ項目は、出力専用パラメーターです。呼び出されるメソッドに入った時点で、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。呼び出されるメソッドの 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。呼び出したメソッドが戻されるとき、呼び出し側に渡される値は、PROCEDURE DIVISION RETURNING データ項目の最終的な値です。

Java で定義されたローカル・オブジェクト・リファレンスおよびグローバル・オブジェクト・リファレンスについては、*Enterprise COBOL プログラミング・ガイド* を参照してください。これらの属性がオブジェクト・リファレンスの存続時間に与える影響、および COBOL でこれらの属性を管理する方法を理解することは重要です。

注: RETURN-CODE 特殊レジスターは、INVOKE ステートメントの実行では設定されません。

RETURNING 句の適合性要件: INVOKE ステートメントに*クラス名-1* NEW を指定するには、RETURNING 句が必要です。RETURNING 項目は、以下のいずれかでなければなりません。

- ユニバーサル・オブジェクト・リファレンス
- クラス名-1 を指定したオブジェクト・リファレンス
- クラス名-1 のスーパークラスを指定したオブジェクト・リファレンス

INVOKE ステートメントに NEW 句を指定しない場合、メソッド呼び出し、および対応するターゲット・メソッドで指定される RETURNING 項目は、以下の要件を満たす必要があります。

- 戻り値の有無は、INVOKE ステートメントおよびターゲット・メソッドと一致していなければなりません。
- RETURNING 項目がオブジェクト・リファレンスではなく、ターゲット・メソッドが COBOL でインプリメントされている場合、ターゲット・メソッド内の対応する RETURNING 項目は、同一のデータ記述記入項目を持っていなければなりません。ターゲット・メソッドが Java でインプリメントされている場合、メソッドの結果のタイプは、RETURNING 句で指定された項目のデータ型に対応する Java 型でなければなりません。詳しくは、336ページの『COBOL と Java の相互運用可能なデータ型』を参照してください。
- RETURNING 項目がオブジェクト・リファレンスの場合、ターゲット・メソッドで指定された RETURNING 項目は、INVOKE ステートメントで指定された RETURNING 項目と同じクラスに合わせて入力されたオブジェクト・リファレンスでなければなりません。

注: これらの要件を満たすのは、プログラマーの責任です。コンパイラーでは、これらの要件は満足されません。

ON EXCEPTION 句

INVOKE ステートメントで指定されたメソッドのシグニチャーと一致するシグニチャーを持つメソッドが、識別されたオブジェクトまたはクラスでサポートされない場合は、例外条件が発生します。例外条件が発生すると、以下のアクションのいずれかが実行されます。

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。
2. ON EXCEPTION 句が指定**されていない**場合は、実行時に重要度-3 の LE 条件が発生します。

NOT ON EXCEPTION 句

例外条件が発生しない (つまり、識別されたメソッドが、指定されたオブジェクトでサポートされている) 場合、制御は呼び出されるメソッドに移ります。呼び出されるメソッドから制御が戻ると、制御は次のものに移されます。

1. 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。
2. INVOKE ステートメントの終わり。ただし、NOT ON EXCEPTION 句が指定されて**いない**場合。

END-INVOKE 句

この明示的範囲終了符号は、INVOKE ステートメントの範囲を区切るために使用されます。END-INVOKE で終了する INVOKE ステートメントとそれに含まれるステートメントは、命令ステートメントのように処理される単位になります。INVOKE ステートメントは、条件ステートメント内の命令ステートメントとして指定できます。たとえば、別のステートメントの例外句の中に指定できます。

COBOL と Java の相互運用可能なデータ型

COBOL データ型のサブセットは、COBOL と Java の相互協調処理のために使用できます。

COBOL INVOKE ステートメントの引き数や RETURNING 項目として、相互運用可能なデータ型を指定できます。同様に、これらのデータ型を Java メソッド呼び出し式から引き数として渡し、USING 句のパラメーターとして、または COBOL メソッドの手続き部ヘッダー内の RETURNING 項目として、それらを受け取ることができます。

337ページの表40 には、相互協調処理でサポートされる基本的な Java 型と COBOL データ型、およびそれらの間の対応関係が示されています。

表 40. 相互運用可能な Java と COBOL のデータ型

Java データ型	COBOL データ型
boolean [1]	形式に対する条件変数と 2 つの条件名 *level-number* *data-name* PIC X. 88 *data-name*-false value X'00'. 88 *data-name*-true value X'01' through X'FF'.
byte	単一バイトの英数字、PIC X または PIC A
short	USAGE BINARY、COMP、COMP-4、または COMP-5。PICTURE 句は S9(n) 形式 (1 <= n <= 4)
int	USAGE BINARY、COMP、COMP-4、または COMP-5。PICTURE 句は S9(n) 形式 (5 <= n <= 9)
long	USAGE BINARY、COMP、COMP-4、または COMP-5。PICTURE 句は S9(n) 形式 (10 <= n <= 18)
float [2]	USAGE COMP-1
double [2]	USAGE COMP-2
char	単一文字 (国別) : PIC N USAGE NATIONAL
クラス型 (オブジェクト・リファレンス)	USAGE OBJECT REFERENCE *クラス名*

注:
[1] Enterprise COBOL では、示された形式の 2 つの正確な条件名が PIC X データ項目の後に続く場合に限って、PIC X 引き数またはパラメーターが Java ブール型として解釈されます。それ以外の場合、PIC X 引き数またはパラメーターは、Java バイト型として解釈されます。
[2] Java 浮動小数点データでは、IEEE 浮動小数点表記が使用されます。Enterprise COBOL では、IBM 16 進数浮動小数点表記が使用されます。COBOL から Java メソッドを呼び出す場合、および Java から COBOL メソッドを呼び出す場合、これらの表記は、必要に応じて自動的に変換されます。

基本タイプに加えて、Java String と Java 基本型の配列も、COBOL と相互運用が可能です。ただし、これを行うには、COBOL ランタイム・システムと Java Native Interface (JNI) で提供される特別なメカニズムが必要となります。

配列データを COBOL に渡したり、COBOL から受け取ったりする場合、Java プログラムでは、通常の Java 構文を使用して、配列の型を宣言します。COBOL プログラムでは、配列をサポートするために提供されている特別なクラスのインスタンスを含むオブジェクト・リファレンスとして、配列を宣言します。メソッドを呼び出す場合、Java 型および COBOL 型間の変換は、自動的に行われます。

String データを COBOL に渡したり、COBOL から受け取ったりする場合、Java プログラムでは、通常の Java 構文を使用して、配列の型を宣言します。COBOL プログラムでは、特別な jstring クラスのインスタンスを含むオブジェクト・リファレンスとして、String を宣言します。メソッドを呼び出す場合、Java 型と COBOL 型の間の変換は、自動的に行われます。338ページの表41 には、Java 配列、String データ型、および対応する特別な COBOL データ型が示されています。

表 41. COBOL と Java の相互運用可能な配列および String データ型

Java データ型	COBOL データ型
boolean[ ]	オブジェクト・リファレンス jboooleanArray
byte[ ]	オブジェクト・リファレンス jbyteArray
short[ ]	オブジェクト・リファレンス jshortArray
int[ ]	オブジェクト・リファレンス jintArray
long[ ]	オブジェクト・リファレンス jlongArray
char[ ]	オブジェクト・リファレンス jcharArray
String	オブジェクト・リファレンス jstring

以下の Java 配列型は、現在サポートされていません。

Java データ型	COBOL データ型
float[ ]	オブジェクト・リファレンス jfloatArray
double[ ]	オブジェクト・リファレンス jdoubleArray
Object[ ]	オブジェクト・リファレンス jobjectArray

Repository 段落では、他のクラスで記入項目が必要なのと同様に、使用する特別なクラスごとに記入項目が必要です。たとえば、jstring を使用する場合は、以下の記入項目が必要です。

```
Configuration Section.
Repository.
    Class jstring is "jstring".
```

また、String 型の場合は、COBOL リポジトリー記入項目で java.lang.String の外部クラス名を指定します。

```
Repository.
    Class jstring is "java.lang.String".
```

Java Native Interface (JNI) では、これらの型の COBOL オブジェクトを COBOL で操作するための呼び出し可能なサービスが提供されています。たとえば、呼び出しサービスを使用すると、jstring オブジェクト内に COBOL の英数字データまたは国別データを設定したり、jstring オブジェクトからデータを抽出したりすることができます。JNI 呼び出しサービスの使用方法については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

また、339ページの『COBOL と Java における各種の引き数の型』では、COBOL INVOKE ステートメントの引き数として使用可能な各種の COBOL データ項目とリテラルの例、およびそれに対応する Java 型を説明しています。

COBOL と Java における各種の引き数の型

表42 では、INVOKE ステートメントの引き数として使用可能な各種の COBOL 項目の例と、それに対応する Java 型を示しています。

表 42. 各種の COBOL 引き数の型とそれに対応する Java 型

COBOL 引き数 (argument)	**対応する Java データ型**
長さが 1 の usage display の参照変更項目	byte
長さが 1 の usage national の参照変更項目	char
SHIFT-IN および SHIFT-OUT 特殊レジスター	byte
usage が binary のときの LINAGE-COUNTER 特殊レジスター	int
LENGTH OF 特殊レジスター	int

表43 には、INVOKE ステートメントの引き数として使用可能な COBOL リテラル型と、それに対応する Java 型を示しています。

表 43. COBOL リテラル引き数の型とそれに対応する Java 型

COBOL リテラル引き数	**対応する Java データ型**
小数部の桁はなく、9 桁以下の固定小数点数字リテラル	int
浮動小数点数字リテラル	double
形象定数 ZERO	int
1 文字の英数字リテラル	byte
1 文字の国別リテラル	char
シンボリック文字	byte
形象定数 SPACE、QUOTE、HIGH-VALUE、または LOW-VALUE	byte

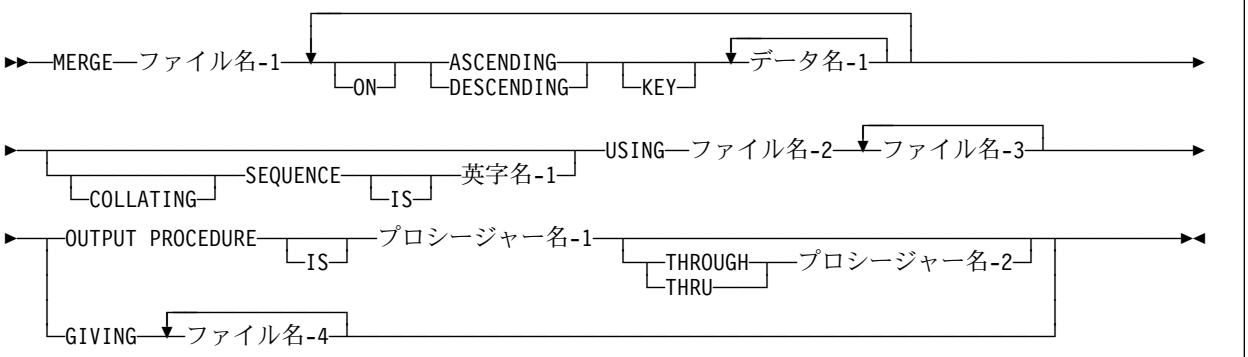
MERGE ステートメント

MERGE ステートメントは、1 つまたは複数のキーに基づいて、2 つ以上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 / 降順キーに従ってすで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、マージしたシーケンスでレコードを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、手続き部のどこにでも置くことができます。

MERGE ステートメントは、THREAD コンパイラー・オプションを指定してコンパイルされたプログラムではサポートされません。

フォーマット



ファイル名-1

マージするレコードを記述する SD 記入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 文節、SAME SORT AREA 文節、または SAME SORT-MERGE AREA 文節の中で指定することはできません。

注: 同じ SAME RECORD AREA 文節内では、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しなければなりません。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 文節を含んでいたり、OCCURS 文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 データ項目は、点在させることはできません。
- KEY データ項目は、可変オカレンス・データ項目を含むグループ項目にすることはできません。
- KEY データ項目は、修飾することができます。
- KEY データ項目には、以下を指定できます。
  - 英字、英数字、英数字編集、数字編集、または数字データ項目
  - 内部浮動小数点データ項目または外部浮動小数点データ項目
  - DBCS または国別データ項目
  - 以下の条件のもとでは、ウィンドウ化日付フィールドを指定できます。
    - USING 句に指定する入力ファイルは、順次、相対、索引付きのどれでも可能ですが、ウィンドウ化日付マージ・キーと同じ位置に RECORD KEY、ALTERNATE RECORD KEY、または RELATIVE KEY を持っていてはなりません。ファイル・システムはウィンドウ化日付フィールドをキーとしてサポートしません。このため、ファイル・システムによって行われる順序付けが、マージ操作のためのウィンドウ化日付フィールド・サポートと矛盾する可能性があります。実際、マージを正常に実行させるためには、ウィンドウ化日付の順序付けも含め、MERGE ステートメントによって指定された同じ順序で入力ファイルを事前にソートしておかなければなりません。
    - GIVING 句に索引付きファイルを指定してはなりません。ファイル・システムが想定または行う (2 進) 順序付けは、マージの出力で提供されるウィンドウ化日付順序付けと矛盾するからです。このような索引付きファイルにウィンドウ化日付マージ出力を書き込もうとしても、ファイルのアクセス方法 (ファイル制御項目の ACCESS MODE) に応じて、失敗するか、または 2 進順序付けが再び行われます。
    - 英数字のウィンドウ化日付フィールドを MERGE ステートメントの KEY として指定する場合は、マージ操作の有効な照合シーケンスが EBCDIC でなければなりません。したがって、MERGE ステートメントの COLLATING SEQUENCE 句、またはこの句を指定しない場合は、OBJECT-COMPUTER 段落の PROGRAM COLLATING SEQUENCE 句に、EBCDIC または NATIVE 以外の照合シーケンスを指定してはなりません。

MERGE ステートメントがこれらの条件を満たしていると、実行環境に世紀ウィンドウ化をサポートするソート製品が組み込まれているものと想定して、SORT 2000 年機能を利用してマージ操作が行われます。

年末尾型ウィンドウ化日付フィールドは MERGE ステートメントの KEY として指定することが可能で、それに対応する機能として、ソート製品の世紀ウィンドウ化機能を利用できます。

ウィンドウ化日付フィールドを KEY データ項目として使用する方法については、*Enterprise COBOL プログラミング・ガイド* 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。

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 (239ページの『比較条件』を参照)。

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。それ以外のキー・データ項目については、比較条件のオペランド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、英数字以外のキーには影響を与えません。

英字名-1

これは SPECIAL-NAMES 段落の ALPHABET 文節で指定されていなければなりません。英字名文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

ASCII 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 (ASCII 照合シーケンスについては、556ページの付録C、『EBCDIC および ASCII の照合シーケンス』を参照)。

STANDARD-2

国際標準 646 で定義された ISO 7 ビット・コードの国際参照バージョンである、情報処理交換用の 7 ビット・コード化文字セットが、すべての英数字比較のために使用されます。

NATIVE

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 照合シーケンスについては、556ページの付録C、『EBCDIC および ASCII の照合シーケンス』を参照)。

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 照合シーケンスについては、556ページの付録C、『EBCDIC および ASCII の照合シーケンス』を参照)。

リテラル
英字名文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 文節 (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 句も PROGRAM COLLATING SEQUENCE 文節を両方とも省略した場合には、EBCDIC 照合シーケンスが使用されます (556ページの付録C、『EBCDIC および ASCII の照合シーケンス』を参照)。

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 操作では、ファイル名-2、ファイル名-3、... (入力ファイル) にあるすべてのレコードはファイル名-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、クローズされますが、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ファイル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、データ部の中の FD 記入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入っている場合、その入力ファイル (ファイル名-2、ファイル名-3、...) に入っているレコードのサイズは、ファイル名-1 で記述されている最小レコードより小さいか、または最大レコードより大きくなければなりません。ファイル名-1 が固定長レコードを含んでいる場合は、入力ファイルに入っているレコードのサイズは、ファイル名-1 で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、 *Enterprise COBOL プログラミング・ガイド* 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。

すべての出力ファイルには、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、データ部の FD 記入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入っている場合、ファイル名-1 に入っているレコードのサイズは、その出力ファイルに記述されている最小レコードより小さいかまたは最大レコードより大きくなければなりません。出力ファイルが固定長レコードを含んでいる場合は、ファイル名-1 に入っているレコードのサイズは、出力ファイルで記述されて

いる最大レコードより大きくすることはできません。詳しくは、 *Enterprise COBOL プログラミング・ガイド* 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、...) をオープンしないでください。出力ファイルは自動的にオープンされ、読み取られ、クローズされますが、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、ファイルはエラーになります。

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 個ずつ使用可能にされるレコードをマージ順に選択、修正、またはコピーするときに必要になる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、出力をプロシージャーの範囲内で CALL、EXIT、GO TO、および PERFORM 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は、MERGE、RELEASE、または SORT のステートメントを実行させ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。

注: OUTPUT PROCEDURE 句は、基本的な PERFORM ステートメントと似ています。たとえば、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 (312ページの『EXIT ステートメント』を参照)。

MERGE 特殊レジスター

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 特殊レジスターで定義します。

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特殊レジスターにある値に優先します。

詳細については、18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、19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、20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

MERGE ステートメントが独立セグメント内にないセクションに置かれている場合は、その MERGE ステートメントで参照されているどの出力プロシージャーも、次のいずれかでなければなりません。

1. 非独立セグメント内に完全に収まっている。
2. 全体が 1 つの独立セグメントに収まっている。

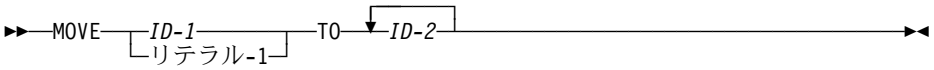
MERGE ステートメントが独立セグメント内に置かれている場合は、その MERGE ステートメントで参照されているどの出力プロシージャーも、次のいずれかでなければなりません。

1. 非独立セグメント内に完全に収まっている。
2. その MERGE ステートメントと同じ独立セグメントの中に全体が収まっている。

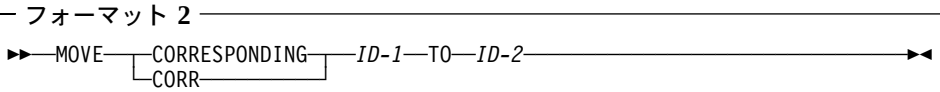
MOVE ステートメント

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



フォーマット 2



ID-1、リテラル-1
送り出し領域。

ID-2
受け取り領域 (複数の場合もある)。

フォーマット 1 を指定するときは、すべての ID はグループ項目か、または基本項目にすることができます。送り出し領域のデータは、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。347ページの『基本移動』および 351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 を指定するときは、ID は両方ともグループ項目である必要があります。CORR は、CORRESPONDING の省略形で、意味は同じです。

CORRESPONDING を指定すると、261 ページの CORRESPONDING 句の規則に従って、ID-1 内の選択された項目が ID-2 に移動します。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。

以下の種類の 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
- OBJECT REFERENCE

INDEX、POINTER、 FUNCTION-POINTER、 PROCEDURE-POINTER、または OBJECT REFERENCEの usage 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 文節の DEPENDING ON 句の影響を受けます (171ページの『OCCURS 文節』を参照)。

送り出しフィールド (ID-1) が参照による修正、添え字付き、もしくは英数字、英字、または国別関数 ID の場合、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。

たとえば、

```
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) へ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。

中間結果の詳細については、 *Enterprise COBOL プログラミング・ガイド* 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。

注: MOVE ステートメント内にオーバーラップする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項目であり、送り出し項目が基本項目またはリテラルである移動のことです。

各基本項目は、次のいずれかのカテゴリーに属します。

英字 - 英字データ項目と形象定数 SPACE が含まれます。

英数字 - 英数字データ項目、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が含まれます (英数字データ項目を必要とするコンテキストで使用される場合)。

英数字編集 - 英数字編集データ項目が含まれます。

数字 - 数字データ項目、数字リテラル、形象定数の ZERO (ZERO が数字項目または数字編集項目に移動されるとき) が含まれます。

数字編集 - 数字編集データ項目が含まれます。

浮動小数点 - 内部浮動小数点項目 (USAGE COMP-1 または USAGE COMP-2 として定義されたもの)、外部浮動小数点項目 (USAGE DISPLAY として定義されたもの)、および浮動小数点リテラルが含まれます。

DBCS - DBCS 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。

国別 - 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 リテラル (リテラルが、国別リテラルの場合)。

データをある 1 つの形式の内部表現のものから別の形式のものに変換する必要がある場合、その変換は、移動中に受け取り項目によって指定された編集またはそれに合わせて暗黙の編集解除を行います。

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

- 境界合わせと必要なスペースの埋め込みは、144ページの『位置合わせの規則』に説明した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**

- 境界合わせと必要なスペースの埋め込みは、144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。
- 送り出し項目が算術演算記号を持つ場合には、記号なしの値が使われます。算術演算記号が独立した 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。

国別:

- 送り出し項目が英字、英数字、英数字編集、DBCS、数字整数、数字編集データ項目、英数字リテラル、または英数字関数の場合、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ない長さの一時国別データ項目に移動されるように扱われます。変換で使用されるソース・コード・ページは、ソース・コードのコンパイル時に CODEPAGE コンパイラー・オプションに対して有効だったコード・ページです。

  結果の国別データ項目は、送り出しデータ項目として扱われます。
- 送り出しデータ項目が国別の場合は、変換されない送り出しデータ項目として使用されます。
- 境界合わせと必要なスペースの埋め込みが、144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。
- 送り出し項目が算術演算記号を持つ場合には、記号なしの値が使われます。算術演算記号が独立した 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 が置き換えられるところを除き、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、144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。送り出し項目が無記号であるときは、受け取り項目に対して正の算術演算記号が生成されます。

- 受け取り項目が無記号であるときは、送り出し項目の絶対値が移動され、受け取り項目に対して何も算術演算記号は生成されません。
- 送り出し項目が英数字であるときは、データは、送り出し項目が記号なしの整数として記述されているかのように移動されます。
- 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。
- 編集解除を行うことによって、数字編集データ項目を数字受け取り項目または数字編集受け取り項目に移動することができます。コンパイラーはこれを次のようにして行います。まず、数字編集項目の未編集の値を設定し、次にその未編集の数字を受け取り側の数字データ項目または数字編集データ項目に移動します。

浮動小数点

- 送り出し項目は、まず内部浮動小数点に変換され、それから移動されます。
- データが外部浮動小数点項目へ移動されるか、または外部浮動小数点項目から移動される場合、そのデータをまず等価の内部浮動小数点値に変換してから移動処理を行います。

DBCS:

- 変換は何も行われません。
- 送り出し項目と受け取り項目が同じサイズではない場合、データの右端が切り捨てられるか、または DBCS スペースで埋め込まれます。

注:

1. 受け取りフィールドが英数字、数字編集、または国別であり、送り出しフィールドが数字の場合、ピクチャー記号 P で記述される桁位置は、値ゼロを持つものとみなされます。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. 受け取りフィールドが数字であり、送り出しフィールドが英数字リテラルまたは ALL リテラルである場合、そのリテラルのすべての文字は数字である必要があります。

350ページの表44 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。図において、「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。

表 44. 有効な基本移動と無効な基本移動

送り出し項目のカテゴリー	受け取り項目のカテゴリー								
	英字	英数字	英数字編集	数字	数字編集	外部浮動小数点	内部浮動小数点	DBCS[1]	国別
英字と SPACE	はい	はい	はい	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 [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	いいえ	いいえ	いいえ	いいえ	はい
数字整数と ZERO [4]	いいえ	はい	はい	はい	はい	はい	はい	いいえ	はい
数字非整数 [5]	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	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	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	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	いいえ	はい

注:

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)、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 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。

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。

表45 は、年末尾型日付フィールド以外のフィールドが関係する移動の動作をまとめたものです。送り出し項目が日付フィールドである場合、受け取り項目は互換日付フィールドにする必要があります。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、ウィンドウ化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

ウィンドウ化日付フィールドの送り出し項目は、165ページの『ウィンドウ化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 45. 日付フィールドが関係する移動

	受け取り項目		
送り出し項目	**非日付データ**	**ウィンドウ化 日付フィールド**	**拡張日付フィールド**
非日付データ	通常	通常	通常
ウィンドウ化日付フィールド	無効	通常	拡張
拡張日付フィールド	無効	通常 [1]	通常

1 拡張日付フィールドからウィンドウ化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「 ウィンドウ化」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化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化日付フィールドがグループ項目 (JUSTIFIED 文節を指定できない) である場合も同様です。

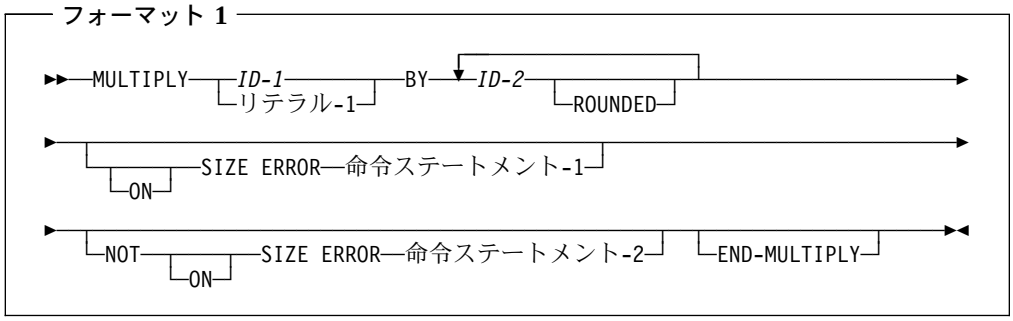
グループ移動

グループ移動とは、送り出しフィールドと受け取りフィールドの両方が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満たされます。ただし、OCCURS 文節で注記されている場合を除きます (171 ページの『OCCURS 文節』を参照)。**すべての**グループ移動が有効です。

MULTIPLY ステートメント

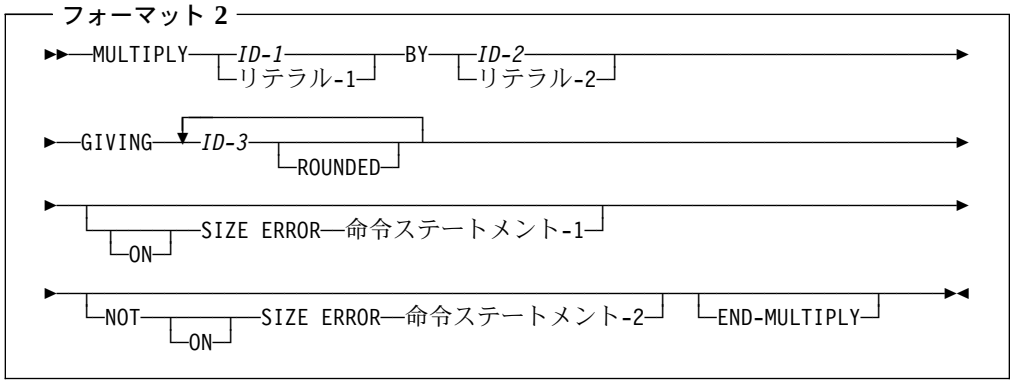
MULTIPLY ステートメントは、数字項目を乗算し、その結果をデータ項目の値として設定します。

フォーマット 1



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



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 の値によって乗算されます。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

ID-1、ID-2
基本数字項目である必要があります。ID-1 とID-2 は、日付フィールドにはできません。

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 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 が日付フィールドを指定する場合、積が ID-3 にどのように保管されるかについては、234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。

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、265ページの『算術ステートメント・オペランド』を参照してください。また、算術の中間結果については、 *Enterprise COBOL プログラミング・ガイド* 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、262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、263ページの『SIZE ERROR 句』を参照してください。

END-MULTIPLY 句

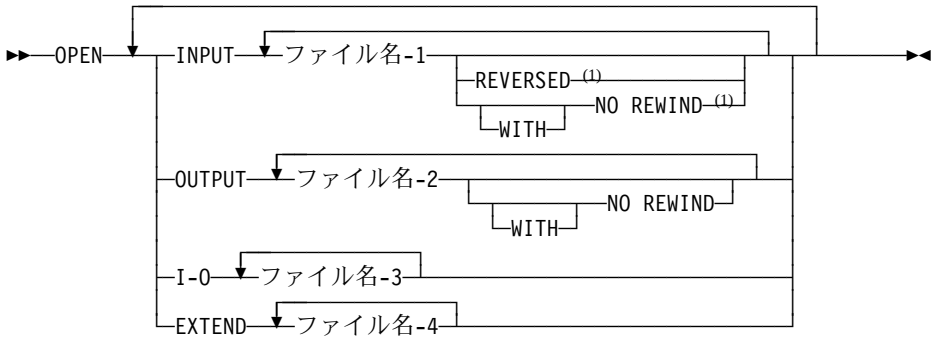
この明示的範囲終了符号は、MULTIPLY ステートメントの範囲を区切るために使用されます。END-MULTIPLY 句を使用することによって、条件的な MULTIPLY ステートメントを他の条件ステートメント内にネストすることができます。END-MULTIPLY は、命令 MULTIPLY ステートメントと共に使用することもできます。

詳細については、260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

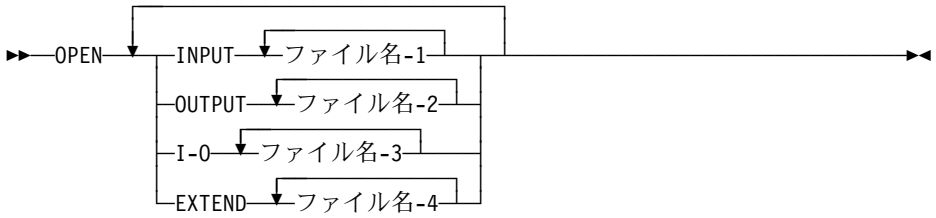
OPEN ステートメントは、ファイルの処理を開始します。ラベルのチェックまたは書き込み (あるいはその両方) も行います。

フォーマット 1 - 順次ファイル

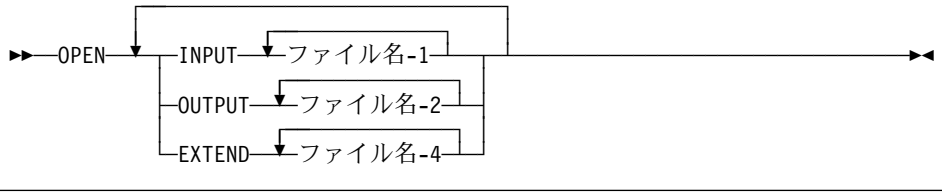


注:
[1] VSAM ファイルの場合、REVERSED 句および WITH NO REWIND 句は無効です。

フォーマット 2 - 索引付きファイルおよび相対ファイル



フォーマット 3 - 行順次ファイル



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
入力操作を行うためにファイルをオープンします。

OUTPUT
出力操作を行うためにファイルをオープンします。この句は、ファイルの作成時に指定することができます。

注: OUTPUT は下記のファイルには指定しないでください。

- レコードが入っているファイル。このファイルは新規データに置き換えられます。すでにレコードが入っているファイルに OUTPUT 句が指定されると、データ・セットは再使用可能として定義しなければならず、代替索引を持つことができなくなります。ファイル内のレコードは新規データに置き換えられ、SELECT ステートメント内の ALTERNATE RECORD KEY 文節はすべて無視されます。
- DD ダミー・カードで定義されたファイル。予測できない結果が生じます。

I-O

入力操作と出力操作の両方を行うためにファイルをオープンできます。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

出力操作を行うためにファイルをオープンします。

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 文節が指定されたファイルで使用可能です。

QSAM ファイルの場合は、複数のファイル・リールに EXTEND 句を指定しないでください。

存在するかどうかが不確実なファイルを OPEN ステートメントに追加したい場合は、EXTEND モードでそのファイルをオープンする前に SELECT OPTIONAL 文節を使用してください。そのファイルが存在しない場合はファイルが作成され、既存の場合は OPEN ステートメントに追加されます。

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、データ部の FD 記入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 記入項目は、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

順次単一リール・ファイルに対してのみ有効です。これは VSAM ファイルには無効です。

たとえば、直接アクセス装置のようにそのストレージ・メディアにリールの概念が意味を成さない場合には、REVERSED 句や NO REWIND 句は適用されません。

NO REWIND

順次単一リール・ファイルに対してのみ有効です。これは VSAM ファイルには無効です。

一般規則

- INPUT 句を付けてオープンしたファイルがオプショナル・ファイルであり、それが存在しない場合、OPEN は、オプショナル入力ファイルが存在し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 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。
- VSAM ファイルの場合、ファイル内にレコードが 1 つも存在しないと、ファイル位置標識は、最初に実行されるフォーマット 1 の READ ステートメントの結果が AT END 条件になるよう設定されます。
- NO REWIND を指定すると、OPEN ステートメントを実行してもファイルは再度位置付けされません。OPEN の実行に先立ち、ファイルはその最初に位置付けされなければなりません。NO REWIND 句が指定されている場合 (または NO REWIND 句と REVERSE 句が両方とも省略されている場合)、ファイルの位置付けは、DD ステートメントの LABEL パラメーターで指定されます。
- REVERSED 句を指定して、OPEN ステートメントを実行すると、QSAM ファイルはその終わりに位置付けられます。それ以降の READ ステートメントは、最後のレコードから開始して逆の順序でデータ・レコードを使用できるようにします。

  OPEN REVERSED を指定する場合は、レコード・フォーマットは固定長でなければなりません。
- REVERSED 句、NO REWIND 句、または EXTEND 句が指定されていない場合、OPEN ステートメントの実行は、ファイルをその先頭に位置付けて行われます。

FILE-CONTROL 記入項目で PASSWORD 文節を指定している場合、OPEN ステートメントを実行する前に、パスワード・データ項目に、有効なパスワードを入れておく必要があります。有効なパスワードが存在しなければ、OPEN を正しく実行できません。

ラベル・レコード

OPEN ステートメントの実行時に、ファイルに対してラベル・レコードが指定されていれば、ラベルは標準ラベルの規則に従って次のように処理されます。

INPUT ファイル 開始ラベルがチェックされます。

OUTPUT ファイル 開始ラベルが書き込まれます。

I-O ファイル ラベルがチェックされ、これにより新規ラベルが作成されます。

EXTEND ファイル 以下のプロシージャーが実行されます。

- これが単一ボリューム・ファイルである場合に限り、開始ファイル・ラベルが処理されます。
- 最後にある既存ボリュームの開始ボリューム・ラベルは、そのファイルが INPUT 句を指定してオープンされたかのように処理されます。

- 既存の終了ファイル・ラベルは、そのファイルが INPUT 句を指定してオープンされたかのように処理されます。これらは、次いで削除されます。
- ファイルは OUTPUT ファイルとしてオープンされたかのように、処理が継続されます。

ラベル・レコードが指定されたにもかかわらずそれが存在しない場合、または存在するのに指定されていない場合、OPEN ステートメントを実行するとどうなるかは予測できません。

OPEN ステートメントに関する注意事項

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。表46 は、使用可能なファイルと使用可能でないファイルに対してオープンを行った結果を示したものです。ファイルの使用可能性については、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

表 46. 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
INPUT (オプショナル・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
I-O	通常のオープン	オープンに失敗する
I-O (オプショナル・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
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
EXTEND (オプショナル・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

2. OPEN ステートメントが正常に実行されると、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が、最初のデータ・レコードは取得または解除されません。
3. 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)。358ページの表47 は、順次ファイルで使用できる入出力ステートメントを示しています。'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 47. 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	オープン・モード			
	INPUT (入力)	**OUTPUT (出力)**	**I/O (入出力)**	**EXTEND (拡張)**
READ	X		X	
WRITE		X		X
REWRITE			X	

表48 は、索引付きファイルと相対ファイルで使用できるステートメントを示しています。'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 OPEN モードで使用できることを意味します。

表 48. 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	オープン・モード			
		INPUT (入力)	OUTPUT (出力)	I-O (入出力)	EXTEND (拡張)
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	
動的	READ	X		X	
	WRITE		X	X	
	REWRITE			X	
	START	X		X	
	DELETE			X	

359ページの表49 は、行順次ファイルに対して使用できる入出力ステートメントを示しています。'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 49. 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	オープン・モード			
	INPUT (入力)	**OUTPUT (出力)**	**I-O (入出力)**	**EXTEND (拡張)**
READ	X			
WRITE		X		X
REWRITE				

4. 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、REEL 句または UNIT 句 (QSAM ファイルのみ)、または 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。
5. FILE-CONTROL 記入項目の中に FILE STATUS 文節が指定されている場合、OPEN ステートメントが実行されると関連付けられた状況キーが更新されます。
6. すでにオープン状態になっているファイルに対して OPEN ステートメントを発行すると、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にある実行可能ステートメントに制御を戻します。

PERFORM ステートメントは、次のいずれかになります。

行外 PERFORM ステートメント
プロシージャー名-1 が指定されます。

行内 PERFORM ステートメント
プロシージャー名-1 は省きます。

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。たとえば、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。

PERFORM ステートメントのフォーマットには、次の 4 種類があります。

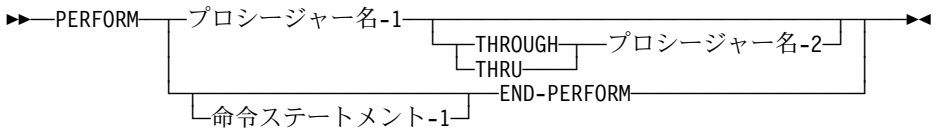
- 基本 PERFORM ステートメント
- TIMES 句付き PERFORM ステートメント
- UNTIL 句付き PERFORM ステートメント
- VARYING 句付き PERFORM

基本 PERFORM ステートメント

基本 PERFORM ステートメント内で参照されるプロシージャー (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 ステートメントなどでは、予測できない結果が生じる可能性があります。

フォーマット 1



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 とプロシージャー名-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 を省略する場合には、命令ステートメントと END-PERFORM 句を指定しなければなりません。

命令ステートメント
行内 PERFORM ステートメントで実行されるステートメント。

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 ～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。

図10 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。

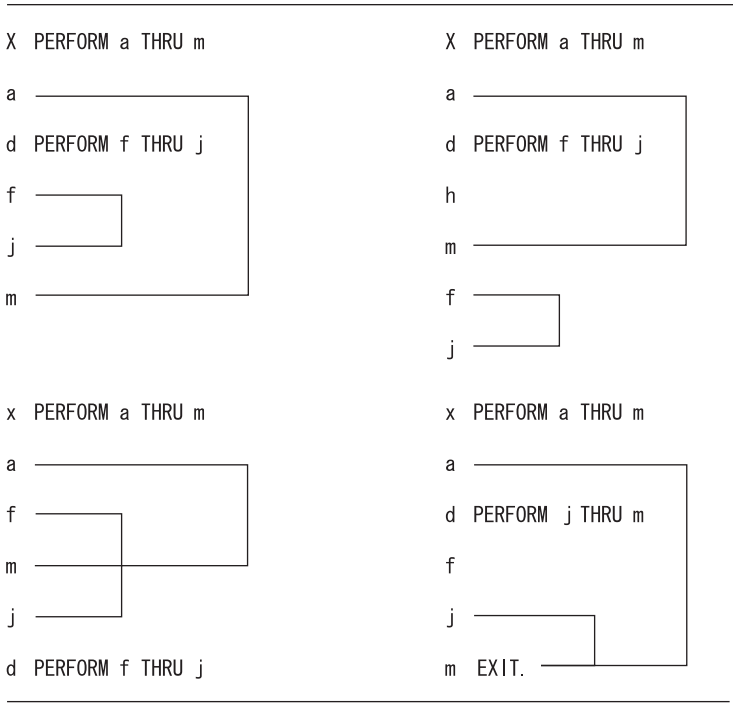


図 10. 有効な PERFORM ステートメントの実行シーケンス

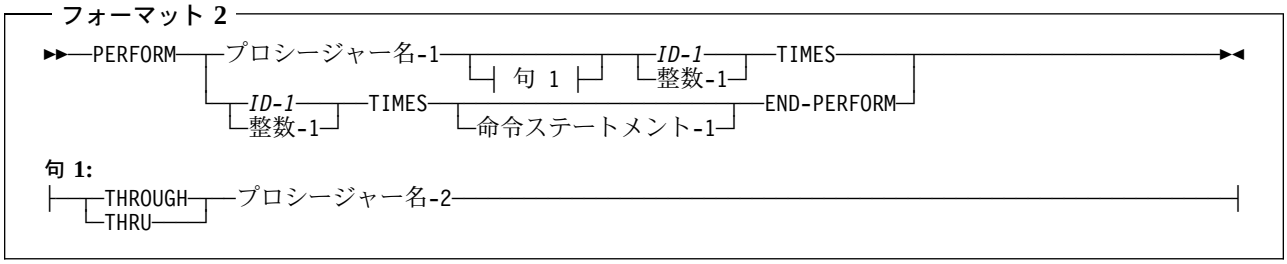
PERFORM ステートメント以外の手段によって一連のプロシージャーに制御が渡される場合、それらのプロシージャーを参照する PERFORM ステートメントが何もないかのように、制御は出口点を通りすぎて次の実行可能ステートメントに渡されます。

END-PERFORM

インライン 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。

TIMES 句を指定した PERFORM

TIMES 句付きの PERFORM ステートメントの中で参照されたプロシージャーは、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。



注: 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 句を指定できません。

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 は、ウィンドウ化日付フィールドにはできません。

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。

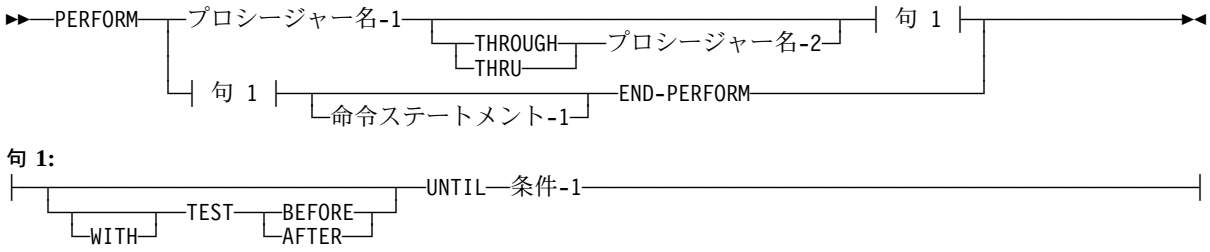
PERFORM ステートメントが開始された後で ID-1 が変更されても、そのプロシージャーを実行する回数は変更されません。

整数-1
正の記号付き整数にできます。

UNTIL 句を指定した PERFORM

UNTIL 句を伴うフォーマットでは、参照される手順は、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



注: 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
235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)。

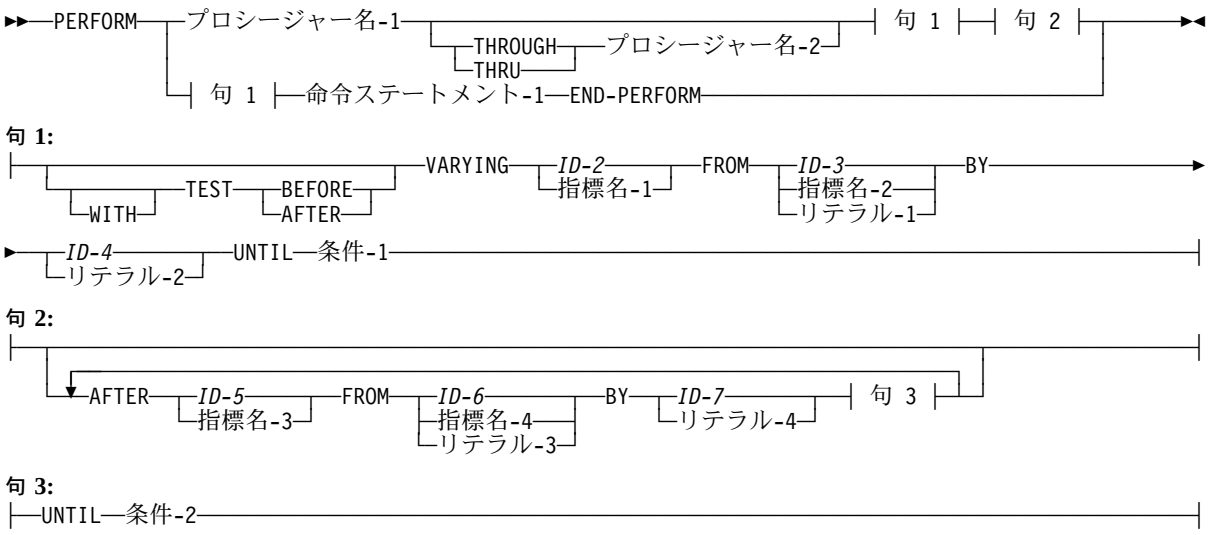
どちらの場合も、条件が真ならば、PERFORM ステートメントの後にある次の実行可能ステートメントに制御は移されます。TEST BEFORE 句も TEST AFTER 句も指定されていない場合、TEST BEFORE 句が想定されます。

VARYING 句を指定した PERFORM

VARYING 句は、ある一定の規則に従って、1 つまたは複数の ID や指標名の値を増やすか、または減らします (369ページの『Varying 句の規則』を参照)。

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



注: 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 句を指定できません。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 ～ 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、ウィンドウ化日付フィールドにはできません。

リテラル-1 ～ 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
235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 に指定される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。

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。

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、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。以下の説明では、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ときは、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、または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ときは、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や増加の操作で使用されるたびに評価されます。

図11 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。

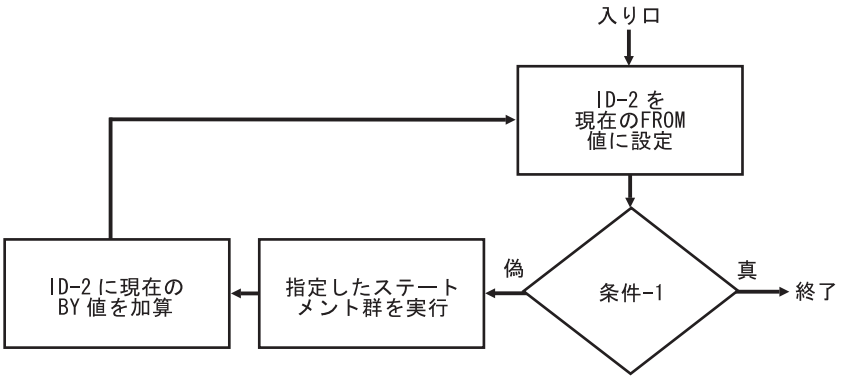


図 11. 1 つの ID の変更 (TEST BEFORE 句指定の場合)

図12 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。

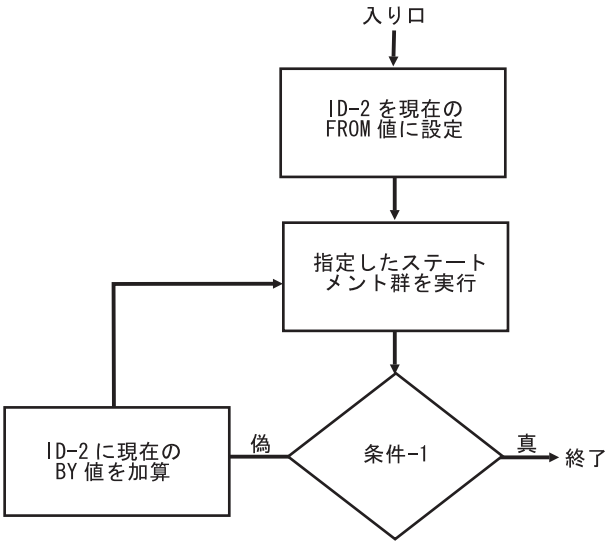


図 12. 1 つの ID の変更 (TEST AFTER 句指定の場合)

2 つの ID の変更

```
PERFORM PROCEDURE-NAME-1 THROUGH PROCEDURE-NAME-2
      VARYING IDENTIFIER-2 FROM IDENTIFIER-3
            BY IDENTIFIER-4 UNTIL CONDITION-1
          AFTER IDENTIFIER-5 FROM IDENTIFIER-6
            BY IDENTIFIER-7 UNTIL CONDITION-2
```

1. **ID-2** と **ID-5** が、その初期値である ID-3 とID-6 にそれぞれ設定されます。
2. **条件-1** が、次のようにして評価されます。
   a. 偽であれば、ステップ 3 ～ 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 ～ 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-1** と **プロシージャー-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 ～ 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 ～ 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**

  ID-6 の現行値が入っています。

- **ID-2**

  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は、このときは、ID-2 には ID-3 の現行値が入っています)。

367ページの図13 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。

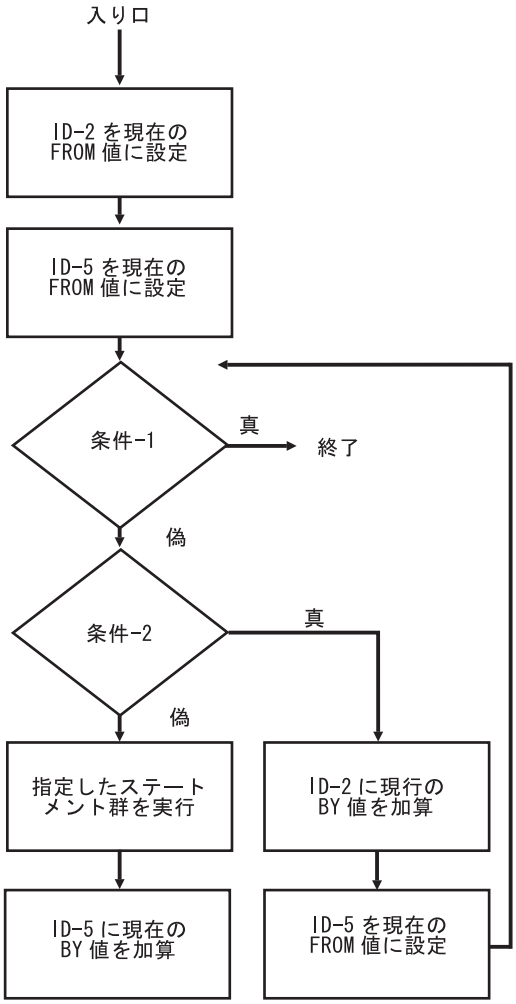


図 13. 2 つの ID の変更 (TEST BEFORE 句指定の場合)

368ページの図14 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。

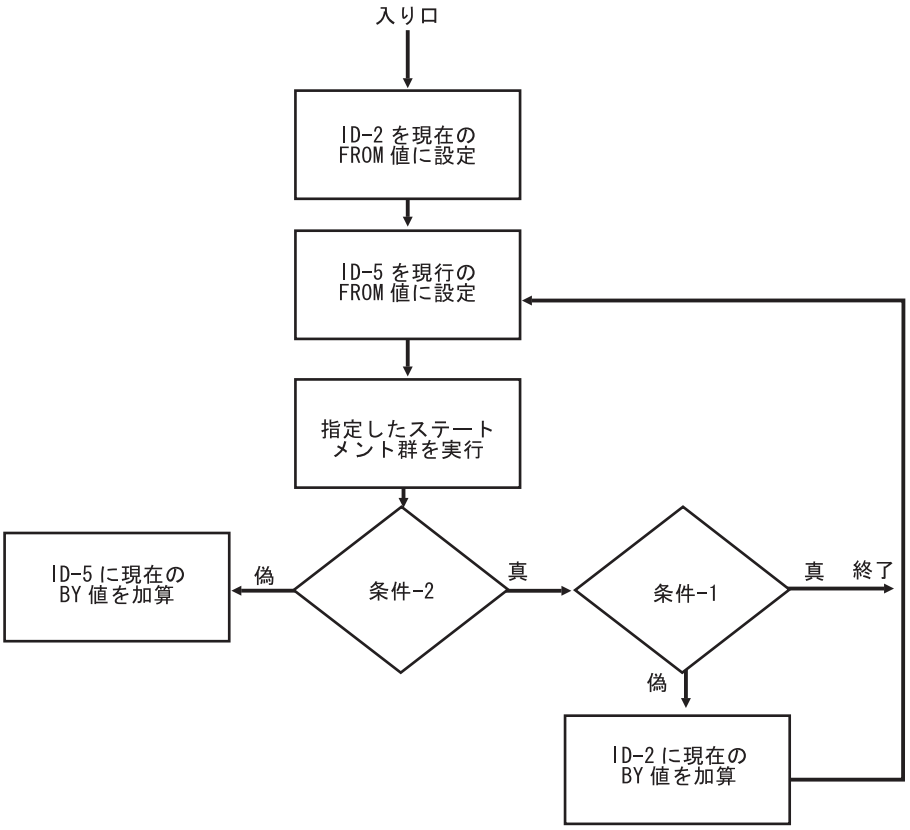


図 14. 2 つの ID の変更 (TEST AFTER 句指定の場合)

3 つの ID の変更

