Micro Focus COBOL"

言語リファレンス

Micro Focus[R]

第 23 版
2003 年 5 月

Micro Focus は、本書の内容が公正かつ正確であるよう万全を期しているが、本書の内容は予告なしに随時変更されることがある。本書に述べられているソフトウェアはライセンスに基づいて提供され、その使用 および複写は、ライセンス契約に基づいてのみ許可される。特に、Micro Focus 社製品のいかなる用途への適合性も明示的に本契約から除外されており、Micro Focus はいかなる必然的損害に対しても一切責任を負うものではない。

AnimatorR、COBOL WorkbenchR、EnterpriseLinkR、Mainframe ExpressR、Micro FocusR、Net ExpressR、REQLR および RevolveR は、Micro Focus International Limited の登録商標である。また、AAI"、Analyzer"、Application to Application Interface"、AddPack"、AppTrack"、AssetMiner"、CCI"、DataConnect"、Dialog System"、EuroSmart"、FixPack"、LEVEL II COBOL"、License Management Facility"、License Server"、Mainframe Access"、Mainframe Manager"、Micro Focus COBOL"、Object COBOL"、OpenESQL"、Personal COBOL"、Professional COBOL"、Server Express"、SmartFind"、SmartFind Plus"、SmartFix"、SourceConnect"、Toolbox"、VS COBOL"、WebSync"、および Xilerator"は同社の商標である。その他の商標は、当該企業の商標または登録商標である。

このドキュメントは、CD-ROM に含まれるソフトウェア製品ユーザ用のものを除き、Micro Focus International Limited による事前の書面による同意の前に、いかなる電子媒体または機会により読み取り可能な書式へ、複写、複製、送付、改訂を行うことはできない。

ライセンス所有者は、CD-ROM に含まれるソフトウェア製品ユーザ用ドキュメントを複製できる。ただし、ライセンス契約によりソフトウェアへのアクセスが認められたユーザのサポートに必要な範囲に限る。ドキュメントを再製する場合は、一部または全部を問わず、この版権の記述の全体が含まれなければならない。

U.S. GOVERNMENT RESTRICTED RIGHTS. It is acknowledged that the Software and the Documentation were developed at private expense, that no part is in the public domain, and that the Software and Documentation are Commercial Computer Software provided with

---

このドキュメントについて

本書では、マイクロコンピュータのためのプログラミングシステムである本システムがサポートする COBOL 言語を解説する。この COBOL 言語は、ANSI COBOL 標準 X3.23-1985、X3.23a-1989 および X3.23b-1993 に基づいており、多くの COBOL 処理系によってサポートされているものである。さらに、ISO/IEC 1989:2002、プログラミング言語 COBOL からのいくつかの機能についてのサポートも追加されている。

COBOL は情報処理業界の標準言語の 1 つである。COBOL 言語は、いかなる企業または企業グループあるいは組織または組織グループの所有物でもない。

プログラミング･システムおよび言語の正確性および機能に関して、CODASYL プログラミング言語委員会も、あるいはいかなる協力者も、明示的にも暗示的にも、何らかの保証をするものではない。さらに、このマニュアルに記載してある内容に関連して、CODASYL プログラミング言語委員会もあるいはいかなる協力者も、何ら責任を負うものではない。

このマニュアルでは、下記の著作物を引用している。

> FLOW-MATIC Programming for the Univac 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, DSI27A5260-2760, copyrighted 1960 by Minneapolis-Honeywell

これらの著作物の著者または著作権保有者により、COBOL 仕様中にこれらの資料を全面的にまたは部分的に引用することが特別に承諾されている。この承諾は、プログラミング･マニュアルまたは類似の刊行物の中で COBOL 仕様を引用する場合にも、延長して適用される。

対象読者

本書は、既に COBOL 言語に精通している読者を対象とする。

関連図書

本書に関連して参照する必要がある図書としては、***入門書***、本 COBOL システムの***ユーザガイド***および***言語リファレンス - 追加トピック***がある。