```
PERFORM PROCEDURE-NAME-1 THROUGH PROCEDURE-NAME-2
      VARYING IDENTIFIER-2 FROM IDENTIFIER-3
            BY IDENTIFIER-4 UNTIL CONDITION-1
         AFTER IDENTIFIER-5 FROM IDENTIFIER-6
            BY IDENTIFIER-7 UNTIL CONDITION-2
         AFTER IDENTIFIER-8 FROM IDENTIFIER-9
            BY IDENTIFIER-10 UNTIL CONDITION-3
```

この場合の動作も、2 つの ID の場合と次の点を除けば同じです。すなわち、ID-5 が ID-7 だけ増やされるたびに、ID-8 が完全なサイクルで処理され、一方、ID-5 は、ID-2 が変更される度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** と **ID-8**

  それぞれ ID-6 と ID-9 の現行値が入っています。

- **ID-2**

  最後に使用された設定値を、1 だけ超えた値または 1 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は、ID-2 には ID-3 の現行値が入っています)。

4 つ以上の ID の変更

AFTER 句を 4 つまで追加して、上記の例と同じように PERFORM ステートメントによる処理ができます。

Varying 句の規則

指定する変数の個数には関係なく、次の規則が適用されます。

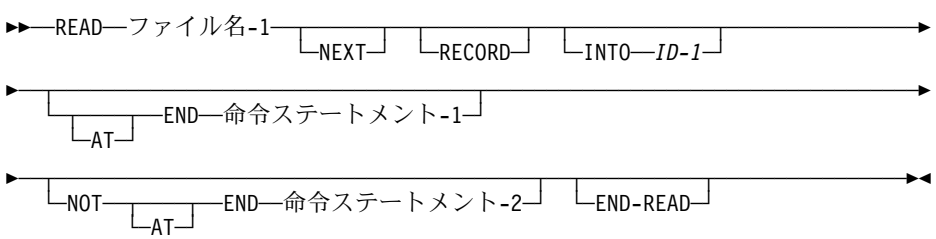
1. VARYING/AFTER 句の中に指標名が指定されている場合
    a. 210ページの『INDEX 句』に示されている規則に従って指標名が初期設定され、それが増分または減少されます (391ページの『SET ステートメント』も参照)。
    b. 関連する FROM 句では、ID を整数として記述し、正の値を持つ必要があります。リテラルは正の整数でなければなりません。
    c. 関連する BY 句内では、ID は整数として記述する必要があります。リテラルは 0 以外の整数でなければなりません。
2. FROM 句の中に指標名が指定されている場合
    a. 関連する VARYING/AFTER 句の中では、ID は整数として記述される必要があります。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。
    b. 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 0 以外の整数でなければなりません。
3. BY 句の中で、ID とリテラルは 0 以外の値を持たねばなりません。
4. VARYING、FROM、および BY 句の中で ID や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

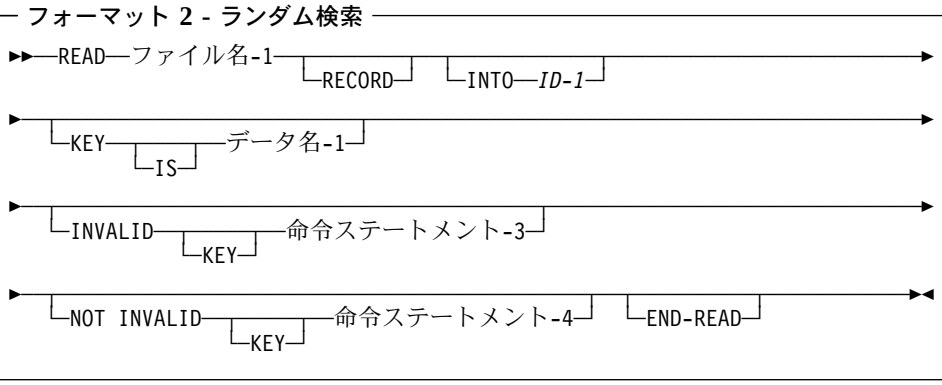
順次アクセスの場合、READ ステートメントはファイル内の次の論理レコードをオブジェクト・プログラムが使用できるようにします。ランダム・アクセスの場合には、READ ステートメントは、オブジェクト・プログラムが直接アクセス・ファイル内の指定したレコードを使用可能にします。

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



フォーマット 2 - ランダム検索



ファイル名-1
データ部の FD 記入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、ACCESS MODE IS SEQUENTIAL 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・モードのファイルの場合に、NEXT RECORD 句を指定する必要があります。その場合、レコードが順次検索されます。

INTO ID-1
ID-1 は受け取りフィールドです。

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。
- 現在のレコードを CORRESPONDING 句を指定しない 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 文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 文節を含む場合には、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、350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、あるいは RECORDING MODE 'S' または 'U' が指定された QSAM ファイルとして記述されている場合は、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。

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。

KEY IS 句

KEY IS 句は、索引付きファイルに対してのみ指定できます。データ名-1 は、ファイル名-1 に関連付けられたレコード・キーと一致しなければなりません。データ名-1 を修飾することができます。添え字付けはできません。

AT END 句

順次アクセスの場合は、AT END 句および利用可能な EXCEPTION/ERROR プロシージャーを両方とも省略できます。

AT END 条件の処理に関する詳細は、373ページの『AT END 条件』を参照してください。

INVALID KEY 句

INVALID KEY 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。

INVALID KEY 句の処理に関する詳細は、271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。詳細については、260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。

複数のレコードの処理

ファイル名-1 に関連付けられた複数のレコード記述項目がある場合、これらのレコードは自動的に同じストレージ域を共用します。つまり、それらは暗黙に再定義されます。READ ステートメントが実行された後、現行レコードの範囲内にあるこれらの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ていません。図15 では、この概念を説明しています。現在のレコードの範囲が、ファイル名-1 のレコード記述項目を超えている場合には、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しく実行され、I-O 状況として (04) が設定されてレコード長に不一致があったことを示します。

図 15. 複数のレコード記述を指定した READ ステートメント

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 の 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 FILE-CONTROL 記入項目の中に SELECT OPTIONAL が指定してあり、オブジェクト・プログラムの実行中にそのファイルが存在しない場合には、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、ファイルが存在しないので、システム定義のファイル終わり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、またはオプショナル入力ファイルが存在しないということを示している場合、次のことが、以下に述べられている順に起こります。

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 に関連する I-O 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。読み込みの失敗に続いて、データにアクセスしたり、データを読み込みレコード域へ移動しようとすると、記憶保護例外という結果になる可能性があります。

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 I-O 状況が更新されます。
2. AT END 条件ではない例外条件が存在する場合、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では、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。読み込みの失敗に続いて、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、記憶保護例外という結果になる可能性があります。

マルチボリューム QSAM ファイル: READ ステートメントの実行時、ボリュームの終わりが認識されても、論理的なファイルの終わりに到達していない場合は、以下の処置が実行されます。

- 標準の終了ボリューム・ラベル・プロシージャー
- ボリュームの切り替え
- 標準の開始ボリューム・ラベル・プロシージャー
- 次のボリュームの最初のデータ・レコードが使用可能にされる

索引付きファイルまたは相対ファイル

NEXT RECORD とは、キーのシーケンスで次に続く論理レコードです。

索引付きファイルでは、キーのシーケンスは、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、キーのシーケンスは、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、ファイル位置標識で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るならば、そのレコードが使用可能にされます。

レコードがすでにアクセス可能でなくなっている場合 (たと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は、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 句を指定しなければなりません。

AT END 条件: この条件は、ファイル位置標識が、次の論理レコードが存在しないことを示している場合、あるいは、オプションの入力ファイルが存在しないことを示している場合に発生します。順次ファイルの場合と同じプロシージャーが実行されます (373ページの『AT END 条件』を参照)。

READ ステートメントの実行の間に、AT END 条件も無効なキー条件も起こらなければ、AT END 句や 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プロシージャーが実行されます (373ページの『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 文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方法は、以下のセクションに説明するように、ファイル編成に応じて異なります。

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 (267ページの『共通の処理機能』にある「無効キー条件」を参照)。

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 ステートメントのために使用されます。

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が指し示される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 (267ページの『共通の処理機能』にある「無効キー条件」を参照)。

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

指標付き編成または相対編成のファイルの場合は、FILE-CONTROL 記入項目の中で動的アクセス・モードを指定することができます。動的アクセス・モードでは、使用する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することができます。

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

FILE-CONTROL 記入項目に FILE-STATUS 文節の指定がある場合は、関連する状況キーが READ ステートメントの実行で更新されます。

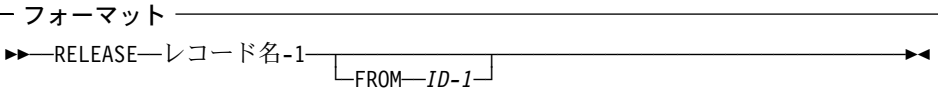
READ ステートメントの実行が失敗した後では、関連するレコード域の内容とファイル位置標識の値は未定義です。読み込みの失敗に続いて、データにアクセスしたり、データをレコード域へ移動しようとすると、記憶保護例外という結果になる可能性があります。

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



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。つまり、ソート操作の初期フェーズに対して使用可能になります。

レコード名-1

ソート / マージ・ファイル記述記入項目 (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 は修飾できます。

FROM 句

FROM ID-1 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 identifier-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

ID-1 は、以下のいずれかを参照する必要があります。

- ワーキング・ストレージ・セクション、ローカル・ストレージ・セクション、またはリンケージ・セクション内の記入項目
- 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フィールドでなければなりません。

ID-1 とレコード名-1 は、同じストレージ域を参照することはできません。

RELEASE ステートメントが実行された後でも、ID-1 の中の情報は使用可能です (267 ページの『共通の処理機能』の「INTO/FROM ID 句」を参照)。

ファイル名-1 に対する SD 記入項目を SAME RECORD AREA 文節で指定せずに 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 記入項目を SAME RECORD AREA 文節の中で**指定した**場合は、レコード名-1 は、その文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。

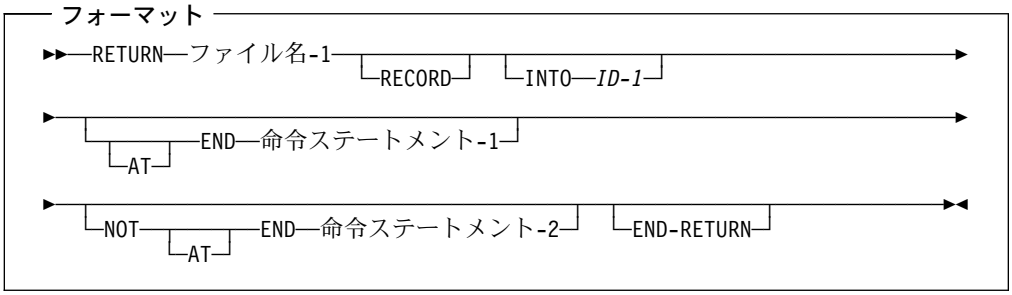
INPUT PROCEDURE から制御を渡されるとき、ソート・ファイルは、RELEASE ステートメントの実行によりその中に入れられたすべてのレコードから構成されています。

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



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

データ部の SD 記入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、こ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場合と同じになります。

- 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。
- 現在のレコードを CORRESPONDING 句を指定しない 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 文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 文節を含む場合には、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に固定長レコードが含まれる場合、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 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。それ以外の場合は、RETURN ステートメントの終わりに制御が渡されます。

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。

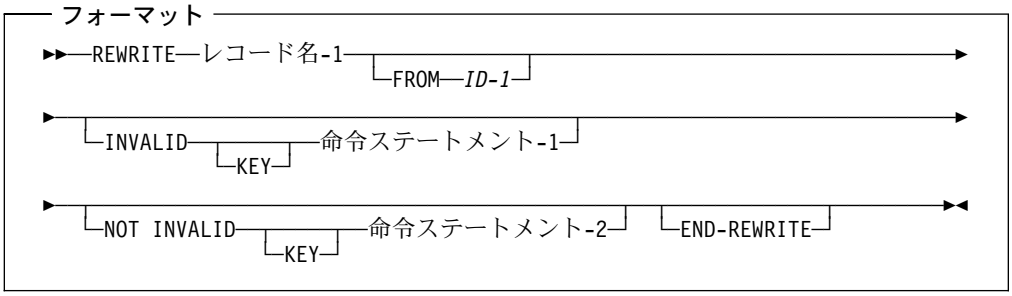
詳細については、260ページの『範囲区切りステートメント』を参照してください。

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



レコード名-1

データ部の FD 記入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 句を指定された 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する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。
- 英数字または国別関数。
- ワーキング・ストレージ・セクション、ローカル・ストレージ・セクション、またはリンケージ・セクション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フィールドでなければなりません。

ID-1 とレコード名-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 の中にある情報は使用可能です (267ページの『共通の処理機能』にある「INTO/FROM ID 句」を参照)。

INVALID KEY 句

(267ページの『共通の処理機能』にある「無効キー条件」を参照。)

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。

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。

詳細については、260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 文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 文節で指定されている場合、レコードは、その SAME RECORD AREA 文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。

FILE-CONTROL 記入項目の中に FILE STATUS 文節が指定されている場合には、REWRITE ステートメントを実行すると関連する状況キーが更新されます。

順次ファイル

ファイルが順次アクセス・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 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もの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。

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。

無効な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 (267ページの『共通の処理機能』にある「無効キー条件」を参照)。

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。

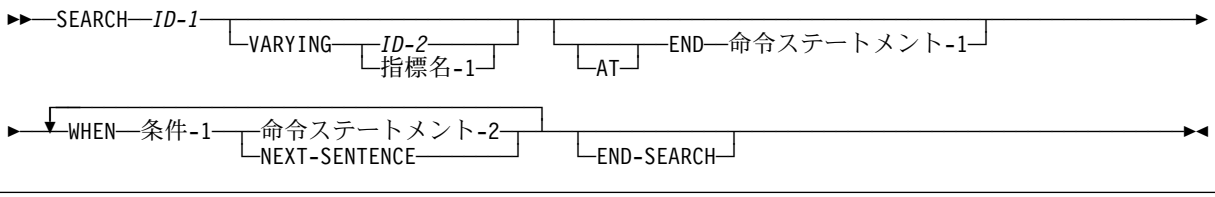
ランダム・アクセス・モードまたは動的アクセス・モードの相対ファイルの場合、INVALID KEY 句および利用可能な EXCEPTION/ERROR プロシージャー (またはそのいずれか) は指定できません。これらは、両方とも省略することができます。

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。ファイルに指定したレコードがない場合、無効なキー条件が生じ、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。(267ページの『共通の処理機能』にある「無効キー条件」を参照)。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。

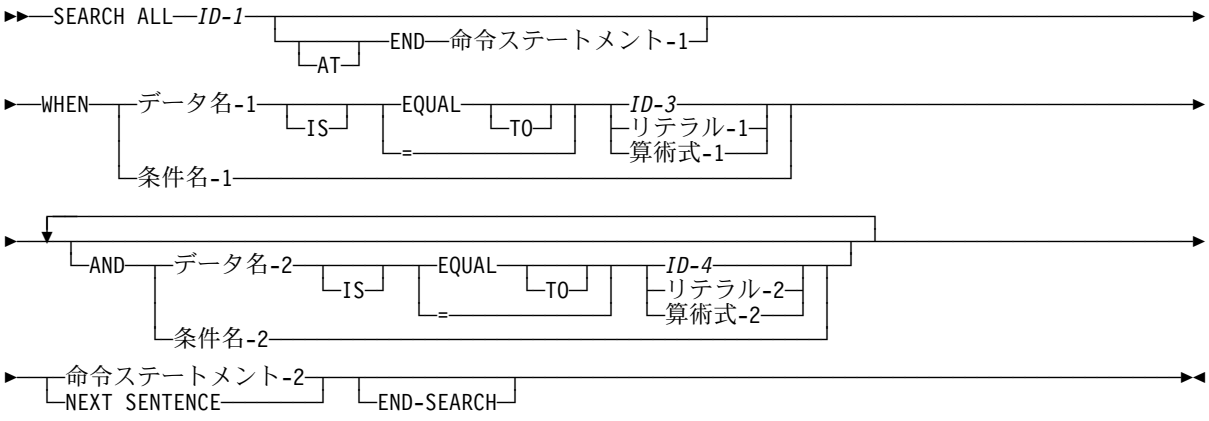
SEARCH ステートメント

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 - シリアル検索



フォーマット 2 - 二分探索



ID-1

OCCURS 文節を含むデータ項目に従属するデータ項目、つまり、多次元のテーブルの一部にできます。この場合、データ記述記入項目では、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、テーブル・エレメント内のすべてのオカレンスを参照する必要があります。したがって、添え字をされたり、参照変更されてはなりません。

データ部の ID-1 に関する記述は、INDEXED BY 句が指定された OCCURS 文節を含まなければなりませんが、テーブルは、適切に記述された別のテーブルに定義された指標を使用して検索できます。

フォーマット 2 では、データ部の記述は、OCCURS 文節に KEY IS 句を含んでいる必要があります。

SEARCH ステートメントが実行されると、ID-1 に関連付けられた指標名の値だけが変更されます (指標名-1 や ID-2 が指定されていれば、その値も変更されます)。したがって、2 ～ 7 次元のテーブルの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。各次元の検索を実行するに先立って、SET ステートメントを実行して関連する指標名を初期設定し直す必要があります。

AT END/WHEN 句

命令ステートメント-1 または命令ステートメント-2 が実行された場合、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。

NEXT SENTENCE

NEXT SENTENCE では、最も近い区切り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。

注: 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は、命令ステートメント-2 も NEXT SENTENCE も必要ありません。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。

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。

詳細については、260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。

シリアル検索

フォーマット 1 の SEARCH ステートメントは、現在の指標設定値からシリアル検索を実行します。検索が開始されると、ID-1 に関連付けられた指標名の値が、可能なオカレンス数の最高値よりも大きくない限り、以下のような処置が取られます。

- 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。
- 条件がいずれも満たされない場合、ID-1 に対する指標名が次のテーブル・エレメントに対応するように増やされ、ステップ 1 が繰り返されます。
- 評価を行って、WHEN 条件の 1 つが満たされた場合、検索は直ちに終了し、その条件に関連する命令ステートメントが実行されます。指標名は、条件を満たしたテーブル・エレメントを指しています。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、増加していった指標名の値が可能なオカレンス番号の最高値よりも大きくなった場合)、下記に記述されているとおり検索は終了します。

検索が開始され、ID-1 に関連付けられた指標名の値が可能なオカレンス数の最高値よりも大きくなると、検索は直ちに終了し、もしも AT END 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。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。

VARYING 句

指標名-1

以下の処置のうちのどちらかが適用されます。

- 指標名-1 が、ID-1 の指標である場合、この指標が検索で使用されます。それ以外の場合は、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には、ID-1 のための最初の (または唯一の) 指標名が検索で使用されます。指標名-1 が示すオカレンス数は、検索指標名の増加分と同じだけ、かつ同時に増加します。

VARYING 指標名-1 句が省略された場合には、ID-1 の最初の (または唯一の) 指標名が、検索のために使用されます。

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。

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 は、ウィンドウ化日付フィールドにはできません。ID-2 に添え字として ID-1 の最初の (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、次のどちらかの処置が取られます。

- 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 (シリアル検索)

条件-1

235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。

図16 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。

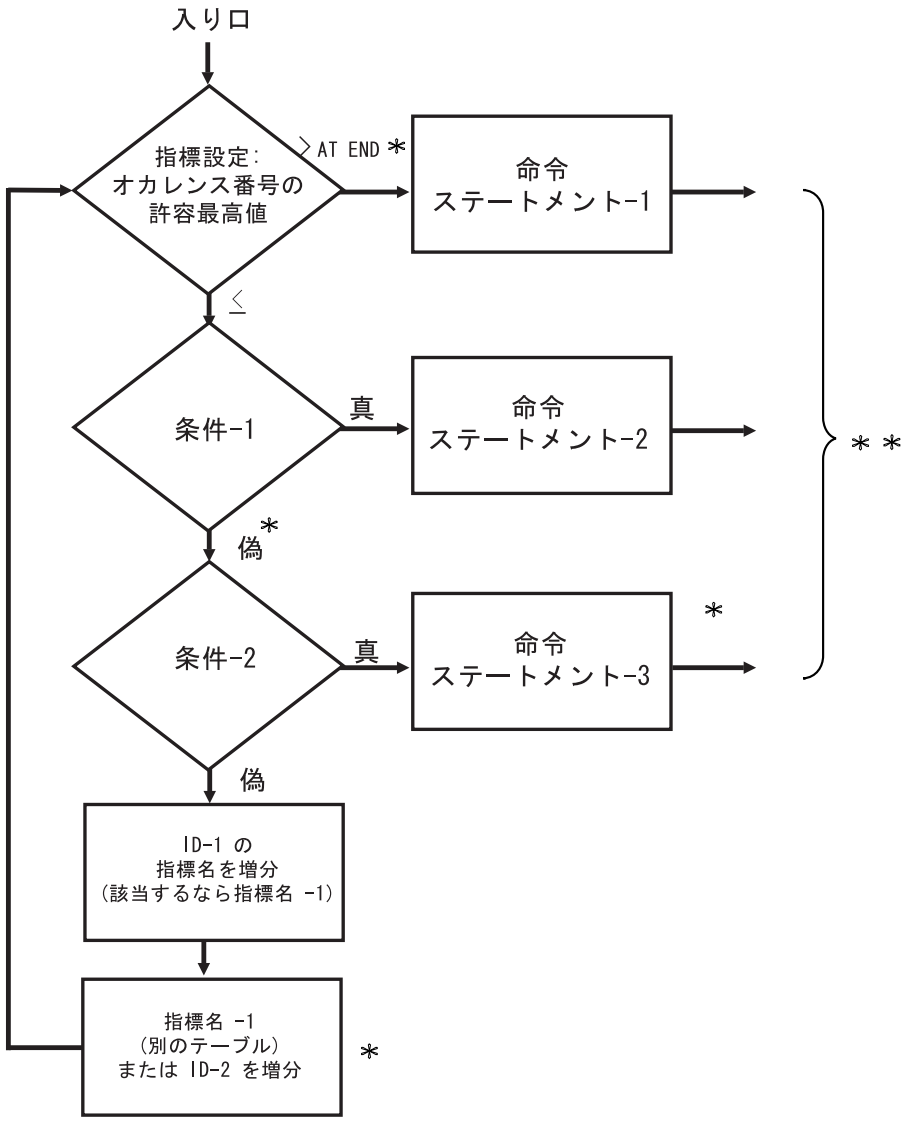


図 16. 2 つの WHEN 句を指定したフォーマット 1 の SEARCH ステートメント

二分探索

フォーマット 2 の 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。SET ステートメントを使用して、検索指標を初期設定する必要はありません。その理由は、検索指標の設定値が、いかなる場合も、最初のテーブル・エレメントの値よりも小さくなったり、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、検索処理中に変更されるからです。使用される指標は、必ず OCCURS 文節で指定された最初の指標名に関連した指標です。

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/DESCENDING KEY の順に並んでいる場合。

- WHEN 文節の中に指定された ASCENDING/DESCENDING KEY の内容が、固有のテーブル参照を提供する場合。

ID-1

ID-1 は、以下が**可能です**。

- OCCURS 文節を含むデータ項目に従属するデータ項目。つまり、2 ～ 7 次元のテーブルの一部。この場合、データ記述記入項目では、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。
- ASCENDING/DESCENDING KEY が DBCS 項目として定義されている場合には、DBCS 項目にすることができます。
- ASCENDING/DESCENDING KEY が国別データ項目として定義されている場合は、国別データ項目にすることができます。

ID-1 は、以下のものにすることは**できません**。

- USAGE IS INDEX
- 浮動小数点データ項目
- USAGE IS POINTER、 USAGE IS FUNCTION-POINTER, USAGE IS PROCEDURE-POINTER、または USAGE IS OBJECT REFERENCE
- ウィンドウ化日付フィールド

ID-1 は、テーブル・エレメント内のすべてのオカレンスを参照する必要があります。つまり、添え字を付けたり、参照変更はしないでください。

データ部の ID-1 に関する記述は、INDEXED BY 句を付けた OCCURS 文節を含まなければなりません。さらに、その OCCURS 文節の中に KEY IS 句も含む必要があります。

AT END

ここには、関連する WHEN 句の中にあるどの条件をも満たせずに検索処理が終了した場合に起こる条件を記述します。

WHEN 句 (二分探索)

WHEN 比較条件が指定されている場合、その比較はデータ名の長さおよび符号に基づきます。たとえば、データ名の長さが検索引き数の長さより短い場合は、比較を行う前に、その検索引き数はデータ名の長さまで切り捨てられます。検索引き数に符号があり、データ名が無符号である場合、比較を行う前に、検索引き数の符号は除去されます。

WHEN 句において、この句の中にある指標のどの設定値についても条件を満たすことが**できない**場合には、検索は失敗に終わります。この場合には、制御は、AT END 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、次の実行可能なステートメントに渡されます。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。

条件名-1
条件名-2
指定する条件名は、単一の値を持たねばならず、このテーブル・エレメントに対して ASCENDING/DESCENDING KEY ID と関連付けられていなければなりません。

データ名-1
データ名-2
ID-1 のテーブル・エレメントの中の ASCENDING/DESCENDING KEY データ項目を指定する必要があります。また、最初の ID-1 指標名によって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。

ID-3、リテラル-1、または算術式-1 と比較するために、データ名-1 は、比較の規則に従う有効なオペランドでなければなりません。

ID-4、リテラル-2、または算術式-2 と比較するために、データ名-2 は、比較の規則に従う有効なオペランドでなければなりません。

データ名-1 およびデータ名-2 は、以下にはなりません。

- 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
- ウィンドウ化日付フィールド

ID-3
ID-4
ID-1 の ASCENDING/DESCENDING KEY データ項目にすることはできません。ID-1 の最初の指標名によって添え字付けされた項目にすることもできません。

ID-3 と ID-4 は、POINTER、 FUNCTION-POINTER、 PROCEDURE-POINTER、または OBJECT REFERENCE の usage で定義されたデータ項目にはできません。

ID-3 と ID-4 は、ウィンドウ化日付フィールドにはできません。

ID-3 またはリテラル-1 がクラス国別の場合は、データ名-1 のクラスを国別にする必要があります。

ID-4 またはリテラル-2 がクラス国別の場合は、データ名-2 のクラスを国別にする必要があります。

算術式
230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/DESCENDING KEY データ項目、または ID-1 の最初の指標名が添え字付けする項目にはしないでください。

WHEN 句の中で ASCENDING/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/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

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。

VARYING 句のオブジェクトが別のテーブル・エレメントの指標名である場合は、フォーマット 1 の SEARCH ステートメントは、同時に 2 つのテーブル・エレメントを検索します。

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ためには、OCCURS DEPENDING ON 文節 (データ名-1) のオブジェクトに、テーブルの現在の長さを示す値が含まれるようにする必要があります。

SEARCH ステートメントの範囲は、次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。
- 分離記号のピリオド。
- 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

SET ステートメントは、次に示す処理のいずれかを実行するために使用されます。

- 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 IS POINTER データ項目をあるデータ・アドレスに設定する。
- USAGE IS PROCEDURE-POINTER データ項目をある記入項目アドレスに設定する。
- USAGE IS FUNCTION-POINTER データ項目をある記入項目アドレスに設定する。
- USAGE OBJECT REFERENCE データ項目を、あるオブジェクト・インスタンスを参照するように設定する。

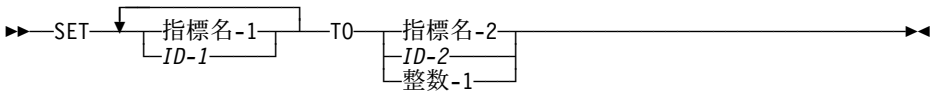
指標名は、OCCURS 文節の INDEXED BY 句を通して付与されたテーブルと関連付けられています。プログラムでさらに定義されることはありません。

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)、そのような 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 - SET (基本的なテーブル処理)



指標名-1、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しなければなりません。受け取りフィールドは、ウィンドウ化日付フィールドにはできません。

指標名-2
送り出しフィールド。

SET ステートメントが実行される前の値は、関連付けられたテーブルのオカレンス数に対応していなければなりません。

ID-2
送り出しフィールド。

指標データ項目または基本数字整数項目のいずれかを指定しなければなりません。送り出しフィールドは、ウィンドウ化日付フィールドにはできません。

整数-1
送り出しフィールド。

これは、正の整数でなければなりません。

表50 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たものです。

表 50. 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

	受け取りフィールド		
送り出しフィールド	**指標名**	**指標データ項目**	**整数データ項目**
指標名	有効	有効*	有効
指標データ項目	有効*	有効*	-
整数データ項目	有効	-	-
整数リテラル	有効	-	-

* 変換は行われない。

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。ID の受け取りフィールドに関連付けられた添え字付けまたは指標付けがある場合には、フィールドが処理される直前にそれが評価されます。

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。

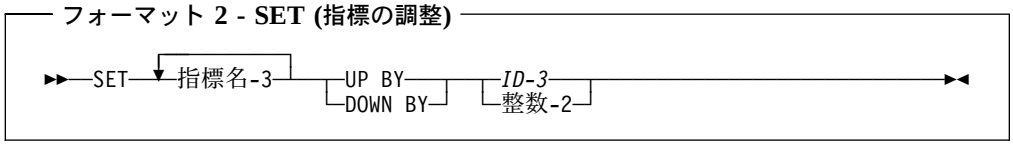
SEARCH または PERFORM ステートメント実行後の指標名の値は未定義となることがあります。したがって、他のテーブル処理操作を行う前に、フォーマット 1 SET ステートメントでそのような指標名を再初期設定してください。

指標名-2 が、OCCURS DEPENDING ON 文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には、未定義の値が ID-1 に受け取りられる可能性があります。

複合 OCCURS DEPENDING ON の詳細については、 *Enterprise COBOL プログラミング・ガイド* 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、受け取りフィールドの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。



受け取りフィールドは、指標名-3 により指定することができます。この指標名の値は、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数に対応していなければなりません。

送り出しフィールドは、ID-3 または整数-2 として定義することができます。そして、ID-3 は基本整数データ項目、整数-2 は 0 以外の整数でなければなりません。データ名-3 をウィンドウ化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 の 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 の値で表される発生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の増加するフィールド値または減少するフィールド値が、すべての受け取りフィールドにおいて使用されます。

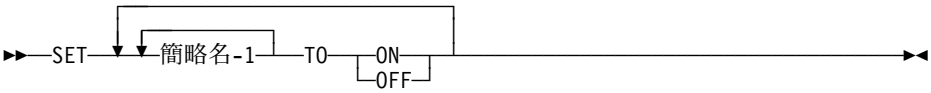
指標名-3 が、OCCURS DEPENDING ON 文節を含む従属項目を持つテーブルに対するものである場合、また ODO のオブジェクトが、フォーマット 2 の SET ステートメントの実行前に変更される場合、指標名-3 には、関連するテーブルの発生数に相当する値が入りません。

複合 OCCURS DEPENDING ON の詳細については、 *Enterprise COBOL プログラミング・ガイド* 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 - SET (外部スイッチ)

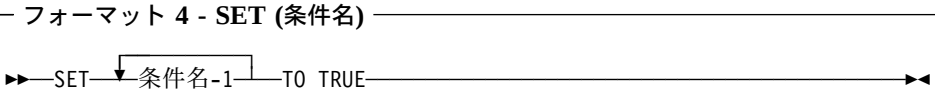


簡略名
外部スイッチと関連付けられていて、その状況は変更可能です。

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 文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 - SET (条件名)



条件名-1
条件変数と関連付けられていなければなりません。

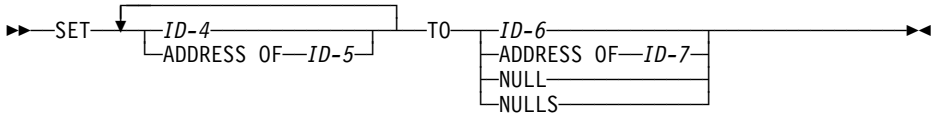
条件名-1 の VALUE 文節の中で複数のリテラルが指定されている場合には、関連付けられた条件変数は、最初のリテラルに等しく設定されます。

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 - SET (使用 POINTER データ項目)



ID-4
受け取りフィールド。

USAGE IS POINTER として記述されている必要があります。

ADDRESS OF ID-5
受け取りフィールド。

ID-5 は、リンケージ・セクション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 として記述されている必要があります。

ADDRESS OF ID-7
送り出しフィールド。ID-7 には、リンケージ・セクション、ワーキング・ストレージ・セクション、またはローカル・ストレージ・セクション内の 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 ID-7** には、ID の内容ではなく、ID のアドレスを入れます。

NULL
NULLS
送り出しフィールド。

受け取りフィールドが、無効なアドレスの値を含むように設定します。

表51 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 51. 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	受け取りフィールド		
	USAGE IS POINTER	ADDRESS OF	NULL/NULLS
USAGE IS POINTER	有効	有効	-
ADDRESS OF	有効	有効	-
NULL/NULLS	有効	有効	-

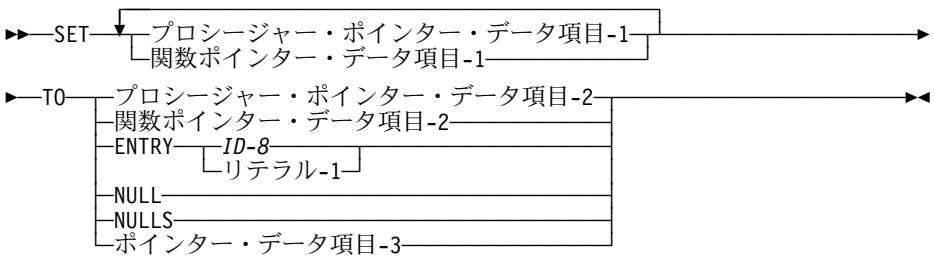
フォーマット 6: プロシージャー・ポインターおよび関数ポインターのデータ項目用の SET

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 - SET (プロシージャーおよび関数ポインター・データ項目)



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であり、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であり、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
その値がプログラム名になり得るような英数字項目として定義されている必要があります。詳細については、86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 は @、#、および $ の各文字を含むことができます。

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、86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、以下に示す種類の入り口点の 1 つを参照しなければなりません。

- PROGRAM-ID ステートメント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、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

NULL
NULLS
受け取りフィールドが、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 を非 COBOL 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

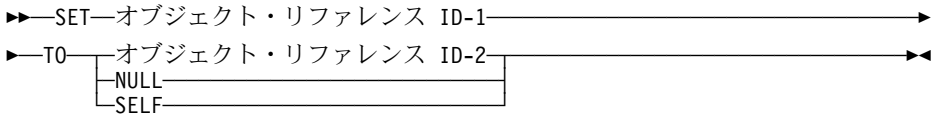
以下の例では、関数ポインターをサービスに戻す C 関数への COBOL CALL を説明しており、サービスへの COBOL CALL が続きます。

```
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

このフォーマットの SET ステートメントを実行すると、受け取り項目の値は送り出し項目の値によって置き換えられます。

フォーマット 7 - SET (オブジェクト・リファレンス・データ項目)



オブジェクト・リファレンス ID-1 とオブジェクト・リファレンス ID-2 は、オブジェクト・リファレンス ID-1 が受け取りで、オブジェクト・リファレンス ID-2 が送り出しとなる USAGE OBJECT REFERENCE として定義される必要があります。オブジェクト・リファレンス ID-1 が特定クラスのオブジェクト・リファレンスとして定義される (「USAGE OBJECT REFERENCE クラス名」として定義される) 場合、オブジェクト・リファレンス ID-2 は、同じクラスの、またはそのクラスから発生するオブジェクト・リファレンスである必要があります。

形象定数 NULL が指定される場合、受け取りオブジェクト・リファレンス ID-1 は NULL 値に設定されます。

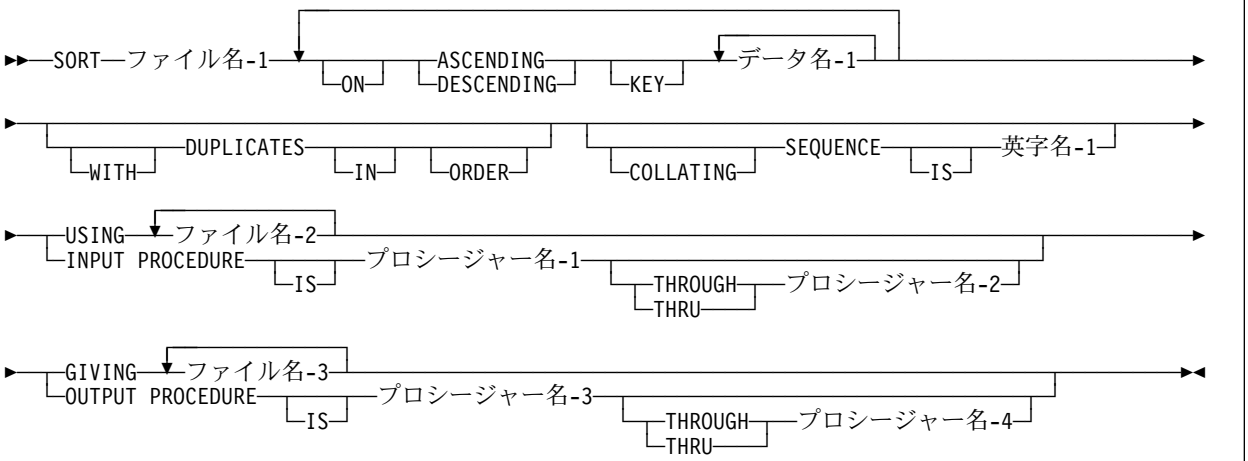
SELF が指定される場合、SET ステートメントが、メソッドの手続き部に表示されなければなりません。この場合、オブジェクト・リファレンス ID-1 は、現在実行中のメソッドが呼び出されたオブジェクトを参照するように設定されます。

SORT ステートメント

SORT ステートメントは、1 つまたは複数のファイルからレコードを受け取り、指定されたキー (複数も可能) に従ってそれらをソートし、OUTPUT PROCEDURE を通して、または出力ファイルの中で、それらのソート済みファイルを利用できるようにします。340ページの『MERGE ステートメント』も参照してください。SORT ステートメントは、宣言部分の中を除き手続き部のどこにでも置くことができます。

SORT ステートメントは、 THREAD オプションを指定してコンパイルされたプログラムではサポートされません。

フォーマット



ファイル名-1
ソートされるレコードを記述している SD 記入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 文節または SAME SORT-MERGE AREA 文節の中で指定することはできません。GIVING 文節に関連したファイル名 (ファイル名-3...) は、SAME AREA 文節には指定できません。ただし、それらを SAME RECORD AREA 文節に関連付けることは可能です。

ASCENDING/DESCENDING KEY 句

この句は、指定されたソート・キーに基づいて、レコードを昇順または降順 (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しなければなりません。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 文節を含んでいたり、OCCURS 文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 データ項目は、点在させることはできません。
- KEY データ項目は、可変オカレンス・データ項目を含むグループ項目にすることはできません。
- KEY データ項目は、修飾することができます。
- KEY データ項目には、以下を指定できます。
  - 任意のクラスおよびカテゴリーのデータ項目。ただし、以下に示すウィンドウ化日付フィールドのものは除きます。
  - 以下の条件のもとでのウィンドウ化日付フィールド。
    - GIVING 句に索引付きファイルを指定してはなりません。ファイル・システムが想定または行う (2 進) 順序付けは、ソートの出力で提供されるウィンドウ化日付順序付けと矛盾するからです。このような索引付きファイルにウィンドウ化日付マージ出力を書き込もうとしても、ファイルのアクセス方法 (ファイル制御項目の ACCESS MODE) に応じて、失敗するか、または 2 進順序付けが再び行われます。
    - 英数字のウィンドウ化日付フィールドを SORT ステートメントの KEY として指定する場合は、マージ操作の有効な照合シーケンスが EBCDIC でなければなりません。したがって、SORT ステートメントの COLLATING SEQUENCE 句、またはこの句を指定しない場合は、OBJECT-COMPUTER 段落の PROGRAM COLLATING SEQUENCE 句に、EBCDIC または NATIVE 以外の照合シーケンスを指定してはなりません。

    SORT ステートメントがこれらの条件を満たしていると、実行環境に世紀ウィンドウ化をサポートするソート製品が組み込まれているものと想定して、SORT 2000 年機能を利用してソート操作が行われます。

    年末尾型ウィンドウ化日付フィールドは SORT ステートメントの KEY として指定することが可能で、それに対応する機能として、ソート製品の世紀ウィンドウ化機能を利用できます。

    KEY データ項目としてウィンドウ化日付フィールドを使用する方法については、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

ファイル名-3 が、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りません。そして、そのデータ名-1 によって参照されるデータ項目は、そのファイルの主レコード・キーに関連付けられたデータ項目として、このレコードの中で同じ文字位置を占めなければなりません。

ソート処理の方向は、次に示すようにキーワードの ASCENDING を指定するか、または DESCENDING 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。
- 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 (235ページの『条件式』の「比較条件」を参照)。
- 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。それ以外のキー・データ項目については、比較条件のオペランド比較規則に従って比較が行われます。

DUPLICATES 句

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 句の詳しい使用方法については、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

COLLATING SEQUENCE 句

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、英数字以外のキーには影響を与えません。

英字名-1

これは SPECIAL-NAMES 段落の ALPHABET 文節で指定されていなければなりません。英字名文節の句のうちいずれか 1 つを指定することができ、次のような結果になります。

STANDARD-1

ASCII 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 (ASCII 照合シーケンスについては、556ページの付録C、『EBCDIC および ASCII の照合シーケンス』を参照)。

STANDARD-2

国際標準 646 で定義された ISO 7 ビット・コードの国際参照バージョンである、情報処理交換用の 7 ビット・コード化文字セットが、すべての英数字比較のために使用されます。

NATIVE

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 照合シーケンスについては、556ページの付録C、『EBCDIC および ASCII の照合シーケンス』を参照)。

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 照合シーケンスについては、556ページの付録C、『EBCDIC および ASCII の照合シーケンス』を参照)。

リテラル
英字名文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 文節 (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 句を両方とも省略した場合には、EBCDIC 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は、これらのファイルを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、読み込み、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、データ部の中の FD 記入項目に記述されている必要があります。

USING 句が指定されており、ファイル名-1 に可変長のレコードが入っている場合、入力ファイル (ファイル名-2、. . .) の中に含まれるレコード・サイズは、ファイル名-1 で記述されている最小レコードより小さいか、または最大レコードより大きくなければなりません。ファイル名-1 が固定長レコードを含んでいる場合は、入力ファイルに入っているレコードのサイズは、ファイル名-1 で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、 *Enterprise COBOL プログラミング・ガイド* 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
INPUT 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
INPUT PROCEDURE 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、および PERFORM 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では、MERGE、RETURN、または SORT のいずれのステートメントも実行させ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 を挿入します。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 句が指定されていると、ファイル名-1 の中のすべてのソートされたレコードは、自動的に出力ファイル (ファイル名-3、. . .) に転送されます。

すべての出力ファイルは、データ部の中の FD 記入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入っている場合、ファイル名-1 に入っているレコードのサイズは、その出力ファイルに記述されている最小レコードより小さいかまたは最大レコードより大きくなければなりません。出力ファイルが固定長レコードを含んでいる場合は、ファイル名-1 に入っているレコードのサイズは、出力ファイルで記述されている最大レコードより大きくすることはできません。詳しくは、*Enterprise COBOL プログラミング・ガイド* 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、. . .) をオープンしてはなりません。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。この終了は、オプションの句が指定されていない CLOSE ステートメントを実行した場合と同様にして行われます。

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

この句には、ソート処理から得られた出力レコードを選択したり修正したりするためのプロシージャーの名前を指定します。

プロシージャー名-3
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-4
OUTPUT PROCEDURE の中の最後のセクションまたは段落を指定します。

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、出力をプロシージャーの範囲内で CALL、EXIT、GO TO、および PERFORM 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は、MERGE、RELEASE、または SORT のステートメントを実行させ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場合、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した後で、制御がそ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。

注: INPUT PROCEDURE 句および OUTPUT PROCEDURE 句は、基本的な PERFORM ステートメント用の句と類似しています。たとえば、OUTPUT PROCEDURE の中で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。INPUT PROCEDURE または OUTPUT PROCEDURE の中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 (312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。ソート制御データ・セットの定義は、SORT-CONTROL 特殊レジスターを使用して行います。

注: 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
19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
18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
19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
20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
18ページの『SORT-CONTROL』を参照してください。

SORT-RETURN 特殊レジスター
20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

SORT ステートメントが独立セグメントにないセクションに置かれている場合、その SORT ステートメントで参照される入力プロシージャーや出力プロシージャーは、次のいずれかでなければなりません。

- 非独立セグメント内に完全に収まっている。
- 全体が 1 つの独立セグメントに収まっている。

SORT ステートメントが独立セグメント内に置かれている場合、その SORT ステートメントで参照される入力プロシージャーや出力プロシージャーは、次のいずれかでなければなりません。

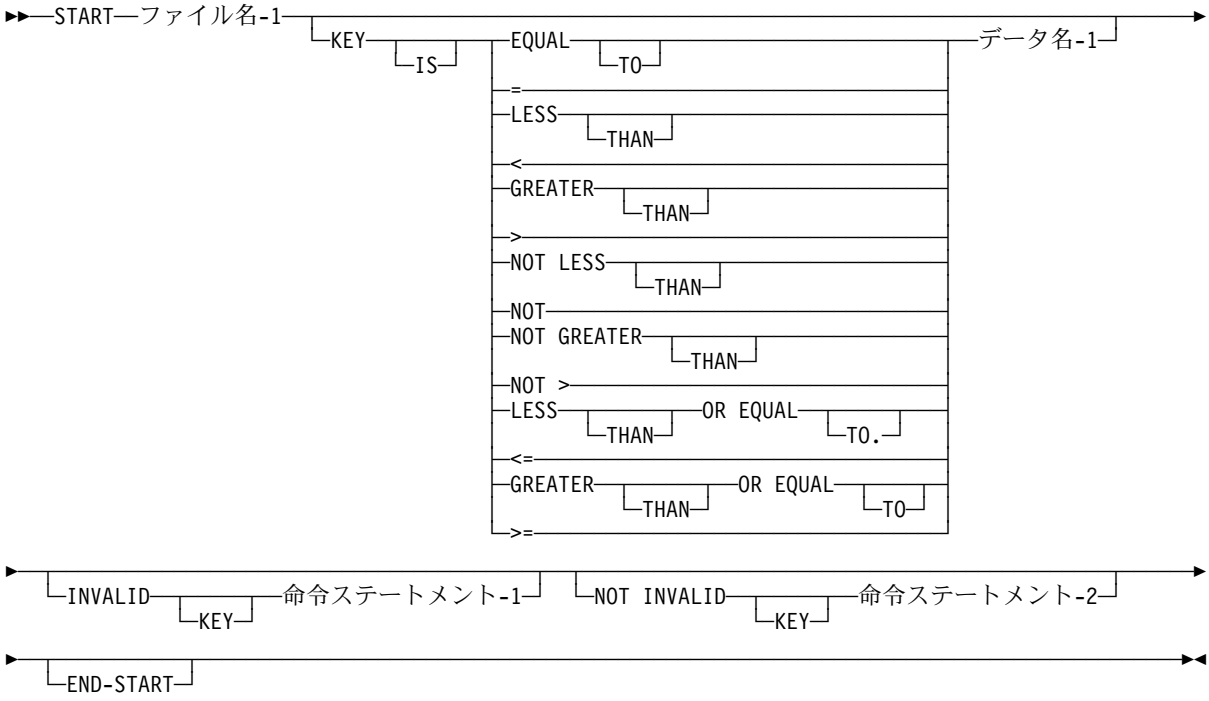
- 非独立セグメント内に完全に収まっている。
- 全体がその SORT ステートメントと同じ独立セグメント内に収まっている。

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。

フォーマット



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、データ部の FD 記入項目の中で定義されている必要があります。またソート・ファイルの名前にすることはできません。

KEY 句

KEY 句を指定すると、ファイル位置標識が、比較で合致するキー・フィールドを持つファイル内の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。

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。

FILE-CONTROL 記入項目の中に FILE STATUS 文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、関連する状況キーが更新されます (267ページの『状況キー』を参照)。

INVALID KEY 句

ファイルのどのレコードも比較が満たされない場合は、無効キー条件が存在します。ファイル位置標識の位置が定義されておらず、(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 (267ページの『共通の処理機能』にある「無効キー条件」を参照)。

INVALID KEY 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。

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。

詳細については、260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 です。

KEY 句を指定しない場合は、EQUAL TO 比較のために使用されるキー・データ項目は、基本 RECORD KEY 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 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

これは以下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 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、比較を満たすレコード・キーを持つファイル内の最初のレコードに位置付けられます。比較の際にオペランドの長さが等しくない場合には、長い方のフィールドの右側が短い方のフィールドに合わせて切り捨てられたかのようにして処理します。PROGRAM COLLATING SEQUENCE 文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常に実行すると、データ名-1 が関連付けられている RECORD KEY は、後続の READ ステートメントの参照キーになります。

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。

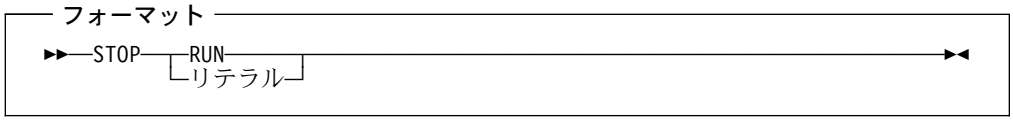
KEY 句の指定の有無にかかわらず、比較の際に使用されるキー・データ項目は、RELATIVE KEY データ項目です。数字比較の規則が適用されます。

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
固定小数点数字リテラル (符号あり、または符号なし)、または英数字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、プログラムの実行途中でオペレーターの介入が必要となるような特別の事態 (ある特定のテープやディスクを装てんする必要があるとき、特定の日常的なコードを入力する必要があるときなど) の際に役立ちます。ただし、オペレーターの介入が必要な場合には、ACCEPT ステートメントや DISPLAY ステートメントを使用してください。

THREAD コンパイラー・オプションを指定してコンパイルされたプログラムでは、STOP リテラル・ステートメントを使用しないでください。

STOP RUN を指定すると、オブジェクト・プログラムの実行が終了し、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、実行単位を構成する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、次の表を参照してください。

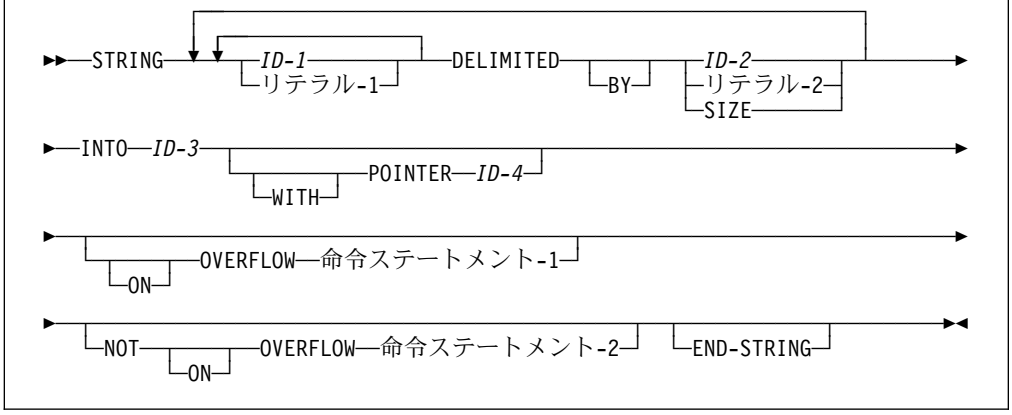
終了ステートメント	メインプログラム	サブプログラム
STOP RUN	呼び出しプログラムへ戻る (それがシステムの場合は、アプリケーションを終了させる)。	メインプログラムを呼び出したプログラムに直接戻る (それがシステムの場合は、アプリケーションを終了させる)。

STRING ステートメント

STRING ステートメントは、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
これは**送り出しフィールド (場合によっては複数)** を表します。

送り出しフィールド、または任意の区切り文字が基本数字項目である場合は、それぞれの数字項目を整数として記述する必要があり、その PICTURE 文字ストリングに記号 P を含めることはできません。

リテラル-1
これは**送り出しフィールド (場合によっては複数)** を表します。

STRING ステートメント内のすべてのリテラルには、以下の規則が適用されます。

- リテラルまたは ID が英数字の場合は、ALL リテラルを除くすべてのリテラルを任意の形象定数として指定できます。それぞれの形象定数は、1 文字の英数字リテラルであるとみなされます。
- リテラルまたは ID が国別の場合は、任意のリテラルを形象定数 SPACE、ZERO、または QUOTE として指定できます。それぞれの形象定数は、1 文字の国別リテラルであるとみなされます。

STRING ステートメント内の任意の ID またはリテラルがクラス DBCS の場合は、ID-4 を除くすべての ID とリテラルがクラス DBCS でなければなりません。

STRING ステートメント内の任意の ID またはリテラルがクラス国別の場合は、ID-4 を除くすべての ID とリテラルがクラス国別でなければなりません。

STRING ステートメントでは、ID をウィンドウ化日付フィールドにすることはできません。

DELIMITED BY 句

DELIMITED BY 句は、ストリングの制限を設定します。

ID-2、リテラル-2
これらは区切りです。つまり、転送するデータを区切る文字 (複数可) です。

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 が同じデータ記述記入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。

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

ID-3
これは**受け取りフィールド**を表します。

ID-3 は、編集済みデータ項目または外部浮動小数点項目を表すことはできません。また、JUSTIFIED 文節で記述することもできません。

ID-3 は、参照変更にすることができます。

ID-3 と ID-4 が同じストレージ域を占めると、たとえそれらの ID が同じデータ記述記入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。

POINTER 句

ID-4
これは**ポインター・フィールド**を表し、受け取りフィールドの中のある文字位置を指し示します。

これは、受け取り領域の長さに 1 を加えた長さに等しい値を入れることができる基本整数データ項目でなければなりません。ポインター・フィールドは、その PICTURE 文字ストリングの中で記号 P を含むことはできません。

ID-3 (受け取りフィールド) が DBCS データ項目である場合、ID-4 は受け取りフィールドの中の DBCS 文字の相対文字位置 (相対バイト位置ではなく) を表します。 ID-3 が国別データ項目の場合、ID-4 は、受け取りフィールド内の相対国別文字位置を表します。

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、実行は命令ステートメント-1 の中に指定された各ステートメントの規則に従って継続されます。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 STRING ステートメントの終了へと移されます。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る場合には、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。

詳細については、260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 ステートメントを実行すると、データが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。
  - 送り出しフィールドが国別の場合は、国別から国別への基本移動の規則を使用して、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。
  - 送り出しフィールドが DBCS の場合は、DBCS から DBCS への基本移動の規則を使用して、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、英数字から英数字への基本移動の規則を使用して、受け取りフィールドに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 (346ページの『MOVE ステートメント』を参照)。
- DELIMITED BY ID / リテラルを指定した場合、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 ID が指定されている場合、それぞれの送り出しフィールドの全体が受け取りフィールドに転送されます。
- 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。(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 を加えた長さと等しい値を入れることができるフィールドとして定義される必要がある点に注意してください。システムが転送の終了時にポインターを更新するときには、このフィールドは算術オーバーフローを除外します。)
- 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。

注: 添え字、参照変更、可変長または可変位置の計算、および関数計算は、STRING ステートメントの実行開始時に 1 回だけ行われます。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、または関数の引き数として使用される場合、つまり、STRING 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、STRING ステートメントの実行開始時に決められるので、STRING 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。**

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。

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、413ページの図17 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
```

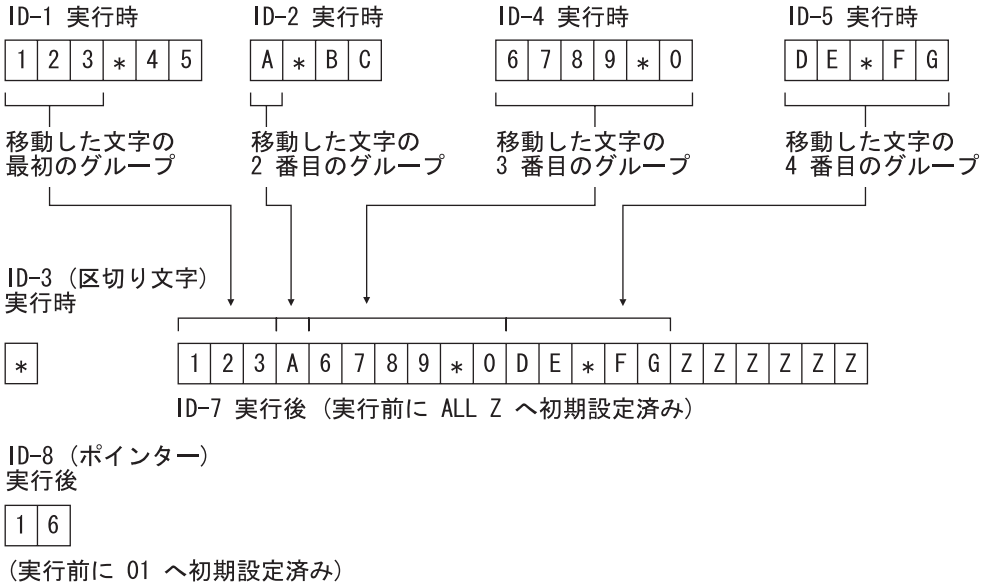
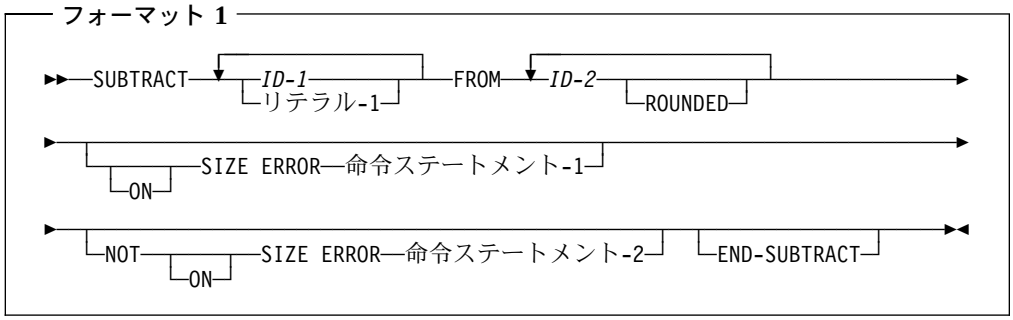


図 17. STRING ステートメントの実行結果

SUBTRACT ステートメント

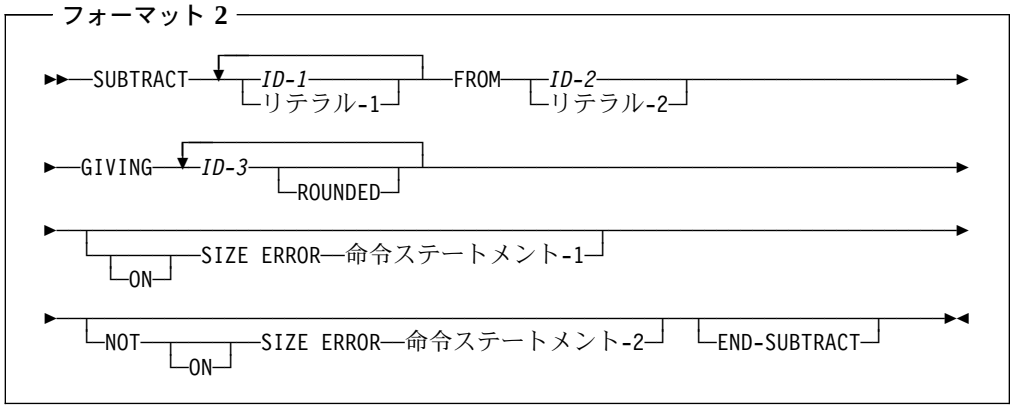
SUBTRACT ステートメントは、1 つまたは複数の数値項目から、1 つの数値項目または 2 つ以上の数値項目の和を減算して、その結果を保管します。

フォーマット 1



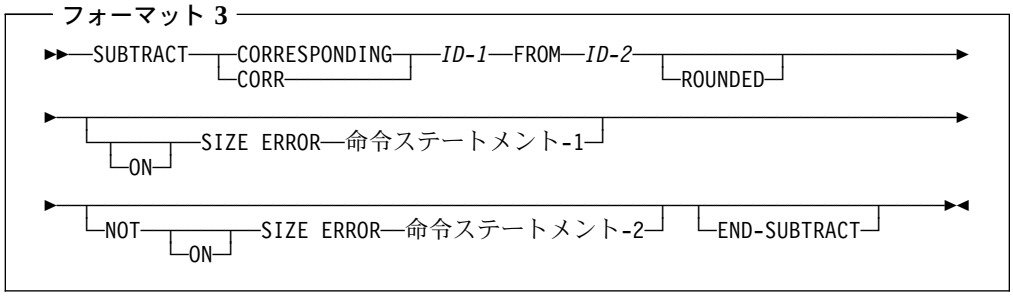
キーワード FROM の前にあるすべての ID またはリテラルは互いに加算され、その和が ID-2 から減算され、ID-2 に直接保管されます。この処理は、ID-2 が連続する場合、それぞれの ID-2 ごとに、ID-2 が指定されている順序で左から右へと繰り返されます。

フォーマット 2



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



ID-1 内の基本データ項目は ID-2 内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。算術の中間結果の詳細については、 *Enterprise COBOL プログラミング・ガイド* 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ようになります。

ID 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、ID がキーワード GIVING の後にある場合を除き、基本数字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項目の名前でなければなりません。

フォーマット 3 では、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、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には、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 文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 つまたは複数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 が指定しない場合、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 つまたは複数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: 233ページの『日付フィールドの関連する減算』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)。詳細については、234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、262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、263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)

261ページの『CORRESPONDING 句』を参照してください。

END-SUBTRACT 句

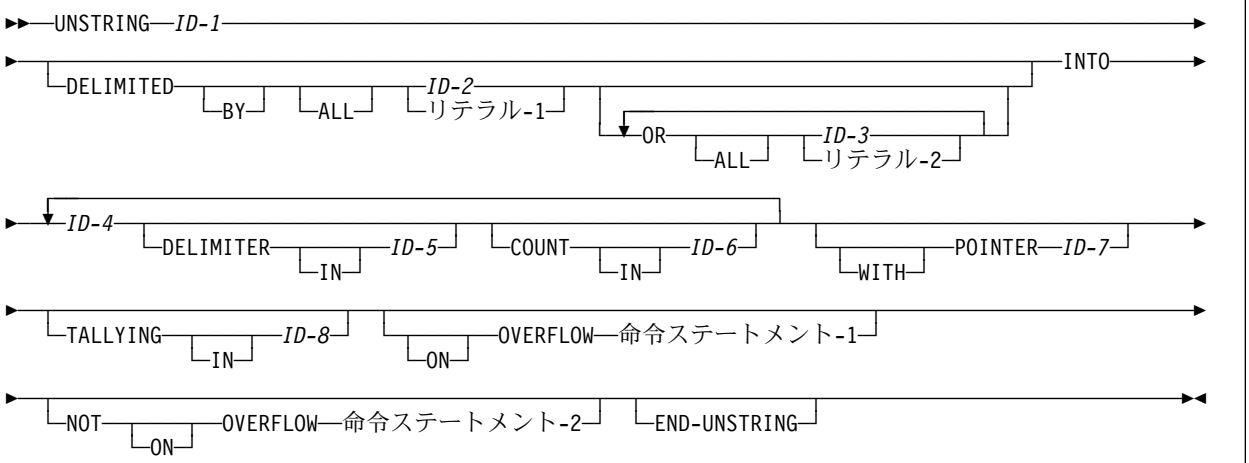
この明示的範囲終了符号は、SUBTRACT ステートメントの範囲を区切るために使用されます。END-SUBTRACT 句を使用することによって、条件 SUBTRACT ステートメントを他の条件ステートメントの中にネストすることができます。END-SUBTRACT 句は、命令 SUBTRACT ステートメントと共に使用することもできます。

詳細については、260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、英数字、英数字編集、英字、国別、または DBCS データ項目でなければなりません。

ID-1 は、参照変更にすることができます。

ID-1、ID-2、リテラル-1、または ID-3、ID-4、ID-5、リテラル-2 のオカレンスのいずれかがクラス DBCS の場合は、すべてがクラス DBCS でなければなりません。

ID-1、ID-2、リテラル-1、または ID-3、ID-4、ID-5、リテラル-2 のオカレンスのいずれかがクラス国別の場合は、すべてがクラス国別でなければなりません。

ID-1 が英数字、英数字編集、または英字の場合は、ID-4、ID-5 か ID-2、ID-3、リテラル-1、リテラル-2 のオカレンスのいずれかが、これらのカテゴリーの 1 つを持つ必要があります。

UNSTRING ステートメントのどの ID もウィンドウ化日付フィールドにはできません。

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。詳細については、423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 (346ページの『MOVE ステートメント』を参照)。たとえば、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。

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。

ID-2
ID-3
以下のいずれかにする必要があります。

- 英数字データ項目
- DBCS データ項目
- 国別データ項目

どちらも区切り文字を表します。

リテラル-1
リテラル-2
以下のいずれかにする必要があります。

- ALL リテラルを除いた英数字リテラル。任意の形象定数も含まれます。
- DBCS リテラル。形象定数 SPACE も含まれます。
- 国別リテラル。形象定数 SPACE、ZERO、または QUOTE も含まれます。

どちらも区切り文字を表します。形象定数を指定するときには、それは 1 文字のリテラルとみなされます。

ALL
任意の区切り文字の 1 つまたは複数の連続するオカレンスは、唯一つ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、指定されている場合は、区切り文字受け取りフィールド (ID-5) 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指定すると、OR 条件が存在し、いずれかの区切り文字がオーバーラップして現れるたびに、指定されたシーケンスで送り出しフィールドの区切り文字として認識されます。

以下に例を示します。

```
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
これは、**データ受け取りフィールド**を表します。

usage には、DISPLAY、DISPLAY-1、または NATIONAL を指定する必要があります。これらのフィールドは、以下のいずれかとして定義できます。

- 英字データ項目
- 英数字データ項目
- 数字データ項目 (PICTURE 文字ストリング内に記号 P は含まれない)
- DBCS データ項目
- 国別データ項目

ID-4 は、浮動小数点項目、英数字編集データ項目、または数字編集データ項目として定義することはできません。

DELIMITER IN
DELIMITED BY 句を指定して*いない* 場合、DELIMITER IN 句を指定しては*なりません*。

ID-5
これは、**区切り文字受け取りフィールド**を表します。次のようにすることができます。

- 英数字データ項目
- DBCS データ項目
- 国別データ項目

COUNT IN
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*。

ID-6
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含まれません。

ID-6 は、PICTURE 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-1 (送り出しフィールド) が DBCS データ項目の場合、ID-6 は、送り出しフィールドで検査された DBCS 文字位置数 (バイト数ではない) を示します。

ID-1 が国別データ項目の場合、ID-6 は、送り出しフィールドで検査された国別文字位置数 (バイト数ではない) を示します。

POINTER 句

POINTER 句を指定した場合、ポインター・フィールドの値は、送り出しフィールドの文字位置が検査されるたびに 1 だけ増加します。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、ユーザーは UNSTRING ステートメントの実行を開始する前に ID-7 を初期設定しなければなりません。

ID-7
ポインター・フィールドです。このフィールドには、送り出しフィールドでの相対文字位置を示す値が入っています。

ID-7 は、PICTURE 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。

これは、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-1 (送り出しフィールド) が DBCS データ項目である場合、ID-7 は送り出しフィールドの中の DBCS 文字の相対文字位置 (相対バイト位置ではなく) を表します。

| ID-1 が国別データ項目の場合、ID-7 は、相対バイト位置ではなく、相対国別文字位置
| を示します。

TALLYING IN 句

TALLYING 句が指定されるときに、フィールド個数フィールドには、(UNSTRING ステートメント実行時の終わりに) 受け取り領域が作用するデータ数を足した初期値と等しい値が含まれます。

この句を指定する場合には、ユーザーは UNSTRING ステートメントの実行を開始する前に ID-8 を初期設定しなければなりません。

ID-8
これは、**フィールド個数フィールド**です。このフィールドは、UNSTRING ステートメントの実行で受け取りフィールドが作用するデータの個数に応じて増加します。

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。

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が生じると、次のような結果となります。

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、実行は、命令ステートメント-1 の中に指定された各ステートメントの規則に従って継続されます。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る場合には、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、実行は命令ステートメント-2 の中に指定された各ステートメントの規則に従って継続されます。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る場合には、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。

詳細については、260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。このサイズは、データ・カテゴリーによって異なります。詳しくは、329ページの表39 を参照してください。

表 52. 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	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 文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、英数字基本項目として扱われ、MOVE ステートメントの規則 (346ページの『MOVE ステートメント』を参照) に従って、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こと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

以下の操作は、1 回だけ、UNSTRING ステートメントの実行開始時に行われます。

- 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

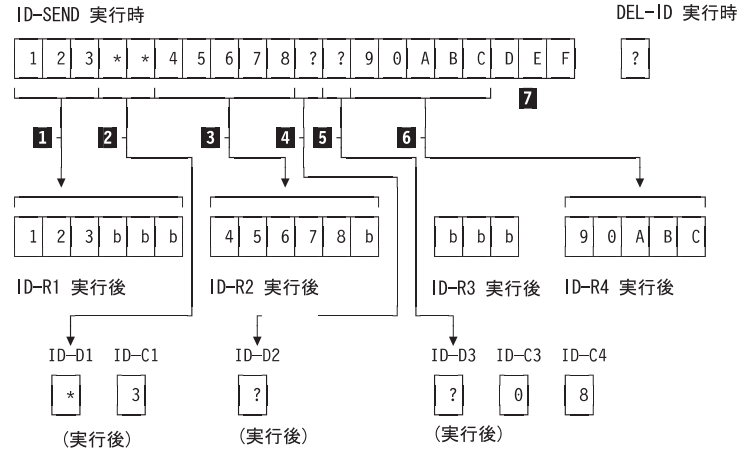
したがって、ID-4、ID-5、ID-6、ID-7、または ID-8 は、UNSTRING ステートメントの中で添え字、参照修飾子、または関数の引き数として使用される場合、つまり、UNSTRING ステートメント内の ID のいずれかの長さまたは位置に影響を与える場合、これらの値は UNSTRING ステートメントの実行開始時に決められるので、UNSTRING ステートメントの実行結果によって影響されることは*ありません*。

UNSTRING ステートメントの例

図18 では、UNSTRING ステートメントの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.
```

（すべてのデータ
受け取りフィールドは
英数字として定義される）



ID-P
（ポインター）

2	1

ID-I
（計算
フィールド）

0	5

（実行後 --
両方とも実行前の
01 に初期化される）

実行の順序は以下のとおりです。

1. ID-R1 に 3 文字が配置される。
2. ALL * が指定されているため、すべての連続するアスタリスクが処理されるが、ID-D1 に置かれるアスタリスクは 1 つだけである。
3. 5 文字が ID-R2 に置かれる。
4. ? が ID-D2 に置かれる。カレント受け取りフィールドは ID-R3 になる。
5. ? が ID-D3 に置かれる。ID-R3 にはスペースが埋め込まれる。文字は転送されないので、0 が ID-C3 に置かれる。
6. 区切り文字は検出されず、5 文字が ID-R4 に入り、8 文字が ID-C4 に入り、最後の区切り文字以降に検出された文字を表す。
7. ID-P は 21 に更新される。送り出しフィールド + 1 の合計長である。ID-T は 5 に更新される。処理されたフィールドに +1 の数。未検査の文字は ID-SEND 内には存在しないため、OVERFLOW EXIT は取られない。

図 18. UNSTRING ステートメントの実行結果

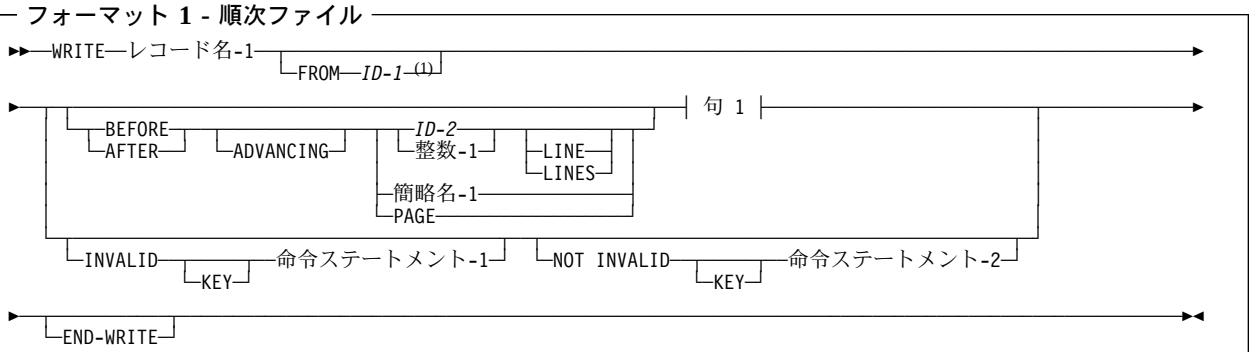
WRITE ステートメント

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のために 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。

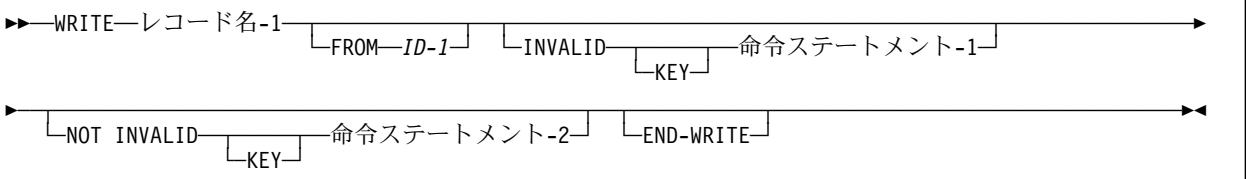
フォーマット 1 - 順次ファイル



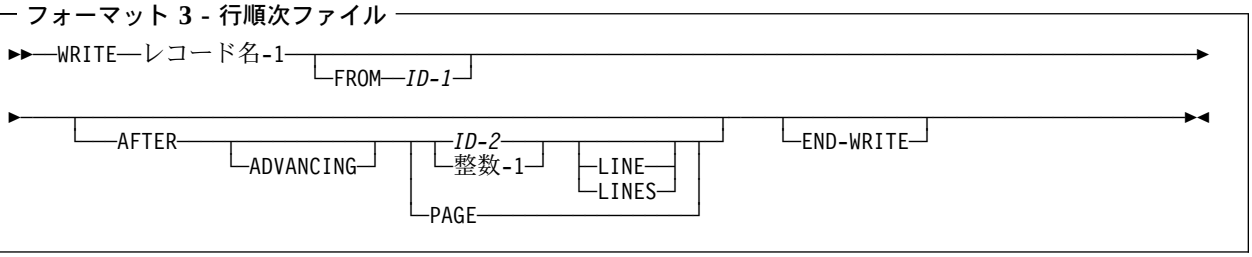
注:

[1] BEFORE、AFTER、INVALID KEY、および AT END OF PAGE の各句は、VSAM ファイルでは無効です。

フォーマット 2 - 索引付きファイルおよび相対ファイル



フォーマット 3 - 行順次ファイル



レコード名-1

データ部の FD 記入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 句を指定した WRITE ステートメントの実行結果は、以下のステートメントを指定した順序で実行する場合と同じになります。

```
MOVE identifier-1 TO record-name-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、以下を参照できます。

- ワーキング・ストレージ・セクション、ローカル・ストレージ・セクション、またはリンケージ・セクションに定義されたデータ項目
- 前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 ステートメントに対して、有効な送り出し項目でなければなりませんん。

ID-1 とレコード名-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 (267ページの『共通の処理機能』の「INTO/FROM ID 句」を参照)。

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。

VSAM ファイルの場合、BEFORE 句および AFTER 句はサポートされていません。QSAM ファイルは順次に編成されます。ADVANCING 句と END-OF-PAGE 句は、印刷ページの各行の縦方向の位置付けを制御します。

単一の WRITE ステートメントには、ADVANCING PAGE 句と END-OF-PAGE 句を指定できます。

印刷ページが中間装置 (たとえば、ディスク) に保持される場合は、これを編集したりブラウズしたりするときのフォーマットとは異なることがあります。

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しなければなりません。ID-2 は、ウィンドウ化日付フィールドにはできません。

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 記入項目に LINAGE 文節が含まれている場合には、その文節で指定された次のページの最初の印刷可能行に位置付けされます。LINAGE 文節を省略すると、位置付けは後に続く次のページの第 1 行目になります。
7. 簡略名を指定すると、チャネル 1 ～ 12 へのスキップ、または行送り抑制が行われます。簡略名は、SPECIAL-NAMES 段落内の環境名-1 と一致しなければなりません。

   簡略名句は、カードせん孔装置ファイルのスタッカー選択のためにも指定することができます。スタッカー選択を使用する場合は、WRITE AFTER ADVANCING を使用しなければなりません。

WRITE ステートメントに ADVANCING 句が指定されているか、ファイルに LINAGE 文節があると、書き出されるレコードの中に紙送り制御文字が生成されます。該当するファイル結合子が EXTERNAL であるときは、実行単位内のすべてのファイル結合子は、書き出されるレコードに紙送り制御文字が生成されるように定義されていなければなりません。つまり、すべてのファイルに LINAGE 文節がある場合は、一部のプログラムが ADVANCING 句指定の WRITE ステートメントを使用し、他のプログラムが ADVANCING 句が指定されていない WRITE ステートメントを使用することができます。ただし、これらのファイルのいずれにも LINAGE 文節がない場合で、しかも ADVANCING 句指定の WRITE を使用するプログラムがある場合には、実行単位内の WRITE ステートメントを使用するすべてのプログラムは、ADVANCING 句指定の WRITE ステートメントを使用しなければなりません。

ADVANCING 句を省略すると、AFTER ADVANCING 1 LINE を指定した場合と同じようにして自動行送りが行われます。

LINAGE-COUNTER の規則

ファイルに対して LINAGE 文節を指定すると、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 または整数を指定している場合、LINAGE-COUNTER は、ID-2 または整数の値だけ増やされます。
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。

注: ADV コンパイラー・オプションが指定されていると、コンパイラーは制御文字を使用できるようにレコード長に 1 バイトを追加します。レコード定義の中に制御文字用に第 1 バイトを確保している場合には、NOADV コンパイラー・オプションを使用してください。LINAGE 文節を使用して定義されているファイルでは、NOADV コンパイラー・オプション

は無効です。コンパイラーはこのようなファイルについては、ADV コンパイラー・オプションが指定されているかのように処理するからです。

END-OF-PAGE 句

VSAM ファイルに関して AT END-OF-PAGE 句はサポートされていません。

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 記入項目には、LINAGE 文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 文節の中で指定します。

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 文節の 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 文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 文節で指定する) 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 文節の 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 文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

VSAM 順次ファイルに関して INVALID KEY 句はサポートされていません。

無効なキー条件は、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、命令ステートメント-1 が実行されます (268 ページの表34 を参照)。
- INVALID KEY 句が指定されていないなら、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 (QSAM ファイルを除く)。さらに、以下のことが発生します。
  - **順次ファイルの場合** - 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。
  - **相対および索引付きファイルの場合** - プログラム実行は、267ページの『状況キー』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、命令ステートメント-4 に制御が移されます。

INVALID KEY 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。

END-WRITE 句

この明示的範囲終了符号は、WRITE ステートメントの範囲を区切るために使用されます。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。

詳細については、260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

ファイルの最大レコード・サイズは、ファイル作成時に設定され、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、論理レコードはレコード名-1 の中でもはや使用することはできません。ただし、次のいずれかの場合は別です。

- 関連するファイルが、SAME RECORD AREA 文節内に指定している場合 (この場合、レコードは SAME RECORD AREA 文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 (188ページの『PICTURE 文節の編集』および 206ページの『USAGE 文節』を参照)。

FILE-CONTROL 記入項目の中で FILE STATUS 文節を指定している場合、WRITE ステートメントが実行されると、それが正常に実行されたかどうかにかかわりなく、関連する状況キーが更新されます。

WRITE ステートメントは、QSAM ファイルでは、OUTPUT モードまたは EXTEND 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。

マルチボリューム・ファイル

マルチボリューム OUTPUT ファイル (テープまたは順次直接アクセス・ファイル) でボリュームの終了が認識されたときは、WRITE ステートメントは次の処理を行います。

- 標準の終了ボリューム・ラベル・プロシージャー
- ボリュームの切り替え
- 標準の開始ボリューム・ラベル・プロシージャー

IBM 3525 でのパンチ機能ファイル

パンチ機能を使用するときは、READ ステートメントの後の入出力操作は、パンチ機能ファイルに対する WRITE ステートメントでなければなりません。

追加データを一部のカードだけにパンチしたい (他のカードにはパンチしない) ときは、最初に出力域を SPACES で満たして、ヌル・カードのためのダミーの WRITE ステートメントを発行する必要があります。

パンチ機能ファイルのためにスタッカー選択をしたい場合は、SPECIAL-NAMES 段落で適切なスタッカー機能名を指定し、次に関連する簡略名を使用して WRITE ADVANCING ステートメントを発行します。

印刷機能ファイル

パンチ機能操作 (指定してある場合) が完了した後、印刷機能ファイルに対して WRITE ステートメントを発行することができます。

追加データを一部のデータ・カードだけに印刷したい (他のカードには印刷しない) ときは、ヌル・カード用の WRITE ステートメントを省略することができます。カードの限界を超えて書き込もうとすると、その結果としてアプリケーションは必ず異常終了します。したがって、END-OF-PAGE 句を指定することはできません。

使用する IBM 3525 の特定のモデルの能力に合わせて、印刷ファイルを 2 行印刷ファイルまたは複数行印刷ファイルのどちらかにすることができます。各行には、64 文字まで印刷することができます。

- 2 行印刷ファイルでは、行は、行 1 (カードの最上端) と行 3 (パンチ段 11 と 12 の間) に印刷されます。行の制御を指定することはできません。自動的な行送りが行われます。
- 複数行印刷ファイルでは、25 行までの文字を印刷することができます。行の制御を指定することができます。行の制御を指定しなければ、自動的な行送りが行われます。

行の制御は、印刷機能ファイルに対して WRITE AFTER ADVANCING ステートメントを発行することによって指定されます。そのようなステートメントの 1 つに行の制御を使用する場合は、そのファイルに対して発行される他のすべての WRITE ステートメントでも行の制御を使用しなければなりません。最大の印刷可能文字数は、スペース文字を含めて 64 です。そのような WRITE ステートメントでは、行送りの抑止を指定することはできません。

ID と整数は、他の WRITE AFTER ADVANCING ステートメントの場合と同じ意味を持っています。ただし、そのような WRITE ステートメントでは、カードの限界を超えて行位置を増やすことはできません。そのようにすると異常終了します。

WRITE AFTER ADVANCING ステートメントの簡略名オプションも指定することができます。SPECIAL-NAMES 段落では、環境名と簡略名を以下のように関連付けることができます。

表 53. SPECIAL NAMES 段落内の環境名の意味

環境名	意味
C02	**行 3**
C03	**行 5**
C04	**行 7**
.	.
.	.
.	.
C12	**行 23**

高機能印刷

環境名 AFP-5A と関連付けられた簡略名で WRITE ADVANCING 句を使用するときには、印刷サービス機能 (PSF) 制御文字が、出力レコードの制御文字の位置に入れられます。この制御文字 (X'5A') を使用すると、高機能印刷 (AFP) サービスを使用することができます。詳細については、印刷サービス機能製品: PSF for OS/390 (5655-B17) の資料を参照してください。

索引付きファイル用 WRITE

WRITE ステートメントを実行する前に、基本レコード・キー (FILE-CONTROL 記入項目の中で定義した RECORD KEY データ項目) を希望の値に設定する必要があります (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)。

FILE-CONTROL 記入項目の中に ALTERNATE RECORD KEY 文節も指定してある場合、DUPLICATES 句が指定されていない限り、各代替レコード・キーも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ときに、保管したとき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。

FILE-CONTROL 記入項目の中に ACCESS IS SEQUENTIAL が指定されている場合、RECORD KEY 値の昇順にレコードを書き出さなければなりません。

FILE-CONTROL 記入項目の中に ACCESS IS RANDOM または ACCESS IS DYNAMIC が指定されている場合、レコードはプログラマーが指定した任意の順序で書き出すことができます。

相対ファイル用 WRITE

OUTPUT ファイルの場合、WRITE ステートメントによって次の処置が取られます。

- ACCESS IS SEQUENTIAL が指定されている場合。

  書き込まれる最初のレコードは、相対レコード番号 1 を持ち、2 番目のレコードは相対レコード番号 2 を持ち、3 番目のレコードは相対レコード番号 3 を持つ、というようになります。

  FILE-CONTROL 記入項目の中で 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。

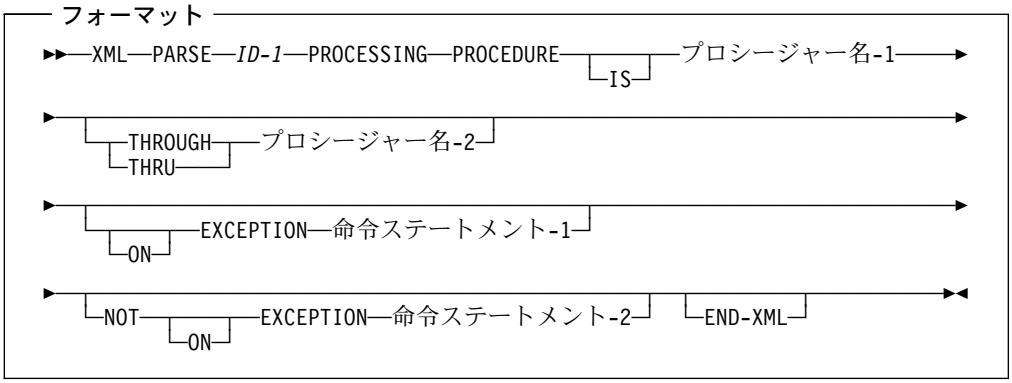
- ACCESS IS RANDOM または ACCESS IS DYNAMIC を指定した場合は、WRITE ステートメントを発行する前に、このレコードの希望する相対レコード番号を RELATIVE KEY に入れておかなければなりません。WRITE ステートメントが実行されると、このレコードはファイル内の指定された相対レコード番号の位置に入れられます。

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 ステートメントを発行する前に、このレコードの希望する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の中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 PARSE ステートメント

XML PARSE ステートメントは、Enterprise COBOL ランタイムに含まれる高速 XML パーサーへの Enterprise 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。

フォーマット



ID-1

ID-1 には、XML 文書の文字ストリームを含む英数字データ項目または国別データ項目を指定します。ID-1 を関数 ID にすることはできません。

ID-1 が英数字の場合、その内容は、436ページの『XML 文書のコード化文字セット』でリスト表示されている単一バイト文字セットの 1 つを使用してエンコードする必要があります。エンコード宣言を含まない EBCDIC XML 文書は、CODEPAGE コンパイラー・オプションで指定されたコード・ページを使用して解析されます。エンコード宣言を含まない ASCII XML 文書は、コード・ページ 819 (ISO 8859-1、Latin 1 オープン・システムとも呼ばれる) を使用して解析されます。

ID-1 が国別の場合、その内容は、CCSID 01200 (Unicode UTF-16) を使用してエンコードする必要があります。この 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。そうした文字を表現する場合は、

"𐄁" または
"#x10101;"

などの文字参照を使用します。

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

処理プロシージャー内にある先頭のまたは唯一のセクション、あるい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。

処理プロシージャーは、XML イベントを処理するステートメントから構成されます。また、処理プロシージャーの範囲の中には、CALL、EXIT、GO TO、GOBACK、INVOKE、および PERFORM ステートメントにより実行されるステートメントがすべて含まれます。

処理プロシージャーで XML PARSE ステートメントを*直接* 実行することはできません。ただし、処理プロシージャーの INVOKE ステートメントまたは CALL ステートメントにより、メソッドまたは最外部のプログラムに制御を渡した場合、ターゲット・メソッドまたはプログラムは、同じ XML PARSE ステートメントまたは別の XML PARSE ステートメントを実行できます。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の最後のステートメントの後には、コンパイラーにより、戻り機構が挿入されます。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。GOBACK ステートメント、または EXIT PROGRAM ステートメントでパーサーに戻ることはできません。

処理プロシージャーの詳細については、435ページの『制御フロー』、および 436ページの『処理プロシージャー』を参照してください。

ON EXCEPTION

ON EXCEPTION 句では、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、例外 XML イベントをシグナル通知します。次にパーサーは、特殊レジスター XML-CODE に数字のエラー・コードを格納します。詳しくは、*Enterprise COBOL プログラミング・ガイド* を参照してください。

XML-CODE に -1 を設定し、処理プロシージャーで解析を故意に終了した後、通常の XML イベントからパーサーに戻る場合も、例外条件が発生します。この場合、パーサーは、EXCEPTION XML イベントをシグナル通知しません。

ON EXCEPTION 句が指定されている場合、パーサーは制御を命令ステートメント 1 に移します。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ます。この場合、制御は、XML PARSE ステートメントの終わりに移ります。

パーサーに制御が戻る前に、XML 処理プロシージャーで例外 XML 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かった場合は、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

この明示的範囲終了符号は、XML PARSE ステートメントの範囲を区切るために使用されます。END-XML を使用すると、条件付き 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させることができます。END-XML は、ON EXCEPTION 句または NOT ON EXCEPTION 句 を指定しない XML PARSE ステートメントと共に使用することもできます。

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、以下の処理時で制御を*プロシージャー名-1* に渡します。

- 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が移動します。

- XML 文書全体が解析され、END-OF-DOCUMENT イベントで終了した場合。
- パーサーで例外が検出され、パーサーに制御が戻る前に処理プロシージャーで特殊レジスター XML-CODE がゼロにリセットされない場合。
- パーサーに制御が戻る前に、XML-CODE に -1 を設定して、解析を故意に終了した場合。

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、パーサーまたは処理プロシージャーによって設定された最新の値が格納されます。

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、XML-EVENT、および XML-TEXT または XML-NTEXT 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。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。EXCEPTION イベントの場合は、*Enterprise COBOL プログラミング・ガイド* に示された XML 例外コードの 1 つが XML-CODE に格納されます。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。XML-TEXT または XML-NTEXT には、イベントに対応する文書の一部が格納されます。詳しくは、22ページの『XML-EVENT』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、11ページの『特殊レジスター』を参照してください。

どのような XML イベントの場合でも、処理プロシージャーがパーサーに制御を戻したときに XML-CODE がゼロでなければ、パーサーはそれ以上 EXCEPTION イベントを発生することなく終了します。XML-CODE に -1 を設定してから、EXCEPTION 以外のイベントで処理プロシージャーからパーサーに戻ると、パーサーは、ユーザー起動の例外条件により、

強制的に終了されます。EXCEPTION イベントによっては、処理プロシージャーで XML-CODE にゼロを設定し、パーサーを強制的に続行できます。ただし、その後の結果は予測できません。XML-CODE がゼロの場合は、XML 文書が完全に解析されるまで、または処理対象外の例外条件が発生するまで、解析が続行されます。

EXCEPTION イベント、および例外処理の詳細については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

処理プロシージャー

処理プロシージャーをコーディングする場合は、以下の点に注意してください。

- XML 処理プロシージャーに EXIT PROGRAM ステートメントまたは GOBACK ステートメントを含めることはできません。
- 処理プロシージャーで ALTER、GO TO、および PERFORM ステートメントを使用すると、処理プロシージャーの外側のプロシージャー名に制御を移すことができます。ただし、GO TO ステートメントまたは PERFORM ステートメントの実行後は、制御を処理プロシージャーに戻す必要があります。
- 処理プロシージャーには、CALL ステートメントまたは INVOKE ステートメントを含めることができます。ターゲット・プログラムまたはメソッドには、XML PARSE ステートメントを含めることができます。

XML PARSE ステートメントおよび処理プロシージャーの使用方法については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

XML 文書のコード化文字セット

XML PARSE では、国別データ項目と英数字データ項目の XML 文書がサポートされます。国別データ項目の文書をエンコードする場合は、Unicode UTF-16、CCSID 01200 を使用します。英数字データ項目の文書をエンコードする場合は、『サポートされる EBCDIC コード・ページ』に示す明示的にサポートされた単一バイト EBCDIC コード・ページの 1 つ、または 437ページの『サポートされる ASCII コード・ページ』に示す ASCII コード・ページの 1 つを使用します。

文書のエンコードの指定、およびパーサーによるエンコードの指定については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

サポートされる EBCDIC コード・ページ

表 54 (1/2). XML 文書でサポートされる EBCDIC コード・ページ

CCSID	説明
01047	Latin 1 / オープン・システム
01140, 00037	米国、カナダなど。ユーロ国別拡張コード・ページ (ECECP)、国別拡張コード・ページ
01141, 00273	オーストリア、ドイツ ECECP、CECP
01142, 00277	デンマーク、ノルウェー ECECP、CECP
01143, 00278	フィンランド、スウェーデン ECECP、CECP
01144, 00280	イタリア ECECP、CECP

表 54 (2/2). XML 文書でサポートされる EBCDIC コード・ページ

CCSID	説明
01145, 00284	スペイン、ラテンアメリカ (スペイン語) ECECP、CECP
01146, 00285	英国 ECECP、CECP
01147, 00297	フランス ECECP、CECP
01148, 00500	国際 ECECP、CECP
01149, 00871	アイスランド ECECP、CECP

サポートされる ASCII コード・ページ

表 55. XML 文書でサポートされる ASCII コード・ページ

CCSID	説明
00813	ISO 8859-7　ギリシャ語 / ラテン語
00819	ISO 8859-1　Latin 1 / オープン・システム
00920	ISO 8859-9　Latin 5 (ECMA-128、トルコ TS-5881)

ASCII XML 文書を解析する場合、特殊レジスター XML-TEXT の処理プロシージャーに渡される文書のフラグメントは、ASCII でエンコードされます。移動や比較などの Enterprise COBOL 演算では、EBCDIC エンコード、または国別文字を使用しないと、正しく演算を行うことができません。したがって、そうした演算を使用する場合は、文書のフラグメントを事前に変換しておく必要があります。文書のフラグメントを変換するには、まず、NATIONAL-OF 組み込み関数を使用して、ASCII コード・ページの XML 文書を国別文字に変換します。次に、必要があれば、DISPLAY-OF 組み込み関数を使用して、結果を国別文字から EBCDIC に変換します。これを行う場合は、明示的な国別データ項目、たとえば、MOVE FUNCTION DISPLAY-OF (FUNCTION NATIONAL-OF(XML-TEXT ASCII - CCSID) EBCDIC - CCSID) TO EBCDIC - TEXT は必要ありません。

ASCII XML フラグメントは、変換を行わなくても、確実にファイルやデータベースに書き込むことができます。

その他のコード・ページ

その他のコード・ページでエンコードされた XML 文書を解析するには、NATIONAL-OF 関数を使用して、XML 文書を国別文字に変換します。特殊レジスター XML-NTEXT の処理プロシージャーに渡される文書テキストの個々の部分は、必要があれば、DISPLAY-OF 関数を使用して、元のコード・ページに変換できます。

組み込み関数

組み込み関数 441
関数の指定 441
関数定義 448
ACOS 452
ANNUITY 453
ASIN 454
ATAN 455
CHAR 456
COS 457
CURRENT-DATE 458
DATE-OF-INTEGER 459
DATE-TO-YYYYMMDD 460
DATEVAL 461
DAY-OF-INTEGER 463
DAY-TO-YYYYDDD 464
| DISPLAY-OF 465
FACTORIAL 467
INTEGER 468
INTEGER-OF-DATE 469
INTEGER-OF-DAY 470
INTEGER-PART 471
LENGTH 472
LOG 473
LOG10 474
LOWER-CASE 475
MAX 476
MEAN 477
MEDIAN 478
MIDRANGE 479
MIN 480
MOD 481
| NATIONAL-OF 482
NUMVAL 483
NUMVAL-C 484
ORD 486
ORD-MAX 487
ORD-MIN 488
PRESENT-VALUE 489
RANDOM 490
RANGE 491
REM 492
REVERSE 493
SIN 494
SQRT 495
STANDARD-DEVIATION 496

© Copyright IBM Corp. 1991, 2001

SUM 497
TAN 498
UNDATE 499
UPPER-CASE 500
VARIANCE 501
WHEN-COMPILED 502
YEAR-TO-YYYY 503
YEARWINDOW 504

組み込み関数

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入手する必要がある値を使用する場合があります。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。

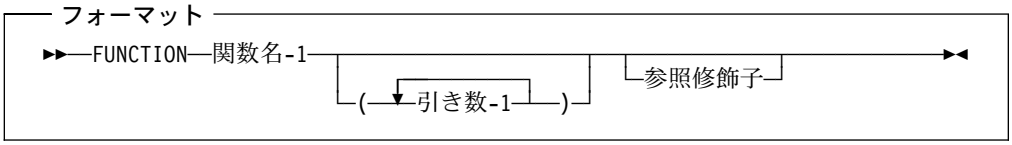
関数は実行されるサービスのタイプに応じて、算術、統計、日時、財務、文字処理、一般という 6 種類に類別できます。

関数は、その名前を必要な引き数と共に指定することによって、手続き部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、英数字値、国別値、数値、または整数値を戻します。関数は、受け取り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

関数 ID の一般的なフォーマットは次のとおりです。



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。

引き数-1
引き数-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。
引き数-1 は、UNDATE 組み込み関数の場合を除いて、ウィンドウ化日付フィールドであってはなりません。

参照修飾子
カテゴリーが英数字または国別である関数についてのみ指定できます。

英数字関数および数字関数について、組み込み関数呼び出しの例を次に示します。

英数字関数の参照で、次のようなステートメント

```
MOVE FUNCTION UPPER-CASE("hello") TO DATA-NAME.
```

があるとします。このソース・ステートメントは、引き数内にあるそれぞれの小文字に対応する大文字に置き換えます。結果として、HELLO が DATA-NAME に移動されます。

数字関数の参照で、次のようなステートメント

```
COMPUTE NUM-ITEM = FUNCTION SUM(A B C)
```

があるとします。これは、値として A、B、C を加算し、結果を NUM-ITEM に置き換えます。

© Copyright IBM Corp. 1991, 2001

手続き部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き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。

- 英数字および国別関数の場合には、戻される値のサイズ。
- 数字関数や整数関数の場合には、戻される値の符号、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。

いくつかの関数の場合、そのクラスと特性は、その関数への引き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、引き数の属性によって影響を受ける場合があります。

関数のタイプ

COBOL の関数には以下のタイプがあります。

- 英数字
- 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。戻される値は、暗黙のうちに USAGE DISPLAY を持ち、標準データ・フォーマットの文字からなります。戻される値の文字位置の数は、関数定義によって決定されます。

国別関数は、国別のクラスとカテゴリーに属します。戻される値は、暗黙のうちに USAGE NATIONAL を持ち、国別文字 (CCSID 01200) で表現されます。戻される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。戻される値は、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、 *Enterprise COBOL プログラミング・ガイド* 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。戻される値は、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。戻される値の桁数は、関数定義によって決定されます。詳しくは、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

使用の規則

英数字関数

英数字関数は、ID を使用することができ、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、次の例外があります。

- 任意のステートメントの受け取り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、および許容値など) 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き数がこれらの特性を持たない場合。

英数字関数の参照変更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。

英数字引き数は使用できる任意の関数の引き数として、英数字関数を使用することができます。

国別関数

国別関数は、国別データ項目を使用でき、しかも一般フォーマットに関連する規則が関数への参照を特に禁止していない一般フォーマットの中であれば、どこでも使用することができます。ただし、次の例外があります。

- 任意のステートメントの受け取り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 (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き数がそれらの特性を持たない場合。

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。

国別引き数は使用できる任意の関数の引き数として、国別関数を使用することができます。

数字関数

数字関数は、算術式が指定できる個所でのみ使用できます。

数字引き数を使用できる関数の引き数として、数字関数を参照することができます。

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、引き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。

整数引き数を使用でき関数の引き数として、整数関数を参照することができます。

特殊な使用に関する注意

CALL ステートメントの ID-2 は、関数 ID である必要はありません。

COPY ステートメントでは、REPLACING 句の中であらゆる種類の関数 ID を使用することができます。

引き数

関数によって戻される値は、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き数によって決定される場合があります。関数の中には引き数が不要なものもあれば、固定数の引き数が必要なもの、さらには可変数の引き数でよいものもあります。

引き数は、次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

ある関数への引き数は、その同じ関数が別の評価を行う場合を含め、何らかの関数、あるいは関数を含む式にすることができます。この場合、引き数となるそれらの関数の評価結果は、この引き数のカテゴリー要件を満たす必要があります。

関数の個別の引き数指定については、448ページの『関数定義』を参照してください。

引き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。引き数の内容は、関数の値を決定するために使用されます。引き数の長さは、関数の値を決定するために使用されることがあります。
- **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。引き数の内容は、関数の値を決定するために使用されます。引き数の長さは、関数の値を決定するために使用されることがあります。
- **DBCS**。DBCS または DBCS リテラルのクラスに属するデータ項目。引き数の内容は、関数の値を決定するために使用されます。引き数の長さは、関数の値を決定するために使用されることがあります (DBCS データ項目または DBCS リテラルは、NATIONAL-OF 関数だけの引き数として使用できます)。
- **国別**。国別または国別リテラルのクラスに属するデータ項目。引き数の内容は、関数の値を決定するために使用されます。引き数の長さは、関数の値を決定するために使用される場合があります。
- **整数**。結果として常に整数値をもたらす算術式。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。
- **数字**。符号も含めその値が、関数の値を決定するために使用される算術式。

関数によっては、その関数自体の有する引き数に関して、制限 (持つことができる値の範囲など) を設けているものがあります。関数に対して引き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、戻される値は未定義になります。

ネストされた関数が引き数として使用される場合、その引き数の評価は、その関数より外側の関数が持つ引き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き数だけが、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。

- 関数の引き数として現れる算術式の計算は、その関数の他の引き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き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、その引き数は、引き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。評価される引き数は、関数 ID、または関数 ID を含む式であることができます。

算術式を引き数として指定し、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、その記号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、数字引き数が使用できる個所ならどこでも使用できます。また、数字引き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。浮動小数点リテラルは、整数引き数が必要とされる個所では使用 *できません*。

外部浮動小数点項目は、数字引き数が使用できる個所ならどこでも使用できます。また、数字引き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。

外部浮動小数点項目は、整数引き数が必要とされる個所や関数 ID の中で英数字クラスの引き数が許可されている個所 (たとえば、LOWER-CASE、REVERSE、UPPER-CASE、NUMVAL、および NUMVAL-C などの関数の中) では使用**できません**。

ALL 添え字

関数が 1 つの引き数を任意の回数を繰り返して使用することができる場合、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って、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 つまたは複数の添え字として ALL の添え字付けを行うことができます。

注: ALL 添え字の評価は、少なくとも 1 つの引き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される値は、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)」としてテーブル引き数を指定する場合、1 つの引き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) の引き数が、「テーブル名(1)」であり、ALL が 1 によって置き換えられます。次の引き数は「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、暗黙の引き数を 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。

たとえば、

```
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、最初の暗黙の引き数は「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。次の引き数は「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き数を作り出します。

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き数が作り

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き数を作り出します。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。

たとえば、