本書の使用方法

本書は、基本 COBOL 言語用 Micro Focus COBOL 製品により提供されるサポートについて説明する。また、***言語リファレンス - 追加トピック*** を併用すること。***言語リファレンス-追加トピック***では、DBCS、Report Writer、SQL および通信などの、その他の言語機能を使用するためのサポートについて説明している。さらに、DOS/VS COBOL や Siemens BS2000 COBOL など、多くの方言のサポートについても、説明している。

表記

本書では、COBOL 文の形式を表わすために、下記の表記法を使用する。

1. 大文字で記し、下線を引いてある語（例:ABC）。その語が係わる機能を使用する際には、必ず指定しなければならない。これに該当する語を指定しないかまたはつづりを間違えるかすると、COBOL コンパイラによってエラーとされる。COBOL 原始プログラムを書く際には、アンダーラインを引く必要はない。
2. 大文字で記してあるが、下線を引いていない語（例:ABC）。COBOL 原始プログラムを読みやすくするためだけに使用する。それらの語は指定しても指定しなくても構わない。ただし、指定する場合は、正しいつづりで書かなければならない。
3. 日本語で記してある語（例:ファイル名）。プログラマが指定すべき名前を表わす。
4. { } いくつかの項目を中かっこで囲んである場合。その中から 1 つを選択して指定しなければならない。中かっこで囲んである項目が 1 つしかない場合は、その項目を指定しなければならない。その場合、中かっこは繰り返しを表わす。
5. { | | } いくつかの項目を中かっこで囲み縦棒で区切ってある場合。その中から 1 つ以上を選択して指定しなければならない。ただし、1 つの項目は 1 回だけしか指定することはできない。
6. [ ]いくつかの項目を角かっこで囲んである場合。その中の項目は任意の選択項目であることを表わす。任意の選択項目は、必要に応じて、指定しても指定しなくても構わない。
7. 項目が箱と角かっこで囲んである場合、

   [ ]

   その項目は ANSI'74 COBOL（American National Standards Institute publication X3.23-1974）の必須項目であるが、言語仕様の機能拡張の結果、省略可能になったことを表わす。端に示されている記号は COBOL の方言を示す。方言を使用するかしないかは自由である。

8. 反復記号（...）。原始プログラムまたは一連の記述の一部を省略していることを表わす。どの部分が省略されているかは、文脈からわかる。

一般形式においては、反復記号が表わすのは、利用者が自分の判断で繰り返しを指定できる部分である。その範囲は次のように読み取る。

句または文の中に反復記号 (...) が出てきたならば、その左側へ順にたどって最初に出てくる}または]を見つける。さらに左側へ順にたどって、}または]に論理的に対応する{または[を見つける。この一組の{ }または[ ]で囲まれた一連の語が反復の対象範囲となる。

9. 一般形式で「整数」という語が使われている場合は、整数の数字定数を意味するものとする。また、その形式において明示的に許可されていないかぎり、符号付きの数字やゼロであってはならない。
10. このリファレンス・マニュアルでは、IBM SAA AD/Cycle COBOL370 (COBOL/370) 文の構文および COBOL/370 を使用してコンパイルする原始プログラムを書く際の規則についても、説明する。

    COBOL/370 と IBM VS COBOL II との違いは、手続きポインタ形式に関して、省略時解釈の長さが 4 バイトではなく 8 バイトであることである。

11. このリファレンス・マニュアルでは、マルチベンダー統合体系 (MIA) の構文も示す。この MIA は COBOL プログラミング言語の技術的な必要条件となっている。
12. 大部分の COBOL 言語には、ANSI X3.23-1974 に定義されている COBOL 言語を拡張した 機能が組み込まれている。組み込まれている拡張機能の度合はコンパイラによって違いがある。つまり、COBOL には「方言」と呼ばれる異なったバージョンがいくつもある。Micro Focus COBOL 製品は、コンパイラをまたがって開発作業を進められるように設計してある。つまり、この製品は、利用者が使用している COBOL システムだけではなく、IBM OS/VS COBOL や IBM VS COBOL II あるいは ANS X3.23-1974 ないしその更新版である ANS X3.23-1985 でプログラムを開発するために使用することができる。この製品では、ANS X3.23-1985 のすべての機能はもちろんのこと、IBM の 2 つの COBOL 製品の大部分の拡張機能をも、使用することができる。