```
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。

たとえば、

```
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、ある引き数に対して指定され、その引き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 文節に関連付けられている場合、その値の範囲は、OCCURS DEPENDING ON 文節のオブジェクトによって決定されます。

たとえば、次のような給与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。

関数定義

449ページの表56 に、各組み込み関数の引き数タイプ、関数タイプ、および戻される値の概要を示します。引き数のタイプと関数のタイプは、次のような略語を使用しています。

A = 英字
D = DBCS
I = 整数
N = 数字
X = 英数字
U = 国別 (汎用文字セット (UCS) および Unicode を表す)
O = 関数定義に指定されているその他のタイプ (ポインター、関数ポインター、プロシージャー・ポインター、またはオブジェクト・リファレンス)

関数のうち、“DP” のマークの付いたものの動作は、DATEPROC または NODATEPROC コンパイラー・オプションのどちら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。

	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) 引き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、表の後に詳しく説明します。

表 56 (1/3). 組み込み関数一覧

関数名	引き数	タイプ	戻される値
ACOS	N1	N	N1 の逆余弦
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
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 (ウィンドウ化西暦年 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 または	I	I1 または X1 に相当する日付フィールド
	X1	X	
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 (ウィンドウ化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)。
DISPLAY-OF	U1 または U1、I2	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 が指定されない場合はコンパイル時に選択されたデフォルトのコード・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。
FACTORIAL	I1	I	I1 の階乗
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

表 56 (2/3). 組み込み関数一覧

関数名	引き数	タイプ	戻される値
INTEGER-PART	N1	I	N1 の整数部分
LENGTH	A1、N1、O1、X1、または U1	I	引き数の長さ (国別文字位置、英数字位置またはバイト数。引き数のタイプによって異なる。)
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	引き数内のすべての文字が小文字に設定される
	U1	U	
MAX	A1...	X	最大引き数の値。関数のタイプは引き数によっ決定されることに注意
	I1...	I	
	N1...	N	
	X1...	X	
	U1...	U	
MEAN	N1...	N	引き数の算術平均
MEDIAN	N1...	N	引き数の中央値
MIDRANGE	N1...	N	引き数の最小値と最大値の平均
MIN	A1... または	X	最小引き数の値。関数のタイプは引き数によって決定されることに注意
	I1... または	I	
	N1... または	N	
	X1... または	X	
	U1...	U	
MOD	I1、I2	I	I1 モジュロ I2
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、国別文字に変換された引き数-1 の文字
	A1、X1、または D1、I2	U	
NUMVAL	X1	N	単純数字ストリングの数値
NUMVAL-C	X1 または X1、X2	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き数の順序位置
ORD-MAX	A1...、N1...、X1...、または U1...	I	最大引き数の順序位置

表 56 (3/3). 組み込み関数一覧

関数名	引き数	タイプ	戻される値
ORD-MIN	A1...、N1...、X1...、または U1...	I	最小引き数の順序位置
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
RANGE	I1...	I	最大引き数の値から最小引き数の値を減算したもの。関数のタイプは引き数によって決定されることに注意
	N1...	N	
REM	N1、N2	N	N1/N2 の剰余
REVERSE	A1 または X1	X	引き数の文字の逆順
	U1	U	
SIN	N1	N	N1 の正弦
SQRT	N1	N	N1 の平方根
STANDARD-DEVIATION	N1...	N	引き数の標準偏差
SUM	I1...	I	引き数の和。関数のタイプは引き数によって決定されることに注意
	N1...	N	
TAN	N1	N	N1 の正接
UNDATE DP	I1 または	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	
UPPER-CASE	A1 または X1	X	引き数内のすべての文字が大文字に設定される
	U1	U	
VARIANCE	N1...	N	引き数の分散
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 (ウィンドウ化西暦年 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。

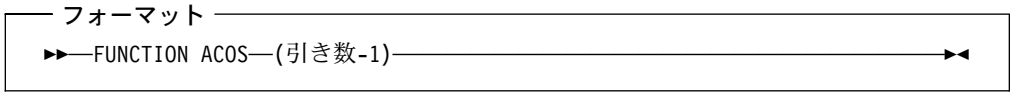
後続のページで、上記の表を要約した各組み込み関数を詳しく定義します。

ACOS

ACOS 関数は、指定された引き数の逆余弦に近似する数値をラジアンで戻します。

関数のタイプは数字です。

フォーマット



引き数-1

この引き数のクラスは数字でなければなりません。引き数-1 の値は、-1 以上 +1 以下でなければなりません。

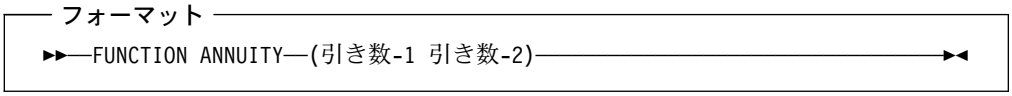
戻される値は引き数の逆余弦の近似値で、0 以上 Pi 以下です。

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。期間の数は引き数-2 によって指定します。金利は引き数-1 によって指定します。たとえば、引き数-1 が 0、引き数-2 が 4 であるとすると、戻される値は比率 1/4 の近似値となります。

関数のタイプは数字です。

フォーマット



引き数-1
この引き数のクラスは数字でなければなりません。整数-1 の値は、0 以上でなければなりません。

引き数-2
これは、正の整数でなければなりません。

引き数-1 が 0 である場合、この関数によって戻される値は、1 / ARGUMENT-2 の近似値となります。

引き数-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。

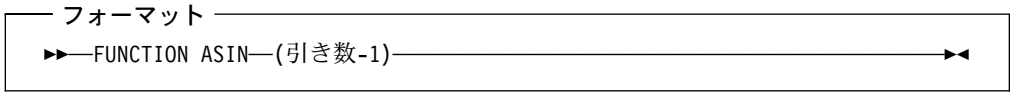
```
ARGUMENT-1 / (1 - (1 + ARGUMENT-1) ** (- ARGUMENT-2))
```

ASIN

ASIN 関数は、指定された引き数の逆正弦に近似する数字をラジアンで戻します。

関数のタイプは数字です。

フォーマット



引き数-1

この引き数のクラスは数字でなければなりません。引き数-1 の値は、-1 以上 +1 以下でなければなりません。

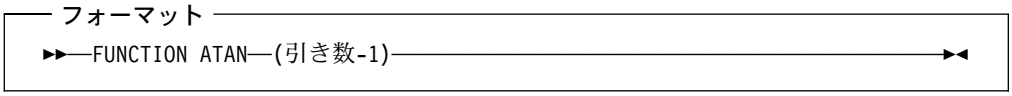
戻される値は、引き数-1 の逆正弦の近似値で、-Pi/2 以上 +Pi/2 以下です。

ATAN

ATAN 関数は、指定された引き数の逆正接に近似する数字をラジアンで戻します。

関数のタイプは数字です。

フォーマット



引き数-1
この引き数のクラスは数字でなければなりません。

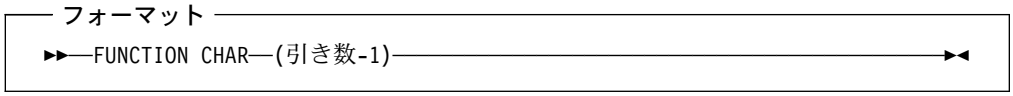
戻される値は、引き数-1 の逆正接の近似値で、-pi/2 以上 +pi/2 以下になります。

CHAR

CHAR 関数は、指定された引き数の値に相当する順序位置にあるプログラム照合シーケンス内の文字である、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



引き数-1
これは整数でなければなりません。この値は、0 以上で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 文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。

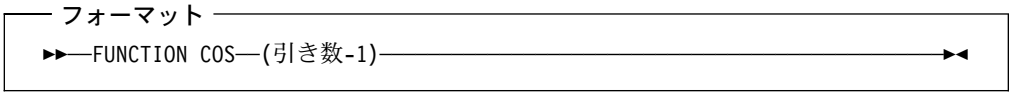
現在のプログラム照合シーケンスが、ALPHABET 文節によって指定されていない場合は、単一バイト EBCDIC 照合シーケンスが使用されます (556ページの付録C、『EBCDIC および ASCII の照合シーケンス』を参照)。

COS

COS 関数は、引き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。

関数のタイプは数字です。

フォーマット



引き数-1
この引き数のクラスは数字でなければなりません。

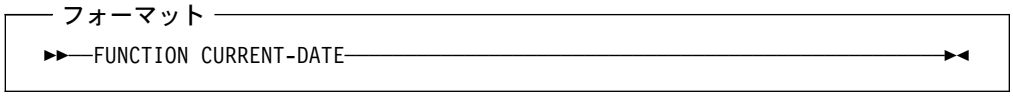
戻される値は引き数の余弦の近似値で、-1 以上 +1 以下の値です。

CURRENT-DATE

CURRENT-DATE 関数は、カレンダー上の日付、時刻、およびこの関数が実行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間差を表す、21 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



戻される値の中の 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 ～ 4	グレゴリオ暦におけるその年を表す 4 桁の数字。
5 ～ 6	月を表す 2 桁の数字で、01 ～ 12 の範囲にある値。
7 ～ 8	日を表す 2 桁の数字で、01 ～ 31 の範囲にある値。
9 ～ 10	午前 0 時からの時間を表す 2 桁の数字で、00 ～ 23 の範囲にある値。
11 ～ 12	分を表す 2 桁の数字で、00 ～ 59 の範囲にある値。
13 ～ 14	秒を表す 2 桁の数字で、00 ～ 59 の範囲にある値。
15 ～ 16	100 分の 1 秒を表す 2 桁の数字で、00 ～ 99 の範囲の値。関数を実行するシステムが、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。この関数を実行するシステムがロケールによる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。
18-19	17 の文字位置が '-' の場合は、時間を表す 00 ～ 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を示す 00 ～ 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。
20-21	前記の時間差に加えるべき分単位の時間を表す 00 ～ 59 の範囲の 2 桁の数字が戻されます。文字位置 17 が '+' か '-' かに応じて、それぞれここに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、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

DATE-OF-INTEGER

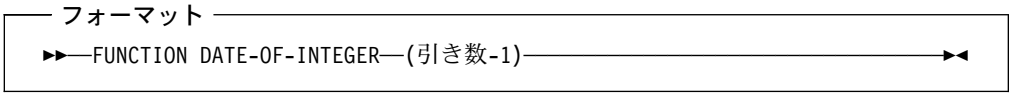
DATE-OF-INTEGER 関数は、グレゴリオ暦の日付を整数で表された日付形式から標準形式の日付 (YYYYMMDD) に変換します。

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・オプションが有効の場合の戻り値は拡張日付フィールドであり、暗黙の DATE FORMAT は YYYYXXXX です。

フォーマット



引き数-1

正の整数で、西暦の 1601 年 1 月 1 日以降の日数を表します。有効な範囲は 1 ～ 3,067,671 で、これは 1601 年 1 月 1 日 ～ 9999 年 12 月 31 日の範囲に対応します。

INTDATE コンパイラー・オプションは、整数日付関数の開始日付に影響します。詳細は、*Enterprise COBOL プログラミング・ガイド* を参照してください。

戻される値は、引き数-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。

戻される値は、YYYYMMDD 形式の整数で、YYYY は西暦の年を、MM はその年の月を、DD はその月の日を表します。

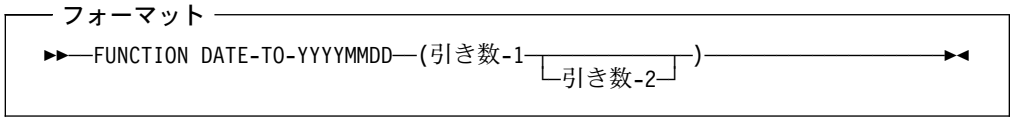
DATE-TO-YYYYMMDD

DATE-TO-YYYYMMDD 関数は、引き数-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。引き数-2 は実行時に年に追加されると、100 年の間隔の終了年を定義するか、または引き数-1 を入れる世紀ウィンドウをどのようにスライドさせるかを定義します。

この関数のタイプは整数です。

DATEPROC コンパイラー・オプションが有効の場合の戻り値は拡張日付フィールドであり、暗黙の DATE FORMAT は YYYYXXXX です。

フォーマット



引き数-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: 値が有効日付であることを確認するための検査は行われません。

引き数-2

これは整数でなければなりません。引き数-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実行されます。

実行時における年と引き数-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。

例

DATE-TO-YYYYMMDD 関数から戻される値の例を以下に示しています。

現在の年	引き数-1 の値	引き数-2 の値	DATE-TO-YYYYMMDD からの戻り値
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、戻される値は引き数-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。

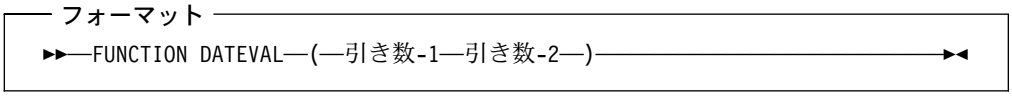
- 算術式の場合、232ページの『日付フィールドでの算術演算』
- 条件式の場合、241ページの『日付フィールド』

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、引き数-1 の値が未変更のまま戻されます。

関数のタイプは、次に示すように引き数-1 のタイプに応じて異なります。

引き数-1 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



引き数-1

以下のいずれかにしなければなりません。

- 引き数-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、引き数-2 で指定された範囲外の値を指定することができます。

引き数-1 の値は、引き数-2 で指定された形式の日付を表します。

引き数-2

164ページの『DATE FORMAT 文節』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化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 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。値の大文字小文字は区別されません。引き数-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	指定する引き数-1 の内容
YY	ウィンドウ化 (2 桁) 年。
YYYY	拡張 (4 桁) 年。
X	1 文字。たとえば、1 学期または四半期 (1 ～ 4) などを表す数字。
XX	2 文字。たとえば、月を表す数字 (01 ～ 12)。
XXX	3 文字。たとえば、1 年のうちの何日目かを表す数字 (001 ～ 366)。
XXXX	4 文字。たとえば、月を表す 2 桁 (01 ～ 12) と月の何日かを表す 2 桁 (01 ～ 31)。

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

DAY-OF-INTEGER

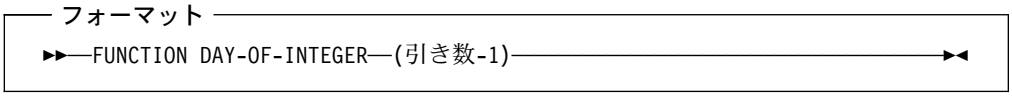
DAY-OF-INTEGER は、グレゴリオ暦の日付を整数で表された日付から年間通算日形式 (YYYYDDD) に変換します。

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・オプションが有効の場合の戻り値は拡張日付フィールドであり、暗黙の DATE FORMAT は YYYYXXX です。

フォーマット



引き数-1

正の整数で、西暦の 1601 年 1 月 1 日以降の日数を表します。有効な範囲は 1 ～ 3,067,671 で、これは 1601 年 1 月 1 日 ～ 9999 年 12 月 31 日の範囲に対応します。

INTDATE コンパイラー・オプションは、整数日付関数の開始日付に影響します。詳細は、*Enterprise COBOL プログラミング・ガイド* を参照してください。

戻される値は、引き数-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。戻される値は、YYYYDDD 形式の整数で、YYYY は西暦の年を、DDD はその年の日を表します。

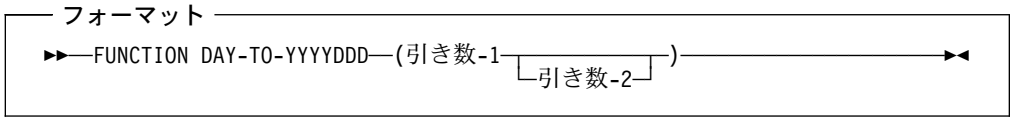
DAY-TO-YYYYDDD

DAY-TO-YYYYDDD 関数は、引き数-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。引き数-2 は実行時に年に追加されると、100 年の間隔の終了年を定義するか、または引き数-1 を入れる世紀ウィンドウをどのようにスライドさせるかを定義します。

この関数のタイプは整数です。

DATEPROC コンパイラー・オプションが有効の場合の戻り値は拡張日付フィールドであり、暗黙の DATE FORMAT は YYYYXXX です。

フォーマット



引き数-1

0 または 99,367 未満の正の整数でなければなりません。

注: 値が有効日付であることを確認するための検査は行われません。

引き数-2

これは整数でなければなりません。引き数-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実行されます。

実行時における年と引き数-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。

例

DAY-TO-YYYYDDD 関数から戻される値の例を以下に示しています。

現在の年	引き数-1 の値	引き数-2 の値	DAY-TO-YYYYDDD からの戻り値
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引き数-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

引き数-1
この引き数のクラスは国別でなければなりません。引き数-1 は変換に使用するソース・ストリングを示します。

引き数-2
これは整数でなければなりません。引き数-2 は変換に使用するターゲット・コード・ページを示します。引き数-2 は、EBCDIC、ASCII、UTF-8、または EUC コード・ページを示す有効な CCSID (Coded Character Set Identifier) 番号でなければなりません。EBCDIC または ASCII の CCSID は、SBCS、DBCS、または MBCS (マルチバイト文字セット) であるコード・ページを示すことができます。

注: UTF-8 を表す CCSID は 01208 です。

引き数-2 を省略した場合、ターゲット・コード・ページは、ソース・コードがコンパイルされたときに CODEPAGE コンパイラー・オプションに対して有効なコード・ページになります。

戻り値は、ターゲット・コード・ページの表現に変換された引き数-1 の文字で構成される英数字ストリングです。ソース文字をターゲット・コード・ページの文字に変換できないときは、単一バイト EBCDIC ターゲット・コード・ページの場合は置換文字 X'3F'、DBCS コード・ページの場合は置換文字 X'FFFE' によってソース文字が置き換えられます。例外条件は発生しません。

戻される値の長さは、引き数-1 の内容およびターゲット・コード・ページの特性によって異なります。

例外: 変換が失敗した場合、重大なランタイム・エラーが発生します。OS/390 Unicode 変換サービス・プログラムがインストールされており、CCSID 01200 からターゲット・コード・ページへの変換用テーブルを含むように構成されているかどうかを検証してください。変換をサポートするためのインストール要件については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

使用上の注意

1. ターゲット・コード・ページが SBCS/DBCS EBCDIC の混合しているコード・ページの場合は、戻される値にシフトアウトおよびシフトイン制御文字で区切られた DBCS サブストリングが含まれる場合があります。
2. DISPLAY-OF 関数は、引き数-2 を指定すると、CODEPAGE コンパライラー・オプションに指定したコード・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。その後そのデータに対する COBOL 操作として、データが

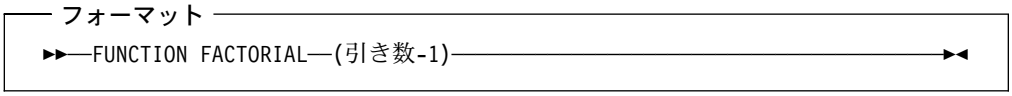
CODEPAGE コンパイラー・オプションで指定された EBCDIC コード・ページで表現されていることを前提とした暗黙の変換を実行できます。単一のプログラム内で複数のコード・ページを使用して表現されたデータを処理する例およびプログラミング手法については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

FACTORIAL

FACTORIAL 関数は、指定された引き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



引き数-1

ARITH(COMPAT) コンパイラー・オプションが有効である場合は、引き数-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、引き数-1 は、0 以上 29 以下の整数でなければなりません。

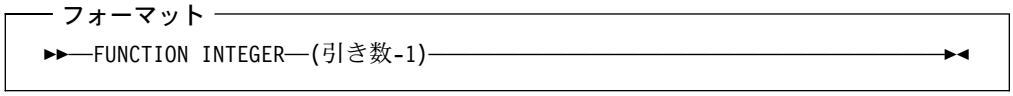
引き数-1 の値が 0 の場合、値として 1 が戻されます。0 以外の場合は、その階乗が戻されます。

INTEGER

INTEGER 関数は、指定された引き数と等しいか、それより小さ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



引き数-1
この引き数のクラスは数字でなければなりません。

戻される値は、引き数-1 の値以下の最大の整数です。たとえば、

```
FUNCTION INTEGER (2.5)
```

は値として 2 を戻します。

```
FUNCTION INTEGER (-2.5)
```

は値として -3 を戻します。

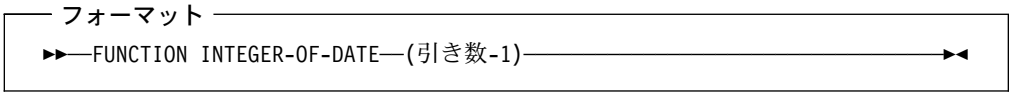
INTEGER-OF-DATE

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 ～ 3,067,671 の範囲の 7 桁の整数です。

フォーマット



引き数-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。

- YYYY は、西暦の年を表します。この値は、1,600 より大きく、9,999 以下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。

戻される値は整数で、引き数-1 によって表された日付を西暦で 1601 年 1 月 1 日以降の日数に換算したものです。

INTDATE コンパイラー・オプションは、整数日付関数の開始日付に影響します。詳細は、*Enterprise COBOL プログラミング・ガイド* を参照してください。

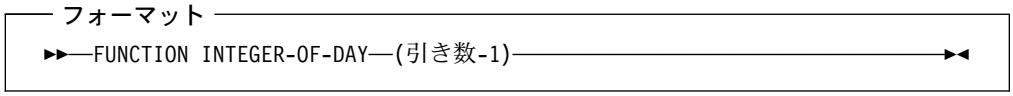
INTEGER-OF-DAY

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



引き数-1

これは YYYYDDD 形式の整数で、この値は、(YYYY * 1000) + DDD という計算から得られます。

- YYYY は、西暦の年を表します。この値は、1,600 より大きく、9,999 以下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。

INTDATE コンパイラー・オプションは、整数日付関数の開始日付に影響します。詳細は、*Enterprise COBOL プログラミング・ガイド* を参照してください。

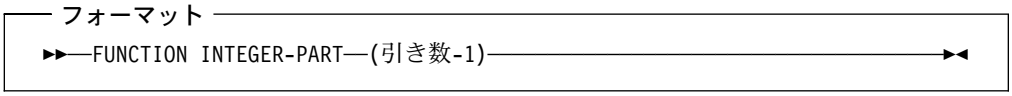
戻される値は整数で、引き数-1 によって表された日付を西暦で 1601 年 1 月 1 日以降の日数に換算したものです。

INTEGER-PART

INTEGER-PART 関数は、指定された引き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



引き数-1
この引き数のクラスは数字でなければなりません。

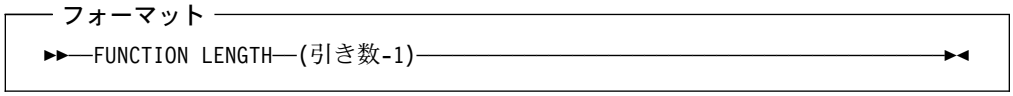
引き数-1 の値が 0 であれば、戻される値は 0 です。引き数-1 の値が正数の場合、戻される値は引き数-1 の値以下の最大の整数となります。引き数-1 の値が負数の場合、戻される値は引き数-1 の値以上の最小の整数となります。

LENGTH

LENGTH 関数は、引き数の長さ (国別クラスの引き数については国別文字位置、その他すべての引き数については英数字位置またはバイト数) 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



引き数-1
以下のようになります。

- 任意のクラスまたはカテゴリーに属する英数字または国別リテラルやデータ項目 (DBCS を除く)。
- 次の USAGE を使用して記述されているデータ項目。POINTER、PROCEDURE-POINTER、FUNCTION-POINTER、または OBJECT REFERENCE。
- 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

戻される値は、以下のようにして決定された 9 桁の整数です。

- 引き数-1 が英数字リテラルまたは英数字データ項目である場合、戻される値は引き数の英数字位置の数と等しくなります。

  引き数-1 がヌル終了英数字リテラルである場合、戻される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。

  注: 英数字データ項目の長さまたは単一バイト文字と 2 バイト文字が混在するリテラルの長さは、各バイトが単一バイト文字であるものとして数えられます。

- 引き数-1 がグループ・データ項目である場合、戻される値はグループの内容に関係なく、引き数-1 の英数字位置分の長さと等しくなります。引き数-1 に従属するいずれかのデータ項目が OCCURS 文節の DEPENDING 句を使用して記述されている場合、引き数-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引き数-1 の長さが決定されます。この評価は、送り出しデータ項目に関する OCCURS 文節における規則に従って実施されます。詳細については、OCCURS 文節および USAGE 文節の説明を参照してください。

  戻される値は、暗黙の FILLER 位置 (ある場合) を含みます。

- 引き数-1 が国別リテラルまたは国別データ項目である場合、戻される値は引き数-1 の国別文字位置分の長さと等しくなります。

  たとえば、引き数-1 が PIC N(3) として定義されている場合、戻される値は 3 です。ただし、引き数のストレージ・サイズは 6 バイトになります。

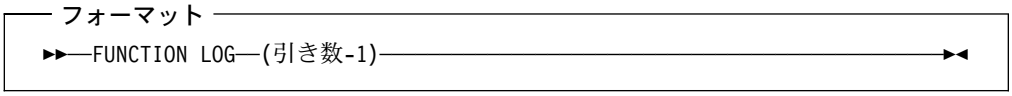
- それ以外の場合は、戻される値は引き数-1 が占めるストレージのバイト数になります。

LOG

LOG 関数は、指定された引き数の ***e*** (natural log) を底とする対数に近似する数字を戻します。

関数のタイプは数字です。

フォーマット



引き数-1

この引き数のクラスは数字でなければなりません。引き数-1 の値は 0 よりも大きくなければなりません。

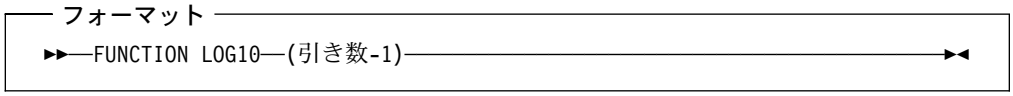
戻される値は ***e*** を底とする引き数-1 の対数の近似値です。

LOG10

LOG10 関数は、指定された引き数の 10 を底とする対数に近似する数字を戻します。

関数のタイプは数字です。

フォーマット



引き数-1

この引き数のクラスは数字でなければなりません。引き数-1 の値は 0 よりも大きくなければなりません。

戻される値は 10 を底とする引き数-1 の対数の近似値です。

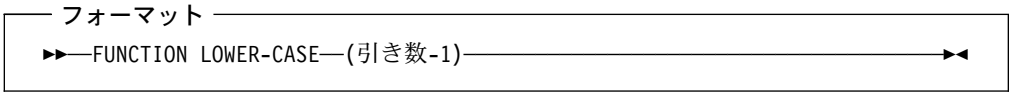
LOWER-CASE

LOWER-CASE 関数は、引き数のそれぞれの大文字を対応する小文字に置き換えて、その引き数と同じ長さの文字ストリングを戻します。

関数のタイプは、次に示すように引き数のタイプに応じて異なります。

引き数のタイプ	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



引き数-1

この引き数のクラスは英字、英数字、または国別でなければならず、少なくとも 1 文字位置の長さを持た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、引き数-1 と同じ文字ストリングが戻されます。

引き数-1 が英字または英数字クラスに属する場合は、'A' ～ 'Z' までの大文字は対応する小文字 'a' ～ 'z' に置き換えられます。ここで 'A' ～ 'Z' の範囲と 'a' ～ 'z' の範囲は、有効なコード・ページにかかわらず 556ページの表60 に示されているようになります。

引き数-1 が国別クラスに属する場合は、各大文字は Unicode データベース UnicodeData.txt (http:/www.unicode.org/ の Unicode Consortium から入手可能) の仕様に基づいて、対応する小文字に置き換えられます。

戻される文字ストリングは、引き数-1 と同じ長さを持ちます。

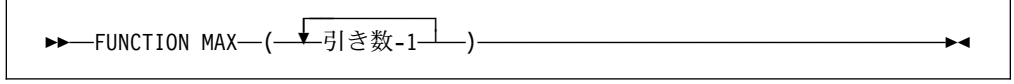
MAX

MAX 関数は、最大値を含む引き数の内容を戻します。

関数のタイプは、次に示すように引き数のタイプに応じて異なります。

引き数のタイプ	関数のタイプ
英字	英数字
英数字	英数字
国別	国別
すべての引き数が整数	整数
数字 (一部の引き数が整数)	数字

フォーマット



引き数-1
この引き数のクラスは英字、英数字、国別、または数値でなければなりません。

すべての引き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き数の組み合わせが許可されています。

戻される値は、最大値を持っている引き数-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。詳細については、235ページの『条件式』を参照してください。

複数の引き数-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き数-1 が戻されます。

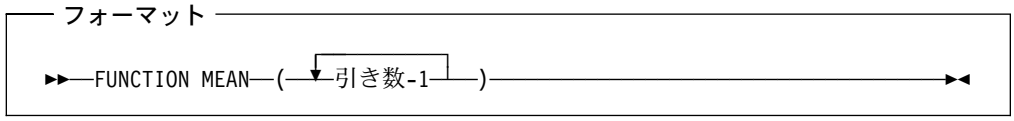
関数のタイプが英数字または国別である場合、戻される値のサイズは選択した引き数-1 のサイズと同じです。

MEAN

MEAN 関数は、その引き数の算術平均に近似する数字を戻します。

関数のタイプは数字です。

フォーマット



引き数-1
この引き数のクラスは数字でなければなりません。

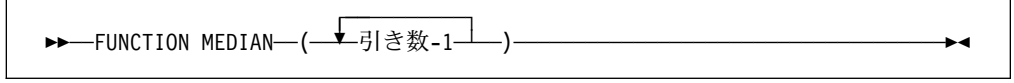
戻される値は一連の引き数-1 の算術平均値です。戻される値は一連の引き数-1 の合計を引き数-1 として参照したオカレンス数で除算したものとして定義されます。

MEDIAN

MEDIAN 関数は、複数の引き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き数の内容を戻します。

関数のタイプは数字です。

フォーマット



引き数-1
この引き数のクラスは数字でなければなりません。

戻される値は、すべての引き数-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き数-1 の内容です。

引き数-1 によって参照されるオカレンス数が奇数である場合、戻される値は次のようなものになります。つまり、引き数-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、戻される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、戻される値よりも小さいかまたはそれに等しくなります。引き数-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、戻される値は中央にある 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。

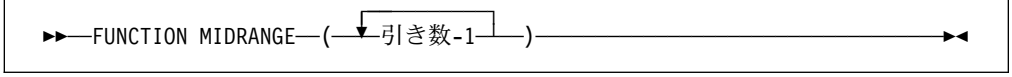
ソート順で引き数値を並べ変えるために使用される比較は、単純条件に関する規則に従って行われます。詳細については、235ページの『条件式』を参照してください。

MIDRANGE

MIDRANGE 関数は、最小の引き数の値と最大の引き数の値の算術平均に近似した数字を戻します。

関数のタイプは数字です。

フォーマット



引き数-1
この引き数のクラスは数字でなければなりません。

戻される値は、最大の引き数-1 の値と最小の引き数-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。詳細については、235ページの『条件式』を参照してください。

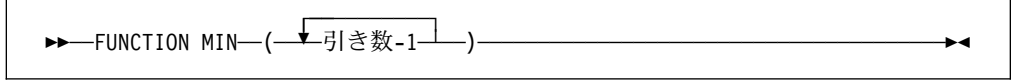
MIN

MIN 関数は、最小値を含む引き数の内容を戻します。

関数のタイプは、次に示すように引き数のタイプに応じて異なります。

引き数のタイプ	関数のタイプ
英字	英数字
英数字	英数字
国別	国別
すべての引き数が整数	整数
数字 (一部の引き数が整数)	数字

フォーマット



引き数-1
この引き数のクラスは英字、英数字、国別、または数値でなければなりません。

すべての引き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き数の組み合わせが許可されています。

戻される値は、最小値を持っている引き数-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。詳細については、235ページの『条件式』を参照してください。

複数の引き数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き数が戻されます。

関数のタイプが英数字または国別である場合、戻される値のサイズは選択した引き数 -1 のサイズになります。

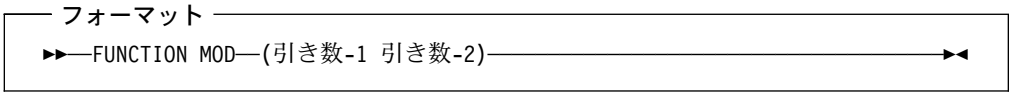
MOD

MOD 関数は、引き数-2 をモジュロとする引き数-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、引き数-1 と引き数-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



引き数-1
これは整数でなければなりません。

引き数-2
これは整数でなければなりません。0 にすることはできません。

戻される値は、引き数-2 をモジュロとする引き数-1 です。戻される値は、次のように定義されます。

引き数-1 - (引き数-2 * FUNCTION INTEGER (引き数-1 / 引き数-2))

次の表は、引き数-1 と引き数-2 のいくつかの値について、予期される結果を示したものです。

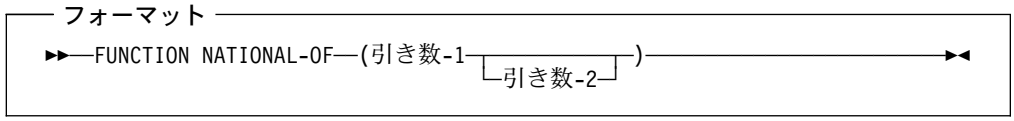
引き数-1	引き数-2	戻される値
11	5	1
-11	5	4
11	-5	-4
-11	-5	-1

NATIONAL-OF

NATIONAL-OF 関数は、引き数-1 の文字の UTF-16 表現から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



引き数-1

英字、英数字、または DBCS のクラスに属さなければなりません。引き数-1 は変換に使用するソース・ストリングを示します。

引き数-2

これは整数でなければなりません。引き数-2 は変換に使用するソース・コード・ページを示します。引き数-2 は、EBCDIC、ASCII、UTF-8、または EUC コード・ページを示す有効な CCSID (Coded Character Set Identifier) 番号でなければなりません。EBCDIC または ASCII の CCSID は、SBCS、DBCS、または MBCS であるコード・ページを示すことができます。

引き数-2 を省略した場合、ソース・コード・ページは、ソース・コードがコンパイルされたときに CODEPAGE コンパイラー・オプションに対して有効なコード・ページになります。

注: UTF-8 を表す CCSID は 01208 です。

戻り値は、国別文字の表現 (CCSID 01200) に変換された引き数-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 X'FFFD' に変換されます。例外条件は発生しません。

戻される値の長さは、引き数-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。

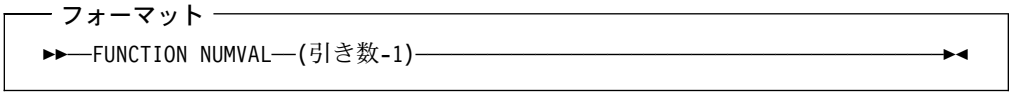
例外: 変換が失敗した場合、重大なランタイム・エラーが発生します。OS/390 Unicode 変換サービス・プログラムがインストールされており、ソース・コード・ページから CCSID 01200 への変換用テーブルを含むように構成されているかどうか検証してください。変換をサポートするためのインストール要件については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

NUMVAL

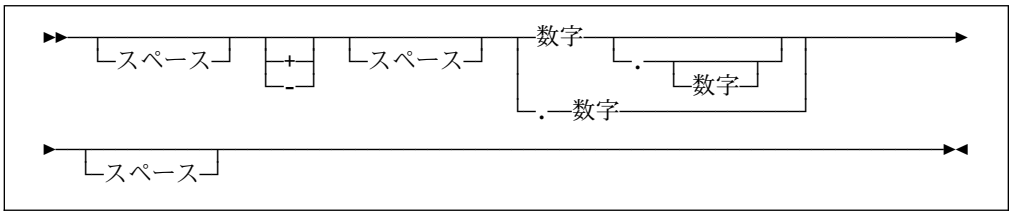
NUMVAL 関数は、引き数の中に指定されている英数字文字ストリングによって表される数字を戻します。この関数は、ストリングの先頭部分と後尾部分にブランクがあるときは、それを取り除き、算術式の中で使用できる数字を作り出します。

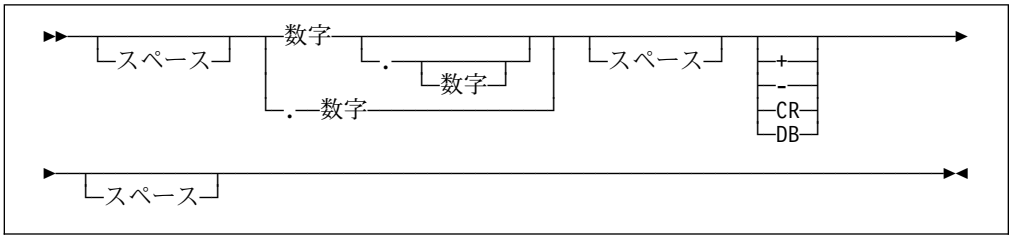
関数のタイプは数字です。

フォーマット



引き数-1
英数字リテラルか、以下のいずれかのフォーマットの内容を持つ英数字データ項目でなければなりません。





スペース
1 つまたは複数のスペースで構成されるストリング。

数字
1 つまたは複数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、数字の合計数は 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、数字の合計数は 31 を超えてはなりません。

DECIMAL-POINT IS COMMA 文節が、SPECIAL-NAMES 段落の中に指定されている場合には、引き数-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。

戻される値は、引き数-1 によって表される数字の近似値です。

NUMVAL-C

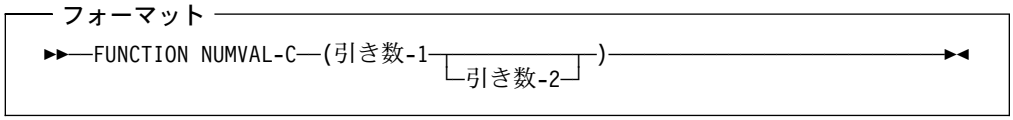
NUMVAL-C 関数は、引き数-1 として指定されている英数字文字ストリングによって表される数値を戻します。引き数-2 によって指定されたオプショナル通貨符号、および小数点に先立つオプショナル・コンマがある場合には、それらは取り除かれ、算術式の中で使用できる数字が作られます。

関数のタイプは数字です。

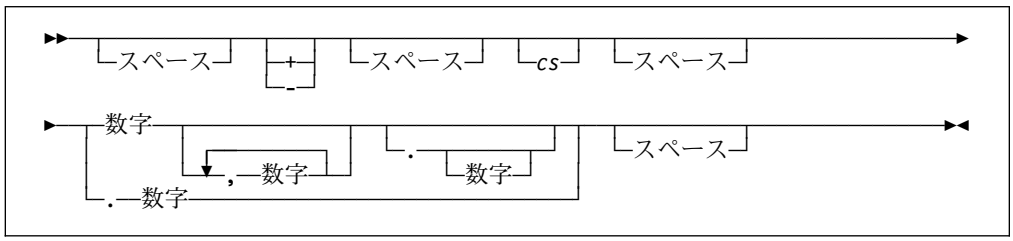
NUMVAL-C 関数は、以下のいずれかの条件が当てはまる場合には使用できません。

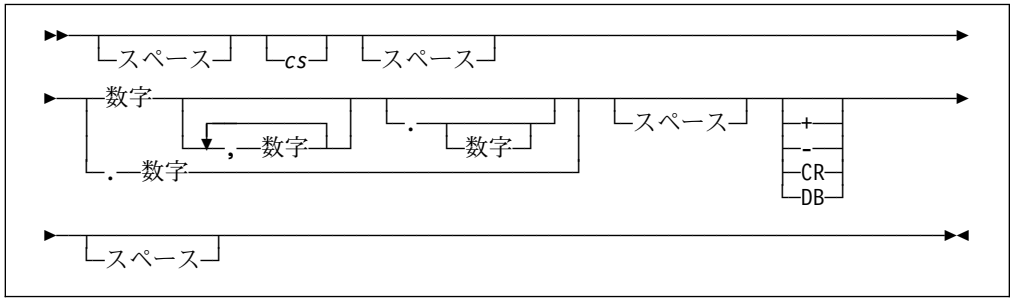
- プログラムの環境部の SPECIAL-NAMES 段落中に複数の CURRENCY SIGN 文節が含まれている。
- CURRENCY SIGN 文節中のリテラル-6 が小文字になっている。
- CURRENCY SIGN 文節中に PICTURE SYMBOL 段落が指定されている。

フォーマット



引き数-1
英数字リテラルか、以下のいずれかのフォーマットの内容を持つ英数字データ項目でなければなりません。





スペース
1 つまたは複数のスペースで構成されるストリング。

cs 引き数-2 によって指定された 1 つまたは複数の文字で構成されるストリング。cs によって指定された文字は、1 回に限り引き数-1 の中で行うことができます。

数字

1 つまたは複数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、数字の合計数は 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、数字の合計数は 31 を超えてはなりません。

DECIMAL-POINT IS COMMA 文節が、SPECIAL-NAMES 段落の中で指定されている場合には、引き数-1 の中のコンマと小数点は、その機能を交換します。

引き数-2

これを指定する場合には、英数字リテラルまたは英数字データ項目でなければならず、以下のような規則に従う必要があります。

- 引き数-2 は、0 ～ 9 の数字、先頭や末尾のスペース、または特殊文字 + や - のいずれをも含むことはできません。
- 引き数-2 は、基本データ項目やグループ・データ項目で有効な長さの場合、長さ 0 を含めどのような長さでも構いません。
- 引き数-2 の指定は、大文字と小文字を区別します。たとえば、引き数-2 に 'Dm' と指定した場合、それは 'DM' とも 'dm' とも、または 'dM' とも一致しません。

引き数-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。

戻される値は、引き数-1 によって表される数字の近似値です。

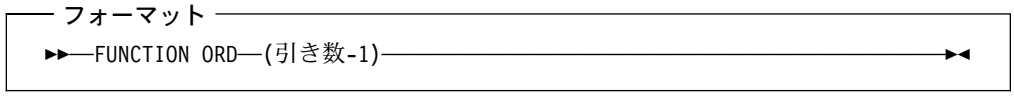
ORD

ORD 関数は、プログラム照合シーケンスの中で、この引き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



引き数-1
この引き数は長さが 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。

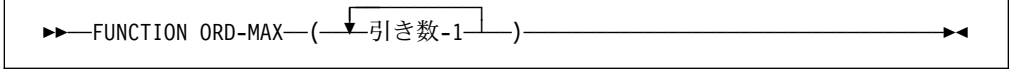
戻される値は、プログラムの照合シーケンス内の引き数-1 の順序位置です。照合シーケンスに応じて、1 ～ 256 の値になります。

ORD-MAX

ORD-MAX 関数は、最大値を持つ引き数の、引き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



引き数-1
この引き数のクラスは英字、英数字、国別、または数値でなければなりません。

すべての引き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き数の組み合わせが許可されています。

戻される値は、一連の引き数-1 の中で最大値を持つ引き数-1 の位置に対応する序数です。

最大値を持つ引き数-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。詳細については、235ページの『条件式』を参照してください。

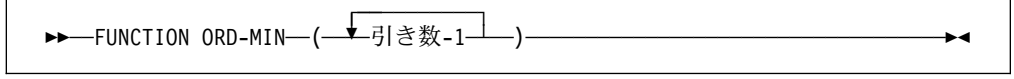
複数の引き数-1 が同一の最大値を持つ場合、その値を持つ引き数-1 のうち左端の位置に対応した引き数-1 の序数が戻されます。

ORD-MIN

ORD-MIN 関数は、最小値を含む引き数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



引き数-1
この引き数のクラスは英字、英数字、国別、または数値でなければなりません。

すべての引き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き数の組み合わせが許可されています。

戻される値は、一連の引き数-1 の中で最小値を持つ引き数-1 の位置に対応する序数です。

最小値を持つ引き数-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。詳細については、235ページの『条件式』を参照してください。

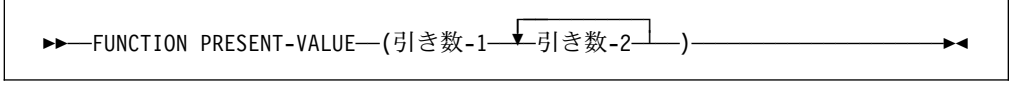
複数の引き数-1 が同一の最小値を持つ場合、その値を持つ引き数-1 のうち左端の位置を占める引き数-1 に対応した序数が戻されます。

PRESENT-VALUE

PRESENT-VALUE 関数は、引き数-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き数-1 によって指定された割引率で戻します。

関数のタイプは数字です。

フォーマット



引き数-1
この引き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。

引き数-2
この引き数のクラスは数字でなければなりません。

戻される値は、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。

引き数-2 / (1 + 引き数-1) ** n

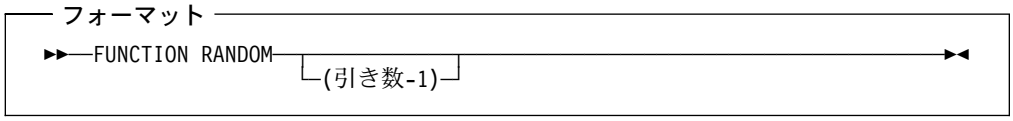
引き数-2 のオカレンスごとに 1 つの期間があります。指数 n は、引き数-2 の一連の指定において各期ごとに 1 から 1 ずつ増加されます。

RANDOM

RANDOM 関数は、一様分布から疑似乱数である数値を戻します。

関数のタイプは数字です。

フォーマット



引き数-1

引き数-1 を指定する場合、0 または正の整数でなければなりません。その値は、PIC 9(18) 固定長項目の中に指定できる最大値である (10**18)-1 以下である必要があります。ただし 0 ～ 2,147,483,645 までの範囲のものだけが、疑似乱数の数列を生み出します。

2 回目以降の参照で引き数-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。

ラン・ユニット内においてこの関数への最初の参照が引き数-1 を指定していない場合、初期値は 0 です。

引き数-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。

戻される値は必ず 0 と 1 の間の数字です。

ある指定された初期値に関しては、疑似乱数のオカレンス数字列は常に同じになります。

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。

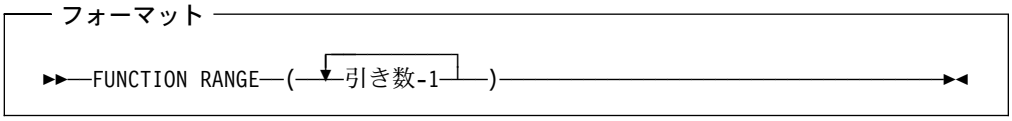
RANGE

RANGE 関数は、最大引き数の値から最小引き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、次に示すように引き数のタイプに応じて異なります。

引き数のタイプ	関数のタイプ
すべての引き数が整数	整数
数字 (一部の引き数が整数)	数字

フォーマット



引き数-1
この引き数のクラスは数字でなければなりません。

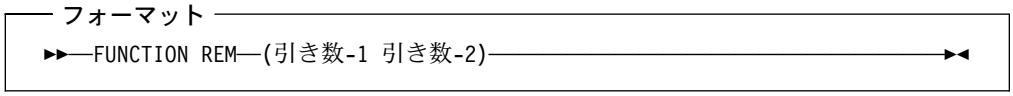
戻される値は、最大値を持つ引き数-1 から最小値を持つ引き数-1 を減算して得られた値に等しくなります。最大値と最小値を決定するために使用される比較は、単純条件に関する規則に従って行われます。詳細については、235ページの『条件式』を参照してください。

REM

REM 関数は、引き数-1 を引き数-2 で除算したその剰余に等しい値を戻します。

関数のタイプは数字です。

フォーマット



引き数-1
この引き数のクラスは数字でなければなりません。

引き数-2
この引き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。

戻される値は、引き数-1 を引き数-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。

引き数-1 - (引き数-2 * FUNCTION INTEGER-PART (引き数-1/ 引き数-2))

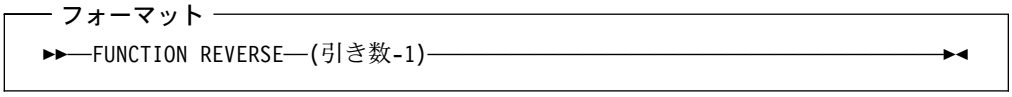
REVERSE

REVERSE 関数は、引き数に使用した文字と同一の文字を使用して、それらの文字の順序を逆にして、引き数と同一の長さの文字ストリングとして戻します。国別引き数の場合は、国別文字位置が逆順になります。

関数のタイプは、次に示すように引き数のタイプに応じて異なります。

引き数のタイプ	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



引き数-1

この引き数のクラスは英字、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。

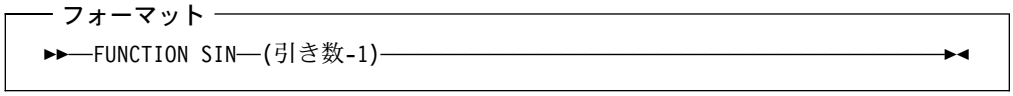
引き数-1 が、長さ n の文字ストリングであるとすれば、戻される値は長さ n の文字ストリングであり、戻される値の文字位置 j が 1 <= j <= n を満たす場合、この文字の引き数 -1 での文字位置は n-j+1 となります。

SIN

SIN 関数は、引き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。

関数のタイプは数字です。

フォーマット



引き数-1
この引き数のクラスは数字でなければなりません。

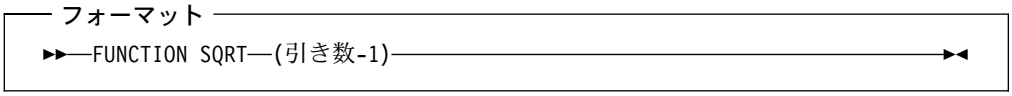
戻される値は引き数-1 の正弦の近似値で、-1 以上 +1 以下の値です。

SQRT

SQRT 関数は、指定された引き数の平方根に近似する数値を戻します。

関数のタイプは数字です。

フォーマット



引き数-1

この引き数のクラスは数字でなければなりません。引き数-1 の値は、0 または正の数でなければなりません。

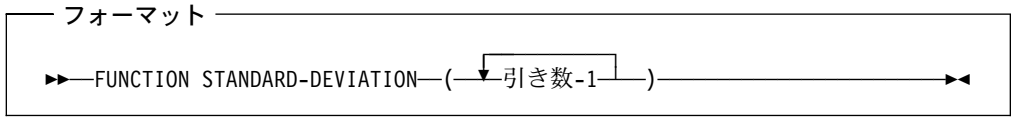
戻される値は、引き数-1 の平方根の近似値の絶対値です。

STANDARD-DEVIATION

STANDARD-DEVIATION 関数は、引き数の標準偏差に近似する数値を戻します。

関数のタイプは数字です。

フォーマット



引き数-1
この引き数のクラスは数字でなければなりません。

戻される値は、一連の引き数-1 の標準偏差の近似値です。戻される値は、以下のようにして計算されます。

1. それぞれの引き数-1 と一連の引き数-1 の算術平均との差を計算し、これを 2 乗します。
2. 得られた値を次にすべて加算します。この値を一連の引き数-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。戻される値は、この平方根の絶対値です。

一連の引き数-1 が唯一の値で構成される場合、または一連の引き数-1 が可変オカレンス・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合計が 1 である場合、戻される値は 0 です。

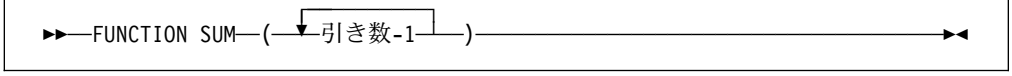
SUM

SUM 関数は、引き数の合計に等しい値を戻します。

関数のタイプは、次に示すように引き数のタイプに応じて異なります。

引き数のタイプ	関数のタイプ
すべての引き数が整数	整数
数字 (一部の引き数が整数)	数字

フォーマット



引き数-1
この引き数のクラスは数字でなければなりません。

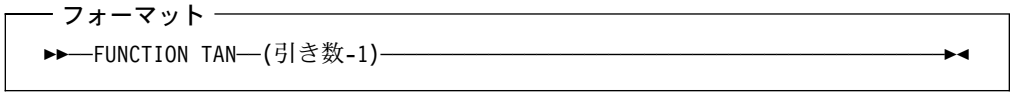
戻される値は、引き数の合計です。一連の引き数-1 がすべて整数である場合には、戻される値は整数です。一連の引き数-1 がすべて整数とは限らない場合には数字が戻されます。

TAN

TAN 関数は、引き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。

関数のタイプは数字です。

フォーマット



引き数-1
この引き数のクラスは数字でなければなりません。

戻される値は、引き数-1 の正接の近似値です。

UNDATE

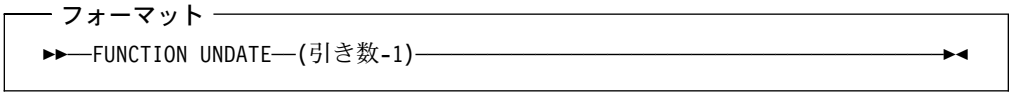
UNDATE 関数は、日付フィールドを非日付データに変換して非日付データで使用してもあいまいとならないようにします。

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、次に示すように引き数-1 のタイプに応じて異なります。

引き数-1 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



引き数-1

日付フィールド。

戻される値は、引き数-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。

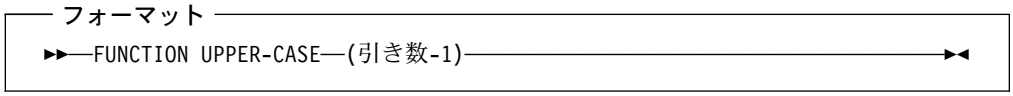
UPPER-CASE

UPPER-CASE 関数は、引き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、同じ長さの文字ストリングとして戻します。

関数のタイプは、次に示すように引き数のタイプに応じて異なります。

引き数のタイプ	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



引き数-1

この引き数のクラスは英字、英数字、または国別でなければならず、少なくとも 1 文字位置の長さを持た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、引き数-1 と同じ長さの文字ストリングが戻されます。

引き数-1 が英字または英数字の場合は、'a' ～ 'z' までの小文字は対応する大文字 'A' ～ 'Z' に置き換えられます。ここで 'a' ～ 'z' の範囲と 'A' ～ 'Z' の範囲は、有効なコード・ページにかかわらず 556ページの表60 に示されているようになります。

引き数-1 が国別の場合は、各小文字は Unicode データベース UnicodeData.txt (http:/www.unicode.org/ の Unicode Consortium から入手可能) の仕様に基づいて、対応する大文字に置き換えられます。

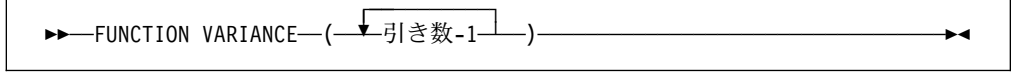
戻される文字ストリングは、引き数-1 と同じ長さを持ちます。

VARIANCE

VARIANCE 関数は、引き数の分散に近似する数値を戻します。

関数のタイプは数字です。

フォーマット



引き数-1

この引き数のクラスは数字でなければなりません。

戻される値は、一連の引き数-1 の分散の近似値です。

戻される値は、一連の引き数-1 の標準偏差の 2 乗として定義されます。この値は次のようにして計算されます。

1. それぞれの引き数-1 と一連の引き数-1 の算術平均との差を計算し、これを 2 乗します。
2. 得られた値を次にすべて加算します。この値を一連の引き数の数で除算します。

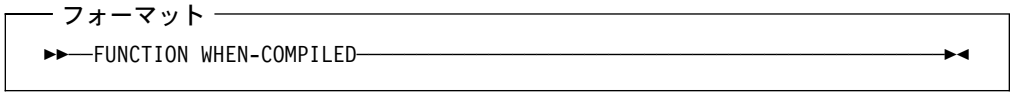
一連の引き数-1 が唯一の値で構成される場合、または一連の引き数-1 が可変オカレンス・データ項目から構成され、それらのデータ項目のオカレンス数の合計が 1 である場合、戻される値は 0 です。

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がコンパイルされた日時を戻します。

この関数のタイプは英数字です。

フォーマット



戻される値の中の 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 ～ 4	グレゴリオ暦におけるその年を表す 4 桁の数字。
5 ～ 6	月を表す 2 桁の数字で、01 ～ 12 の範囲にある値。
7 ～ 8	日を表す 2 桁の数字で、01 ～ 31 の範囲にある値。
9 ～ 10	午前 0 時からの時間を表す 2 桁の数字で、00 ～ 23 の範囲にある値。
11 ～ 12	分を表す 2 桁の数字で、00 ～ 59 の範囲にある値。
13 ～ 14	秒を表す 2 桁の数字で、00 ～ 59 の範囲にある値。
15 ～ 16	100 分の 1 秒を表す 2 桁の数字で、00 ～ 99 の範囲の値。関数を実行するシステムが、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。この関数を実行するシステムがロケールによる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 ～ 19	17 の文字位置が '-' の場合は、時間を表す 00 ～ 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を示す 00 ～ 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 ～ 21	前記の時間差に加えるべき分単位の時間を表す 00 ～ 59 の範囲の 2 桁の数字が戻されます。文字位置 17 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。

戻される値は、この関数を含むソース・プログラムのコンパイル日時です。その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、戻される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。

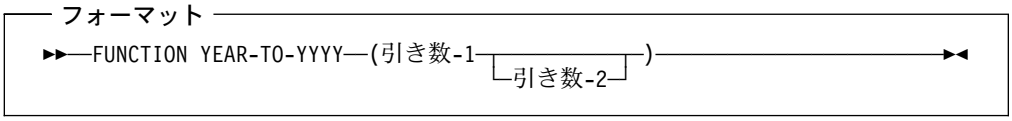
YEAR-TO-YYYY

YEAR-TO-YYYY 関数は、引き数-1、すなわち 2 桁の年を 4 桁の年に変換します。引き数-2 は実行時に年に追加されると、100 年の間隔の終了年を定義するか、または引き数-1 を入れる世紀ウィンドウをどのようにスライドさせるかを定義します。

この関数のタイプは整数です。

DATEPROC コンパイラー・オプションが有効の場合の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。

フォーマット



引き数-1

100 未満の負ではない整数でなければなりません。

引き数-2

これは整数でなければなりません。引き数-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実行されます。

実行時における年と引き数-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。

例

YEAR-TO-YYYY 関数からの戻り値の例を 2 つ、以下に示します。

現在の年	引き数-1 の値	引き数-2 の値	YEAR-TO-YYYY からの戻り値
1995	4	23	2004
1995	4	-15	1904
2008	98	23	1998
2008	98	-15	1898

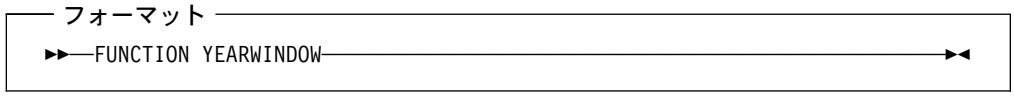
YEARWINDOW

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、暗黙の DATE FORMAT が YYYY の拡張日付フィールドです。

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



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 506
BASIS 506
CBL (PROCESS) ステートメント 508
*CONTROL (*CBL) ステートメント 509
COPY ステートメント 511
DELETE ステートメント 518
EJECT ステートメント 519
ENTER ステートメント 520
INSERT ステートメント 520
READY または RESET TRACE ステートメント 521
REPLACE ステートメント 522
SERVICE LABEL ステートメント 526
SERVICE RELOAD ステートメント 526
SKIP1/2/3 ステートメント 526
TITLE ステートメント 528
USE ステートメント 529

© Copyright IBM Corp. 1991, 2001

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ものです。

コンパイラー指示ステートメントは、以下の目的で使用します。

- 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)
- コンパイラー・リストの制御 (*CONTROL/*CBL、EJECT、TITLE、および SKIP1/2/3 ステートメント)
- コンパイラー・オプションの指定 (CBL/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)

SERVICE LABEL ステートメントは言語環境プログラム条件処理で使用され、このステートメントも CICS プリプロセッサーによって生成されます。

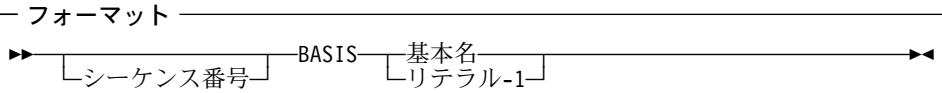
次のコンパイラー指示には効果がありません。ENTER、READY、または RESET TRACE、SERVICE RELOAD。

BASIS

BASIS ステートメントは、ソース・プログラムのライブラリー拡張用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、ユーザーのライブラリーの中に 1 つの記入項目として含めておくことができ、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 (またはその一方) 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 ～ 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 ～ 72 桁のどこにでも記入できます。後に BASIS 名を続けます。このステートメントの中に他のテキストは含まれません。

BASIS 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記入項目を認識します。

形成規則と処理規則については、511ページの『COPY ステートメント』のリテラル-1 とテキスト名-1 の説明を参照してください。

© Copyright IBM Corp. 1991, 2001

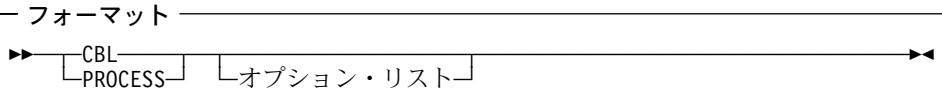
ソース・ファイルは、BASIS ステートメントの実行後も変更されません。

注: 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。

CBL (PROCESS) ステートメント

CBL (PROCESS) ステートメントを使用することによって、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。CBL (PROCESS) ステートメントは、最外部のプログラムの見出し部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 つまたは複数のコンパイラー・オプションで、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なければなりません。

コンパイラー・オプションの詳細については、 *Enterprise COBOL プログラミング・ガイド* を参照してください。

CBL (PROCESS) ステートメントは、1 ～ 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。CBL や PROCESS 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、CBL や PROCESS は 1 桁目から始めることができます。

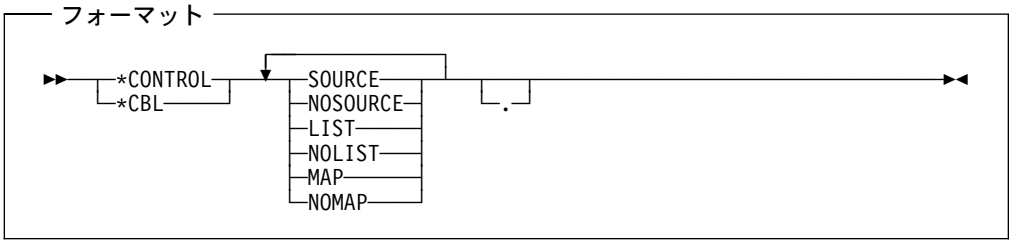
CBL (PROCESS) ステートメントは、72 桁以前に終わらなければなりません。また、複数の CBL (PROCESS) ステートメントにオプションを続けることはできません。ただし、複数の CBL (PROCESS) ステートメントを使用することはできます。複数の CBL (PROCESS) ステートメントを使用する場合には、それらのステートメント間に他のタイプのステートメントを入れずに続けなければなりません。

CBL (PROCESS) 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ことによって、ソース・コードやオブジェクト・コードのリスト、およびソース・プログラムの全体にわたるストレージ・マップのリストを選択して表示したり、表示を抑止したりすることができます。

フォーマット



これらのオプションを指定することによって得られる出力についての詳しい説明は、 *Enterprise COBOL プログラミング・ガイド* 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 が使用できるところはどこでも、*CBL を使用することができます。

*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。たとえば、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、この固定オプションは次に挙げるすべてのものを優先します。

- 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、リスト出力が抑止されるソース・プログラム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・コードのリスト

入力ソース・プログラム行のリストは、次のどちらかのステートメントによって制御されます。

```
*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。PRINTING OF THE SOURCE HAS BEEN SUPPRESSED 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・コードのリスト

生成されるオブジェクト・コードのリストは、手続き部の中に書かれた次のどちらかのステートメントによって制御されます。

```
*CONTROL LIST             [*CBL LIST]
*CONTROL NOLIST           [*CBL NOLIST]
```

*CONTROL NOLIST ステートメントで、かつコンパイラー・オプションとして LIST が要求されていた場合、ここ以降、生成されたオブジェクト・コードのリストは抑止されます。

ストレージ・マップのリスト

ストレージ・マップの記入項目のリストは、データ部の中にある次のどちらかのステートメントによって制御されています。

```
*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・マップの記入項目のリストは抑止されます。

たとえば、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

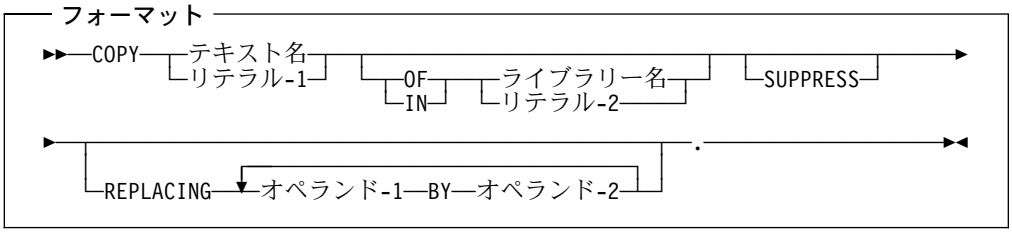
COPY ステートメントは、事前に書かれたテキストを COBOL コンパイル単位に入れるライブラリー・ステートメントです。

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、これらの記入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・テキストを示します。ライブラリー名は、コピー・テキストが存在する場所を示します。

- 1 ～ 30 文字の長さでなければなりません。
- 文字 A ～ Z、a ～ z、0 ～ 9、およびハイフンを含めることができます。
- 先頭の文字は英字でなければなりません。
- 最後の文字にハイフンは使用できません。

テキスト名とライブラリー名はユーザー定義語として同一にすることができます。

テキスト名は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。

JCL または TSO からコンパイルする場合は、最初の 8 文字だけが識別名として使用されます。cob2 コマンドを使用してコンパイルし、階層ファイル・システム (HFS) に入っている COPY テキストを処理する場合は、すべての文字が意味を持ちます。

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。

JCL または TSO からコンパイルする場合:

- 1 ～ 30 文字の長さでなければなりません。
- 文字 A ～ Z、a ～ z、0 ～ 9、およびハイフンを含めることができます。
- 先頭の文字は英字でなければなりません。
- 最後の文字にハイフンは使用できません。
- 最初の 8 文字だけが識別名として使用されます。

cob2 コマンドを使用してコンパイルし、HFS に入っている COPY テキストを処理する場合は、リテラルの長さは 1 ～ 160 文字の範囲です。

テキスト名と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。

処理規則については、 *Enterprise COBOL プログラミング・ガイド* 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) のいずれかにすることができます。

ライブラリー・テキストと疑似テキストは、DBCS ユーザー定義語および DBCS リテラルから構成する (またはこれらを含める) ことができます。DBCS ユーザー定義語は ID として指定することができ、DBCS リテラルはリテラル・オペランドが許可されている場所で指定することができます。国別リテラルはリテラル・オペランドが許可されている場所で指定できます。

DBCS 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、DBCS ワードを部分語で置き換えることはできません。

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、区切り文字の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・プログラムの中で 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。ただし、ネストされた COPY ステートメントでは REPLACING 句を指定することはできません。また、REPLACING 句の指定のある COPY ステートメントの中にネストされた COPY ステートメントを含めることはできません。

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。たとえば、ソース・プログラムに `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、ライブラリー・テキスト Y には `COPY X` ステートメントまたは `COPY Y` ステートメントが含まれていてはなりません。

デバッグ行は、ライブラリー・テキストと疑似テキストの中で使用することができます。デバッグ行の中のテキスト・ワードは、D が標識域に表示されていない場合と同様にマッチング規則の適用対象になります。デバッグ行が疑似テキスト内で指定されるのは、そのデバッ

グ行がソース・プログラムにおいて、疑似テキスト開始の区切り文字の後、対応する疑似テキスト終了の区切り文字の前で始まる場合です。

COPY ステートメントを実行した結果としてソース・プログラムの中に余分な行が入れられると、その COPY ステートメントがデバッグ行で開始される場合、または挿入されるテキスト・ワードがライブラリー・テキスト中のデバッグ行上にある場合には、その挿入されるテキスト・ワードはデバッグ行上に置かれることになります。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・プログラム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 ワードがコメント記入項目に置かれているか、またはコメント記入項目を指定できるところにある場合には、その COPY ワードはコメント記入項目の一部とみなされます。

すべての COPY ステートメントと REPLACE ステートメントの処理が終了すると、デバッグ行は、SOURCE-COMPUTER 段落の中に WITH DEBUGGING MODE 文節が指定されていない場合には、コメント行としてのすべての特性を持っているものとみなされます。

コメント行やブランク行は、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やブランク行は、結果ソース・プログラム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やブランク行は、そのコメント行や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ある場合はコピーされません (515ページの『置換と比較に関する規則』を参照)。

*CONTROL (*CBL)、EJECT、SKIP1/2/3、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・プログラム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、標準の COBOL フォーマットの規則に適合している必要があります。

SUPPRESS 句

SUPPRESS 句は、ライブラリー・テキストをソース・プログラム・リストに印刷させないように指定します。

REPLACING 句

以下の説明では、各**オペランド**は、次のいずれかで構成されています。

- 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

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 は、それが完全に一致するたびに関連するオペランド-2 によって置き換えられます。

疑似テキスト

疑似テキスト区切り文字-1 (==) 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-1 区切り文字の両方の文字は 1 行に記述しなければなりません。ただし、疑似テキスト-1 の文字ストリングは次に続けることができます。

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。詳細については、2ページの『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。

疑似テキストに COPY ワードを含めることはできません。

疑似テキスト-1 は、区切り文字のコンマまたは区切り文字のセミコロンだけ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、スペース文字またはコメント行あるいはその両方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。

ID データ部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

DBCS 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。

区切り文字以外の COBOL 文字 (たとえば +、*、/、$、<、>、および =) 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、ハイフンをその語の最初の文字または最後の文字に使用することもできます。

マッチングを行うために、ID-1、リテラル-1、またはワード-1 は、それぞれ ID-1、リテラル-1、またはワード-1 だけを含む疑似テキストとして扱われます。

置換と比較に関する規則

1. 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・オプションによってのみ置き換えることができます。
2. 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。
3. 形象定数がオペランド-1 であるときには、それが指定されたとおりに現れる場合に限り一致したものとみなします。たとえば、ALL “AB” がライブラリー・テキストに指定されると、“ABAB” は一致とみなされません。ALL “AB” だけが一致とみなされます。
4. PICTURE 文字ストリングを置き換える場合、疑似テキスト・オプションを使用します。あいまいさを除くため、疑似テキスト-1 では、PICTURE や PIC といったキーワードを含め、 PICTURE 文節全体を指定するようにします。
5. ライブラリー・テキストの中の左端の語の前にある区切り文字のコンマ、セミコロン、スペースが、ソース・プログラムの中にコピーされます。左端のライブラリー・テキスト・ワードおよび REPLACING オプションの中に指定されている最初のオペランド-1 から開始して、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。
6. オペランド-1 の中の順序付けられた一連のテキスト名が、順序付けられた一連のライブラリー・ワードと 1 文字 1 文字がすべて等しい場合に限り、オペランド-1 はライブラリー・テキストと一致するものとみなします。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。マッチングにおいて、区切り文字のコンマやセミコロン、および 1 つまたは複数の区切り文字のスペースは、それぞれ現れるたびに 1 つのスペースであるとみなされます。しかし、疑似テキスト-1 が区切り文字のコンマまたはセミコロンだけで構成されている場合、それは突き合わせにおいてテキスト・ワードとして扱われます (この場合、コンマやセミコロンの後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、その直後に区切り文字としてスペース、コンマ、セミコロン、またはピリオドがない場合、その終了引用符は区切り文字の引用符とみなされます。
7. 一致しなければ、一致するものが検索されるまで、または REPLACING オペランドがそれ以上検索されないという時点まで、後続のそれぞれのオペランド-1 (指定されていれば) に関して比較が繰り返されます。
8. オペランド-1 とライブラリー・テキストが一致するごとに、関連するオペランド-2 がソース・プログラムの中にコピーされます。
9. REPLACING 句のついた COPY ステートメントを使用すれば、語の一部を置き換えることができます。コロンによってプログラム・テキストに区切ったダミー・オペランドを挿入すると、コンパイラーは、指定したテキストをそのダミー・オペランドに置き換えます。例 3 では、**:TAG:** というダミー・オペランドを例に、その使用方法を示します。

   注: 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。

10. すべてのオペランドが比較され、一致するものが検索されなければ、左端のライブラリー・テキスト・ワードがソース・プログラムの中にコピーされます。
11. さらに、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・ワードとみなされ、最初のオペランド-1 から始めて、比較処理が繰り返されます。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けられます。
12. ライブラリー・テキストと疑似テキスト-1 に置かれているコメント行やブランク行は、マッチングのために無視されます。ライブラリー・テキストと疑似テキスト-1 内のテキスト・ワードの並びは、参照フォーマットの規則によって判別されます。疑似テキスト-2 に現れるコメント行またはブランク行は、テキスト置き換えの結果として疑似テキスト-2 がソース・プログラムの中に入れられるとき、結果として得られたプログラムに変更を加えずにコピーされます。ライブラリー・テキストに置かれているコメント行やブランク行は、結果ソース・プログラム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やブランク行が、疑似テキスト-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、そのコメント行やブランク行はコピーされません。
13. 置き換え後のテキスト・ワードは、標準の COBOL フォーマットの規則に従って、ソース・プログラムの中に入れられます。
14. COPY REPLACING は、コンパイラー指示ステートメントの EJECT、SKIP1/2/3、または TITLE に影響を与えません。テキスト・ワードがソース・プログラムの中に入れられる場合、すでにスペース (ソース・コード行間の想定上のスペースを含む) が存在する場所にだけ、テキスト・ワードの間に追加のスペースが挿入されます。

コードの並び (ファイルおよびデータ記述、エラーおよび例外ルーチンなど) で、複数のプログラムに共通のものは、ライブラリーに保管することができ、COPY ステートメントと関連して使用することができます。そのような共通のコードに関して命名規則が確立されていれば、REPLACING 句を指定する必要はありません。プログラムごとに別の名前に変更する場合には、そのプログラムにとって意味のある名前を与えるために REPLACING 句を使用することができます。

例 1

この例では、ライブラリー・テキスト PAYLIB は、次のようなデータ部の記入項目から構成されています。

```
01  A.
  02  B    PIC S99.
  02  C    PIC S9(5)V99.
  02  D    PIC S9999 OCCURS 1 TO 52 TIMES
      DEPENDING ON B OF A.
```

プログラマーは、プログラムのデータ部の中で COPY ステートメントを次のように使用することができます。

```
    COPY PAYLIB.
```

このプログラムでは、ライブラリー・テキストがコピーされます。結果テキストは、次のように書き込ま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・テキストの中のいくつかの (またはすべての) 名前を変更するために、プログラマーは 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.
```

このプログラムでは、ライブラリー・テキストがコピーされます。結果テキストは、次のように書き込ま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だけ行われます。ライブラリーの中にあるテキストは変更されません。

例 3

ライブラリー・テキストの中で次のような基準に従っている場合、名前の一部 (たとえばデータ名の接頭部) 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、次のようなデータ部の記入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:.
```

プログラマーは、プログラムのデータ部の中で COPY ステートメントを次のように使用することができます。

```
COPY PAYLIB REPLACING ==:TAG:== BY ==Payroll==.
```

注: この例で特に注意する点は、ライブラリー・テキストの中で区切り文字としてコロンや括弧を使用する必要があることです。括弧は、添え字やテーブルの場合でも使用されるため、わかりやすさを考慮してコロンを使用してください。

このプログラムでは、ライブラリー・テキストがコピーされます。結果テキストは、次のように書き込ま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だけ行われます。ライブラリーの中にあるテキストは変更されません。

例 4

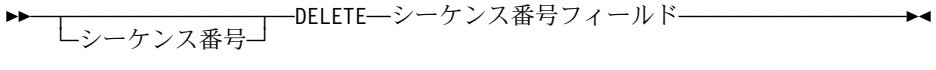
この例では、PICTURE 文節の番号を置き換えずに、レベル番号を選択して置き換える方法を示します。

```
COPY xxx REPLACING ==(01)== BY ==(01)==
                   == 01 == BY == 05 ==.
```

DELETE ステートメント

DELETE ステートメントは、ソース・ライブラリー用の拡張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 ～ 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
これは、第 1 ～ 72 桁のどこにでも記述できます。1 つのスペースとシーケンス番号フィールドの後に記述する必要があります。このステートメントの中に他のテキストがある必要はありまん。

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第 1 ～ 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指定する番号は、シーケンス番号の数字の小さいものから順に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 つまたは複数) と番号の範囲 (1 つまたは複数) 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。以下に例を示します。

```
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 ステートメントとみなします。

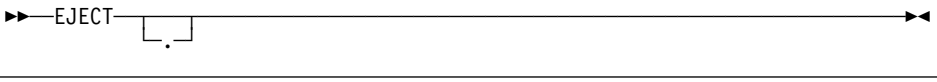
DELETE ステートメントの後に続く新しいソース・プログラム・ステートメントは、DBCS データ項目を参照することができます。

注: 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の小さいものから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、また区切り文字の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。たとえば、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・プログラム自体のコンパイルには影響しません。

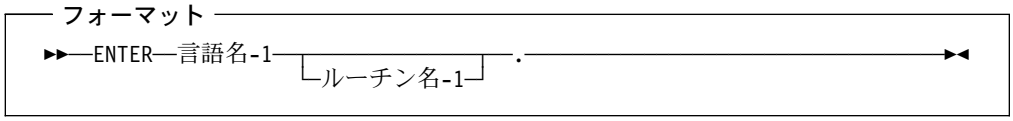
ENTER ステートメント

ENTER ステートメントを使用すると、同じソース・プログラムの中で複数のソース言語を使用できます。

Enterprise COBOL の場合、ソース・プログラムの中で使用できるのは COBOL だけです。

注: ENTER ステートメントは、コンパイル時に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



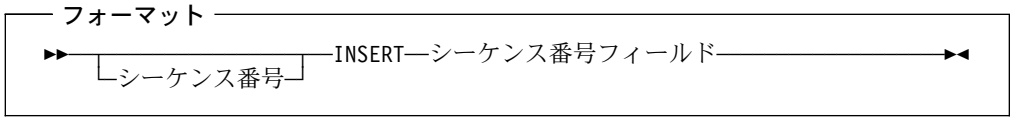
言語名-1
定義された意味を持たないシステム名。これは、正しく作られたユーザー定義語か、「COBOL」ワードのどちらかにしなければなりません。少なくとも 1 文字は英字でなければなりません。

ルーチン名-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 ～ 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
これは、第 1 ～ 72 桁のどこにでも記述できます。1 つのスペースとシーケンス番号フィールドを後に記述する必要があります。このステートメントの中に他のテキストがある必要はありません。

シーケンス番号フィールド
BASIS ソース・プログラムの中にあるシーケンス番号と同じ番号。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第 1 ～ 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指定する番号は、数字の小さいものから順に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、たとえば、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。

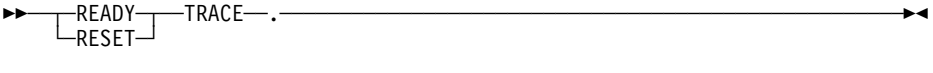
INSERT ステートメントの後に続く新しいソース・プログラム・ステートメントには、DBCS　データ項目を参照することができます。

注: 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の小さいものから順番になっていなければなりません。

READY または RESET TRACE ステートメント

READY ステートメントまたは RESET TRACE ステートメントは、手続き部だけに記述できますが、プログラムには影響しません。

フォーマット



USE FOR DEBUGGING 宣言、DISPLAY ステートメント、DEBUG-ITEM 特殊レジスターを使えば、READY TRACE の関数を複製できます。たとえば、次のようになります。

```
          .
          .
ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
  SOURCE-COMPUTER. IBM-390 WITH DEBUGGING MODE.
          .
DATA DIVISION.
          .
  WORKING-STORAGE SECTION.
  01 TRACE-SWITCH        PIC 9 VALUE 0.
     88  READY-TRACE           VALUE 1.
     88  RESET-TRACE           VALUE 0.
          .
PROCEDURE DIVISION.
  DECLARATIVES.
  COBOL-II-DEBUG SECTION.
    USE FOR DEBUGGING ON ALL PROCEDURES.
  COBOL-II-DEBUG-PARA.
    IF READY-TRACE THEN
        DISPLAY DEBUG-NAME
    END-IF.
  END DECLARATIVES.
  MAIN-PROCESSING SECTION.
          .
  PARAGRAPH-3.
          .
    SET READY-TRACE TO TRUE.
  PARAGRAPH-4.
          .
  PARAGRAPH-6.
          .
    SET RESET-TRACE TO TRUE.
  PARAGRAPH-7.
```

この場合、DEBUG-NAME は、DEBUG-ITEM 特殊レジスターのフィールドであり、これに、デバッグ・プロシージャーを実行するプロシージャー名が表示されます (この例では、オブジェクト・プログラムは、制御がプロシージャー PARAGRAPH-4 ～ PARAGRAPH-6 の範囲内の各プロシージャーに達すると、そのプロシージャーの名前を表示します)。

実行時に、DEBUG ランタイム・オプションを指定して、このデバッグ・プロシージャーをアクティブにする必要があります。この方法を使用すれば、デバッグ宣言をアクティブまたはアクティブを解除するためにプログラムを再コンパイルする必要はありません。

REPLACE ステートメント

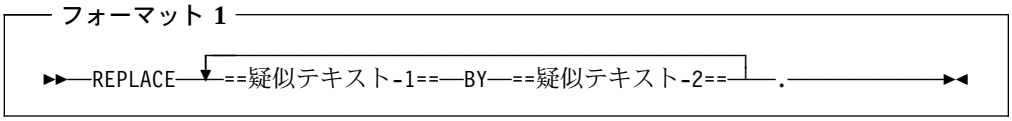
REPLACE ステートメントは、ソース・プログラムのテキストを置き換えるために使用されます。

REPLACE ステートメントは、ソース・プログラム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。これが別個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、前に区切り文字のピリオドを置かなければなりません。また、区切り文字のピリオドで終わる必要があります。

REPLACE ステートメントは、1 つの COBOL ソース・プログラム全体を変更するために使用することも、またはソース・プログラム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・プログラム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。

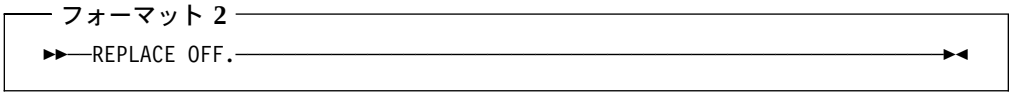
REPLACE ワードがコメント記入項目に置かれているか、またはコメント記入項目が指定できるところにある場合には、その REPLACE ワードはコメント記入項目の一部とみなされます。

フォーマット 1



ソース・プログラム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。フォーマット 2 を指定しない場合、ある 1 つの REPLACE ステートメントの実行は、それが指定されたところから有効になり、次の REPLACE ステートメントが現れるか、または別個にコンパイルされたプログラムの終了まで、それぞれ処理が行われます。

疑似テキスト-1
これは 1 つまたは複数のテキスト・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、区切り文字のコンマまたは区切り文字のセミコロンだけ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 と疑似テキスト-2 には、国別リテラル、 DBCS リテラル、および DBCS ユーザー定義語を含めることができます。

| DBCS 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、DBCS 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 と疑似テキスト-2 には、DBCS文字をコメント行またはコメント記入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 文字までが可能です。ただし、DBCS 文字を含むストリングを継続することはできません。

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・プログラムの REPLACE ステートメントを処理します。完全なソース・プログラム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、そのプログラムを修正できます。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す。

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

疑似テキストを含む文字ストリングや区切り文字は、領域 A からでも領域 B からでも開始できます。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)。

比較演算

テキスト置換を判別する比較操作は、REPLACE ステートメントの後に置かれたソース・プログラム・テキスト・ワードの左端と最初の疑似テキスト-1 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・プログラム・テキスト・ワードと比較されます。疑似テキスト-1 を形成するテキスト・ワードの順序付けられた並びが、文字ごとに比較してプログラム・テキスト・ワードの順序付けられた並びと等しい場合に、疑似テキスト-1 は、ソース・プログラム・テキストと一致したものとみなします。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。

マッチングにおいて、区切り文字のコンマ、セミコロン、およびスペースは、それが出現するたびに 1 つのスペースとみなされます。また、1 つまたは複数のスペース区切り文字の並びは、1 つのスペースとして扱われます。

しかし、疑似テキスト-1 が区切り文字のコンマまたはセミコロンだけで構成されている場合、それは突き合わせにおいてテキスト・ワードとして扱われます (この場合、コンマやセミコロンの後に続くスペースは省略可能です)。

一致するものが現れないときは、連続している疑似テキスト-1 のそれぞれについて比較が繰り返されます。これは、一致するものが見つかるか、または比較する次の疑似テキスト-1 がなくなるまで行われます。

すべての疑似テキスト-1 の並びの比較が終了し、しかも一致するものがない場合には、並んでいる次のソース・プログラム・テキスト・ワードが最も左端のソース・プログラム・テキスト・ワードとみなされ、比較サイクルが再び最初の疑似テキスト-1 との間で開始されます。

疑似テキスト-1 とソース・プログラム・テキストの間で一致したものがあると、対応している疑似テキスト-2 が、ソース・プログラムの中の一致したテキストと置き変わります。突き合わせが行われたテキスト・ワードの右端の直後に続くソース・プログラム・テキスト・ワードが、今度は左端のソース・プログラム・テキスト・ワードとみなされます。比較サイクルは、再び疑似テキスト-1 の最初のものとの間で開始されます。

比較操作は、REPLACE ステートメントの範囲内にあるソース・プログラム・テキストの中の右端のテキスト・ワードが、突き合わせの対象となるか、またはそれが左端のソース・プログラム・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。

REPLACE ステートメントに関する注意事項

ソース・プログラムおよび疑似テキスト-1 の中に現れるコメント行とブランク行は、マッチングの際には無視されます。ソース・プログラム・テキストと疑似テキスト-1 の中のテキスト・ワードの並びは、参照フォーマットに関する規則によって判別されます (41ページの『参照フォーマット』を参照)。疑似テキスト-2 の中にあるコメント行とブランク行は、疑似テキスト-2 がテキスト置換の結果ソース・プログラムの中に入れられるときには常に、結果として得られるプログラムの中に変更されずにそのまま入れられます。ソース・プログラム・テキストに置かれているコメント行やブランク行は、そのまま保存されます。ただし、ソース・プログラム・テキストに置かれているコメント行やブランク行が、疑似テキスト-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、そのコメント行やブランク行は保存されません。

*CONTROL (*CBL)、EJECT、SKIP1/2/3、または TITLE の各ステートメントを持つ行は、ソース・プログラム・テキストの中に現れることができます。これらの行は、REPLACE 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。

疑似テキストには、デバッグ行を入れることができます。デバッグ行の中のテキスト・ワードは、D が標識域に表示されていない場合と同様にマッチング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 D がないものとして扱われます。

すべての COPY ステートメントと REPLACE ステートメントの処理が終了すると、デバッグ行は、SOURCE-COMPUTER 段落の中に WITH DEBUGGING MODE 文節が指定されていない場合には、コメント行としてのすべての特性を持っているものとみなされます。

COPY ステートメントと REPLACE ステートメントを除き、ソース・プログラム・テキストの構文が正しいかどうかは、すべての COPY ステートメントと REPLACE ステートメントの処理が完了するまで判別することはできません。

REPLACE ステートメントの処理の結果としてソース・プログラムの中に挿入されたテキスト・ワードは、参照フォーマットに関する規則に従ってソース・プログラムの中に入れられます。疑似テキスト-2 のテキスト・ワードがソース・プログラムに挿入されるとき、スペース (ソース行間の想定上のスペースを含む) がすで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・プログラムに挿入される場合、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。

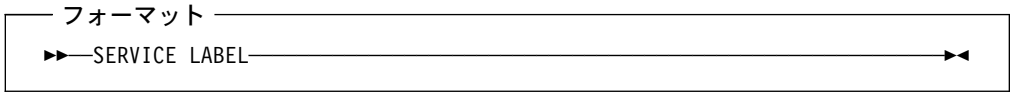
疑似テキスト-2 内にあるリテラルが長すぎて、結果として得られたプログラムの中で次の行に継続しない限り 1 行に収まらない場合は、そのリテラルがデバッグ行に入れられるのでない限り、そのリテラルの余分な部分を入れる追加の継続行が挿入されます。デバッグ行において次の行に継続しなければならないリテラルを置換する要求が生じると、プログラムにエラーが発生します。

注: 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 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 の中にあるときと同じ領域から開始されます。

SERVICE LABEL ステートメント

このステートメントは CICS プリプロセッサーが、制御フローを示すために生成します。また、言語環境プログラム条件処理を使用する際 CEE3SRP への呼び出し後に使用されます。CEE3SRP についての詳細は、*言語環境プログラム プログラミング・ガイド* を参照してください。

フォーマット



SERVICE LABEL ステートメントは、宣言セクションを除く手続き部にのみ記述できます。

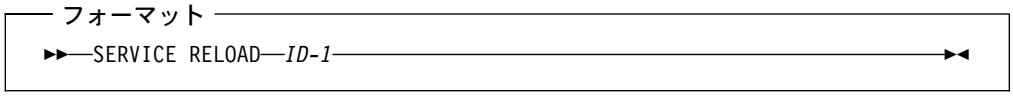
SERVICE LABEL ステートメントの次のステートメントで、有効ではなくなったすべてのレジスターが再ロードされます。

詳しくは、*Enterprise COBOL プログラミング・ガイド* を参照してください。

SERVICE RELOAD ステートメント

SERVICE RELOAD ステートメントはコメントとして扱われます。

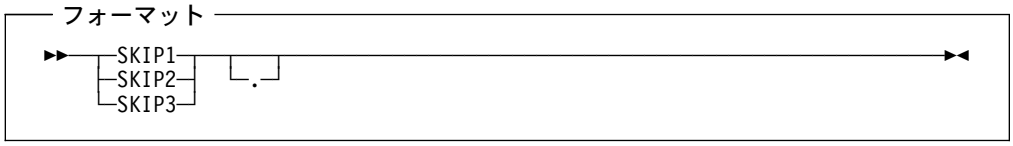
フォーマット



SKIP1/2/3 ステートメント

SKIP1/2/3 ステートメントは、ソース・プログラムのリストを印刷する際にコンパイラーが追加すべきブランク行を指定します。SKIP ステートメントは、ソース・プログラム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。

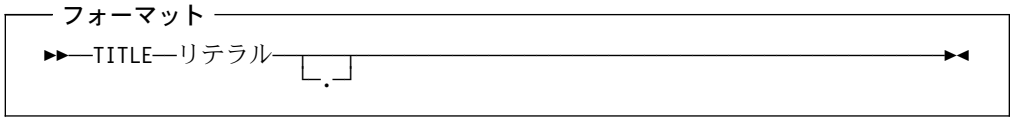
SKIP1、SKIP2、または SKIP3 は、領域 A または領域 B のどこにでも書き込むことができ、区切り文字のピリオドを付けて終了することができます。このステートメントは、その行にある唯一のステートメントでなければなりません。

SKIP1/2/3 ステートメントは、プログラムのソース・コード中に埋め込まなければなりません。たとえば、バッチ・アプリケーションの場合には、SKIP1/2/3 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において各ページの上部に印刷す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、コンパイラーと現在のリリース・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。区切り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、次のような情報が印刷されます。

- 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)。
- 現在のページ番号
- 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。
- このステートメント自体は、ソース・プログラムのリストには印刷されません。
- コンパイルに対して他の影響を及ぼしません。
- 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、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。たとえば、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 CBL (PROCESS) 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 ステートメントのフォーマットには、次のものがあります。

EXCEPTION/ERROR 宣言
LABEL 宣言
DEBUGGING 宣言

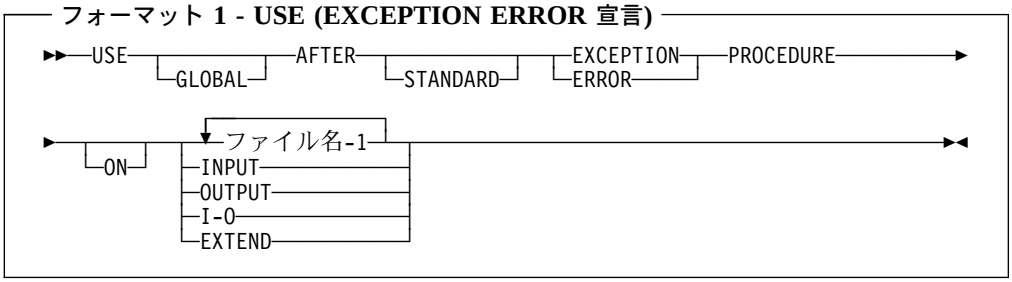
宣言の全般的な情報については、227ページの『宣言部分』を参照してください。

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 - USE (EXCEPTION ERROR 宣言)



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。
- 状況キー 1 を 9 に設定する IBM 定義の条件が検出された場合 (267ページの『状況キー』を参照)。

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、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、267ページの『共通の処理機能』を参照してください。

次の規則が宣言型プロシージャーに適用されます。

- 宣言型プロシージャーは、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の GO TO ステートメントの中で参照できます。
- 宣言型プロシージャーは、宣言型プロシージャーの GO TO ステートメントで参照できます。

以前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、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に含まれるときは、特別の優先規則に従います。これらの規則を適用するときは、実行のために選択される最初の修飾宣言だけは、その宣言の実行に関する規則に従っていなければなりません。宣言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)。

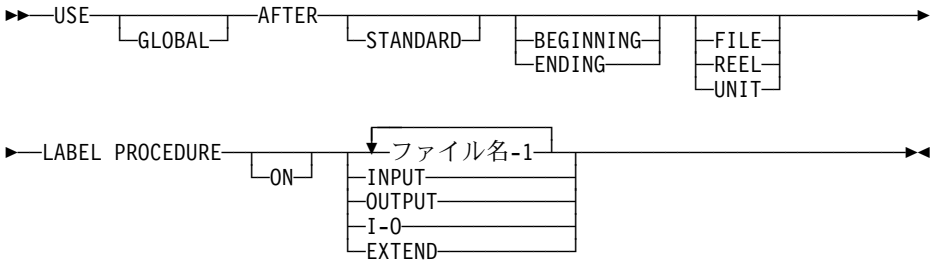
3. GLOBAL 句を指定しており、かつ修飾条件に関して最後に調べられたプログラムを直接的に含んでいるプログラム内にあるファイル特定の宣言。
4. GLOBAL 句を指定しており、かつ修飾条件に関して最後に調べられたプログラムを直接的に含んでいるプログラム内にあるモード特定の宣言。

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。

LABEL 宣言

LABEL 宣言を使用することによって、ユーザーのラベル処理手順を指定することができます。

フォーマット 2 - USE (LABEL 宣言)



AFTER
標準ファイル・ラベルの後にユーザー・ラベルが付き、それが処理されます。

ラベルは、そのファイルに対するファイル記述項目の中の LABEL RECORDS 文節でデータ名としてリストされていなければならず、ファイル記入項目に従属するレベル 01 のデータ項目として記述されていなければなりません。

BEGINNING も ENDING も指定しなければ、指定のプロシージャーは、開始ラベルと終了ラベルの両方について実行されます。

FILE、REEL、または UNIT が指定されていなければ、指定のプロシージャーは、REEL または UNIT のどちらかの適切なラベルと、FILE ラベルの 2 つについて実行されます。

FILE
指定のプロシージャーは、ファイルの開始 (最初のボリューム) とファイルの終了 (最後のボリューム) 、またはそのどちらか一方で実行されます。

REEL
指定のプロシージャーは、ボリュームの開始 (最初のボリュームを除く各ボリューム) とボリュームの終了 (最後のボリュームを除く各ボリューム) またはそのどちらか一方で実行されます。

REEL オプションは、直接アクセス・ファイルには適用できません。

UNIT
指定のプロシージャーは、ボリュームの開始 (最初のボリュームを除く各ボリューム) とボリュームの終了 (最後のボリュームを除く各ボリューム) 、またはそのどちらか一方で実行されます。

UNIT 句は、ランダム・アクセス・モードのファイルには適用できません。このモードでは FILE ラベルだけが処理されるためです。

ファイル名-1
さまざまな配置のフォーマットに指定することができます。ただし、USE ステートメントにおけるファイル名の指定が、2 つ以上の USE 宣言の実行を同時に要求するものであってはなりません。

ファイル名-1 は、ソート・ファイルを表すことはできません。

ファイル名-1 のオプションを使用する場合、ファイル名に対するファイル記述項目は、LABEL RECORDS ARE OMITTED 文節を指定できません。

INPUT、OUTPUT、または I-O のオプションを指定している場合、ユーザー・ラベル・プロシージャーは、次のように実行されます。

- INPUT を指定したときは、INPUT としてオープンしたファイルに対してだけ実行されます。
- OUTPUT を指定したときは、OUTPUT としてオープンしたファイルに対してだけ実行されます。
- I-O を指定したときは、I-O としてオープンしたファイルに対してだけ実行されます。
- EXTEND を指定したときは、EXTEND としてオープンしたファイルに対してだけ実行されます。

INPUT、OUTPUT、または I-O のいずれかの句を指定した場合で、それぞれ対応して入力ファイル、出力ファイル、または入出力ファイルが、LABEL RECORDS ARE OMITTED 文節を指定して記述されているならば、USE プロシージャーは適用されません。標準システム・プロシージャーが実行されるのは、次のような時点です。

- 開始または終了の入力ラベル・チェック・プロシージャーが実行される前。
- 開始または終了の出力ラベルが作成される前。
- 開始または終了の出力ラベルが作成されてから、テープに書き込まれるまでの間。
- 開始または終了の入出力ラベル・チェック・プロシージャーが実行される前。

USE 宣言のプロシージャーにおいて、その中の USE 文が**ファイル名**以外のオプションを指定している場合は、共通ラベル項目を参照する際にファイル名で修飾する必要はありません。共通ラベル項目は、プログラムのすべてのラベル・レコードの中に現れる基本データ項目であっても、そのプログラムのデータ・レコードには現れません。そのような項目は、各ラベル・レコードの中で同一の記述と位置を持つ必要があります。

宣言セクションの中では、どのような宣言型プロシージャーを参照することもできません。逆に、非宣言部分では、宣言セクションの中に現れるプロシージャー名を参照することができません。ただし、PERFORM ステートメントは、USE プロシージャーまたはそれに関連したプロシージャーを参照することができます。

コンパイラーは、宣言セクションからの出口をそのセクションの最後のステートメントの後に挿入します。そのセクション内のすべての論理プログラム・パスは、その出口点に導かれなければなりません。

ただし例外が 1 つあります。特殊出口は、GO TO MORE-LABELS ステートメントで指定できます。このステートメントによって、宣言セクションからの出口が作られると、システムは次のどちらかの処置を行います。

1. 現在の開始ラベルまたは終了ラベルを書き込み、さらにラベルを作成するために USE セクションの先頭に再入します。最後のラベルを作成し終わると、ユーザーはそのセクションの最後のステートメントを実行して終了しなくてはなりません。
2. 追加の開始ラベルまたは終了ラベルを読み取り、さらにラベルを検査するために USE セクションの先頭に再入します。ユーザー・ラベルを処理しているときは、検査すべき別のユーザー・ラベルがある場合にのみ、このセクションに再入します。したがって、セクションの最後のステートメントを通って流れるプログラム・パスがある必要はありません。

ユーザー・ラベルに対して GO TO MORE-LABELS ステートメントを実行しない場合には、すぐ後に検査または作成すべきユーザー・ラベルがあっても宣言セクションに再入することはありません。

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・セッションは、以下のものは無効です。

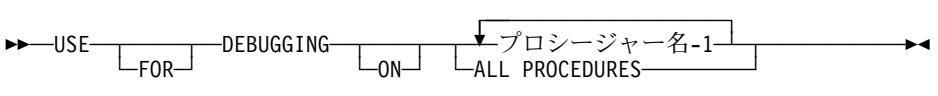
- RECURSIVE 属性で定義されたプログラムまたはメソッド
- THREAD コンパイラー・オプションを指定してコンパイルしたプログラム

SOURCE コンパイラー・ステートメントの WITH DEBUGGING MODE 文節は、コンパイルされてオブジェクト・プログラム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。詳細については、564ページの付録D、『ソース言語のデバッグ』を参照してください。

SOURCE コンパイラー・ステートメントのこのオプションを指定せずにデバッグ・モードを抑止したときは、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 3 - USE (DEBUGGING 宣言)



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。

534ページの表57 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たものです。

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 57. 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すべての非デバッグ・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すべての 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。

付録

© Copyright IBM Corp. 1991, 2001

付録A. IBM 拡張

IBM 拡張とは、577ページの付録G、『業界仕様』 にリストされた 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。

表58には、IBM 拡張が要旨と共に記載されています。拡張については、このブック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する必要があります。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・オプションについて詳しくは、*Enterprise COBOL プログラミング・ガイド* 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。たとえば、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文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58 では、標準の振る舞いが明らかでない場合はそれを大括弧 [ ] で囲んで示しています。

表 58 (1/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
COBOL ワード	DBCS 文字で書かれたユーザー定義語
	DBCS 文字で書かれたコンピューター名
	クラス名 (オブジェクト指向用)
	メソッド名
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
	国別リテラル (基本および 16 進数)
	形象定数 SPACE、ZERO、QUOTE、ALL リテラル
	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

© Copyright IBM Corp. 1991, 2001

表 58 (2/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
暗黙の項目	特殊オブジェクト・リファレンス SELF SUPER 特殊レジスター ADDRESS OF JNIENVPTR 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 ポインターおよびオブジェクト・リファレンス用の NULL/NULLS

表 58 (3/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
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 DBCSおよび英数字の混在 (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。N" および N' は、NSYMBOL(DBCS) コンパイラー・オプションが有効であるときは DBCS として定義されます。 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単一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形成します。 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'。N" および N' は、NSYMBOL(NATIONAL) コンパイラー・オプションが有効である場合は国別として定義されます。 19 (～ 31) 桁の固定小数点数字リテラル。 [標準 COBOL は最大桁数は 18 です。] 浮動小数点数字リテラル。
コメント	見出し部ヘッダーの前のコメント行。 DBCS 文字を含むコメント行およびコメント記入項目。
終了マーカー	END CLASS END FACTORY END METHOD END OBJECT
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 文節 ウィンドウ化日付フィールド、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。 組み込み関数 DATEVAL UNDATE YEARWINDOW DATE-TO-YYYYMMDD DAY-TO-YYYYDDD YEAR-TO-YYYY

表 58 (4/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
プログラム識別部	IDENTIFICATION に対する省略形 ID RECURSIVE 文節 PROGRAM-ID、AUTHOR、INSTALLATION、DATE-WRITTEN、DATE-COMPILED、および SECURITY段落ヘッダーの後のオプショナル区切りピリオド。 [標準 COBOL では PROGRAM-ID の後にピリオドが必要です。] PROGAM-ID 段落内のプログラム名の後のオプショナル区切りピリオド。 [標準 COBOL 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。最外部プログラムの名前の中の文字 $、#、および @。長さ 60 文字までのプログラム名。 [標準 COBOL 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 [標準 COBOL 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。]
オブジェクト指向の構造	クラス定義 CLASS-ID 段落、INHERITS 文節、および END CLASS マーカー メソッド定義 METHOD-ID 段落 (および IS METHOD 句)、EXIT METHOD ステートメント、および END METHOD マーカー
構成セクション	リポジトリー段落

表 58 (5/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
SPECIAL-NAMES 段落	文節のオプショナル順序。 [標準 COBOL では、文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。] 文節がコーディングされていない場合に、最後の文節の後にピリオドを加えるかどうかのオプション。 [標準 COBOL では、文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 文節。 [標準 COBOL では、単一の CURRENCY SIGN 文節は許可されません。] CURRENCY SIGN 文節の WITH PICTURE SYMBOL 句 CURRENCY SIGN 文節での複数文字および大小混合通貨符号 (WITH PICTURE SYMBOL 句が指定されている場合)。 [標準 COBOL 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。これ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 ～ 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。 [標準 COBOL では大文字のみが許可されます。]

表 58 (6/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
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)。 [標準 COBOL では、"INPUT-OUTPUT SECTION." がコーディングされている場合、"FILE-CONTROL." がコーディングされる必要があります。] "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。 [標準 COBOL では、"INPUT-OUTPUT SECTION." がコーディングされている場合、ファイル制御段落がコーディングされる必要があります。] PASSWORD 文節 FILE STATUS 文節の 2 番目のデータ名 ALTERNATE RECORD KEY 文節の RECORD のオプション [標準 COBOL では、ワード RECORD が必要です。] ORGANIZATION IS LINE SEQUENTIAL 文節 数値、数値編集、英数字編集、英字、内部浮動小数点、外部浮動小数点、あるいは DBCS 基本または代替レコード・キー・データ項目。 [標準 COBOL では、キーが英数字であることが必要です。] 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。 [標準 COBOL 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。] FILE STATUS 文節における、USAGE DISPLAY の数値データ項目 [標準 COBOL 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 文節および行順次ファイル制御形式
INPUT-OUTPUT SECTION I-O-CONTROL 段落	APPLY WRITE ONLY 文節 以下の形式の I-O-control 記入項目の SAME 文節に 1 つのファイル名のみを指定 順次 索引付き ソート・マージ [標準 COBOL では、最低でも 2 つのファイル名が必要です。] RERUN 文節のキーワード ON のオプション [標準 COBOL では、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 記入項目 ソート・マージ I-O-control 記入項目の RERUN 文節

表 58 (7/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
データ部	LOCAL-STORAGE SECTION リンケージ・セクションの GLOBAL 文節 データ記述記入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 [標準 COBOL 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。] データ・カテゴリー内部浮動小数点、外部浮動小数点および DBCS TRUNC コンパイラー・オプションの 2 進数値項目の値に対する影響
ファイル・セクション	LABEL RECORDS 文節のデータ名 (ユーザー・ラベルの指定用) RECORDING MODE 文節 行順次形式ファイル記述項目
ソート・マージ・ファイル記述記入項目	BLOCK CONTAINS 文節 LABEL RECORDS 文節 VALUE OF 文節 LINAGE 文節 CODE-SET 文節 WITH FOOTING 文節 LINES AT 文節
BLOCK CONTAINS 文節	QSAM ファイルに対する BLOCK CONTAINS 0 [標準 COBOL では、最低でも 1 CHARACTER または RECORD を BLOCK CONTAINS 文節に指定する必要があります。]
VALUE OF 文節	VALUE 文節がないと、SD の元で指定された場合の実行に影響します。
DATA RECORDS 文節	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 [標準 COBOL 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 文節	EXTEND モードでオープンされたファイルに対する LINAGE の指定
データ記述記入項目	DATE-FORMAT 文節
BLANK WHEN ZERO 文節	ZERO に対する代替スペル ZEROS および ZEROES
GLOBAL 文節	リンケージ・セクションでの GLOBAL の指定

表 58 (8/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
INDEXED BY 句	固有でない非参照索引名
OCCURS 文節	可変長テーブルに対する "integer-1 TO" の省略 複合 OCCURS DEPENDING ON [標準 COBOL では、OCCURS DEPENDING ON を含む記入項目の後ろには従属記入項目のみが続き、OCCURS DEPENDING ON を含む記入項目が OCCURS DEPENDING ON を含む記入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、および COMPUTATIONAL-4 のキー 参照されない固有でない索引名の受け入れ
PICTURE 文節	31 ～ 50 までの文字を含む PICTURE 文字ストリング [標準 COBOL 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 文節がデータ記述記入項目の最後の文節ではない場合の、直後に区切りコンマまたは区切り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 [標準 COBOL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 文節は記入項目内の最後の文節でなければならず、その直後に区切り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 外部浮動小数点フォーマット [標準 COBOL では最大桁数は 18 です。] TRUNC および NUMPROC コンパイラー・オプションの、2 進、計算用、計算用-4 データ項目の値に対する影響
REDEFINES 文節	再定義データ項目の REDEFINES の指定
SYNCHRONIZED 文節	レベル 01 記入項目に対する SYNCHRONIZED の指定

表 58 (9/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
USAGE 文節	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 OBJECT REFERENCE NATIONAL POINTER、 PROCEDURE-POINTER、および FUNCTION-POINTER USAGE INDEX の項目に対する SYNCHRONIZATION 文節の使用
VALUE 文節 条件名記入項目用	ファイルおよびリンケージ・セクションにおける VALUE 文節 (条件名記入項目以外) DISPLAY 以外の USAGE を持つグループでの条件名記入項目に対する VALUE 文節 VALUE IS NULL/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 クラス定義のフォーマット メソッド手続き部 手続き部のヘッダーで USING 句を使用しないリンケージ・セクションのデータ項目の参照 (データ項目が 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) ステートメント ENTRY EXIT METHOD GOBACK INVOKE XML PARSE
手続き部のヘッダー	BY VALUE 句 RETURNING 句 データ項目の記述に REDEFINES 文節がある場合の USING 文節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 クラスのフォーマット

表 58 (10/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
宣言型プロシージャー	LABEL 宣言 宣言型プロシージャーからの非宣言型プロシージャーの実行 宣言型プロシージャーの GO TO ステートメントにおける宣言型プロシージャーまたは非宣言型プロシージャーの参照 [標準 COBOL では、宣言型プロシージャーは非宣言型プロシージャーを参照してはなりません。別の宣言型プロシージャーまたは非宣言型プロシージャーからの宣言型プロシージャーの参照は、PERFORM ステートメントを使用した場合のみ許可されます。] アクティブ宣言の実行
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 [標準 COBOL では、符号なし整数でなければなりません。] 宣言の後ろ、または宣言がない場合のセクション・ヘッダーの省略 [標準 COBOL では、"DECLARATIVES." 構文の後ろ、および "END DECLARATIVES." 構文の後ろにセクション・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 [標準 COBOL 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 [標準 COBOL では、ステートメントが段落内にある必要があります。] 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) [標準 COBOL 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。標準 COBOL 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 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、およびオブジェクト・リファレンス・フォーマット 索引名の演算式との比較 NUMPROC コンパイラー・オプションの符号条件の結果に対する影響
比較条件	簡略複合比較条件内の括弧の使用

表 58 (11/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
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 [標準 COBOL では、最低でも上記のいずれかが必要です。]
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
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・オペランド ADDRESS OF 句 LENGTH OF 句 OMITTED 句 BY VALUE 句 RETURNING 句 引き数としての *file-name* の指定 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 BY CONTENT を渡すときの、従属グループ項目として指定された引き数の指定 [標準 COBOL では、引き数はレベル 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
COMPUTE ステートメント	= 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

表 58 (12/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
EXIT ステートメント	EXIT に続くステートメントを持つ文での EXIT の指定 [標準 COBOL では、EXIT は文内に単独で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 [標準 COBOL 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 [標準 COBOL 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。] MORE-LABELS フォーマット
IF ステートメント	END-IF と NEXT SENTENCE 句との併用 [標準 COBOL では、END-IF を NEXT SENTENCE と併用することはできません。]
INITIALIZE ステートメント	REPLACING 句での DBCS、EGCS、および NATIONAL OCCURS 文節の DEPENDING 句を含むデータ項目の初期化
MERGE ステートメント	SAME 文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	行順次フォーマット LINAGE 文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	空のインライン 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	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	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

表 58 (13/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
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	USAGE ポインター・フォーマット
	USAGE 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
	USAGE オブジェクト・リファレンス・フォーマット
SORT ステートメント	SAME 文節での 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
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
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
	行順次フォーマット
	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
	ファイルに書き込まれるレコードの長さに対する ADV コンパイラー・オプションの影響
	カード・パンチ・ファイルに対するスタッカー選択と共に WRITE ADVANCING を使用
	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

表 58 (14/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
組み込み関数	DATE-OF-INTEGER および DAY-OF-INTEGER 関数に対する DATEPROC および INTDATE コンパイラー・オプションの影響 関数 DATE-TO-YYYYMMDD DATEVAL DAY-TO-YYYYDDD DISPLAY-OF NATIONAL-OF UNDATE YEAR-TO-YYYY YEARWINDOW
FACTORIAL 関数	引き数で許可される値の範囲に対する ARITH(EXTEND) コンパイラー・オプションの影響
INTEGER-OF-DATE 関数	関数の開始日付に対する INTDATE コンパイラー・オプションの影響
INTEGER-OF-DAY 関数	関数の開始日付に対する INTDATE コンパイラー・オプションの影響
LENGTH 関数	関数への引き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
NUMVAL 関数	引き数で許可される最大桁数に対する ARITH(EXTEND) コンパイラー・オプションの影響
NUMVAL-C 関数	引き数で許可される最大桁数に対する ARITH(EXTEND) コンパイラー・オプションの影響
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 CBL(PROCESS) *CONTROL および *CBL DELETE EJECT INSERT READY または RESET TRACE SERVICE LABEL SERVICE RELOAD SKIP/1/2/3 TITLE

表 58 (15/15). IBM 拡張言語エレメント

言語領域	拡張機能エレメント
COPY ステートメント	テキスト名修飾子を指定するための "OF library-name" の構文のオプション 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許可 (COBOL 区切り文字を除く) [標準 COBOL では、ユーザー定義語の構成で使用される文字のみが許可されます。]
USE ステートメント	LABEL 宣言フォーマット

付録B. コンパイラーの制限事項

以下の表には、Enterprise COBOL プログラムのコンパイラー限界値がリストされています。

表 59 (1/4). コンパイラーの制限事項

言語エレメント	コンパイラーの制限事項
プログラムのサイズ	999,999 行
リテラルの数	4,194,303 [1]
リテラルの全長	4,194,303 バイト[1]
予約語テーブルの項目の数	1536
COPY REPLACING ... BY ... (COPY ステートメント当たりの項目数)	なし
COPY ライブラリーの数	なし
COPY ライブラリーのブロック・サイズ	32,767 バイト
見出し部	
環境部	
構成セクション	
SPECIAL-NAMES 段落	
関数名 IS	18
UPSI-n ... (スイッチの数)	0 ～ 7
英字名 IS ...	なし
リテラル THRU/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 ...	なし
ALTERNATE RECORD KEY データ名 ...	253
RECORD KEY の長さ	なし[3]
RESERVE 整数 (バッファー)	255 [4]
I-O-CONTROL 段落	
RERUN ON システム名 ...	32,767
整数 RECORDS	16,777,215
SAME RECORD AREA	255
FOR ファイル名	255

© Copyright IBM Corp. 1991, 2001

表 59 (2/4). コンパイラーの制限事項

言語エレメント	コンパイラーの制限事項
SAME SORT/MERGE AREA	なし[2]
MULTIPLE FILE ... ファイル名	なし[2]
データ部	
ファイル・セクション	
FD ファイル名 ...	65,535
LABEL データ名 ... (オプショナル文節がない場合)	255
ラベル・レコード長	80 バイト
DATA RECORD データ名 ...	なし[2]
BLOCK CONTAINS 整数	2,147,483,647 [8]
RECORD CONTAINS 整数	1,048,575 [5]
項目長	1,048,575 バイト[5]
LINAGE 文節値	99,999,999
SD ファイル名 ...	65,535
DATA RECORD データ名 ...	なし[2]
ソート・レコード長	32,751 バイト
ワーキング・ストレージおよびローカル・ストレージ・セクション	
EXTERNAL 属性を持たない項目	134,217,727 バイト
EXTERNAL 属性を持つ項目	134,217,727 バイト
77 データ名	16,777,215 バイト
01 ～ 49 データ名	16,777,215 バイト
88 条件名 ...	なし
VALUE リテラル ...	なし
66 RENAMES ...	なし
PICTURE 文字ストリング	50
数字項目の数字桁数	ARITH(COMPAT) コンパイラー・オプションが有効な場合: 18 ARITH(EXTEND) コンパイラー・オプションが有効な場合: 31
数字編集文字の文字数	249
PICTURE の複製 (　　)	16,777,215
PICTURE 複製 (編集)	32,767
PICTURE 複製 (　)、DBCS 項目	8,388,607
PICTURE 複製 (　)、国別項目	8,388,607

表 59 (3/4). コンパイラーの制限事項

言語エレメント	コンパイラーの制限事項
グループ項目サイズ: ファイル・セクション	1,048,575 バイト
基本項目のサイズ	16,777,215 バイト
VALUE 初期設定 (すべての値リテラルの合計長)	16,777,215 バイト
OCCURS 整数	16,777,215
ODO の合計数	4,194,303 [1]
テーブル・サイズ	16,777,215 バイト
テーブル・エレメントのサイズ	8,388,607 バイト
ASC/DES KEY ... (OCCURS 文節ごと)	12 KEYS
合計長	256 バイト
INDEXED BY ... (指標名) (OCCURS 文節ごと)	12
指標 (指標名) の合計数	65,535
相対指標のサイズ	32,765
リンケージ・セクション	134,217,727 バイト
01 + 77 の合計 (データ項目)	なし
手続き部	
プロシージャー + 定数域	4,194,303 バイト[1]
USING ID ...	32,767
プロシージャー名	1,048,575 [1]
ステートメントごとの添え字付きデータ名	32,767
行ごとの動詞 (TEST)	7
ACCEPT: 入力装置のレコード長	32,760
ADD ID ...	なし
ALTER pn1 TO pn2 ...	4,194,303 [1]
CALL ... BY CONTENT ID	2,147,483,647 バイト
CALL ID / リテラル USING ID / リテラル...	16380
CALL リテラル ...	4,194,303 [1]
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)	なし
CANCEL ID/ リテラル ...	なし
CLOSE ファイル名 ...	なし
COMPUTE ID ...	なし
DISPLAY ID/ リテラル ...	なし

表 59 (4/4). コンパイラーの制限事項

言語エレメント	コンパイラーの制限事項
DIVIDE ID ...	なし
ENTRY USING ID/ リテラル ...	なし
EVALUATE ... サブジェクト	64
EVALUATE ... WHEN 文節	256
GO pn ... DEPENDING	255
INSPECT TALLY/REPL 文節	なし
MERGE ファイル名 ASC/DES KEY ...	なし
キー合計長	4,092 バイト[6]
USING ファイル名 ...	16[7]
MOVE ID/ リテラル TO ID ...	なし
MULTIPLY ID ...	なし
OPEN ファイル名	なし
PERFORM	4,194,303
SEARCH ... WHEN ...	なし
SET 指標 /ID ... TO	なし
SET 指標 ... UP/DOWN	なし
SORT ファイル名 ASC/DES KEY	なし
キー合計長	4,092 バイト[6]
USING ファイル名 ...	16[7]
STRING ID ...	なし
DELIMITED ID / リテラル...	なし
UNSTRING DELIMITED ID / リテラル OR ID / リテラル...	255
UNSTRING INTO ID / リテラル...	なし
USE ... ON ファイル名 ...	なし

表の注:

1 4,194,303 バイトに含まれる項目は、プロシージャーとそれに定数域を加えたものに制限されます。
2 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。 制限はありません。
3 コンパイラーの制限はありませんが、VSAM がこれを 255 バイトに制限しています。
4 QSAM。
5 コンパイラーの制限は表記のとおりですが、QSAM がこれを 32,767 バイトに制限しています。
6 QSAM と VSAM で OPTION 制御ステートメントに EQUALS が指定されている場合、4088 バイトに制限されます。

7 QSAM および VSAM での SORT の制限。

8 OS/390 DFSMS バージョン 2 リリース 10.0 以降でサポートされる、ラージ・ブロック・インターフェース (LBI) サポートが必要です。それ以前のリリースの DFSMS を使用する OS/390 システムでの制限は、32,767 バイトです。大きなブロック・サイズの使用についての詳細は、*Enterprise COBOL プログラミング・ガイド* を参照してください。

付録C. EBCDIC および ASCII の照合シーケンス

この付録では、単一バイト EBCDIC (拡張 2 進化 10 進コード) および単一バイト 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

表60は、単一バイト EBCDIC コード・ページ 1140 の照合シーケンスを示しています。

表 60 (1/4). EBCDIC 照合シーケンス

序数	記号	意味	10 進表記	16 進表記
65	ƀ	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
109	%	パーセント記号	108	6C

© Copyright IBM Corp. 1991, 2001

表 60 (2/4). EBCDIC 照合シーケンス

序数	記号	意味	10 進表記	16 進表記
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
⋮				

表 60 (3/4). EBCDIC 照合シーケンス

序数	記号	意味	10 進表記	16 進表記
160	€	ユーロ通貨符号	159	9F
⋮				
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

表 60 (4/4). EBCDIC 照合シーケンス

序数	記号	意味	10 進表記	16 進表記
213			212	D4
214	N		213	D5
215	O		214	D6
216	P		215	D7
217	Q		216	D8
218	R		217	D9
⋮				
225	\	バックスラッシュ	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 (ISO 646)

表 61 (1/4). 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	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

表 61 (2/4). ASCII 照合シーケンス

序数	記号	意味	10 進表記	16 進表記
62	=	等号	61	3D
63	>	不等号 (より大きい)	62	3E
64	?	疑問符	63	3F
65	@	単価記号	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	92	5C

表 61 (3/4). ASCII 照合シーケンス

序数	記号	意味	10 進表記	16 進表記
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
99	b		98	62
00	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	¦	分割垂直バー	124	7C

表 61 (4/4). ASCII 照合シーケンス

序数	記号	意味	10 進表記	16 進表記
126	}	右中括弧	125	7D
127	~	波形記号	126	7E

付録D. ソース言語のデバッグ

デバッグ機能を持つ COBOL 言語エレメントは、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 文節)
- オブジェクト時スイッチ

デバッグ行のコーディング

デバッグ行は、コンパイル時スイッチがアクティブな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、たとえば、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、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。ただし、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、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、そのデバッグ行はブランク行として扱われます。

デバッグ・セクションのコーディング

デバッグ・セクションは、最外部プログラムでのみ可能です。ネストされたプログラム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、529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、たとえば、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は、デバッグ・セクションの中で定義されたプロシージャーを参照できません。

© Copyright IBM Corp. 1991, 2001

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。

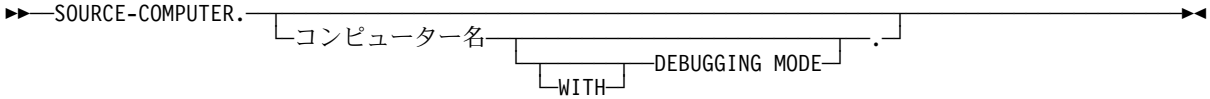
DEBUG-ITEM 特殊レジスター

DEBUG-ITEM 特殊レジスターの詳細については、『DEBUG-ITEM 特殊レジスター』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。

フォーマット



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。

注: 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表示しなければなりません。Enterprise COBOL 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セットされます (NODEBUG は IBM のデフォルト値です)。

フォーマットの詳細については、*言語環境プログラム プログラミング・ガイド* 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 を指定した場合は動作を禁止されます。

デバッグ行 (7 桁目の D) は、DEBUG/NODEBUG オプションの影響を受けません。コンパイル後は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アクティブ解除にするために、ソース・プログラムを再コンパイルする必要はありません。

付録E. 予約語

下記のリストは、Enterprise COBOL において予約されているワード、および Enterprise COBOL の将来のリリースで予約される可能性のあるワードを示しています。

- *Enterprise COBOL* の欄にマークのあるワードは、Enterprise COBOL にインプリメントされている機能用のものです。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* の欄にマークのあるワードは、COBOL 85 予約語のうち、Enterprise COBOL にはインプリメントされていない機能用のものです。(これらの機能の中には報告書作成プログラム・プリコンパイラーにインプリメントされているものもあります。) これらのワードをユーザー定義名として使用すると、S -LEVEL メッセージのフラグが付きます。
- *RFD* の欄にマークのあるワードは、Enterprise COBOL の将来のリリースで予約される可能性があるワードです。この欄には、COBOL 標準の草案である ISO/IEC FCD 1989:2001 で予約されているワードが記載されています。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。
- *X* [1] のマークのあるワードは Enterprise COBOL において新規に予約されたワードであり、他の IBM COBOL コンパイラーでは予約されていない可能性があります。

注: WORD コンパイラー・オプションを使用して、予約語テーブルを切り替えて使用することができます。代替予約語テーブルを指定する方法については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

表 62 (1/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
ACCEPT	X		
ACCESS	X		
ACTIVE-CLASS			X [1]
ADD	X		
ADDRESS	X		
ADVANCING	X		
AFTER	X		
ALIGNED			X [1]
ALL	X		
ALLOCATE			X [1]
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 [1]

表 62 (1/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
APPLY	X		
ARE	X		
AREA	X		
AREAS	X		
AS			X [1]
ASCENDING	X		
ASSIGN	X		
AT	X		
AUTHOR	X		
AUTOMATIC			X
B-AND			X
B-NOT			X
B-OR			X
B-XOR			X [1]
BASED			X [1]
BASIS	X		
BEFORE	X		
BEGINNING	X		
BINARY	X		
BINARY-CHAR			X [1]
BINARY-DOUBLE			X [1]
BINARY-LONG			X [1]

© Copyright IBM Corp. 1991, 2001

表 62 (2/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
BINARY-SHORT			X [1]
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 [1]
COLLATING	X		
COLS			X [1]
COLUMN		X	
COLUMNS			X [1]
COM-REG	X		
COMMA	X		
COMMON	X		
COMMUNICATION		X	
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		

表 62 (2/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
COMPUTATIONAL-4	X		
COMPUTATIONAL-5	X		
COMPUTE	X		
CONDITION			X [1]
CONFIGURATION	X		
CONSTANT			X [1]
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 [1]
DATA	X		
DATA-POINTER			X [1]
DATE	X		
DATE-COMPILED	X		
DATE-WRITTEN	X		
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		

表 62 (3/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
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 [1]
EGCS	X		
EGI		X	
EJECT	X		
ELSE	X		
EMI		X	
ENABLE		X	
END	X		
END-ACCEPT			X
END-ADD	X		
END-CALL	X		
END-COMPUTE	X		
END-DELETE	X		
END-DISPLAY			X [1]
END-DIVIDE	X		
END-EVALUATE	X		
END-EXEC	X		
END-IF	X		
END-INVOKE	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		

表 62 (3/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
END-STRING	X		
END-SUBTRACT	X		
END-UNSTRING	X		
END-WRITE	X		
END-XML	X [1]		
ENDING	X		
ENTER	X		
ENTRY	X		
ENVIRONMENT	X		
EO			X [1]
EOP	X		
EQUAL	X		
ERROR	X		
ESI		X	
EVALUATE	X		
EVERY	X		
EXCEPTION	X		
EXCEPTION-OBJECT			X [1]
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 [1]
FLOAT-LONG			X [1]
FLOAT-SHORT			X [1]
FOOTING	X		
FOR	X		
FORMAT			X
FREE			X
FROM	X		
FUNCTION	X		
FUNCTION-ID			X [1]

表 62 (4/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
FUNCTION-POINTER	X [1]		
GENERATE		X	
GET			X
GIVING	X		
GLOBAL	X		
GO	X		
GOBACK	X		
GREATER	X		
GROUP		X	
GROUP-USAGE			X [1]
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 [1]
INTERFACE-ID			X [1]
INTO	X		
INVALID	X		
INVOKE	X		
IS	X		
JNIENVPTR	X [1]		
JUST	X		
JUSTIFIED	X		
KANJI	X		

表 62 (4/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
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 [1]
LOCK	X		
LOW-VALUE	X		
LOW-VALUES	X		
MEMORY	X		
MERGE	X		
MESSAGE		X	
METHOD	X		
METHOD-ID	X		
MINUS			X [1]
MODE	X		
MODULES	X		
MORE-LABELS	X		
MOVE	X		
MULTIPLE	X		
MULTIPLY	X		
NATIONAL	X [1]		
NATIONAL-EDITED			X [1]
NATIVE	X		
NEGATIVE	X		
NESTED			X [1]
NEXT	X		
NO	X		
NOT	X		
NULL	X		

表 62 (5/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
NULLS	X		
NUMBER		X	
NUMERIC	X		
NUMERIC-EDITED	X		
OBJECT	X		
OBJECT-COMPUTER	X		
OBJECT-REFERENCE			X [1]
OCCURS	X		
OF	X		
OFF	X		
OMITTED	X		
ON	X		
OPEN	X		
OPTIONAL	X		
OPTIONS			X [1]
OR	X		
ORDER	X		
ORGANIZATION	X		
OTHER	X		
OUTPUT	X		
OVERFLOW	X		
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
PREVIOUS			X
PRINTING		X	
PROCEDURE	X		
PROCEDURE-POINTER	X		

表 62 (5/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
PROCEDURES	X		
PROCEED	X		
PROCESSING	X		
PROGRAM	X		
PROGRAM-ID	X		
PROGRAM-POINTER			X [1]
PROPERTY			X [1]
PROTOTYPE			X [1]
PURGE		X	
QUEUE		X	
QUOTE	X		
QUOTES	X		
RAISE			X [1]
RAISING			X [1]
RANDOM	X		
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		

表 62 (6/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
RESET	X		
RESUME			X [1]
RETRY			X [1]
RETURN	X		
RETURN-CODE	X		
RETURNING	X		
REVERSED	X		
REWIND	X		
REWRITE	X		
RF		X	
RH		X	
RIGHT	X		
ROUNDED	X		
RUN	X		
SAME	X		
SCREEN			X [1]
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 [1]
SHIFT-IN	X		
SHIFT-OUT	X		
SIGN	X		
SIZE	X		
SKIP1	X		
SKIP2	X		
SKIP3	X		
SORT	X		
SORT-CONTROL	X		

表 62 (6/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
SORT-CORE-SIZE	X		
SORT-FILE-SIZE	X		
SORT-MERGE	X		
SORT-MESSAGE	X		
SORT-MODE-SIZE	X		
SORT-RETURN	X		
SOURCE		X	
SOURCE-COMPUTER	X		
SOURCES			X [1]
SPACE	X		
SPACES	X		
SPECIAL-NAMES	X		
SQL	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	
SUB-SCHEMA			X
SUBTRACT	X		
SUM		X	
SUPER	X		
SUPPRESS	X		
SYMBOLIC	X		
SYNC	X		
SYNCHRONIZED	X		
SYSTEM-DEFAULT			X [1]
TABLE		X	
TALLY	X		
TALLYING	X		
TAPE	X		
TERMINAL		X	
TERMINATE		X	
TEST	X		
TEXT		X	
THAN	X		

表 62 (7/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
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 [1]
UNIT	X		
UNIVERSAL			X [1]
UNLOCK			X [1]
UNSTRING	X		
UNTIL	X		
UP	X		
UPDATE			X
UPON	X		
USAGE	X		
USE	X		
USER-DEFAULT			X [1]
USING	X		
VAL-STATUS			X [1]
VALID			X
VALIDATE			X
VALIDATE-STATUS			X [1]
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 [1]		
XML-CODE	X [1]		

表 62 (7/7). 予約語

予約語	Enterprise COBOL	標準のみ	RFD
XML-EVENT	X [1]		
XML-NTEXT	X [1]		
XML-TEXT	X [1]		
ZERO	X		
ZEROES	X		
ZEROS	X		
<	X		
<=	X		
+	X		
*	X		
**	X		
-	X		
/	X		
>	X		
>=	X		
=	X		
&			X [1]
*>			X [1]
::			X [1]
>>			X [1]

注:

X　Enterprise COBOL、COBOL (OS/390 および VM 版)、COBOL Set for AIX、および VisualAge COBOL で予約されたワード。その他のワードが COBOL (OS/390 および VM 版)、COBOL Set for AIX、または VisualAge COBOL で予約またはインプリメントされる可能性があります。

X [1]　Enterprise COBOL で新規に予約されたワード。

付録F. ASCII に関する考慮事項

このコンパイラーは、情報交換用米国標準コード (ASCII) をサポートしています。したがって、プログラマーは、次の規格に従って記録されるテープ・ファイルを作成し、処理することができます。

- 情報交換用米国標準コード、X3.4-1977
- 情報交換用米国標準磁気テープ・ラベル、X3.27-1978
- 情報交換用米国標準磁記録気テープ (800 CPI、NRZI)、X3.22-1967

単一バイト ASCII コード化テープ・ファイルは、システムの中に読み込まれたとき、自動的にバッファーの中で単一バイトEBCDIC に変換されます。データの内部操作は、あたかもそれらの ASCII ファイルが単一バイトの EBCDIC コード・ファイルであるかのように行われます。出力ファイルの場合は、システムがバッファーの中で EBCDIC 文字を単一バイトの ASCII に変換してから、ファイルをテープに書き出します。したがって、ASCII コード化ファイルを COBOL で処理するときには、特別に考慮する必要があります。

この付録は、情報処理交換用国際標準 646、7 ビット・コード化文字に定義されている ISO 7 ビット・コードの国際参照版 (ISCII) にも (適切な修正を加えた上で) 適用されます。ISCII コード・セットが、ASCII と異なるのは、次の 2 つのコード・ポイントにある図形表現だけです。

- 序数 37。これは ASCII ではドル記号 ($) になっていますが、ISCII ではひし形になっています。
- 序数 127。これは ASCII では波形記号 (~) になっていますが、ISCII では上線記号 (オプションにより波形記号ともなる) になっています。

注: 以下の説明では、STANDARD-1 についての説明内容は、特に断りがない限り、STANDARD-2 にも適用されます。

以下の段落では、ASCII (または ISCII) コード化ファイルに関して特に考慮すべき点について説明します。

環境部

環境部では、OBJECT-COMPUTER、SPECIAL-NAMES、FILE-CONTROL の各段落が影響を受けます。

OBJECT-COMPUTER 段落と SPECIAL-NAMES 段落

プログラム内でファイルが 1 つでも ASCII コード化ファイルである場合には、SPECIAL-NAMES 段落の英字名文節を指定しなければなりません。その英字名は、STANDARD-1 または STANDARD-2 と関連付ける必要があります (それぞれ、ASCII または ISCII の照合シーケンスやコード・セット)。

オブジェクト・プログラムの中の英数字比較で ASCII 照合シーケンスを使用するときには、OBJECT-COMPUTER 段落の PROGRAM COLLATING SEQUENCE 文節を指定しなければなりません。その英字名も、SPECIAL-NAMES 段落の英字名として指定し、STANDARD-1 と関連付ける必要があります。以下に例を示します。

© Copyright IBM Corp. 1991, 2001

```
Object-computer.  IBM-390
    Program collating sequence is ASCII-sequence.
Special-names.  Alphabet ASCII-sequence is standard-1.
```

両方の文節が指定されている場合、このプログラムでは、ASCII 照合シーケンスを使用して次のような英数字比較の真の値を判定します。

- 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。
- 任意の英数字ソート・キーまたはマージ・キー (MERGE ステートメントまたは SORT ステートメントの中で COLLATING SEQUENCE 句が指定されていない場合)。

PROGRAM COLLATING SEQUENCE 文節を省略した場合には、そのような比較を行うために EBCDIC 照合シーケンスが使用されます。

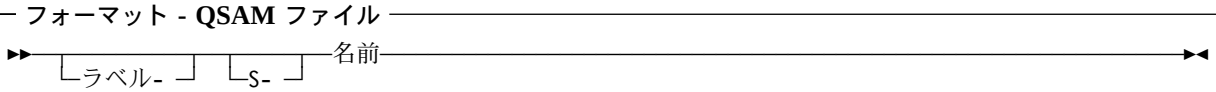
PROGRAM COLLATING SEQUENCE 文節を英字名文節と共に使用すれば、ASCII コード・テープ・ファイルについて EBCDIC による英数字比較を指定したり、EBCDIC コード・テープ・ファイルについて ASCII による英数字比較を指定したりできます。

英字名文節のリテラル・オプションを使用すれば、内部データを NATIVE または STANDARD-1 以外の照合シーケンスで処理することができます。

FILE-CONTROL 段落

ASCII ファイルの場合、ASSIGN 文節の割り当て名は次のフォーマットとなります。

フォーマット - QSAM ファイル



ファイルは、磁気テープ装置に割り当てられた QSAM ファイルでなければなりません。

ラベル-
ファイルの割り当て先の装置と装置クラスを記述します。このラベルを指定する場合、これはハイフンで終わらなければなりません。

S- 編成フィールド。QSAM ファイル用のオプションは、必ず順次編成です。

名前
このファイルの外部名を指定する 1 ～ 8 文字の必須フィールド。

I-O-CONTROL 段落

RERUN 文節の中の割り当て名には、ASCII コード化ファイルを指定することはできません。

チェックポイント・レコードが含まれている ASCII コード化ファイルを処理することはできません。

データ部

データ部では、FD 記入項目およびデータ記述記入項目に関して特別の考慮事項があります。

環境部の中で定義されているそれぞれの論理ファイルごとに、対応する FD 記入項目とレベル 01 のレコード記述項目がデータ部のファイル・セクションの中になければなりません。

FD 記入項目 - CODE-SET 文節

ASCII コード化ファイルの FD 記入項目には、CODE-SET 文節が入っていなければなりません。英字名は、SPECIAL-NAMES 段落の STANDARD-1 (ASCII コード・セットの場合) と関連付ける必要があります。以下に例を示します。

```
Special-names.  Alphabet ASCII-sequence is standard-1.
   .
   .
   .
FD  ASCII-file label records standard
  Recording mode is f
  Code-set is ASCII-sequence.
```

データ記述記入項目

ASCII ファイルの場合、次に示すようなデータ記述に関する考慮事項が適用されます。

- PICTURE 文節の指定は、以下のカテゴリーのデータについて有効です。

  英字

  英数字

  英数字編集

  数字

  数字編集

- 符号付き数字項目の場合、SEPARATE CHARACTER 句を指定した SIGN 文節を指定する必要があります。
- USAGE 文節では、DISPLAY 句だけが有効です。

手続き部

ASCII 照合シーケンスを使用するソート・マージ操作は、次のように 2 とおりの方法で指定することができます。

- OBJECT-COMPUTER 段落の中の PROGRAM COLLATING SEQUENCE 文節を使用する。

  この場合には、ASCII 照合シーケンスは、比較条件と条件名条件の中に明示的に指定された英数字比較で使用されます。

- SORT ステートメントまたは MERGE ステートメントで指定する COLLATING SEQUENCE 句を使用する。

  この場合には、このソート・マージ操作だけが、ASCII 照合シーケンスを使用します。

どちらの場合も、英字名は、SPECIAL-NAMES 段落の中の STANDARD-1 (ASCII 照合シーケンスを表す) に関連付ける必要があります。

このソート・マージ操作の場合、COLLATING SEQUENCE オプションは、PROGRAM COLLATING SEQUENCE 文節を優先します。

PROGRAM COLLATING SEQUENCE 文節と COLLATING SEQUENCE 句を両方とも省略した場合 (またはどちらか一方が有効で、それが EBCDIC 照合シーケンスを指定している場合)、ソート・マージ操作は EBCDIC 照合シーケンスを使用して行われます。

付録G. 業界仕様

Enterprise COBOL は、以下の業界標準をサポートします。

1. ISO 1989:1985、プログラミング言語 - COBOL。

   ISO 1989:1985 は、ANSI X3.23:1985 と同一です。

   ISO/IEC 1989/改訂1:1992、プログラミング言語 - COBOL - 改訂 1: 組み込み関数モジュール。

   ISO 1989/改訂1:1992 は、ANSI X3.23a:1985 と同一です。

   ISO/IEC 1989/改訂2:1994、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 1989/改訂2:1992 は、ANSI X3.23b:1985 と同一です。

   すべての必要なモジュールは、ANSI により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)

   標準の報告書作成プログラム・オプショナル・モジュールが、オプションのIBM COBOL 報告書作成プログラム・プリコンパイラーおよびライブラリー (5798-DYR) と共にサポートされています。

   以下に示すような ANSI 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)

2. 情報システム用米国標準規格 X3.23-1985、プログラミング言語 - COBOL

   情報システム用米国標準規格 X3.23a-1989、プログラミング言語 - COBOL 用組み込み関数モジュール。

   情報システム用米国標準規格 X3.23b-1993、プログラミング言語 - COBOL 用修正改訂。

   すべての必要なモジュールは、ANSI により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 ANSI 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)

3. FIPS 資料 21-4 (連邦情報処理規格 21-4) COBOL 高水準サブセット

© Copyright IBM Corp. 1991, 2001

4. *国際標準規格 646、情報処理交換用 7 ビット・コード化文字セット* に定義されている国際標準化機構 ISO 7 ビット・コードの国際参照版。
5. *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。

Enterprise COBOL には、業界標準に関連して次のような制約事項があります。

- OPEN EXTEND が ASCII エンコード・テープ (CODESET STANDARD-1 または STANDARD-2) に対してサポートされていない。
- 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。

上記の標準をサポートするために**必要な**コンパイラー・オプションおよび言語環境プログラム・ランタイム・オプションの指定については、*Enterprise COBOL プログラミング・ガイド* を参照してください。

付録H. 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あり、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 以外の製品と組み合わせた場合、その操作の評価と検証については、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。実施権の許諾については、下記の宛先に書面にてご照会ください。

〒106-0032 東京都港区六本木 3 丁目 2-31
IBM World Trade Asia Corporation
Intellectual Property Law & Licensing

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。

本書は定期的に見直され、必要な変更 (たとえば、技術的に不適切な表現や誤植など) 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。

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 IBM Enterprise COBOL for z/OS and OS/390 サービスを使用するためのプログラミング・インターフェースが記述されています。

商標

以下は、IBM Corporation の商標です。

Advanced Function Printing
AFP
AIX
DB2

© Copyright IBM Corp. 1991, 2001

CICS
DFSMS
DFSORT
IBM
IMS
IMS/ESA
Language Environment
MVS
OS/390
Print Services Facility
SOM
System/390
VisualAge
z/OS

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

UNIX は、The Open Group がライセンスしている米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標または登録商標です。

付録I. リソース・リスト

Enterprise COBOL for z/OS and OS/390

コンパイラーおよび実行時プログラム 移行ガイド、 GC88-9118

カスタマイズ・ガイド、 GC88-9119

Debug Tool User's Guide、SC27-1573

Debug Tool Reference Manual and Messages、SC27-1575

Diagnosis Guide、 GC26-9047

Fact Sheet、GC27-1407

言語解説書、SC88-9117

Licensed Program Specifications、GC27-1411

2000 年言語拡張の手引き、GD88-7092

プログラミング・ガイド、SC88-9121

関連資料

VisualAge COBOL

Fact Sheet、GC26-9052

ご使用になる前に、GD88-7042

言語解説書、SD88-7071

プログラミング・ガイド、SC88-8665

COBOL ビジュアル・ビルダー 使用者の手引き、SD88-7072

COBOL Set for AIX

Fact Sheet、GC26-8484

Getting Started、GC26-8425

言語解説書、SD88-7071

LPEX User's Guide and Reference、SC09-2202

Program Builder User's Guide、SC09-2201

プログラミングの手引き、SC88-7491

CICS Transaction Server

アプリケーション・プログラミング・ガイド、SC88-7689

アプリケーション・プログラミング・リファレンス、SC88-7690

カスタマイズ・ガイド、SC88-7686

外部インターフェース・ガイド、SD88-7026

© Copyright IBM Corp. 1991, 2001

z/OS C/C++

ランタイム・ライブラリー・リファレンス、SA88-8515

DB2 UDB (OS/390 および z/OS 版)

アプリケーション・プログラミングおよび SQL ガイド、SC88-8763

コマンド解説書、SC88-8764

SQL 解説書、SC88-8772

z/OS DFSMS

Access Method Services for Catalogs、SC26-7394

Checkpoint/Restart、SC26-7401

Macro Instructions for Data Sets、SC26-7408

Program Management、SC27-1130

Using Data Sets、SC26-7410

ユーティリティー、SC88-8979

DFSORT

アプリケーション・プログラミングの手引き、SC88-7061

導入およびカストマイズ、SC88-7163

IMS/ESA

Application Programming: Client Server Object Manager Client/Server Application Programming Guide and Reference、SC26-3483

Application Programming: Client Server Object Manager データストア・アプリケーション・プログラミングの手引きと解説書、SC88-7498

Application Programming: Database Manager Summary、SC26-8037

アプリケーション・プログラミング: データベース管理プログラム、SC88-7552

アプリケーション・プログラミング: 設計の手引き、SC88-7542

アプリケーション・プログラミング: EXEC DLI コマンド (CICS および IMS)、SC88-7554

アプリケーション・プログラミング: トランザクション管理プログラム、SD88-7121）

z/OS ISPF

ダイアログ開発者ガイドとリファレンス、SC34-8964

ユーザーズ・ガイド 第 1 巻、*SC 88-8965 および 第 2 巻*、SC88-8966

z/OS 言語環境プログラム

概念、SA88-8555

カスタマイズ、SA88-8552

デバッグのガイド、GA88-8548

ランタイム・メッセージ、SA88-8554

プログラミング・ガイド、SA88-8549

プログラミング・リファレンス、SA88-8550

ランタイム マイグレーション・ガイド、GA88-8553

ILC アプリケーションの作成、SA88-8551

z/OS MVS

JCL Reference、SA22-7597

JCL ユーザーズ・ガイド、SA88-8570

System Commands、SA22-7627

z/OS UNIX システム・サービス

UNIX システム・サービス コマンド解説書、SA88-8641

UNIX システム・サービス プログラミング: アッセンブラー呼び出し可能サービス 解説書、SA88-8642

UNIX システム・サービス ユーザーズ・ガイド、SA88-8640

z/OS TSO/E

コマンド解説書、SA88-8628

ユーザーズ・ガイド、SA88-8638

z/Architecture

Principles of Operation、SA22-7832

ソフトコピー資料 (z/OS 版)

次のコレクション・キットには、z/OS および関連製品の文献が含まれています。

IBM オンライン・ライブラリー MVS、*OS/390*、*z/OS コレクション・キット CD-ROM (12/2001)*、*SK88-8002-34*

Unicode および文字表現

OS/390 Unicode サポート: 変換サービスの使用法、SD88-6163、
publibfp.boulder.ibm.com/pubs/pdfs/os390/cunuge00.pdf (英文資料)

Program Directory for OS/390 Support for Unicode、GI10-9760、
publibfp.boulder.ibm.com/pubs/pdfs/os390/cunpde00.pdf (英文資料)

Character Data Representation Architecture Reference and Registry、SC09-2190

オンライン資料

Java 言語仕様 第 2 版 (Gosling 他著)、
http://software.nikkeibp.co.jp/software/books/books90.html

Java Native Interface、http:java.sun.com/j2se/1.3/ja/docs/ja/guide/jni/

Java on the OS/390 platform、www-1.ibm.com/servers/eserver/zseries/software/java/

Java 2 Enterprise Edition Developer's Guide、
java.sun.com/j2ee/j2sdkee/devguide1_2_1.pdf

Unicode、www.unicode.org

XML 仕様、www.w3c.org/XML/

用語集

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。

- 米国標準規格 (ANS) Programming Language COBOL, ANSI X3.23-1985 (著作権 1985 米国規格協会 (ANSI)) (ISO 1989: 1985)、X3.23a-1989 (ISO 1989/修正 1) および X3.23b-1993 (ISO1989/修正 2) によって改訂されました。
- 米国標準規格 (ANS) Dictionary for Information Systems ANSI X3.172-1990、著作権 1990 米国規格協会 (ANSI)。

American National Standards Institute, 11 West 42nd Street, New York, New York 10036 から、コピーを購入することができます。

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。

ア

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。

遊びバイト (slack bytes). 一部の数値項目の位置合わせが正しく行われるように、データ項目相互間またはレコード相互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。場合によっては、コンパイラーによって遊びバイトが挿入されるが、それ以外の場合、遊びバイトの挿入はプログラマーが行う。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 文節によって、コンパイラーに遊びバイトを挿入させる。レコード間遊びバイトは、プログラマーが挿入する。

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、その範囲を区切る区切り文字のピリオド。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。

インスタンス・データ (instance data). オブジェクト・インスタンスの状態を定義するデータ。インスタンス・データは、クラス定義の OBJECT 段落のワーキング・ストレージ・セクションで宣言される。*オブジェクト・インスタンス・データ* とも呼ばれる。各オブジェクト・インスタンスはインスタンス・データの独自のコピーを所有する。インスタンス・データは Java クラスの private 非静的メンバー・データと等価である。

インスタンス・メソッド (instance method). クラス定義の OBJECT 段落で定義されるメソッド。インスタンス・メソッドは、Java の public 非静的メソッドと等価である。

隠蔽 (メソッド) (hide (a method)). 親クラスで同じメソッド名により定義されたファクトリーまたは静的メソッドを (サブクラスで) 再定義すること。したがって、サブクラスのメソッドは親クラスのメソッドを*隠蔽する*。

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。

ウィンドウ化西暦年 (windowed year). 2 桁の年だけで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。たとえば、05 は 2005 と解釈できる。「*世紀ウィンドウ (century window)*」も参照。

「*拡張西暦年 (expanded year)*」と対比。

ウィンドウ化日付フィールド (windowed date field). ウィンドウ化 (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「*ウィンドウ化西暦年 (windowed year)*」も参照。

埋め込み文字 (* padding character). 物理レコードの中の未使用の文字位置を埋めるために使用される英数字。

英字 (* alphabetic character). 英字またはスペース文字。

英字 (* 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

英字名 (* alphabet-name). 環境部の SPECIAL-NAMES 段落の中で定義される、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ユーザー定義語。

英数字 (alphanumeric characte). コンピューターの有する 1 バイト文字セット中の任意の文字。

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). USAGE DISPLAY およびシンボル X を含む PICTURE 文字ストリングの両方で記述されるデータ項目。

エンクレーブ (enclave). 言語環境プログラム製品のもとで稼働するエンクレーブは、実行単位と同様である。これは、LINK によって、および C の system () 関数を使用することによって他のエンクレーブを作成できる。

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。

© Copyright IBM Corp. 1991, 2001

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。

オーバーライド (override). 親クラスから継承したインスタンス・メソッドを (サブクラスで) 再定義すること。

オープン・モード (* open mode). OPEN ステートメントを実行してから、REEL または UNIT 句を指定せずに CLOSE ステートメントを実行するまでの、ファイルの状態。特定のオープン・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。

オブジェクト (object). 状態 (そのデータ値) および演算 (そのメソッド) を持つエンティティー。オブジェクトは状態と動作をカプセル化する手段である。

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。この用語は、「実行時 (execution time)」と同義。

オブジェクト・インスタンス (object instance). 単一の、場合によっては多数のオブジェクト。COBOL クラス定義の Object 段落における指定に基づいてインスタンス生成される。オブジェクト・インスタンスは、そのクラス定義に記述されたデータおよびすべての継承データのコピーを保持する。オブジェクト・インスタンスに関連付けられたメソッドには、そのクラス定義で定義されたメソッドおよびすべての継承メソッドが含まれる。

オブジェクト・インスタンスは Java クラスのインスタンスにすることができる。

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。

オブジェクト・デック (object deck). リンケージ・エディターへの入力として適切なオブジェクト・プログラムの部分。この用語は「*オブジェクト・モジュール (object module)*」および「*テキスト・デック (text deck)*」と同義。

オブジェクト・プログラム (* object program). データと相互作用して問題解決を行うように設計された、実行可能なマシン言語命令および他のデータの集合あるいはグループ。このコンテキストでは、オブジェクト・プログラムは一般に、COBOL コンパイラーがソース・プログラムに操作した結果のマシン言語である。あいまいになるおそれがない場合には、「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」という語だけが使用される。

オブジェクト・モジュール (object module). 「*オブジェクト・デック (object deck)*」または「*テキスト・デック (text deck)*」と同義。

オブジェクト・リファレンス (object reference). オブジェクトを起動したり、オブジェクトを参照するために必要な情報を含めることができるデータ項目。オブジェクト・リファレンスは、COBOL ではデータ記述記入項目の USAGE 文節の OBJECT REFERENCE 句を使用して定義される。「*型式化オブジェクト・リファレンス (typed object reference)*」 および「*汎用オブジェクト・リファレンス (universal object reference)*」も参照。

オプショナル・ファイル (* optional file). オブジェクト・プログラムの実行ごとに、必ずしも存在する必要がないものとして宣言されるファイル。オブジェクト・プログラムが実行されると、そのファイルの存在の有無が調べられる。

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フォーマットに含められている予約語。そのようなワードが現れるフォーマットをソース・プログラムの中で使用する場合、そのワードの有無は、ユーザーのオプションにゆだねられる。

オペランド (operand). 操作の対象となるデータ。本書では、ステートメントや記入項目のフォーマット中に現れる小文字または日本語で書かれたワード (またはワード群) は、そのワード (またはワード群) が示すデータへの参照という点でオペランドである。

カ

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。

外部 10 進数項目 (external decimal item). ビット 4 ～ 7 に数字が含まれ、右端のバイトのビット 0 ～ 3 に符号が含まれる、数を表現する形式。他のすべてのバイトのビット 0 ～ 3 には 1 (16 進数の F) が入る。たとえば、10 進値 +123 は 1111 0001 1111 0010 1111 0011 として表される (「*ゾーン 10 進数項目 (zoned decimal item)*」とも呼ばれる)。

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。

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ているデータ。

外部データ項目 (* external data item). ある実行単位の 1 つまたは複数のプログラムの中で外部レコードの一部として記述され、それを記述しているどのプログラムからでも参照できるデータ項目。

外部データ・レコード (* external data record). ある実行単位の 1 つまたは複数のプログラムの中で記述されている論理レコードで、そのコンポーネントであるデータ項目がそれらを記述しているどのプログラムからでも参照できるもの。

外部ファイル結合子 (* external file connector). 実行単位内にある 1 つまたは複数のオブジェクト・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 item). 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、実数は、固定小数点部分 (最初の数表示) と、暗黙の浮動小数点の底を指数 (2 番目の数表示) で累乗することで得られる値との積である。

たとえば、0.0001234 という数の浮動小数点表示は 0.1234 -3 である。ここで、0.1234 は小数部を、-3 は指数部を表す。

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、数または数の表現を収めるために使用されるデータ項目。

拡張機能 (extensions). 「*IBM 拡張* (IBM extensions)」を参照。

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 (たとえば、1998)。「*ウィンドウ化西暦年 (windowed year)*」と対比。

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。

拡張モード (* extend mode). ファイルに対して EXTEND 句の指定のある OPEN ステートメントが実行された後から、そのファイルに対する REEL 句または 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。

型式化オブジェクト・リファレンス (typed object reference). 指定されたクラスまたはそのいずれかのサブクラスのみを参照できるオブジェクト・リファレンス・データ項目。

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・コード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を移行できる能力。

可変位置グループ (* variably-located group). 同じレベル-01 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。

可変位置項目 (* variably-located item). 同じレベル-01 レコード内の可変長テーブルに続くデータ項目。可変長テーブルに従属するわけではない。

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、繰り返される回数が可変であるテーブル・エレメント。そのような項目は、OCCURS DEPENDING ON 文節をそのデータ記述記入項目の中に持つか、またはこの文節を持つ項目に従属していなければならない。

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・ファイル記述記入項目が、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。

環境部 (Environment Division). COBOL ソース単位の部の 1 つ。環境部では、ソース・プログラムをコンパイルするためのコンピューターおよびオブジェクト・コードを実行するためのコンピューターを記述する。また、ファイルおよびそのレコードの論理的概念と、ファイルが保管される装置の物理的特徴との間の関係について記述する。

環境変数 (environment variable). オペレーティング・システムの稼働方法、およびオペレーティング・システムが認識する装置を記述する変数。

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、環境部において簡略名と関連付けられている場合、その簡略名は、置き換えが有効であれば、どのようなフォーマットにも置き換えることができる。

関係 (* relation). 「*比較演算子 (relational operator)*」または「*比較条件 (relation condition)*」を参照。

関数 ID (* function-identifier). 関数を参照する、文字ストリングと区切り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、関数名と引き数 (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。

関数 (* 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。

関数ポインター (function-pointer). FUNCTION-POINTER の使用によって記述される、プロシージャーまたは関数のアドレスを含むことができるデータ項目。

関数名 (function-name). 必要な引き数を伴って関数の値を決定する呼び出しを行うのためメカニズムに付けられる名前を表すワード。

簡略名 (* mnemonic-name). 環境部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。

キー (* key). レコードの位置を識別するデータ項目。またはデータの順番を識別するために使用されるデータ項目の集合。

キーワード (* keyword). 予約語または関数名で、そのワードの現れるフォーマットがソース・プログラムの中で使用されるときには必須である。

疑似テキスト (* pseudo-text). ソース・プログラムまたは COBOL ライブラリーの中にあって、その境界が疑似テキスト区切り文字によって区切られたテキスト・ワード、コメント行、または区切り文字のスペースから構成される列 (その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 (==)。

記入項目 (* entry). COBOL プログラムの見出し部、環境部、またはデータ部に置かれる一連の連続する記述文節。

記入項目のオブジェクト (* object of entry). COBOL プログラムのデータ部記入項目内にあり、記入項目のサブジェクトの直後に置かれるオペランドおよび予約語の集合。

記入項目のサブジェクト (* subject of entry). データ部の記入項目内において、レベル標識やレベル番号の直後に現れるオペランドまたは予約語。

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。

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。Enterprise COBOL では、基本文字セットは単一バイトの EBCDIC (拡張 2 進化 10 進コード) でインプリメントされる。拡張文字セットには DBCS (2 バイト文字セット) 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。

この用語は 標準 COBOL における「*COBOL 文字セット (COBOL character set)*」と同義。

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。

共通プログラム (* common program). 別のプログラムの中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の中に直接的または間接的に含まれているどのプログラムからでも呼び出すことができるプログラム。

キロバイト (KB) (kilobyte(KB)). 1 キロバイトは 1024 バイトに相当する。

句 (* phrase). 句は、COBOL プロシージャー・ステートメントまたは COBOL 文節の一部を形成する連続的な COBOL 文字ストリングを、1 つまたは複数個並べた集合。

空白文字 (white space characters). 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 (LF)
- 次の行

Unicode 標準では上記のように呼ばれる。

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。

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用される、後に 1 つのスペースが続く 1 つのコンマ (,)。

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用される、後に 1 つのスペースが続く 1 つのセミコロン (;)。

区切り文字のピリオド (* separator period). 文字ストリングを区切るために使用される、後に 1 つのスペースが続く 1 つのピリオド (.)。

句読記号 (* 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
b	スペース
=	等号

国別データ項目 (national data item). USAGE NATIONAL を使用して暗黙的または明示的に記述されるデータ項目。

国別文字 (national character). 国別データ項目または国別リテラルの文字。

国別文字位置 (national character position). 「*文字位置 (character position)*」を参照。

組み込み関数 (intrinsic function). COBOL 言語の一部として定義された関数。一部のプログラム言語では組み込み関数と呼ばれる。

クラス (オブジェクト指向) (class (object-oriented)). 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、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。

クラス定義 (class definition). クラスを定義する COBOL ソース単位。

クラス名 (オブジェクト指向)(class-name (object-oriented)). オブジェクト指向のクラス定義の名前。*クラス名* は、COBOL クラス名または Java クラス名を指す場合がある。

クラス名 (データの)(* class-name (of data)). SPECIAL-NAMES 段落の中で定義されたユーザー定義語であり、データ項目の内容がクラス名定義の中にリストされた文字だけからなり、真の値を定義できる命題を参照する。

グループ項目 (* group item). 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。

グローバル名 (* global name). 1 つのプログラムの中でだけ宣言されているが、そのプログラムからだけではなく、そのプログラムに含まれるプログラムからも参照できる名前。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。

継承 (* inheritance). クラス (*スーパークラス*) のインプリメンテーションを新規クラス (*サブクラス*) の基本として使用するメカニズム。各 *サブクラス* は厳密に 1 つのクラスから継承する。継承されたクラスはそれ自身がサブクラスになり、別のクラスから継承する。

注: Enterprise COBOL は多重継承をサポートしない。Enterprise COBOL は、単一継承を提供する Java オブジェクト・モデルをサポートする。

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。

桁 (* column). 行の中の文字位置。桁は、行の左端の文字位置から始めて行の右端の文字位置まで、1 から 1 ずつ増やして番号が付けられる。 参照形式では、列は 1 つの 1 バイト文字を保持する。

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記入項目に指定された用途によって異なる。

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。

現行コード (* 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。

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。

コード化文字セット (coded character set). 文字および制御機能の集合。特定のコード・ポイントに対する曖昧でない割り当て (エンコード) を伴う。EBCDIC はコード化文字セットの例である。エンコードの特定のインスタンスをコード・ページと呼ぶ。IBM によって指定されたコード・ページは、CCSID によって識別される。

コード・ページ (code page). コード化文字セット内のコード・ポイントに対して図形文字および制御機能の意味を割り当てたもの。たとえば、単一バイト EBCDIC (拡張 2 進化 10 進コード) の 256 のコード・ポイントに文字と意味を割り当てたもの。コード化文字セットとコード・ページは互換性がある。

コード・ポイント (code point). コード・ページに定義された固有のビット・パターン。グラフィック・シンボルおよび制御機能はコード・ポイントに割り当てられる。

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。

構成セクション (* Configuration section). 環境部の中にあるセクションであり、ソース・プログラムおよびオブジェクト・プログラムおよびメソッド定義の全般的な仕様を記述するもの。

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。

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。

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。

- **データ部** (Data Division)
  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 つのデータ・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化日付フィールドである (一方がウィンドウ化西暦年である DATE FORMAT YY だけから構成される)。
  - 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 だけから構成される)。
  - 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。

ウィンドウ化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、従属の方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。

  - 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** (Procedure Division)
  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。年号部分については、ウィンドウ化でも拡張することも可能である。たとえば、DATE FORMAT YYXXX のウィンドウ化日付フィールドは、以下と互換性がある。
  - DATE FORMAT YYXXX の別のウィンドウ化日付フィールド
  - DATE FORMAT YYYYXXX の拡張日付フィールド

固定小数点数 (fixed-point number). オプショナル符号の位置、含んでいる桁の数、およびオプショナル小数点の位置を指定する PICTURE 文節で定義された数値データ項目 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。

固定長レコード (* fixed-length record). ファイル記述項目またはソート・マージ記述記入項目が、すべてのレコードの文字位置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確立され、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。

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見出し部内の記入項目で、実行に影響しない。

コメント行 (* comment line). ソース・テキストを記述する行の標識域にアスタリスクを書き入れることによって表現されるソース・プログラムの行で、その行の領域 A と領域 B にコンピューターの有する文字セットから任意の文字を使用して表現される。コメント行は、プログラムの文書化にのみ役立つ。行の標識域にスラッシュを記入し、領域 A と領域 B にはコンピューターの有する文字セットから任意の文字を書き入れることによって表現した特別の形式のコメント行は、そのコメントを印刷する前に改ページが行われる。

固有照合シーケンス (* native collating sequence).
OBJECT-COMPUTER 段落の中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。

固有文字セット (* native character set).
OBJECT-COMPUTER 段落の中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。

コンパイラー指示ステートメント (compiler directing statement). COBOL ソース・テキストの処理時にコンパイラーによって取られるアクションを指定するステートメント。たとえば、COPY ステートメントと REPLACE ステートメント。

コンパイル時 (* compile time). COBOL ソース・コードが COBOL コンパイラーによって COBOL オブジェクト・モジュールに変換される時点。

サ

再帰 (recursion). それ自体を呼び出すプログラム、または、自分で呼び出したプログラムによって直接あるいは間接に呼び出されるプログラム。

最高位の桁 (* high order end). 文字ストリングの左端の文字。

最後に使われた状態 (last-used state). 内部値が、プログラムまたはメソッドが終了したときと同じままである (再入時に初期値にリセットされない) ストレージの状態。

最低位の桁 (* low order end). 文字ストリングの右端の文字。

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 つのコピーをコピーのユーザーが共用できる。

索引付きファイル (* indexed file). 指標付き編成のファイル。

サブクラス (subclass). 別のクラスから継承するクラス。継承関係にある 2 つのクラスをまとめて考える場合、サブクラスが継承する側のクラスであり、*スーパークラス* が継承される側のクラスである。

また、サブクラスは子クラスまたは派生クラスとも呼ばれる。

サブプログラム (* subprogram). 任意の呼び出されるプログラム。

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き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
**	累乗

算術演算符号 (* operational sign). 値が正であるか負であるかを示すために数字データ項目または数字リテラルに付けられる代数符号。

算術式 (* arithmetic expression). 数字基本項目の ID、数字リテラルなど、算術演算子により分けられた ID とリテラル、1 つの算術演算子により分けられた 2 つの算術式、または括弧で囲まれた算術式。

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。

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。

参照形式 (* reference format). COBOL ソース・コードを記述する際に、標準的な方式を提供するフォーマット。

参照修飾子 (* reference-modifier). 固有のデータ項目を定義する、文字ストリングと区切り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。

参照変更 (reference modification). 新しいデータ項目を定義する方法。左端の文字位置と別のデータ項目の左端の文字位置までの相対的な長さを指定することによって行う。

式 (* expression). 算術式または条件式。

シグニチャー (signature). メソッドの名前およびその仮パラメーターの数と型。

指数 (exponent). 別の数 (底) を乗ずる指数を示す数。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 において、指数表現は、指数の前に記号 '**' を付けて示す。

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。

実行時 (execution time). 「*実行時 (run time)*」を参照。

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。

実行時環境 (execution-time environment). 「*ランタイム環境 (run-time environment)*」を参照。

実行単位 (* run unit). スタンドアロンのオブジェクト・プログラム。あるいは、実行時にエンティティーとして機能する COBOL CALL ステートメントによって相互に作用するいくつかのオブジェクト・プログラム。

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。

失敗した実行 (* 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。

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。

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。

指標データ名 (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。

指標名 (* 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。

修飾子 (qualifier).

1. データ名またはレベル標識と関連付けられた名前。修飾子に従属する項目の名前である別のデータ名と共に、あるいは条件名と共に、参照する際に使用される。
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。

修飾されたデータ名 (* qualified data-name). データ名の修飾子を連結語の OF または IN を使って 1 組または数組を従えているデータ名から構成された ID。

出力ファイル (* output file). OUTPUT モードまたは EXTEND モードのいずれかでオープンされるファイル。

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の処理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してから制御が渡されるステートメントの集合。

出力モード (* output mode). ファイルに対して OUTPUT 句または EXTEND 句の指定のある OPEN ステートメントが実行された後から、そのファイルに対する REEL 句または 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。

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。

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。用語 'condition' (condition-1, condition-2,...) が一般フォーマットの '条件' (条件-1、条件-2、...) の中またはそれに関連した言語仕様に表示される場合、これは任意で括弧で囲まれる単純条件か、または単一条件、論理演算子、および括弧を構文的に正確に結合した複合条件かのいずれか一方からなる条件式。これにより真の値が判別できる。

条件 (例外) (condition (exception)). 言語環境プログラムによって使用可能にされる、あるいは認識される例外。したがって、ユーザー条件処理ルーチンと言語条件処理ルーチンの活動化に適している。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。条件は、言語固有生成コードや言語ライブラリー・コードによって検出されることもある。

条件句 (* conditional phrase). 条件句は、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の値の判定に基づいてとられるべき処置を指定する。

条件式 (* conditional expression). EVALUATE、IF、PERFORM、または SEARCH のステートメントで指定される、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)* 」を参照)。

条件ステートメント (* conditional statement). 条件の真の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の値によって決定され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). 条件変数がとることの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たユーザー定義語。「条件名」が一般フォーマットの中で使用されるときは、これは、参照の固有性を保つために必要になる修飾子や添え字と「条件名」とが構文的に正しく組み合わされたものからなる固有のデータ項目参照を表す。

条件名条件 (* condition-name condition). 条件変数の値が、条件変数に関連する条件名に帰する値の集合のうちの 1 つであるかどうかという命題であり、この命題に関して真の値を判別できる。

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。

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。

省略による簡易複合比較条件 (* 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。

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。

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。

シリアル検索 (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。

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。

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。

スーパークラス (superclass). 別のクラスによって継承されるクラス。継承関係にある 2 つのクラスをまとめて考える場合、サブクラスが継承する側のクラスであり、*スーパークラス* が継承される側のクラスである。

スーパークラスは親クラスとも呼ばれる。

スイッチ状況条件 (switch-status condition). 「オン」または「オフ」状況に設定できる UPSI スイッチが特定の状況に設定されているかという命題。この命題に対して真の値を判別できる。

数字 (digit). 0 ～ 9 の任意の数字。COBOL では、この用語は他の何らかの記号を指すために使用されることはない。

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。

数字項目 (* numeric item). データ項目。記述の内容は、'0' ～'9' の数字によって表される値に限定される。符号付きの場合は、'+'、'-'、または他の演算符号を項目に入れることもできる。

数字編集項目 (numeric-edited item). 印刷出力に使用できる形式になっている英数字項目。外部 10 進数字の 0 ～ 9 の数字、小数点、コンマ、ドル符号、編集符号制御文字、その他の編集記号から構成される。

数字リテラル (* numeric literal). 1 つまたは複数の数字からなるリテラル。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。

ステートメント (* statement). 実行する 1 つまたは複数のアクションを指定する COBOL 言語構成体。ステートメントには、プロシージャー・ステートメントとコンパイラー指示ステートメントがある。プロシージャー・ステートメントの例としては ADD ステートメント、コンパイラー指示ステートメントの例としては USE ステートメントがある。

世紀ウィンドウ (century Windows). 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 プログラマーが使用できる世紀ウィンドウには、次のものがある。

1. ウィンドウ化日付フィールドの場合は、YEARWINDOW コンパイラー・オプションによって指定される。
2. ウィンドウ化組み込み関数 DATE-TO-YYYYMMDD、DAY-TO-YYYYDDD、および YEAR-TO-YYYY の場合は、引き数-2 によって指定される。
3. 言語環境プログラム呼び出し可能サービスの場合は、CEESCEN で指定される。

整数 (* integer). (1) 小数点の右側に桁位置がない数値リテラル。
(2) データ部に定義されている数字データ項目であり、小数点の右側に桁位置を持たないもの。
(3) 関数の計算で戻される値の、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。

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。

セクション名 (* section-name). 手続き部のセクションの名前を表すユーザー定義語。

セクション・ヘッダー (* section header). セクションの開始を示すために組み合わされた語で、後に区切りピリオドを伴う。たとえば、WORKING-STORAGE SECTION がある。

宣言部分 (* declaratives). 手続き部の始めに書かれる特殊な目的を有する 1 つまたは複数のセクションの集合。最初のセクションの前にはキーワード DECLARATIVES と書き込み、最後のセクションの後にはキーワード END DECLARATIVES を記述する。1 つの宣言は、セクション・ヘッダーと次に続く USE コンパイラー指示文と、次に続く関連する段落から構成される。この段落は、なくてもよいし、1 個でも複数個でもよい。

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。プログラムまたはクラス定義。この用語は、「*コンパイル単位 (compilation unit)*」と同義。

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合。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・ファイル記述記入項目 (* sort-merge file description entry). データ部のファイル・セクションの中の記入項目であり、レベル標識 SD、それに続くファイル名、およびその後に付ける、ソート・マージ・ファイルの属性を記述する文節で構成される。

ゾーン 10 進数項目 (zoned decimal item). 「*外部 10 進数項目 (external decimal item)*」を参照。

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。

相対ファイル (* relative file). 相対編成のファイル。

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。

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。想定小数点は、論理的な意味を有するが物理的な表示は行われない。

添え字 (* subscript). 整数、データ名 (およびその後にオプションで続く演算子 + または - のある整数)、または索引名 (およびその後にオプションで続く演算子 + または - のある整数) のいずれかで表現されるオカレンス番号であり、テーブルの特定エレメントを識別する。引き数の個数が可変の関数の引き数として添え字付きの ID を使用する場合は、添え字にワード 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。

タ

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。

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). 大記憶容量を持った装置。磁気ディスクや磁気ドラムなど。

大容量記憶ファイル (* mass storage file). 大容量記憶メディアに割り当てられたレコードの集合。

多重定義 (overload). 同じクラス内で使用可能な別のメソッドと同じ名前、異なるシグニチャーを使用してメソッドを定義すること。「*シグニチャー (signature)*」も参照。

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。

単純条件 (* simple condition). 以下の集合から選択された単一の条件。

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

単純否定条件 (* negated simple condition). 論理演算子 'NOT' とそのすぐ後にある単純条件。

段落 (* paragraph). 手続き部では、段落名とその後の区切りピリオド、およびゼロ、1 つ、または複数の文。見出し部および環境部では、段落ヘッダーとそれに続く記入項目のこと。記入項目は省略することも 1 つまたは複数であることもある。

段落名 (* paragraph-name). 手続き部の中の段落を識別し開始するユーザー定義語。

チェックポイント (checkpoint). 後でジョブ・ステップを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。

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。

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。

通貨記号 (currency symbol). 数字編集項目内の*通貨符号値* の部分を示す PICTURE 文節で使用される文字。通貨記号は CURRENCY コンパイラー・オプションで定義したり、環境部の SPECIAL-NAMES 段落中の CURRENCY SIGN 文節で定義したりできる。CURRENCY SIGN 文節が指定されない場合に NOCURRENCY コンパイラー・オプションが有効な場合、デフォルトの通貨符号値および通貨記号としてドル符号 ($) が使用される。通貨記号と通貨符号値は複数定義可能。「*通貨符号値 (currency sign value)*」も参照。

通貨符号値 (currency sign value). 数字編集項目に格納されている通貨単位を示す文字ストリング。典型的な例としては、'$'、'USD'、'EUR' などがある。通貨符号値は CURRENCY コンパイラー・オプションで定義したり、環境部の SPECIAL-NAMES 段落中の CURRENCY SIGN 文節で定義したりできる。CURRENCY SIGN 文節が指定されておらず、NOCURRENCY コンパイラー・オプションが有効な場合は、ドル記号 ($) がデフォルトの通貨符号値として使用される。「*通貨記号 (currency symbol)*」も参照。

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。

次の実行可能な文 (* 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。

データ記述記入項目 (* data description entry). データ部の記入項目であり、その構成は、レベル番号、データ名 (必要な場合)、データ項目またはレコードの属性を記述する文節の集合 (必要な場合) が置かれる。

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)。

データ部 (Data Division). 実行時に処理されるデータおよびファイルを記述する COBOL 部。

データ文節 (* data clause). データ部のデータ記述記入項目の中に置かれ、データ項目の特定の属性を記述している情報を提供する文節。

データ名 (* data-name). データ記述記入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るときは、「データ名」は、形式の規則で特に許される場合を除き、参照変更、添え字付け、または修飾を行ってはならない語を表す。

テーブル (* table). データ部の中で OCCURS 文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・デック (text deck). 「*オブジェクト・デック (object deck)*」または「*オブジェクト・モジュール (object module)*」の同義語。

テキスト・ワード (* text word). COBOL ライブラリー、COBOL ソース・プログラム、または疑似テキストの中の A マージンと R マージンの間の 1 文字または連続する文字の並び。テキスト・ワードには以下のものがある。

- スペース以外の区切り文字、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。右括弧と左括弧の文字は、それがライブラリー、ソース・プログラム、または疑似テキストのいずれの中にあるかに関係なく、常にテキスト・ワードとみなされる。
- 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。
- コメント行と区切り文字によって限定されたワード 'COPY' を除く、区切り文字でもリテラルでもないその他すべての連続する COBOL 文字の並び。

手続き部 (Procedure Division). 実行時に操作を実行するためのプロシージャー・ステートメントを含むプログラムまたはメソッドの部。

デバッグ行 (* debugging line). 行の標識域に 'D' を書き入れた任意の行。

デバッグ・セクション (* debugging section). USE FOR DEBUGGING ステートメントを含むセクション。

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。

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
/	スラッシュ
=	等号
$	通貨符号
,	コンマ (小数点)
;	セミコロン
.	ピリオド (小数点、終止符)
"	引用符
(	左括弧
)	右括弧
>	不等号 (より大きい)
<	不等号 (より小さい)
:	コロン

特殊文字ワード (* special-character word). 算術演算子または比較文字である予約語。

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。

独立項目 (* noncontiguous items). ワーキング・ストレージ・セクションとリンケージ・セクションの中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。

トレーラー・ラベル (trailer-label). (1) 記録メディアの装置上のデータ・レコードに続くファイル・ラベルまたはデータ・セット・ラベル。(2) ファイル終了ラベルと同義語。

ナ

内部 10 進項目 (internal decimal item). フィールドの各バイト (右端バイトを除く) が 2 桁の数字を表すフォーマット。右端バイトには 1 桁の数字と符号が入る。たとえば、10 進値 +123 は 0001 0010 0011 1111 として表される (パック 10 進数とも呼ばれる)。

内部データ (* internal data). プログラムに記述されるデータであり、外部データ項目および外部ファイル結合子はすべて除く。プログラムのリンケージ・セクションに記述されている項目は、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 つのプログラムに記述されているデータ項目。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。

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。

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、そのファイル用のファイル制御記入項目にある FILE STATUS 文節を使用することによりプログラムで使用できる。

入出力ステートメント (* input-output statement). 個々のレコードに、または 1 単位としてのファイルに操作を実行することにより、ファイルが処理されるようにするステートメント。入出力ステートメントは、ACCEPT (ID 句指定)、CLOSE、DELETE、DISPLAY、OPEN、READ、REWRITE、SET (TO ON または TO OFF 句指定)、START、WRITE。

入出力セクション (* Input-Output section). 環境部のセクションであり、プログラムまたはメソッドが必要とするファイル

と外部メディアを指定し、プログラムまたはメソッドの実行時におけるデータの転送および処理に必要な情報を提供する。

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。

入出力モード (* I-O-Mode). ファイルに対して I-O 句を指定して OPEN ステートメントを実行した後から、そのファイルに対して REEL 句や UNIT 句を指定せずに CLOSE ステートメントを実行する前までのそのファイルの状態。

入力ファイル (* input file). INPUT モードでオープンされるファイル。

入力プロシージャー (* input procedure). SORT ステートメントの実行中に、ソート対象に指定されたレコードの RELEASE を制御するために制御を与えられる一連のステートメント。

入力モード (* input mode). ファイルに対して INPUT 句を指定して OPEN ステートメントを実行した後から、そのファイルに対して REEL 句や UNIT 句を指定せずに CLOSE ステートメントを実行する前までのそのファイルの状態。

ヌル (null). ポインター・データ項目に無効なアドレス値を割り当てるために使用される形象定数。NULL が使えるところならばどこでも NULLS が使用できる。

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。

ハ

廃止される言語エレメント (* obsolete element). COBOL 85 の COBOL 言語エレメントのうち、標準 COBOL の次回の改訂版から削除される予定のもの。

バイト (byte). いくつかのビット (通常は 8 ビット) で構成されるビット・ストリングで、1 つの単位として扱われ、1 つの文字を表現する。

配列 (array). 言語環境プログラムでは、データ・オブジェクトからなる集合体。これを構成するそれぞれのデータ・オブジェクトは、添え字付けにより固有のものとして参照できる。広い意味では、COBOL テーブルと類似している。

パスワード (password). プログラム、コンピューターのオペレーター、またはユーザーが、データにアクセスする前にセキュリティー上の要件を満たすために与えなければならない固有な文字ストリング。

パック 10 進数 (packed decimal item). 「*内部 10 進項目 (internal decimal item)*」を参照。

パッケージ (package). Java では、個別にまたは総括してインポートできる関連したクラスのグループ。

バッファー (buffer). 入力データや出力データを一時的に保管しておくストレージ部分。

パラメーター (parameter). パラメーターは、呼び出しプログラムと呼び出されるプログラムの間でデータの値を渡すために使用される。

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。

範囲終了符号 (scope terminator). 手続き部の特定のステートメントの終わりのマークを付ける COBOL 予約語。これは明示的なもの (END-ADD など)であることもあれば、暗黙のもの (区切りピリオドなど) であることもある。

汎用オブジェクト・リファレンス (universal object reference). 任意のクラスのオブジェクトへの参照を含むことができるオブジェクト・リファレンス・データ項目。

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。

演算子	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

比較条件 (* relation condition). ある算術式、データ項目、英数字リテラル、または指標名の値が、他の算術式、データ項目、英数字リテラル、または指標名の値と特定の関係を持つかどうかという命題で、それに関して真の値が判別され得る (「*比較演算子 (relational operator)*」も参照)。

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
=	等しい

引き数 (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 または INVOKE ステートメントの USING 句のオペランドで、呼び出されるプログラムまたは呼び出されたメソッドに値を渡すために使用される。

ビッグ・エンディアン (big-endian). メインフレームおよび AIX ワークステーションが バイナリー・データ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、最小重み数字が最高位アドレスにある。

日付フィールド (date field). 次のうちのいずれか。

- データ記述記入項目が DATE FORMAT 文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、および DAY YYYYDDD。
- 特定の算術演算の結果 (詳細については、232ページの『日付フィールドでの算術演算』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 (expanded date filed)*」と「*ウィンドウ化日付フィールド (windowed date filed)*」の両方を指す (「*非日付データ (non-date)*」も参照。)

日付フォーマット (date format). 日付フィールドの日付パターン。次のうちいずれかの方法で指定される。

- DATE FORMAT 文節または DATEVAL 組み込み関数引き数-2 によって明示的に指定される。

または

- 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 (詳細については、「*非日付データ (nondate)*」を参照)。

非日付データ (nondate). 次のいずれか。

- 日付記述記入項目が DATE FORMAT 文節を含んでいないデータ項目。
- 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。たとえば、2 つの互換日付フィールドの差。

標準 COBOL (* Standard COBOL). 577ページの付録 G、『業界仕様』に示された ANSI および ISO 標準によって定義された COBOL 言語。

標準データ・フォーマット (* standard data format). COBOL プログラムのデータ部においてデータの特性を記述する際に使用される概念。この概念の下では、データの特性は、そのデータがコンピューターの中または特定の外部メディア上に内部的に保管される方式に合わせた形式ではなく、無限の長さと幅を持つ印刷ページ上のデータの外観に合わせた形式で表現される。

ファイル (* file). 論理レコードの集合。

ファイル位置標識 (*file position indicator). 索引付きファイルの場合は、その参照キー内の現行キーの値が、順次ファイルの場合は、現行レコードのレコード番号が、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収められている概念上のエンティティー。あるいは、次の論理レコードが存在しないこと、オプショナル入力ファイルが存在しないこと、AT END 条件がすでに存在すること、または有効な次のレコードが存在しないことを示すための概念上のエンティティー。

ファイル記述項目 (* file description entry). データ部のファイル・セクションの中の記入項目であり、レベル標識 FD、それに続くファイル名、およびその後に付けるファイルの属性を含む文節の集合で構成される。

ファイル結合子 (* file connector). ファイルに関する情報が収められ、ファイル名と物理ファイルをリンクするものとして、またファイル名とその関連レコード域をリンクするものとして使用されるストレージ域。

ファイル制御記入項目 (* file control entry). あるファイルの関連物理属性を宣言する SELECT 文節とそれに従属するすべての文節。

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。

ファイル名 (* file-name). データ部のファイル・セクションの中のファイル記述項目またはソート・マージ・ファイル記述記入項目で記述されるファイル結合子に付ける名前を表すユーザー定義語。

ファイル・コピー (* copy file). 一連のコードを含むファイルまたはライブラリー・メンバーであり、COPY ステートメント

によってコンパイル時にソース・テキストに組み込まれるもの。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。

ファイル・システム (file system). ディスケットやミニディスクなど、物理大容量記憶装置または論理大容量記憶装置上の、ファイルおよびファイル管理構造体の集合体。

ファイル・セクション (* File Section). ファイル記述項目とソート・マージ・ファイル記述記入項目を、それらに関連するレコード記述と共に含むデータ部のセクション。

ファクトリー・データ (factory data). ファクトリー・オブジェクトのデータ。ファクトリー・データは 1 つのクラスにつき 1 回割り振られ、そのクラスのすべてのインスタンスによって共用される。ファクトリー・データは、クラス定義の FACTORY 段落内のワーキング・ストレージ・セクションで宣言される。ファクトリー・データは Java の private 静的データと等価である。

ファクトリー・メソッド (factory method). どのオブジェクト・インスタンスとも無関係にクラスによってサポートされるメソッド。ファクトリー・メソッドはクラス定義の FACTORY 段落で定義され、Java の public 静的メソッドと等価である。これらは通常オブジェクト生成のカスタマイズに使用される。

フォーマット (* format). データの集合の特定の配列。

複合 ODO (complex ODO). 次のような OCCURS DEPENDING ON 文節の特定の形式。

- 可変位置項目またはグループ。DEPENDING ON 句を持つ OCCURS 文節を使用して記述されたデータ項目。後ろに非従属データ項目またはグループが付く。
- 可変位置テーブル。DEPENDING ON 句を持つ OCCURS 文節を使用して記述されたデータ項目。後ろに OCCURS 文節を使用して記述された非従属データ項目が付く。
- 可変長エレメントを持つテーブル。OCCURS 文節を使用して記述されたデータ項目で、この場合、従属データ項目は DEPENDING ON 句を持つ OCCURS 文節を使用して記述される。
- 可変長エレメントを持つテーブルの索引名。
- 可変長エレメントを持つテーブルのエレメント。

複合条件 (* combined condition). AND または OR の論理演算子を使って 2 つまたはそれ以上の条件を結び合わせた結果作られる条件。

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。(「*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。)

複合否定条件 (* negated combined condition). 論理演算子 'NOT' とそのすぐ後にある括弧で囲まれた複合条件。

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、それに関して真の値が判別できる。

物理レコード (* physical record). 「*ブロック (block)*」を参照。

浮動小数点数 (floating-point number). 小数部と指数を含んでいる数値データ項目。その値は、指数によって指定されただけ累乗したその数字データ項目の基数に小数部を乗算することにより得る。

部の見出し (* division header). ワードとその後に続く、部の先頭を示す区切り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.
```

プログラム名 (* program-name). 見出し部と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。

プロシージャー (* procedure). 手続き部内にある 1 つの段落または論理的につながりのある段落のグループ、あるいは 1 つのセクションまたは論理的につながりのあるセクションのグループ。

プロシージャー名 (* procedure-name). 手続き部の段落またはセクションを指名す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・ブランチ・ステートメント (* procedure branching statement). ステートメントがソース・プログラムに書かれている順序で次の実行可能なステートメントではないステートメントに制御を明示的に移させるステートメント。プロシージャー・ブランチ・ステートメントは、ALTER、CALL、EXIT、EXIT PROGRAM、GO TO、MERGE (OUTPUT PROCEDURE 句指定)。PERFORM、SORT (INPUT PROCEDURE または OUTPUT PROCEDURE 句指定)。および XML PARSE。

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 文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。

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)」と同義語。

文 (* sentence). 1 つのステートメントまたは複数のステートメントの並びで、その最後のものは、区切り文字のピリオドで終了する。

文節 (* 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 COBOL 文字ストリング。

ページ (page). 出力データの物理的分割を表している縦方向のデータの分割。この分割は、内部論理上の要件または外部出力メディアの特性 (またはその両方) に基づいて行われる。

米国規格協会 (American National Standards Institute: ANSI). 生産者、消費者、および一般利害関係グループから構成された組織体で、それが確立した各種プロシージャーに基づき、その承認を得た種々の組織体が米国内において自発的な業界標準を作成して維持している。

別個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。

ヘッダー・ラベル (header label). (1) 記録メディア装置上のデータ・レコードの前に付いているファイル・ラベルまたはデータ・セット・ラベル。(2) ファイル開始ラベルと同義語。

編集解除 (* de-edit). 数字編集データ項目からすべての編集文字を論理的に取り除き、その項目の未編集数字を判別すること。

編集済みデータ項目 (edited data item). 0 の抑止または編集用文字の挿入 (またはその両方の処理) により修正を受けたデータ項目。

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
b	スペース
0	0
+	正符号
-	負符号
CR	貸方
DB	借方
Z	0 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)

変数 (* variable). 実行時にアプリケーションによって値が変更される可能性のあるデータ項目。

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。USAGE IS POINTER 文節を使用すれば、データ項目は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。

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。

ボリューム切り替えプロシージャー (volume switch procedures). ファイル終わり (EOF) に達する前にユニットまたはリールの終わりに達したとき、自動的に実行されるシステム・プロシージャー。

マ

マージ・ファイル (* merge file). MERGE ステートメントでマージされるレコードの集合。このファイルは、マージ機能を使ってのみ作成することと使用することができる。

見出し部 (Identification Division). COBOL プログラム、クラス定義、またはメソッド定義の 4 つの主コンポーネントの 1 つ。見出し部では、プログラム、クラス、またはメソッドを指定する。見出し部には、作成者名、インストール・システム、または、日付を含めることができる。

無効キー条件 (* invalid key condition). プログラムの実行時に、索引付きファイルまたは相対ファイルに関連する特定のキーの値が無効であると判定されたときに起こる条件。

明示的範囲終了符号 (* explicit scope terminator). 個々の手続き部ステートメントの範囲を終わらせる予約語。たとえば、END-READ。

命令ステートメント (imperative statement). 行うべき無条件処置を指定するステートメント、または明示的範囲終了符号 (範囲区切りステートメント) で区切られた条件ステートメント。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。

メインプログラム (main program). プログラムとサブルーチンの階層内において、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。

メガバイト (M) (* megabyte (M)). 1 メガバイトは 1,048,576 バイトに相当する。

メソッド (method). オブジェクトによってサポートされる操作の 1 つを定義するプロシージャー・コード。メソッド・プロシージャー・コードは COBOL INVOKE ステートメントによって特定のオブジェクト・インスタンスに対して実行される。メソッドは Java 起動式によって起動することができる。メソッドは *ファクトリー・メソッド* または *インスタンス・メソッド* のいずれかにすることができる。

メソッドの隠蔽 (method hiding). 「*隠蔽 (hide)* 」を参照。

メソッドのオーバーライド (method overriding). 「*オーバーライド (override)*」を参照。

メソッドの起動 (method invocation). (1) メソッドを起動する行為。
(2) メソッドを起動するために使用されるプログラム言語構文 (COBOL では INVOKE ステートメント、Java では メソッド起動式)。

メソッドの多重定義 (method overloading). 「*多重定義 (overload)*」を参照。

メソッド名 (* method-name). メソッドを識別する名前。METHOD-ID 段落では英数字または国別リテラルの内容として指定され、INVOKE ステートメントでは英数字リテラル、国別リテラル、英数字データ項目、または国別データ項目として指定される。

文字 (* character). 言語のそれ以上分割できない基本単位。

文字位置 (character position). 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を指す概念的な用語。この用語はどのような文字にも適用される。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。

- *英数字位置* は、USAGE DISPLAY で表現される文字を指す。
- *DBCS 文字位置* は、USAGE DISPLAY-1 で表現される DBCS 文字を指す。
- *国別文字位置* は、USAGE NATIONAL で表現される文字を指す。UTF-16 の文字エンコード・ユニットと同義。

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。

USAGE NATIONAL の場合は、1 つの文字エンコード・ユニットは Unicode 文字セットの UTF-16 形式における 1 つの 2 バイト・コード・ポイントに相当し、*文字エンコード・ユニット* は「*国別文字 (national character)*」および「*国別文字位置 (national character position)*」と同義である。

USAGE DISPLAY の場合は、1 つの文字エンコード・ユニットは 1 バイトに相当する。

USAGE DISPLAY-1 の場合は、1 つの文字エンコード・ユニットは DBCS 文字セットにおける 1 つの 2 バイト・コード・ポイントに相当する。

文字ストリング (* character string). COBOL 語、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隣接する文字。区切り文字により区切る必要がある。

文字セット (character set). 「*基本文字セット (basic character set)* 」および「*コード化文字セット (basic character set)* 」を参照。

ヤ

ユーザー定義語 (* user-defined word). 文節やステートメントのフォーマット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。

優先順位番号 (* priority-number). セグメント化の目的で、手続き部のセクションを分類するユーザー定義語。セグメント番号は、'0'、'1'、...、'9' の文字だけを含むことができる。セグメント番号は、1 桁でも 2 桁でもどちらでも表現できる。

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。

呼び出し可能サービス (callable services). 言語環境プログラムにおいて、通常の言語環境プログラム定義の呼び出しインターフェースを使用して COBOL プログラムから呼び出すことができるサービスの集合で、言語環境プログラムの規約を順守するすべてのプログラムから使用することができる。

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し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。

ラ

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。任意のソース・プログラムをコンパイルするに当たりコンパイラーによって使用される。

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。

ランタイム環境 (run-time environment). COBOL プログラムが実行される環境。

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。

リール (reel). 1 つのファイルの 1 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ストレージ・メディアの独立部分。「ユニット (unit)」および「ボリューム (volume)」と同義。

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。

リテラル (literal). ストリングを構成するために並べられた文字によって、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。

リンケージ・セクション (Linkage section). 呼び出されるプログラムのデータ部の中にあるセクション。ここには呼び出しプログラムから使用可能なデータ項目が記述されている。呼び出しプログラムも呼び出されるプログラムも、共にこれらのデータ項目を参照できる。

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。言語環境プログラムでは、プロシージャー、関数、またはサブルーチンのいずれかを指す。

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。

レコード (* record). 「*論理レコード (logical record)*」を参照。

レコード域 (* record area). データ部のファイル・セクション (File Section) にあるレコード記述項目で記述されたレコードを処理する目的で割り振られたストレージ域。ファイル・セクションでは、レコード域にある現在の文字位置は、明示的もしくは暗黙の RECORD 文節によって判別される。

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記入項目全体。「レコード記述 (record description)」と同義。

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。

レコード名 (* record-name). COBOL プログラムのデータ部にあるレコード記述項目で記述されたレコードを指定するユーザー定義語。

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・モード (recording mode). ファイル内の論理レコードの形式。レコード・モードは、F (固定長)、V (可変長)、S (スパン)、または U (不定フォーマット) とすることができる。

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 ～ 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 ～ 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、データ記述記入項目の特性を識別する。

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。データ部のレベル標識は CD、FD、SD である。

連続項目 (* contiguous items). データ部の中で連続する記入項目により記述され、相互に一定の階層関係を持つ項目。

ローカル・ストレージ・セクション (Local-storage section). データ部のセクションであり、ストレージを定義する。ここで定義されるストレージは、その VALUE 文節に割り当てられている値に基づいて、呼び出しのたびに割り振られたり解放されたりする。

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。

ワ

ワーキング・ストレージ・セクション (* Working-storage section). データ部にあるセクションで、独立項目またはワーキング・ストレージ・レコードあるいはその両方から構成されたワーキング・ストレージ・データ項目を記述する。

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文字ストリング。

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。

英数字

1 バイト文字セット (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。「EBCDIC (拡張 2 進化 10 進コード) (Extended Binary-Coded Decimal Interchange Code)」も参照。

2 進数項目 (binary item). 2 進表記 (基数 2 の表記体系) で表される数値データ項目。バイナリー項目には、0 ～ 9 までの 10 進数字で構成される 10 進数と同等の数と演算符号がある。項目の左端のビットは、演算符号。

2 バイト文字セット (Double-Byte Character Set: DBCS). IBM のコード化文字セット。各文字は 2 バイトで表される。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 バイトが必要になるので、DBCS 文字を入力、表示、および印刷するときは、DBCS 機能を持つハードウェアとこれをサポートしたソフトウェアが必要である。

77 レベル記述記入項目 (* 77-level-description-entry). レベル番号 77 を持つ非連続データ項目を記述するデータ記述記入項目。

A

ABEND. プログラムの異常終了。

ASCII. 情報交換用米国標準コード。7 ビット・コード化文字 (パリティー・チェックを含めて 8 ビット) から構成されるコード化文字セットを使用した、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のために使用する標準コード。ASCII セットは、制御文字と図形文字から構成されている。

注: IBM は、ASCII に対する拡張 (文字 128 ～ 255) を定義している。

AT END 条件 (* AT END condition). 次のような条件。

1. 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、そのファイルの中に次の論理レコードがない場合、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、あるいはオプショナルの入力ファイルが存在しない場合。
2. RETURN ステートメントの実行中に、関連付けられたソート・ファイルまたはマージ・ファイルに対して次の論理レコードが存在しない場合。
3. SEARCH ステートメントの実行中に、関連付けられている WHEN 句に指定されたどの条件も満たさず、検索処理が終了する場合。

C

CCSID. 「*Coded Character Set Identifier (CCSID)*」を参照。

COBOL 文字セット (* COBOL character set). 「*基本文字セット (basic character set)*」を参照。

COBOL ワード (* COBOL word). 「*ワード (word)* 」を参照。

Coded Character Set Identifier (CCSID). 特定のコード・ページを識別するための IBM が 定義した 5 桁の数CCSID は 16 ビットで内部的に表現される。

CONSOLE. オペレーター・コンソールに関連する COBOL 環境名。

D

DBCS. 「*2 バイト文字セット (Double-Byte Character Set: DBCS)*」を参照。

DBCS 文字 (DBCS character). IBM の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。

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 の場合、TEST AFTER 句を PERFORM ステートメントと共に使用すれば、同じ機能が得られる。

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 の場合、TEST BEFORE 句を PERFORM ステートメントと共に使用すれば、同じ機能が得られる。

E

*** EBCDIC (拡張 2 進化 10 進コード) (Extended Binary-Coded Decimal Interchange Code)**. 8 ビットのコード化文字で構成されるコード化文字セット。

EBCDIC 文字 (EBCDIC character). EBCDIC コード化文字セットでエンコードされた図形文字または制御機能のいずれか。

Extensible Markup Language. 「*XML*」を参照。

I

IBM 拡張機能 (IBM extensions). 標準 COBOLではなく IBM によって指定された COBOL 構文およびセマンティクス。

ID (* identifier). データ項目などのリソースを参照する構文。データ項目を参照する ID は、データ名およびオプションで修飾子、添え字付け、および参照変更を含む。

J

Java Native Interface (JNI). Java 仮想マシン (JVM) 内部で実行している Java コードを他のプログラム言語で作成されたアプリケーションおよびライブラリーと相互運用できるようにするプログラミング・インターフェース。

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。

O

ODO オブジェクト (ODO object). 以下の例では、

```
WORKING-STORAGE SECTION
01  TABLE-1.
    05  X                  PICS9.
    05  Y OCCURS 3 TIMES
          DEPENDING ON X    PIC X.
```

X は、OCCURS DEPENDING ON 句のオブジェクト (ODO オブジェクト)。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 サブジェクト (ODO subject). 上記の例では、Y が OCCURS DEPENDING ON 句のサブジェクト (ODO サブジェクト)。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。

P

private. オブジェクト指向において、データを定義するクラスのメソッドによってのみアクセス可能なデータ。インスタンス・データはインスタンス・メソッドによってのみアクセス可能。ファクトリー・データはファクトリー・メソッドによってのみアクセス可能。したがって、インスタンス・データは同じクラス定義で定義されたインスタンス・メソッドにとって private である。ファクトリー・データは同じクラス定義で定義されたファクトリー・メソッドにとって private である。

Q

QSAM (待機順次アクセス方式) (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。

S

SBCS (1 バイト文字セット) (Single Byte Character Set). 「1 バイト文字セット (Single Byte Character Set: SBCS)」を参照。

SPECIAL-NAMES. 環境部の段落の名前。ここで、環境名がユーザー指定の簡略名と関連付けられる。

U

Unicode. 現代のさまざまな言語の記述表現に必要なすべての文字をコード化したコード化文字セット。Unicode 表現には、UTF-8、UTF-16、および UTF-32 など多数の形式がある。Enterprise COBOL では、国別データ型を表現するものとして UTF-16 を使用して Unicode をサポートしている。UTF-16 は CCSID 01200 によって、ビッグ・エンディアン形式でサポートされる。

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 ～ UPSI-7 の 8 つのスイッチがある。

X

XML. Extensible Markup Language。SGML から派生した SGML のサブセットであるマークアップ言語を定義するためのメタ言語。XML は、SGML からより複雑で使用頻度の少ない部分を省略して以下の作業を容易に行えるようにする。

- 文書タイプを処理するアプリケーションの作成
- 構造化された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された情報の伝送と共用

XML は、W3C (World Wide Web Consortium) のサポートのもとで開発されている。

索引

日本語、英字、数字、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〕

アクセス・モード
  順次
    説明　120
    DELETE ステートメント　299
    READ ステートメント　372
  説明　119
  動的
    説明　120
    DELETE ステートメント　300
    READ ステートメント　375
  ランダム
    説明　120
    DELETE ステートメント　300
    READ ステートメント　375
アスタリスク (*)
  コメント行　47
  挿入文字　191
遊びバイト
  間　204
  内部　201
暗黙の
  再定義、ストレージ域の　150, 193
  範囲終了符号　260
暗黙の属性、データの　54
異常終了　602
位置合わせ、データ　199
一致規則
  SET...USAGE OBJECT REFERENCE　396
移動、制御ステートメント
  XML PARSE ステートメント　433
移動、制御の
  暗黙の　64
  明示的　64
  ALTER　281
  ALTER ステートメント　282
  GO TO ステートメント　316
  IF ステートメント　319
  PERFORM ステートメント　360
印刷ファイル、WRITE ステートメント　431
インスタンス定義
  フォーマットと説明　80
インスタンス変数　79
インスタンス・データ　78, 81
インスタンス・メソッド　77, 81
インプリメンター名　8
隠蔽　91
引用符 (") 文字　45
インライン PERFORM ステートメント　360
ウィンドウ化日付フィールド
  拡張、使用前の　165
  定義　67
  参照：日付フィールド
受け取りフィールド
  複数の演算結果をもたらす場合の規則　266
  COMPUTE ステートメント　296
  MOVE ステートメント　346
  SET ステートメント　391
  STRING ステートメント　410
  UNSTRING ステートメント　419
英字、ACCEPT 内の　274
英字項目　183, 184
  位置合わせの規則　144
  基本移動の規則　348
  PICTURE 文節　183
英字名　7, 50
  説明　100
  MERGE ステートメント　342
  PROGRAM COLLATING SEQUENCE 文節　96
  SORT ステートメント　400
英数字オペランドの比較　246
英数字関数　442
英数字クラスとカテゴリー
  位置合わせの規則　144
  説明　142
英数字項目
  位置合わせの規則　144
  基本移動の規則　348
  PICTURE 文節　185
英数字引き数　444
英数字編集項目
  位置合わせの規則　144
  基本移動の規則　348
  PICTURE 文節　185
英数字リテラル　25, 28
英数字リテラル、制御文字についての制限　145
英数字リテラル、DBCS 文字　26
英数字リテラル、16 進表記　27
エンコード・ユニット　3

© Copyright IBM Corp. 1991, 2001

オーバーライド　91
オーバーラップ、オペランドの無効な
　算術ステートメント　266
　データ処理ステートメント　266
置き換えフィールド、INSPECT REPLACING ステートメントの　325
送り、ページ　47
送り出しフィールド
　MOVE ステートメント　346
　SET ステートメント　391
　STRING ステートメント　409
　UNSTRING ステートメント　417
オブジェクト、EVALUATE ステートメント内の　308
オブジェクト指向 COBOL
　一致規則
　　SET...USAGE OBJECT REFERENCE　396
　影響、GLOBAL 属性の　136
　オブジェクト定義　80
　クラス定義　77
　構成セクションの指定　94
　サブクラスとメソッド　89
　手続き部 (クラスおよびメソッド)　223
　比較の規則　244
　ファクトリー定義　80
　見出し部 (クラスおよびメソッド)　84
　メソッド定義　81
　メソッド名　49
　INHERITS 文節　88
　INVOKE ステートメント　332
　OO クラス名　50
　REPOSITORY 段落　105
　SELF および SUPER 特殊文字　9
　USAGE OBJECT REFERENCE 文節　211
オブジェクト指向クラス名　50
オブジェクト定義
　OBJECT 段落　90
オブジェクト見出し部　84, 90
オブジェクト・データ部　134
オブジェクト・プログラム　72
オブジェクト・リファレンス　79
　SET ステートメント内の　391
オブジェクト・ワーキング・ストレージ　136
オプショナル・ファイル
　参照 : SELECT OPTIONAL 文節
オプショナル・ワード、構文表記法　xi
オペランド
　オーバーラップ　266
　合成　265
　比較、英数字の　246
　比較、数字の　244
親クラス　79

〔カ行〕

ガーベッジ・コレクション　77
解決、名前の　51
回線アドバンス　426
外部 10 進数項目
　DISPLAY ステートメント　301
外部クラス名　8, 106
外部浮動小数点
　位置合わせの規則　144
　DISPLAY ステートメント　301
　PICTURE 文節と　187
拡張、使用前のウィンドウ化日付フィールドの　165
拡張機能言語エレメント　536
拡張西暦年
　定義　67
　参照 : 日付フィールド
拡張日付フィールド
　定義　67
　参照 : 日付フィールド
拡張文字セット　2
型の一致
　SET...USAGE OBJECT REFERENCE　396
括弧
　算術式内の　231
　複合条件、使用法　253
カテゴリー、データの
　英字項目　183
　英数字項目　185
　英数字編集項目　185
　関係、データのクラスとの　142
　国別項目　186
　数字項目　183, 184
　数字編集項目　184
　DBCS 項目　186
可変長テーブル　175
紙送り制御文字　427
環境部
　構成セクション
　　ALPHABET 文節　100
　　CURRENCY SIGN 文節　104
　　OBJECT-COMPUTER 段落　96
　　REPOSITORY 段落　105
　　SOURCE-COMPUTER 段落　95
　　SPECIAL-NAMES 段落　97, 103
　　SYMBOLIC CHARACTERS 文節　103
　入出力セクション
　　FILE-CONTROL 段落　109
　ASCII に関する考慮事項　573
　REPOSITORY 段落　105

環境名　8, 275, 431
　SPECIAL-NAMES 段落　99
漢字　237
関数
　規則、使用の　442
　クラスとカテゴリー　143
　説明　441
　タイプ、関数の　442
　引き数　443
関数 ID　63
関数、クラスとカテゴリー　143
関数定義　448
関数のタイプ　442
関数引き数　443
関数ポインター　164, 209
　SET ステートメント内の　391
関数ポインター・データ項目
　比較条件　243
　SET ステートメント　395
関数名　8
簡略複合比較条件　592
　使用、括弧の　255
　例　257
簡略名　7, 50, 274
　ACCEPT ステートメント　274
　DISPLAY ステートメント　302
　SET ステートメント　393
　SPECIAL-NAMES 段落　99
　WRITE ステートメント　427
キー、参照の　117
記号、PICTURE 文節内の　178
疑似テキスト
　継続規則　524
　説明　47
　COPY ステートメントのオペランド　514
疑似テキスト区切り文字　38
規則、構文表記法の　xi
記入項目
　定義　39
基本移動の規則　347
基本国別リテラル　32
基本項目　140
　位置合わせの規則　144
　英数字オペランドの比較　248
　基本的小部分、レコードの　140
　クラスとカテゴリー　142
　サイズの決定、ストレージで　145
　サイズの決定、プログラムで　145
　MOVE ステートメント　347
基本名　49
基本文字セット　2
行外 PERFORM ステートメント　361
業界仕様　577
行順次ファイル編成　117
兄弟プログラム　72
共通の処理機能　267
共用、データの　170
共用、ファイルの　151
切り捨て、データの
　算術項目　145
　JUSTIFIED 文節　171
　ROUNDED 句　262
　TRUNC コンパイラー・オプション　145
句　40
句、定義　40
区切り文字　36, 217
　INSPECT ステートメント　327
　UNSTRING ステートメント　418
区切り文字、規則　36
区切り文字、クラス条件　237
国別関数　442, 443
国別クラスとカテゴリー　142
国別項目
　基本移動の規則　348
国別データ項目　186, 211
　クラス条件内の　236
　の位置合わせ　144
　SEARCH ステートメント内の　388
　UNSTRING ステートメント内の　417
国別引き数　444
国別リテラル　2, 32
組み込み関数　441
　英数字関数　442
　国別関数　442
　数字関数　442
　整数関数　442
　浮動小数点リテラル　445
　まとめ　449
　ACOS　452
　ANNUITY　453
　ASIN　454
　ATAN　455
　CHAR　456
　COS　457
　CURRENT-DATE　458
　DATEVAL　461
　DATE-OF-INTEGER　459
　DATE-TO-YYYYMMDD　460
　DAY-OF-INTEGER　463
　DAY-TO-YYYYDDD　464
　DISPLAY-OF　465

組み込み関数 *(続き)*
FACTORIAL 467
INTEGER 468
INTEGER-OF-DATE 469
INTEGER-OF-DAY 470
INTEGER-PART 471
LENGTH 472
LOG 473
LOG10 474
LOWER-CASE 475
MAX 476
MEAN 477
MEDIAN 478
MIDRANGE 479
MIN 480
MOD 481
NATIONAL-OF 482
NUMVAL 483
NUMVAL-C 484
ORD 486
ORD-MAX 487
ORD-MIN 488
PRESENT-VALUE 489
RANDOM 490
RANGE 491
REM 492
REVERSE 493
SIN 494
SQRT 495
STANDARD-DEVIATION 496
SUM 497
TAN 498
UNDATE 499
UPPER-CASE 500
VARIANCE 501
WHEN-COMPILED 502
YEARWINDOW 504
YEAR-TO-YYYY 503
クラス 77
クラス、データの 142
クラス条件 236, 237
クラス定義
クラス手続き部 223
構成セクション 94
指標テーブルの要件 174
説明 77
見出し部 85
CLASS-ID 段落 88
SELF と SUPER の影響 332
クラス手続き部 223
クラスとカテゴリー
関数の 143
グループ項目の 143
リテラルの 143
クラス見出し部 84, 88
クラス名 7, 8, 50
クラス名、OO 50
クラス名テスト 237
繰り返しワード、構文表記法 xii
グループ移動の規則 351
グループ項目 140
英数字オペランドの比較 248
説明 140
MOVE ステートメント 351
グループ項目、クラスとカテゴリー 143
継承 79, 89
形象定数 9
シンボリック文字 10
ALL リテラル 10
DISPLAY ステートメント 302
HIGH-VALUE/HIGH-VALUES 10
LOW-VALUE/LOW-VALUES 10
NULL/NULLS 11
QUOTE/QUOTES 10
SPACE/SPACES 9
STOP ステートメント 408
STRING ステートメント 409
UNSTRING ステートメント 418
ZERO/ZEROS/ZEROES 9
継続
行 44, 46
領域 41
桁 7
指定、コメントの 46
標識域 44
結果フィールド
GIVING 句 262
NOT ON SIZE ERROR 句 263
ON SIZE ERROR 句 263
ROUNDED 句 262
限界値、日付フィールド 166
言語環境プログラム呼び出し可能サービス
説明 283
言語名 8
高機能印刷 432
合成、オペランド 265
構成セクション
説明 (プログラム、クラス、メソッド) 94
REPOSITORY 段落 105
SOURCE-COMPUTER 段落 95
SPECIAL-NAMES 段落 97

構造、COBOL 言語の　2
構造化プログラミング
　DO-WHILE および DO-UNTIL　363
構文表記法の規則　xi
構文表記法の見方　xi
項目　39
互換日付フィールド
　定義　68
　参照：日付フィールド
固定挿入による編集　189
固定長
　項目、最大長　161
　レコード　151
コメント　34
コメント行
　説明　46
　見出し部　91
　ライブラリー・テキストで　513
固有照合シーケンス　100
固有性、参照の　53
固有の 2 進データ項目　208
固有文字セット　100
コロン文字
　説明　37
　必須、使用　517
コンパイラー指示ステートメント　47, 506
　BASIS　506
　CBL (PROCESS)　508
　COPY　511
　DELETE　518
　EJECT　519
　ENTER　520
　INSERT　520
　PROCESS (CBL)　508
　READY TRACE　521
　REPLACE　522
　RESET TRACE　521
　SERVICE LABEL　526
　SERVICE RELOAD　526
　SKIP1/2/3　526
　TITLE　528
　USE　529
　USE ステートメント　529
　*CBL (*CONTROL)　509
　*CONTROL (*CBL)　509
コンパイラーの制限事項　551
コンパイラー・オプション　508
　指定　508
　制御、出力の　509
　ADV　428
　DATEPROC　67
　NUMPROC　250
コンパイラー・オプション *(続き)*
　THREAD　174
　TRUNC　145
コンピューター名　8, 95, 96
コンマ (,)
　構成セクション　94
　挿入文字　188
　DECIMAL-POINT IS COMMA 文節　105
コンマ (,)、PICTURE 文節の記号

〔サ行〕

最外部プログラム、デバッグ　533
再帰的プログラム　87
　要件、指標項目の　174
再帰的メソッド　332
再使用、論理レコードの　382
サイズ・エラー条件　263
最大指標値　60
再定義、暗黙の　150
索引付きファイル
　使用可能なステートメント　358
　編成　117
　CLOSE ステートメント　293
　DELETE ステートメント　300
　FILE-CONTROL 段落のフォーマット　110
　I-O-CONTROL 段落のフォーマット　126
　READ ステートメント　374
　REWRITE ステートメント　383
　START ステートメント　406
索引編成
　説明　117
　FILE-CONTROL 段落のフォーマット　110
　I-O-CONTROL 段落のフォーマット　126
サブクラス　79
サブクラスとメソッド　89
サブジェクト、EVALUATE ステートメント内の　308
サブストリング、指定 (参照変更)　61
サブプログラム終了
　CANCEL ステートメント　291
　EXIT PROGRAM ステートメント　314
　GOBACK ステートメント　315
サブプログラムのリンケージ
　CALL ステートメント　283
　CANCEL ステートメント　290
　ENTRY ステートメント　307
算術演算子　9
　説明　231
　対にできる記号　232
算術演算符号
　代数、説明　145
　SIGN 文節と　145

算術演算符号*（続き）*
USAGE 文節と 145
算術式
説明 230
比較条件 239
COMPUTE ステートメント 296
EVALUATE ステートメント 309
算術ステートメント
オペランド 265
共通の句 261
複数の演算結果 266
プログラミングに関する注意 265
リスト 264
ADD 278
COMPUTE 296
DIVIDE 304
MULTIPLY 352
SUBTRACT 414
参照、方法
単純なデータ 55
参照変更 61
シーケンス番号域 (1 ～ 6 桁) 41
式、算術 230
字下げ 44, 142
指数計算
指数式 230
システム情報の転送、ACCEPT ステートメント 275
システムに関する考慮事項、サブプログラムのリンケージ
CALL ステートメント 283
CANCEL ステートメント 290
システム入力装置、ACCEPT ステートメント 274
システム名 8, 96
コンピューター名 95
SOURCE-COMPUTER 段落 95
実行単位
終了、CANCEL ステートメントで 291
説明 72
実行の一時停止 408
実行フロー
ALTER ステートメント変更 281
PERFORM ステートメント変更 360
指標
相対指標付け 60
データ項目 248
SET ステートメント 60
指標付け
説明 60
相対 60
MOVE ステートメントの評価 347
OCCURS 文節 60, 171
SET ステートメントと 60
指標名 50
値の割り当て 391
データ項目の定義 210
比較 248
OCCURS 文節 174
PERFORM ステートメント 369
SET ステートメント 391
修飾 53
終了、実行の
EXIT METHOD ステートメント 313
EXIT PROGRAM ステートメント 314
GOBACK ステートメント 315
STOP RUN ステートメント 408
終了符号、範囲 260
終了マーカー 43
出力ファイル、ラベル処理 356
順次アクセス・モード
説明 120
データ編成と 121
DELETE ステートメント 299
READ ステートメント 372
REWRITE ステートメント 382
順次ファイル
アクセス・モード、可能な 121
使用可能なステートメント 357
説明 116
ファイル記述項目 147
CLOSE ステートメント 292, 293
FILE-CONTROL 段落のフォーマット 110
LINAGE 文節 156
OPEN ステートメント 354
PASSWORD 文節を使用できる場所 124
READ ステートメント 373
REWRITE ステートメント 382
SELECT OPTIONAL 文節 112
順序、記入項目の
文節、FILE-CONTROL 段落内の 109
IO CONTROL 段落 126
順序、複合条件における評価の 254
状況キー
値と意味 267
共通の処理機能 267
ファイル処理 530
条件
関係 239
クラス 236
条件名 238
省略形で結合した比較 255
スイッチ状況 251
単純 236
単純否定 252
複合 253

条件 *(続き)*
複合体 251
符号 250
EVALUATE ステートメント 309
IF ステートメント 318
PERFORM UNTIL ステートメント 363
SEARCH ステートメント 386
条件式
括弧、省略形で結合した比較条件内の 255
順序、オペランドの評価 254
説明 235
比較、指標名と指標データ項目の 248
比較、DBCS オペランドの 249
条件ステートメント
説明 259
リスト 259
GO TO ステートメント 316
IF ステートメント 318
PERFORM ステートメント 363
条件変数 162
条件名 7, 49
規則、値に関する 217
条件変数 162
スイッチ状況条件 99
説明とフォーマット 238
SEARCH ステートメント 389
SET ステートメント 393
SPECIAL-NAMES 段落 99
照合シーケンス
指定、OBJECT-COMPUTER 段落での 96
指定、SPECIAL-NAMES 段落での 100
ASCENDING/DESCENDING KEY 句と 173
ASCII 560
EBCDIC 556
小数点 (.) 262
商標 579
初期状態、プログラムの 88
シリアル検索
PERFORM ステートメント 363
SEARCH ステートメント 385
真の値
条件ステートメントと 259
複合条件 251
複合条件の 252
符号条件 250
EVALUATE ステートメント 309
IF ステートメント 318
シンボリック文字 7, 10, 50
スーパークラス 79
スイッチ状況条件 251
数字オペランド、比較 244
数字関数 442, 443
数字クラスとカテゴリー 142
数字引き数 443, 444, 445
数字編集項目
位置合わせの規則 144
基本移動の規則 348
編集符号 146
PICTURE 文節 184
数字リテラル 29
スキップ、次のページへ 47
図形文字 3
ステートメント 40
カテゴリー 257
条件付き 259
説明 40, 230
データ操作 266
入出力 266
範囲区切り 260
プロシージャーのブランチ 273
命令ステートメント 257
ステートメントによる操作
共通の句 261
ファイル位置標識 273
INTO/FROM ID 句 272
ストレージ
マップのリスト 510
MEMORY SIZE 文節 96
REDEFINES 文節 192
スラッシュ (/)
記号、PICTURE 文節内の 180
コメント行 46
挿入文字 188
世紀ウィンドウ
定義 69
参照：日付フィールド
制御の移動 64
制限事項、コンパイラーの 551
整数関数 442, 443
整数引き数 443, 444
静的データ 78
静的メソッド 78
正符号 (+)
固定挿入記号 189
挿入文字 191
浮動挿入記号 190, 191
SIGN 文節 198
セクション 39, 229
セクションのヘッダー
仕様 42
説明 229

セクション名　7, 49
　説明　229
　EXCEPTION/ERROR 宣言内で　529
セグメント化に関する考慮事項　282
ゼロ
　埋め込み、基本移動　347
　消去と置換による編集　191
宣言型プロシージャー
　説明とフォーマット　227
　PERFORM ステートメント　360
　USE ステートメント　228
宣言セクション　227
宣言部分
　優先規則、ネストされたプログラムの　530
　DEBUGGING　533
　EXCEPTION/ERROR　529
　LABEL　531
選択オブジェクト、EVALUATE ステートメント内で　308
選択サブジェクト、EVALUATE ステートメント内で　308
ソース言語のデバッグ　564
ソース・コード
　ライブラリー、プログラミングに関する注意　516
　リスト　510
ソース・プログラム
　標準 COBOL 参照フォーマット　41
ソート / マージ機能
　I-O-CONTROL 段落のフォーマット　126
　MERGE ステートメント　340
　RELEASE ステートメント　377
　RERUN 文節　128
　RETURN ステートメント　379
　SAME SORT AREA 文節　130
　SAME SORT-MERGE AREA 文節　130
　SORT ステートメント　398
ソート / マージ・ファイル・ステートメント句
　ASCENDING/DESCENDING KEY 句　340
　COLLATING SEQUENCE 句　342
　GIVING 句　343
　OUTPUT PROCEDURE 句　344
　USING 句　343
ソート・ファイル記述記入項目
　参照 : SD (ソート・ファイル記述) 記入項目
相対ファイル
　アクセス・モード、可能な　121
　使用可能なステートメント　358
　編成　117
　CLOSE ステートメント　293
　DELETE ステートメント　300
　FILE-CONTROL 段落のフォーマット　110
　I-O-CONTROL 段落のフォーマット　126
　READ ステートメント　372
　RELATIVE KEY 文節　121, 123
相対ファイル *(続き)*
　REWRITE ステートメント　383
　START ステートメント　407
相対編成
　アクセス・モード、可能な　121
　説明　117
　FILE-CONTROL 段落のフォーマット　110
　I-O-CONTROL 段落のフォーマット　126
挿入編集
　固定 (数字編集項目)　189
　単純　188
　特別 (数字編集項目)　189
　浮動 (数字編集項目)　190
添え字付け
　使用、指標名の (指標付け)　60
　使用、整数の　59
　使用、データ名の　59
　テーブル参照　58
　定義とフォーマット　58
　INDEXED BY 句、OCCURS 文節の　174
　MOVE ステートメントの評価　347
　OCCURS 文節指定　171

〔タ行〕

タイプ、関数の　442
多重定義　90
単項演算子　231
単純条件
　説明と種類　236
　否定　252
　複合　253
単純挿入による編集　188
単純なデータ参照　55
単純否定条件　252
段落
　終了、EXIT ステートメント　312
　説明　39, 229
　ヘッダー、仕様　42
段落名　7, 49
　仕様　42
　説明　229
チェックポイント処理、RERUN 句　127
置換規則、COPY ステートメントの　515
置換による編集　191
重複データ名、優先順位　134
通貨記号
　指定、CURRENCY SIGN 文節で　104
　PICTURE 文節　180
通貨記号、デフォルト ($)　189

通貨符号値　104
データ
　位置合わせ　144
　階層構造、修飾で使用される　139
　カテゴリー　143, 183
　切り捨て　145, 171
　クラス　142, 143
　フォーマット、標準　145
　符号付き　145
　編成　116
データ型
　インスタンス・データ　139
　オブジェクト・インスタンス・データ　139
データ記述記入項目
　字下げと　142
　説明とフォーマット　161
　データ名　163
　レベル 66 のフォーマット (すでに定義済みの項目)　162
　レベル 88 のフォーマット (条件名)　162
　レベル番号記述　162
　BLANK WHEN ZERO 文節　163
　DATE FORMAT 文節　164
　FILLER 句　163
　GLOBAL 文節　170
　JUSTIFIED 文節　171
　OCCURS 文節　171
　OCCURS DEPENDING ON (ODO) 文節　175
　PICTURE 文節　177
　REDEFINES 文節　192
　RENAMES 文節　196
　SIGN 文節　198
　SYNCHRONIZED 文節　199
　USAGE 文節　206
　VALUE 文節　214
データ項目
　記述記入項目の説明　135
　データ記述記入項目　161
　レコード記述項目　161
　EXTERNAL 文節　169
データ項目記述記入項目　136
データ処理ステートメント
　オーバーラップ、オペランド　266
　リスト　266
　ACCEPT　274
　INITIALIZE　320
　MOVE　346
　READ　370
　RELEASE　377
　RETURN　379
　REWRITE　381
　SET　391
　STRING　409
データ処理ステートメント *(続き)*
　UNSTRING　417
　WRITE　425
データ単位
　ファイル・データ　138
　ファクトリー・データ　139
　プログラム・データ　138
　メソッド・データ　139
データ転送　274
　ACCEPT ステートメント　274
　MOVE ステートメント　346
　UNSTRING ステートメント　417
データ部
　オブジェクト定義で　134
　ソート記述 (SD) 記入項目　150
　データ記述記入項目　161
　データの関係　139
　ファイル記述 (FD) 記入項目　150
　ファクトリー定義で　134
　リンケージ・セクション　137
　レベル、データの　139
　ローカル・ストレージ・セクション　137
　ワーキング・ストレージ・セクション　135
　ASCII に関する考慮事項　575
データ部の名前　55
データ変換、DISPLAY ステートメント　301
データ編成
　アクセス・モードと　121
　行順次　117
　索引付き　117
　順次　116
　相対　117
データ名　49
　データ記述記入項目　163
　優先順位、重複する場合の　134
データ・カテゴリー
　英字項目　183
　英数字項目　185
　英数字編集項目　185
　国別項目　186
　数字項目　183, 185
　数字編集項目　184
　DBCS 項目　186
データ・フロー
　STRING ステートメント　411
　UNSTRING ステートメント　421
テーブル参照
　指標付け　60
　添え字付け　58
テキスト名　7, 50
　リテラル-1　511

テキスト・ワード　512
手続き部
　ステートメント　274
　宣言型プロシージャー　227
　フォーマット (プログラム、メソッド、クラス)　223
　ヘッダー　224
　ASCII に関する考慮事項　575
手続き部のヘッダー
　RETURNING 句　227
手続き部名　55
デバッグ　564
デバッグ行　47, 95, 564
デバッグ・セクション　564
転送、データの
　STRING ステートメント　409
等号 (=)　239
動的アクセス・モード
　説明　120
　データ編成と　121
　DELETE ステートメント　300
　READ ステートメント　375
登録商標　579
特殊オブジェクト ID
　SELF　9
　SUPER　9
特殊レジスター　11
　ADDRESS OF　13
　DEBUG-ITEM　13
　JNIENVPTR　14
　LENGTH OF　15
　LINAGE-COUNTER　16
　RETURN-CODE　16
　SHIFT-OUT、SHIFT-IN　17
　SORT-CONTROL　18
　SORT-CORE-SIZE　18
　SORT-FILE-SIZE　19
　SORT-MESSAGE　19
　SORT-MODE-SIZE　20
　SORT-RETURN　20
　TALLY　20
　WHEN-COMPILED　21
　XML-CODE　21
　XML-EVENT　22
　XML-NTEXT　24
　XML-TEXT　24
特別挿入による編集　189
独立したものとしてコンパイルされるプログラム　72
特記事項　579
トリガー値、日付フィールド　166

〔ナ行〕

内部浮動小数点
　位置合わせの規則　144
　DISPLAY ステートメント　301
二分探索　387
入出力ステートメント
　一般的説明　266
　共通の処理機能　267
　ACCEPT　274
　CLOSE　292
　DELETE　299
　DISPLAY　301
　EXCEPTION/ERROR プロシージャー　530
　OPEN　354
　READ　370
　REWRITE　381
　START　405
　WRITE　425
入出力セクション
　説明　108
　ファイル制御段落　108
　フォーマット　108
　FILE-CONTROL 段落　109
入力ファイル、ラベル処理　356
ヌル終了英数字リテラル　28
ヌル・ブロック・ブランチ、CONTINUE ステートメント　298
ネストされた IF 構造
　説明　319
　EVALUATE ステートメント　308
ネストされたプログラム
　説明　72
　優先規則　530
年末尾型日付フィールド
　定義　68
　参照：日付フィールド

〔ハ行〕

廃止される言語エレメント　x
ハイフン (-)、標識域の　44
派生クラス　79
バッチ・コンパイル　73
範囲区切りステートメント　260
範囲終了符号
　暗黙の　260
　明示的　260
パンチ・ファイル、WRITE ステートメント　430
汎用オブジェクト・リファレンス　211
比較
　英数字オペランド　246
　規則、COPY ステートメントの　515

比較 *(続き)*
国別オペランドとその他のオペランドの 249
国別オペランドの 249
指標データ項目の 248
指標名の 248
周期、INSPECT ステートメントの 330
数字オペランド 244
数字オペランドと英数字オペランド 248
DBCS オペランド 249
EVALUATE ステートメントにおける 310
比較演算子 9
意味、個々の比較演算子の 240
省略形で結合した比較条件の中で 255
比較条件の使用 239
比較条件
オペランドのサイズが等しい場合 246
オペランドのサイズが等しくない場合 246
省略形で結合した 255
説明 239
比較、数字オペランドと英数字オペランドの 244
比較、第 2 の英数字オペランドとの 246
比較、第 2 の数字オペランドとの 245
比較文字
COPY ステートメント 514
INITIALIZE ステートメント 320
INSPECT ステートメント 325
引き数 443
ピクチャー記号 178
日付関数 448
日付フィールド
ウィンドウ化日付フィールドの条件変数 239
拡張、使用前のウィンドウ化日付フィールドの 165
加算 233
グループ項目、日付フィールドである 167
減算 233
互換性 68
サイズ・エラー 234, 263
算術演算 232
算術演算結果の保管 234
制約事項 166
定義 67
トリガー値 166
比較条件での 241
非日付データ 69
符号条件での 251
目的 66
DATE FORMAT 文節 164
DATEPROC コンパイラー・オプション 67
DATEVAL 関数 461
MOVE ステートメントの動作 350
UNDATE 関数 499
日付フォーマット
定義 68
参照 : DATE FORMAT 文節
必須ワード、構文表記法 xi
非日付データ
定義 69
参照 : 日付フィールド
評価の規則
ネストされた IF ステートメント 319
複合条件 254
EVALUATE ステートメント 310
標識域 41
標準 577
標準 COBOL フォーマット 41
標準位置合わせ
規則 144
JUSTIFIED 文節 171
標準データ・フォーマット 145
非リール・ファイル、定義 293
ピリオド (.)
実際の小数点 189
ファイル
定義 138
ラベル 155
ファイル位置標識
説明 273
READ ステートメント 374
ファイル終わり処理 292
ファイル記述項目 135
参照 : FD (ファイル記述) 記入項目
ファイル編成
行順次 117
種類 116
定義 120
LINAGE 文節 156
ファイル名 49
ファイル名、SELECT 文節で指定 112
ファイル・セクション 135, 149
EXTERNAL 文節 150
RECORD 文節 153
ファクトリー定義
フォーマットと説明 80
FACTORY 段落 89
ファクトリー手続き部のヘッダー 224
ファクトリー見出し部 84, 89
ファクトリー・データ 78
ファクトリー・データ部 134
データ部 134
ファクトリー・メソッド 78, 82
ファクトリー・ワーキング・ストレージ 136

フォーマットの表記法、規則　xi
複合 OCCURS DEPENDING ON (CODO)　177
複合条件
　順序、評価の　254
　省略形で結合した比較　255
　説明　251, 253
　単純否定　252
　評価の規則　254
　複合条件　253
　許されるエレメントのシーケンス　253
　論理演算子と評価結果　253
複合否定条件　253
複数の演算結果、算術ステートメント　266
複数のレコードの処理、READ ステートメント　372
含まれているプログラム　72
符号条件　250
符号付き
　算術演算符号　145
　数字項目、定義　184
符号なし数字項目、定義　184
物理レコード
　定義　138
　ファイル記述項目と　138
　ファイル・データ　138
　BLOCK CONTAINS 文節　151
　RECORDS 句　152
浮動小数点
　内部　144
　DISPLAY ステートメント　301
浮動小数点リテラル　29
浮動挿入による編集　190
部のヘッダー
　仕様　42
　フォーマット、環境部　94
　フォーマット、手続き部　224
　フォーマット、見出し部の　84
負符号 (-)
　固定挿入記号　189
　浮動挿入記号　190, 191
　COBOL 文字　2
　SIGN 文節　198
部分的リスト　509
ブランク行　47
ブランチ
　行外 PERFORM ステートメント　361
　GO TO ステートメント　316
プログラミング構造　363
プログラミングに関する注意事項
　算術ステートメント　265
　データ処理ステートメント　409, 417
　変更された GO TO ステートメント　281
　ACCEPT ステートメント　274
プログラミングに関する注意事項 *(続き)*
　DELETE ステートメント　299
　DISPLAY ステートメント　303
　EXCEPTION/ERROR プロシージャー　530
　OPEN ステートメント　357
　PERFORM ステートメント　362
　RECORDS 文節　153
　STRING ステートメント　409
　UNSTRING ステートメント　417
プログラミング・インターフェース情報の　579
プログラム、再帰的　87
プログラム、独立したものとしてコンパイルされる　72
プログラム識別部　84
プログラム終了
　GOBACK ステートメント　315
　STOP ステートメント　408
プログラム手続き部のヘッダー　224
プログラム名　49
プログラム名、参照の規則　75
プログラム・データ部　134
プログラム・ローカル・ストレージ　137
プロシージャー、説明　228
プロシージャーのブランチ
　ステートメント、順次に実行される　273
　GO TO ステートメント　316
プロシージャーのブランチ・ステートメント　273
プロシージャー名
　GO TO ステートメント　316
　MERGE ステートメント　344
　PERFORM ステートメント　360
　SORT ステートメント　401
プロシージャー・ポインター・データ項目
　定義された、USAGE 文節で　213
　比較条件　243
　SET ステートメント　395
文　40
　説明　230
　COBOL、定義　40
文節　40
ページ送り　47
変更される GO TO ステートメント　317
編集
　固定挿入　189
　単純挿入　188
　置換　191
　特別挿入　189
　符号　146
　浮動挿入　190
　抑止　191
編集解除　349

編集符号制御記号　180
ポインター・データ項目
　定義された、USAGE 文節で　212
　比較条件　242
　SET ステートメント　394

〔マ行〕

マルチボリューム・ファイル
　READ ステートメント　374
　WRITE ステートメント　430
見出し部
　オプションの段落　91
　フォーマット (プログラム、クラス、メソッド)　84
　CLASS-ID 段落　88
　FACTORY 段落　89
　METHOD-ID 段落　90
　OBJECT 段落　90
　PROGRAM-ID 段落　86
無効キー条件　271
無条件 GO TO ステートメント　316
明示的
　範囲終了符号　260
明示的な属性、データの　54
命令ステートメント　257
メソッド
　再帰的再入　87
　再使用　88
　終了　313
　使用可能、サブクラスに　89
　呼び出し　332
メソッド隠蔽　91
メソッド多重定義　90
メソッド定義
　継承規則　89
　データ部　134
　フォーマット　223
　フォーマットと説明　81
　見出し部　85
　メソッド手続き部　223
　METHOD-ID 段落　90
　SELF と SUPER の影響　332
メソッド手続き部　223
メソッド手続き部のヘッダー　224
メソッドのオーバーライド　91
メソッド見出し部　84, 90
メソッド名　49
メソッド・データ部　134
メソッド・ローカル・ストレージ　137
メソッド・ワーキング・ストレージ　136
文字、COBOL プログラムで有効　2
文字エンコード・ユニット　3
文字コード・セット、指定　100
文字ストリング　5
　サイズの決定　145
　表現、PICTURE 文節の　182
　COBOL ワード　5

〔ヤ行〕

ユーザー定義語　6
ユーザー・ラベル
　DEBUGGING 宣言　533
　LABEL 宣言　531
ユーロ通貨符号　558
　指定、CURRENCY SIGN 文節で　104
有効範囲、名前の　48
優先順位番号　97
ユニット・ファイル、定義　293
用語集　585
抑止、出力の　509
抑止による編集　191
呼び出されるプログラムと呼び出しプログラム、説明　283
予約語　8, 566

〔ラ行〕

ライブラリー名　7, 49
　COPY ステートメント　511
ラベル処理、OPEN ステートメント　356
ランダム・アクセス・モード
　説明　120
　データ編成と　121
　DELETE ステートメント　300
　READ ステートメント　375
リソース・リスト　581
リテラル
　英数字オペランドの比較　248
　説明　25
　と算術式　230
　ヌル終了英数字　28
　ASSIGN 文節　112
　CODE-SET 文節と ALPHABET 文節　102
　CURRENCY SIGN 文節　104
　STOP ステートメント　408
　VALUE 文節　216
リテラル、クラスとカテゴリー　143
リポジトリー段落　107
領域 A (8 ～ 11 桁)　42
領域 B (12 ～ 72 桁)　44

リンケージ・セクション
説明　137
要件、指標項目の　174
呼び出されるサブプログラム　227
VALUE 文節　214
レコード
基本項目　140
固定長　151
物理、定義　138
領域記述　153
論理、定義　138
レコード記述項目
レベル、データの　140
論理レコード　138
レコード名　49
レコード・キー、索引ファイル内の　300
レベル
標識、定義　139
01 項目　140
02 ～ 49 項目　140
66 項目　142
77 項目　142
88 項目　142
レベル番号　46, 161
説明とフォーマット　162
定義　139
FILLER 句　163
(01 および 77)　43
レベル標識
(FD および SD)　43
ローカル・ストレージ　137
定義、RECURSIVE 文節で　87
要件、指標項目の　174
論理演算子
複合条件　251
複合条件の評価における　253
リスト　251
論理レコード
定義　138
ファイル・データ　138
プログラム・データ　138
レコード記述項目と　138
RECORDS 句　152

〔ワ行〕

ワーキング・ストレージ・セクション　135
割り当て、指標値　391
割り当て名　8
ASSIGN 文節　112
RERUN 文節　127

A

A、PICTURE 文節の記号　178
ACCEPT ステートメント
オペランドのオーバーラップ、予想できない結果　266
簡略名　274, 275
システム情報の転送　275
説明とフォーマット　274
FROM 句　274
ACCESS MODE 文節　119
ACOS 関数　452
ADD ステートメント
共通の句　261
説明とフォーマット　278
CORRESPONDING 句　280
END-ADD 句　280
GIVING 句　278
NOT ON SIZE ERROR 句　280
ON SIZE ERROR 句　280
ROUNDED 句　280
ADDRESS OF 特殊レジスター　13
ADVANCING 句　426
AFTER 句
INSPECT ステートメント　329
PERFORM ステートメント　363
REPLACING の指定された　326
TALLYING の指定された　325
WRITE ステートメント　426
ALL
句、INSPECT ステートメントの　325, 326
SEARCH ステートメント　387
UNSTRING ステートメント　418
ALL 添え字　445
ALL リテラル　10
STOP ステートメント　408
STRING ステートメント　409
UNSTRING ステートメント　418
ALPHABET 文節　100
ALPHABETIC クラス・テスト　237
ALPHABETIC-LOWER クラス・テスト　237
ALPHABETIC-UPPER クラス・テスト　237
ALSO 句
ALPHABET 文節　101
EVALUATE ステートメント　308
ALTER ステートメント
セグメント化に関する考慮事項　282
説明とフォーマット　281
GO TO ステートメントと　317
ALTERNATE RECORD KEY 文節　122
AND 論理演算子　251

ANNUITY 関数 453
ANSI COBOL 標準 577
APPLY WRITE-ONLY 文節 131
ASCENDING KEY 句
照合シーケンス 173
説明 340
MERGE ステートメント 340
OCCURS 文節 173
SORT ステートメント 398
ASCII
指定、SPECIAL-NAMES 段落での 100
照合シーケンス 560
ASCII に関する考慮事項
環境部 573
手続き部 575
ASSIGN 文節 574
CODE-SET 文節 575
ASCII 標準 577
ASIN 関数 454
ASSIGN 文節
説明 112
フォーマット 110
ASCII に関する考慮事項 574
SELECT 文節と 112
AT END 句
READ ステートメント 371
RETURN ステートメント 380
SEARCH ステートメント 385
AT END 条件
READ ステートメント 374
RETURN ステートメント 380
AT END-OF-PAGE 句 428
ATAN 関数 455
AUTHOR 段落
説明 91
フォーマット 84

B

B
挿入文字 188
B、PICTURE 文節の記号 178
BASIS ステートメント 506
BEFORE 句
INSPECT ステートメント 329
PERFORM ステートメント 363
REPLACING の指定された 326
TALLYING の指定された 325
WRITE ステートメント 426
BINARY 句、USAGE 文節内の 207
BLANK WHEN ZERO 文節
説明とフォーマット 163
USAGE IS INDEX 文節 210
BLOCK CONTAINS 文節
説明 151
フォーマット 147
BY CONTENT 句
CALL ステートメント 285
BY REFERENCE 句
CALL ステートメント 285
BY VALUE 句
CALL ステートメント 286
INVOKE ステートメント 334

C

CALL ステートメント
移動、制御の 65
サブプログラムのリンケージ 283
説明とフォーマット 283
手続き部のヘッダー 224, 227
プログラム終了ステートメント 283
リンケージ・セクション 227
CANCEL ステートメントと 290
ON OVERFLOW 句 283
USING 句 227
CANCEL ステートメント 290
CBL (PROCESS) ステートメント 508
CHAR 関数 456
CHARACTERS 句
BLOCK CONTAINS 文節 152
INSPECT ステートメント 325
MEMORY SIZE 文節 96
USAGE 文節と 152
CHARACTERS BY 句 326
CLASS 文節 103
CLASS-ID 段落 88
CLOSE ステートメント
フォーマットと説明 292
COBOL
クラス定義 77
言語の構造 2
参照フォーマット 41
プログラムの構造 72
メソッド定義 81
COBOL オブジェクト 77
COBOL クラス 77
COBOL 標準 577
COBOL ワード 5
DBCS 文字が含まれる 5
1 バイト文字が含まれる 5

CODE-SET 文節
説明 160
フォーマット 147
ALPHABET 文節と 102
ASCII に関する考慮事項 575
NATIVE 句と 160
COLLATING SEQUENCE 句 96
ALPHABET 文節 100
MERGE ステートメント 342
SORT ステートメント 400
COMMON 文節 88
COMPUTATIONAL データ項目 207
COMPUTE ステートメント
共通の句 262
説明とフォーマット 296
COMP-1 ～ COMP-5 データ項目 208
CONTINUE ステートメント 298
CONTROL ステートメント (*CONTROL) 509
CONVERTING 句 328
COPY ステートメント
説明とフォーマット 511
置換規則 515
比較の規則 515
例 516
REPLACING 句 514
SUPPRESS オプション 513
COPY ライブラリー 54
CORRESPONDING (CORR) 句
説明 280
ADD ステートメント 280
MOVE ステートメント 346
ON SIZE ERROR 句と 264
SUBTRACT ステートメント 414
COS 関数 457
COUNT IN 句、UNSTRING ステートメント 419
CR (借方)
記号、PICTURE 文節内の 180
挿入文字 189
cs (通貨記号)
PICTURE 文節 178
CURRENCY SIGN 文節
説明 104
ユーロ通貨符号 104
NUMVAL-C 関数、制約事項 484
CURRENT-DATE 関数 458

D

DATA RECORDS 文節
説明 156
フォーマット 147
DATE 276
DATE FORMAT 文節 164
結合、他の文節との 167
DATE YYYYMMDD 276
DATEPROC コンパイラー・オプション 67
DATEVAL 関数 461
DATE-COMPILED 段落
説明 91
フォーマット 84
DATE-OF-INTEGER 関数 459
DATE-TO-YYYYMMDD 関数 460
DATE-WRITTEN 段落
説明 91
フォーマット 84
DAY 277
DAY YYYYDDD 277
DAY-OF-INTEGER 関数 463
DAY-OF-WEEK 277
DAY-TO-YYYYDDD 関数 464
DB (貸方)
記号、PICTURE 文節内の 180
挿入文字 189
DBCS クラス条件 237
DBCS の表記 xii
DBCS 引き数 444
DBCS 文字
リテラルで 27
COBOL ワードで 6
DBCS 文字セット 2
DBCS リテラル 30
DBCS (2 バイト文字セット)
基本移動の規則 349
クラスとカテゴリー 142
使用、コメントで 91
使用、比較演算子と共に 242
PICTURE 文節と 186
参照 : DBCS 文字
DEBUGGING 宣言 529, 533
DEBUGGING MODE 文節 95, 533, 564
DEBUG-CONTENTS 14
DEBUG-ITEM 特殊レジスター 13, 565
DEBUG-LINE 13
DEBUG-NAME 13
DECIMAL-POINT IS COMMA 文節
説明 105
DECLARATIVES キーワード
開始、領域 A で 43
説明 228
DELETE ステートメント
順次アクセス 299
説明とフォーマット 518

DELETE ステートメント *(続き)*
動的アクセス 300
フォーマットと説明 299
ランダム・アクセス 300
INVALID KEY 句 300
DELIMITED BY 句
STRING 410
UNSTRING ステートメント 418
DELIMITER IN 句、UNSTRING ステートメント 419
DEPENDING 句
GO TO ステートメント 316
OCCURS 文節 175
DESCENDING KEY 句 173
照合シーケンス 173
説明 340
MERGE ステートメント 340
SORT ステートメント 398
DISPLAY 句、USAGE 文節内の 209
DISPLAY ステートメント
外部 144, 187
説明とフォーマット 301
プログラミングに関する注意 303
DISPLAY-OF 関数 465
DIVIDE ステートメント
共通の句 262
説明とフォーマット 304
REMAINDER 句 306
DOWN BY 句、SET ステートメント 392
DO-UNTIL 構造、PERFORM ステートメント 363
DO-WHILE 構造、PERFORM ステートメント 363
DUPLICATES 句
KEY 句 405
SORT ステートメント 400
START ステートメント 405

E

E、PICTURE 文節の記号 179
EBCDIC
コード・ページ 1140 556
指定、SPECIAL-NAMES 段落での 100
照合シーケンス 556
CODE-SET 文節と 160
EJECT ステートメント 519
ELSE NEXT SENTENCE 句 318
END DECLARATIVES キーワード 228
END PROGRAM 73
END PROGRAM マーカー 43
END-ADD 句 280
END-CALL 句 289
END-IF 句 318
END-INVOKE 句 336
END-OF-PAGE 句 428
END-PERFORM 句 362
END-SUBSTRACT 句 416
END-XML 句 435
ENTER ステートメント 520
ENTRY ステートメント
サブプログラムのリンケージ 307
説明とフォーマット 307
EOP 句 428
EQUAL TO 比較演算子 239
EVALUATE ステートメント
決定、真の値の 309
比較、オペランドの 310
フォーマットと説明 308
EXCEPTION/ERROR 宣言 529
説明とフォーマット 529
CLOSE ステートメント 293
DELETE ステートメント 299
EXIT ステートメント
フォーマットと説明 312
PERFORM ステートメント 361
EXIT METHOD ステートメント
フォーマットと説明 313
EXIT PROGRAM ステートメント
フォーマットと説明 314
EXTEND 句
OPEN ステートメント 354
EXTERNAL 文節
データ項目で 169
ファイル名で 150

F

FACTORIAL 関数 467
FACTORY 段落 89
FALSE 句 309
FD (ファイル記述) 記入項目
説明 149
フォーマット 147
レベル標識 139
BLOCK CONTAINS 文節 151
DATA RECORDS 文節 156
GLOBAL 文節 151
LABEL RECORDS 文節 155
VALUE OF 文節 156
FILE STATUS 文節
状況キー 267
説明 125
フォーマット 110

FILE STATUS 文節 *(続き)*
DELETE ステートメントと 299
INVALID KEY 句と 271
FILE-CONTROL 段落
説明とフォーマット 109
ASSIGN 文節 112
FILE STATUS 文節 125
ORGANIZATION 文節 116
PADDING CHARACTER 文節 118
RECORD KEY 文節 121
RELATIVE KEY 文節 123
RESERVE 文節 115
SELECT 文節 112
FILLER 句
データ記述記入項目 163
CORRESPONDING 句 163
FILLER 文節 161
FIPS 標準 577
FOOTING 句、LINAGE 文節の 156
FOR REMOVAL 句 292, 293
FROM 句
ACCEPT ステートメント 274
ID の指定された 272
REWRITE ステートメント 381
SUBTRACT ステートメント 414
WRITE ステートメント 426
FUNCTION-POINTER 句、USAGE 文節内の 209

G

G、PICTURE 文節の記号 179
GIVING 句
算術演算 262
ADD ステートメント 278
DIVIDE ステートメント 306
MERGE ステートメント 343
MULTIPLY ステートメント 352
SORT ステートメント 402
SUBTRACT ステートメント 415
GLOBAL 文節
データ項目で 170
ファイル名で 151
GO TO、DEPENDING ON 句 281
GO TO ステートメント 281
条件付き 316
フォーマットと説明 316
変更される GO TO 317
無条件 316
MORE-LABELS 317
SEARCH ステートメント 385
GOBACK ステートメント 315
GREATER THAN 記号 (>) 239
GREATER THAN OR EQUAL TO 記号 (>=) 239

H

HIGH-VALUE(S) 形象定数 101
HIGH-VALUE/HIGH-VALUES 10

I

IBM 拡張機能 x, 536
ID 55, 230
IF ステートメント 318
INDEX 句、USAGE 文節内の 210
INDEXED BY 句 174
INHERITS 文節 88
INITIAL 文節 88
INITIALIZE ステートメント
オペランドのオーバーラップ、予想できない結果 266
フォーマットと説明 320
INPUT 句
OPEN ステートメント 354
USE ステートメント 529
INPUT PROCEDURE 句
RELEASE ステートメント 377
SORT ステートメント 401
INPUT-OUTPUT セクション
I-O-CONTROL 段落 126
INSERT ステートメント 520
INSPECT ステートメント
オペランドのオーバーラップ、予想できない結果 266
比較の周期 330
AFTER 句 327
BEFORE 句 327
CONVERTING 句 328
REPLACING 句 325
INSTALLATION 段落
説明 91
フォーマット 84
INTEGER 関数 468
INTEGER-OF-DATE 関数 469
INTEGER-OF-DAY 関数 470
INTEGER-PART 関数 471
INTO 句
DIVIDE ステートメント 304
ID の指定された 272
READ ステートメント 370
RETURN ステートメント 379
STRING ステートメント 410
UNSTRING ステートメント 419

INVALID KEY 句
DELETE ステートメント 300
READ ステートメント 371
REWRITE ステートメント 382
START ステートメント 406
WRITE ステートメント 428
INVOKE ステートメント
フォーマットと説明 332
BY VALUE 句 334
LENGTH OF 特殊レジスター 334
NEW 句 333
NOT ON EXCEPTION 句 336
ON EXCEPTION 句 336
RETURNING 句 334
SELF 特殊オブジェクト・リファレンス 332
SUPER 特殊オブジェクト・リファレンス 332
USING 句 334
ISCII 処理に関する考慮事項 573
ISCII 標準 577
ISO COBOL 標準 577
I-O-CONTROL 段落
順序、記入項目の 126
説明 108, 126
チェックポイント処理 127
APPLY WRITE-ONLY 文節 131
MULTIPLE FILE TAPE 文節 130
RERUN 文節 127
SAME AREA 文節 128
SAME RECORD AREA 文節 129
SAME SORT AREA 文節 130
SAME SORT-MERGE AREA 文節 130

J

Java
クラス名 106
パッケージ 106
Java インターオペラビリティー
データ型 334, 336, 339
リテラルの型 334
Java オブジェクト 77
Java クラス 77
Java ストリング・データ 77
Java との相互運用性 (インターオペラビリティー) 77
Java Native Interface 602
Java Native Interface (JNI) 14, 77, 79
java.lang.Object 79
JNI 環境ポインター 79
JNIENVPTR 特殊レジスター 14, 79
JUSTIFIED 文節
影響、初期設定値への 171
JUSTIFIED 文節 *(続き)*
切り捨て、データの 171
説明とフォーマット 171
STRING ステートメント 410
USAGE IS INDEX 文節と 171
UUSAGE IS NATIONAL 文節と 171
VALUE 文節と 215

K

KEY 句
OCCURS 文節 173
READ ステートメント 371
SEARCH ステートメント 384
SORT ステートメント 398
START ステートメント 405

L

LABEL 宣言 529, 531
LABEL RECORDS 文節
説明 155
フォーマット 147
LEADING 句
INSPECT ステートメント 325, 326
SIGN 文節 198
LENGTH 関数 472
LENGTH OF 特殊レジスター 15
INVOKE ステートメント 334
LESS THAN 記号 (<) 239
LESS THAN OR EQUAL TO 記号 (<=) 239
LINAGE 文節
図、句に関する 157
説明 156
フォーマット 147
LINAGE-COUNTER 特殊レジスター
説明 16
WRITE ステートメント 427
LINES AT BOTTOM 句 157
LINES AT TOP 句 157
LINE/LINES、WRITE ステートメント 426
LOG 関数 473
LOG10 関数 474
LOWER-CASE 関数 475
LOW-VALUE(S) 形象定数 101
LOW-VALUE/LOW-VALUES 10

M

MAX 関数 476

MEAN 関数　477
MEDIAN 関数　478
MEMORY SIZE 文節　96
MERGE ステートメント
　フォーマットと説明　340
　ASCENDING/DESCENDING KEY 句　340
　COLLATING SEQUENCE 句　342
　GIVING 句　343
　OUTPUT PROCEDURE 句　344
　USING 句　343
METHOD-ID 段落　90
MIDRANGE 関数　479
MIN 関数　480
MOD 関数　481
MORE-LABELS GO TO ステートメント　317
MOVE ステートメント
　基本移動　347
　グループ移動　351
　フォーマットと説明　346
　CORRESPONDING 句　346
MULTIPLE FILE TAPE 文節　130
MULTIPLY ステートメント
　共通の句　262
　フォーマットと説明　352

N

N、PICTURE 文節の記号　179
NATIONAL 句、USAGE 文節内の　211
NATIONAL-OF 関数　482
NEGATIVE　250
NEW 句
　INVOKE ステートメント　333
NEXT RECORD 句、READ ステートメント　370
NEXT SENTENCE 句
　IF ステートメント　318
　SEARCH ステートメント　385
NO ADVANCING 句、DISPLAY ステートメント　302
NO REWIND 句　292
　OPEN ステートメント　354
NOT AT END 句
　READ ステートメント　371
　RETURN ステートメント　380
NOT INVALID KEY 句
　DELETE ステートメント　300
　READ ステートメント　371
　REWRITE ステートメント　382
　START ステートメント　406
NOT ON EXCEPTION 句
　CALL ステートメント　288
　INVOKE ステートメント　336
NOT ON EXCEPTION 句 *(続き)*
　XML PARSE ステートメント　434
NOT ON OVERFLOW 句
　STRING ステートメント　411
　UNSTRING ステートメント　421
NOT ON SIZE ERROR 句
　一般的説明　263
　ADD ステートメント　280
　DIVIDE ステートメント　306
　MULTIPLY ステートメント　353
　SUBTRACT ステートメント　415, 416
NSYMBOL コンパイラー・オプション　2
NULL　219
NULL/NULLS　11, 210, 394, 396
NUMERIC クラス・テスト　237
NUMVAL 関数　483
NUMVAL-C 関数　484

O

OBJECT 段落　90
OBJECT REFERENCE 句　211
OBJECT-COMPUTER 段落　96
OCCURS 文節
　可変長テーブルのフォーマット　175
　制約事項　172
　説明　171
　ASCENDING/DESCENDING KEY 句　173
　INDEXED BY 句　174
OCCURS DEPENDING ON (ODO) 文節
　オブジェクト　175
　サブジェクト　172, 175
　サブジェクトとオブジェクト　175
　説明　175
　添え字付け　58
　フォーマット　175
　複合体　177
　RECORD 文節　153
　REDEFINES 文節と　172
　SEARCH ステートメントと　172
OFF 句、SET ステートメント　393
OMITTED　285, 286
ON 句、SET ステートメント　393
ON EXCEPTION 句
　CALL ステートメント　288
　INVOKE ステートメント　336
　XML PARSE ステートメント　434
ON OVERFLOW 句
　CALL ステートメント　289
　DISPLAY ステートメント　303
　STRING ステートメント　410, 421

ON SIZE ERROR 句
算術ステートメント 263
ADD ステートメント 280
COMPUTE ステートメント 297
DIVIDE ステートメント 306
MULTIPLY ステートメント 353
SUBTRACT ステートメント 415, 416
OPEN ステートメント
句 354
システム依存事項 358
新規 / 既存ファイルの場合 355
フォーマットと説明 354
プログラミングに関する注意 357
ラベル処理 356
I-O 句 354
ORD 関数 486
ORD-MAX 関数 487
ORD-MIN 関数 488
ORGANIZATION 文節
説明 116
フォーマット 110
ORGANIZATION IS INDEXED 文節 116
ORGANIZATION IS LINE SEQUENTIAL 文節 116
ORGANIZATION IS RELATIVE 文節 116
ORGANIZATION IS SEQUENTIAL 文節 116
OUTPUT 句 354
OUTPUT PROCEDURE 句
MERGE ステートメント 344
RETURN ステートメント 379
SORT ステートメント 403
OVERFLOW 句
CALL ステートメント 289
STRING ステートメント 410, 421

P

P、PICTURE 文節の記号 179
PACKED-DECIMAL 句、USAGE 文節内の 207
PADDING CHARACTER 文節 118
PASSWORD 文節
システム依存事項 124
説明 124
PERFORM ステートメント
行外 361
行内 361
実行シーケンス 362
条件付き 363
フォーマットと説明 360
ブランチ 361
END-PERFORM 句 362
EVALUATE ステートメント 308
PERFORM ステートメント *(続き)*
EXIT ステートメント 312
TIMES 句 362
VARYING 句 363, 366
PICTURE 文節
記号、使用される 178
計算用項目と 207
シーケンス、記号の 181
説明 177
データ・カテゴリー 183
とクラス条件 237
フォーマット 177
編集 188
CURRENCY SIGN 文節 104
DECIMAL-POINT IS COMMA 文節 105, 178
PICTURE 文字ストリング 34
PICTURE SYMBOL 句 105
POINTER 句
STRING ステートメント 410
UNSTRING ステートメント 420
POSITIVE 250
PRESENT-VALUE 関数 489
PREVIOUS RECORD 句、READ ステートメント 370
PROCESS (CBL) ステートメント 508
PROCESSING PROCEDURE 句、XML PARSE 内の 433
PROGRAM COLLATING SEQUENCE 文節
ALPHABET 文節 100
SPECIAL-NAMES 段落と 96
PROGRAM-ID 段落
説明 86
フォーマット 84

Q

QUOTE/QUOTES 10

R

RANDOM 関数 490
RANGE 関数 491
READ ステートメント
オペランドのオーバーラップ、予想できない結果 266
動的アクセス・モード 375
フォーマットと説明 370
複数のレコードの処理 372
プログラミングに関する注意 376
マルチボリューム・ファイル 374
ランダム・アクセス・モード 375
AT END 句 371
INTO ID 句 272, 370
INVALID KEY 句 271, 371

READ ステートメント *(続き)*
KEY 句 371
NEXT RECORD 句 370
READY TRACE ステートメント 521
RECORD 文節
省略 153
説明とフォーマット 153
RECORD CONTAINS 0 CHARACTERS 153
RECORD KEY 文節
説明 121
フォーマット 110
RECORDING MODE 文節 158
RECORDS 句
BLOCK CONTAINS 文節 152
RERUN 文節 128
RECURSIVE 文節 87
REDEFINES 文節
一般的考慮事項 194
説明 192
フォーマット 192
予想外の結果 196
例 195
OCCURS 文節の制約事項 193
SYNCHRONIZED 文節と 199
VALUE 文節と 194
REEL 句 292, 293
RELATIVE KEY 文節
説明 123
フォーマット 110
RELEASE ステートメント 266, 377
REM 関数 492
REMAINDER 句、DIVIDE ステートメントの 306
RENAMES 文節 142
説明とフォーマット 196
レベル 66 の項目 142, 196
INITIALIZE ステートメント 320
PICTURE 文節 178
REPLACE ステートメント
継続規則、疑似テキストの 524
説明とフォーマット 522
注意事項 525
比較操作 524
REPLACING 句
COPY ステートメント 514
INITIALIZE ステートメント 321
REPOSITORY 段落 105
RERUN 文節
説明 127
ソート / マージ 128
チェックポイント処理 127
フォーマット 126
RERUN 文節 *(続き)*
RECORDS 句 127
RESERVE 文節
説明 115
フォーマット 110
RESET TRACE ステートメント 521
RETURN ステートメント
オペランドのオーバーラップ、予想できない結果 266
説明とフォーマット 379
AT END 句 380
RETURNING 句
手続き部のヘッダー 227
CALL ステートメント 287
INVOKE ステートメント 334
RETURN-CODE 特殊レジスター 16
REVERSE 関数 493
REWRITE ステートメント
説明とフォーマット 381
FROM ID 句 272
INVALID KEY 句 382
ROUNDED 句
サイズ・エラー検査 264
説明 262
ADD ステートメント 280
COMPUTE ステートメント 296
DIVIDE ステートメント 306
MULTIPLY ステートメント 353
SUBTRACT ステートメント 415, 416

S

S、PICTURE 文節の記号 179
SAME 文節 128
SAME RECORD AREA 文節
説明 129
フォーマット 126
SAME SORT AREA 文節
説明 130
フォーマット 126
SAME SORT-MERGE AREA 文節
説明 130
フォーマット 126
SD (ソート・ファイル記述) 記入項目
説明 147, 149
データ部 150
レベル標識 139
DATA RECORDS 文節 156
SEARCH ステートメント
シリアル検索 385
説明とフォーマット 384
二分探索 387

SEARCH ステートメント *(続き)*
AT END 句 385
SET ステートメント 384
USAGE IS INDEX 文節 210
VARYING 句 386
WHEN 句 385
SECURITY 段落
説明 91
フォーマット 84
SEGMENT-LIMIT 文節 97
SELECT 文節
指定、ファイル名の 112
フォーマット 110
ASSIGN 文節と 112
SELECT OPTIONAL 文節
指定、順次入出力ファイルの 112
説明 112
フォーマット 110
CLOSE ステートメント 293
SELF 9
SELF 特殊オブジェクト・リファレンス 332
SELF 特殊文字ワード 9
SEPARATE CHARACTER 句、SIGN 文節の 198
SERVICE LABEL ステートメント 526
SERVICE RELOAD ステートメント 526
SET ステートメント
オペランドのオーバーラップ、予想できない結果 266
関数ポインター・データ項目 395
説明とフォーマット 391
プロシージャー・ポインター・データ項目 395
ポインター・データ項目 394
要件、指標項目の 174
割り当てられた関数ポインター・データ項目値 209
割り当てられた指標データ項目値 210
DOWN BY 句 392
OFF 句 393
ON 句 393
SEARCH ステートメント 392
TO 句 391
TO TRUE 句 393
UP BY 句 392
USAGE IS INDEX 文節 210
USAGE OBJECT REFERENCE 396
SHIFT-IN 特殊レジスター 17
SHIFT-OUT 特殊レジスター 17
SIGN 文節 198
SIGN IS SEPARATE 文節 198
SIN 関数 494
SKIP1/2/3 ステートメント 526
SORT ステートメント
説明とフォーマット 398
ASCENDING KEY 句 398
COLLATING SEQUENCE 句 400
DESCENDING KEY 句 398
DUPLICATES 句 400
GIVING 句 402
INPUT PROCEDURE 句 401
OUTPUT PROCEDURE 句 403
USING 句 401
SORT-CONTROL 特殊レジスター 18, 404
SORT-CORE-SIZE 特殊レジスター 18, 404
SORT-FILE-SIZE 特殊レジスター 19, 404
SORT-MESSAGE 特殊レジスター 19, 404
SORT-MODE-SIZE 特殊レジスター 20, 404
SORT-RETURN 特殊レジスター 20, 404
SOURCE-COMPUTER 段落 95
SPACE/SPACES 9
SPECIAL-NAMES 段落
簡略名 99
説明 97
フォーマット 97
ACCEPT ステートメント 274
ALPHABET 文節 100
ASCII コードのファイル仕様 160
CLASS 文節 103
CODE-SET 文節と 160
CURRENCY SIGN 文節 104
DECIMAL-POINT IS COMMA 文節 105
SQRT 関数 495
STANDARD-DEVIATION 関数 496
STANDARD-1 句 100
STANDARD-2 句 100
START ステートメント
索引付きファイル 406
状況キーに関する考慮事項 405
説明とフォーマット 405
相対ファイル 407
INVALID KEY 句 271, 406
STOP ステートメント 408
STOP RUN ステートメント 408
STRING ステートメント
オペランドのオーバーラップ、予想できない結果 266
実行 411
説明とフォーマット 409
SUBTRACT ステートメント
共通の句 261
説明とフォーマット 414
SUM 関数 497

SUPER 9
SUPER 特殊オブジェクト・リファレンス 332
SUPER 特殊文字ワード 9
SUPPRESS オプション、COPY 513
SYMBOLIC CHARACTERS 文節 103
SYNCHRONIZED 文節 199
VALUE 文節と 215

T

TALLY 特殊レジスター 20
TALLYING 句
INSPECT ステートメント 325
UNSTRING ステートメント 420
TAN 関数 498
THREAD コンパイラー・オプション 174
要件、指標項目の 174
THROUGH (THRU) 句
ALPHABET 文節 101
CLASS 文節 104
EVALUATE ステートメント 309
PERFORM ステートメント 360
RENAMES 文節 196
VALUE 文節 217
TIME 277
TIMES 句、PERFORM ステートメントの 362
TITLE ステートメント 528
TO 句、SET ステートメント 391
TO TRUE 句、SET ステートメント 393

U

UNDATE 関数 499
Unicode 2, 3
Unicode UTF-16 2
UNIT 句 292
UNSTRING ステートメント
受け取りフィールド 419
送り出しフィールド 417
オペランドのオーバーラップ、予想できない結果 266
実行 421
説明とフォーマット 417
UP BY 句、SET ステートメント 392
UPON 句、DISPLAY 302
UPPER-CASE 関数 500
UPSI-0 ～ UPSI-7、プログラム・スイッチ
条件名 99
処理、特別条件の 99
スイッチ状況条件 251
SPECIAL-NAMES 段落 99
USAGE 文節
基本項目のサイズ 145
算術演算符号と 145
説明 206
フォーマット 206
BINARY 句 207
CODE-SET 文節と 160
COMPUTATIONAL 句 208
DISPLAY 句 209
DISPLAY-1 句 209
FUNCTION-POINTER 句 209
INDEX 句 210
NATIONAL 句 211
PACKED-DECIMAL 句 207
USAGE IS PROCEDURE-POINTER 213
VALUE 文節と 215
USAGE DISPLAY
クラス条件 ID 236
STRING ステートメントと 409
USAGE DISPLAY-1
STRING ステートメントと 409
USAGE IS COMPUTATIONAL 句 208
USAGE IS OBJECT REFERENCE 構文 206
USAGE IS POINTER 212
USAGE IS PROCEDURE-POINTER 213
USAGE OBJECT REFERENCE 句 332
USE ステートメント
フォーマットと説明 529
USE FOR DEBUGGING 宣言部 565
USING 句
サブプログラムのリンケージ 227
手続き部のヘッダーで 224
CALL ステートメント 284
INVOKE ステートメント 334
MERGE ステートメント 343
SORT ステートメント 401
UTF-16 2, 3

V

V、PICTURE 文節の記号 179
VALUE 文節
影響、オブジェクト指向プログラムへの 136
規則、条件名記入項目の 217
規則、リテラルの値に関する 216
条件名 216
フォーマット 214, 216
レベル 88 の項目 142
NULL 219
NULL/NULLS 210

VALUE OF 文節
  説明 156
  フォーマット 147
VARIANCE 関数 501
VARYING 句
  PERFORM ステートメント 363
  SEARCH ステートメント 386

W

WHEN 句
  EVALUATE ステートメント 308
  SEARCH ステートメント 385
WHEN-COMPILED 関数 502
WHEN-COMPILED 特殊レジスター 21
WITH DEBUGGING MODE 文節 95, 533, 564
WITH DUPLICATES 句、SORT ステートメント 400
WITH FOOTING 句 157
WITH NO ADVANCING 句 302
WITH NO REWIND 句、CLOSE ステートメント 293
WITH POINTER 句
  STRING ステートメント 410
  UNSTRING ステートメント 420
WRITE ステートメント
  順次ファイル 426
  説明とフォーマット 425
  AFTER ADVANCING 426, 430
  ALTERNATE RECORD KEY 432
  BEFORE ADVANCING 426, 430
  END-OF-PAGE 句 428
  FROM ID 句 272

X

X、PICTURE 文節の記号 179
XML 603
XML 処理
  XML-CODE 特殊レジスター 21
  XML-EVENT 特殊レジスター 22
  XML-NTEXT 特殊レジスター 24
  XML-TEXT 特殊レジスター 24
XML PARSE
  サポートされるコード・ページ 436
XML PARSE ステートメント
  処理プロシージャー 436
  制御フロー 435
  例外イベント 434
XML-CODE 特殊レジスター 434
XML-EVENT 特殊レジスター 435
XML-NTEXT 特殊レジスター 435
XML-TEXT 特殊レジスター 435
X'00' ～ X'1F' 制御文字 145

Y

YEARWINDOW 関数 504
YEARWINDOW コンパイラー・オプション
  世紀ウィンドウ 69
YEAR-TO-YYYY 関数 503

Z

Z
  挿入文字 191
Z、PICTURE 文節の記号 180
ZERO、符号条件における 250
ZERO/ZEROS/ZEROES 9

数字

0
  記号、PICTURE 文節内の 182
  挿入文字 188
0、PICTURE 文節の記号 180
16 進表記、国別リテラルの 33
2 項算術演算子 231
2 進数データ項目、DISPLAY ステートメント 301
2 バイト文字セット (DBCS)
  クラスとカテゴリー 142
  使用、コメントで 91
  使用、比較演算子と 242
  PICTURE 文節と 186
  参照：DBCS 文字
2000 年言語拡張
  構文 66
2000 年言語拡張 (MLE)
  説明 66
  参照：日付フィールド
2000 年問題
  参照：日付フィールド
66, RENAMES データ記述記入項目 196
77、項目記述記入項目 142
88、条件名データ記述記入項目 162
88 のレベル項目 142
9、PICTURE 文節の記号 180, 182

特殊文字

. (ピリオド)、PICTURE 文節の記号 180
(/) コメント行 47
+ (正符号)
  記号、PICTURE 文節内の 182

+ (正符号)*（続き）*
挿入文字　189, 190, 191
SIGN 文節　198
- (負符号)
記号、PICTURE 文節内の　182
挿入文字　189, 190
SIGN 文節　198
$ (デフォルトの通貨記号)
挿入文字　189, 190
PICTURE 文節　180, 182
*、PICTURE 文節の記号　180
*CBL (*CONTROL) ステートメント　509
*CONTROL (*CBL) ステートメント　509
< (より小さい)　240
<= (より小さいか等しい)　240
> (より大きい)　240
>= (より大きいか等しい)　240
/ (スラッシュ)
記号、PICTURE 文節内の　182
挿入文字　188
/ (スラッシュ)、PICTURE 文節の記号
, (コンマ)
記号、PICTURE 文節内の　180, 182
挿入文字　188
= (等しい)　240
{ : }
説明　37
必須、使用　517

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12