    方言を使用している利用者の便宜を図るために、このマニュアルでは、機能ごとに方言があれば明示している。ANS X3.23-1974 を超える機能に関する記述は四角で囲んで、端に下記の楕円形の記号を付している。セクション全体が別の方言である場合は、セクションの見出しの後に記号のみの行がある。別の方言の一部である語句は、そのようにマークが付けられ、それのみで 1 段落となっている。構文の表記では、ANS X3.23-1974 以外の機能は、線で囲まれ、その後またはその行末に記号が付けられている。

    使用される記号は、以下のとおり。

OSVS	IBM OS/VS COBOL 中の ANS X3.23-1974 に対する拡張機能を表わす。
VSC2	IBM VS COBOL II 中の ANS X3.23-1985 に対する拡張機能を表わす。
COB370	IBM SAA AD/Cycle COBOL/370 中の ANS X3.23-1985 に対する

拡張機能であって IBM VS COBOL II には含まれない機能を表わす。

OS/390 IBM COBOL for OS/390 & V2R2 でサポートされているが、IBM SAA AD/Cycle COBOL/370 ではサポートされていない、ANS X3.23-1985 に対する拡張機能を表す。

ANS85 ANS X3.23-1985 で新たに定義された機能で ANS X3.23-1974 にはなかった機能を表わす。

XOPEN X/Open CAE 仕様の COBOL 言語(XPG-4)に固有の ANS X3.23-1985 に対する拡張機能を表わす。

MF Micro Focus COBOL に固有の ANS X3.23-1985 に対する拡張機能を表わす。

ISO2002 ANS X3.23-1985 ではサポートされていない、ISO/IEC 1989:2002 で定義された機能を表す。

注:

- Micro Focus、 VS COBOL II、COBOL/370、OS/390、および X/Open 方言はすべて、ANS X3.23-1985 COBOL 標準に基づく。したがって、ANS85 に有効と記されている構文 (または規則) はどれも、とくに記述されていないかぎり、Micro Focus、VS COBOL II、COBOL/370、OS/390、および X/Open にも有効である。
- COBOL/370 方言は、本書中で VSC2 の記号が付けられている VS COBOL II に含まれる構文をすべて含んでいる。本書で唯一 COB370 の記号が記載されているのは、COBOL370 (NOCMPR2 コンパイラオプションを伴う) が、VS COBOL II (NOCMPR2 コンパイラオプションを伴う) とともに含まれるものテキスト以外を含む場合のみである。

これらの記号は、特定の構文および意味論のサポートを示す。特定の方言に関する予約語を変更するには、コンパイラの指令を使用する必要がある。本書の付録、*予約語*および、*Compatibility Guide* には、さまざまな方言に影響される予約語の一覧がある。

たとえば、IBM OS/VS COBOL 上で使用するプログラムを開発している場合は、記号が付けられていない機能と、OSVS の記号が付けられている機能を使用できる。自分の COBOL 環境専用に開発を行っている場合は、どの機能でも使用可能である。システムソフトウェアを起動するときに、 FLAG コンパイラ指令を発行すると、その COBOL システムソフトウェアにより、指定された方言のもの以外のあらゆる機能にフラグが付けられる。また、FLAGAS コンパイラ指令を使用して、フラグのメッセージをエラーメッセージに変換することもできる。

13. 方言により異なる効力を持つ機能もあり、方言制御指令を指定して、互換性を持たせる方言を選択することができる。このような機能は、二重線で囲まれている。

    ANS X3.23-1985 以外の機能に必要なその他の予約語は、適切な方言制御指令が存在する場合にのみ予約されている。これにより、確実に、指定した方言での予約語のみが目的のシステムで予約されているようにすることができる。その他の予約語を必要とせず、その効果がすべての方言で同じである機能のみを使用する場合は、方言制御指令を指定する必要はない。

    dialect-control 指令に ANS X3.23-1985 方言を指定すると、ANS X3.23-1985 内では使用できないいくつかの ANS X3.23-1974 の機能に警告メッセージが付けられる。

14. 本書中に「注記にとどまる」と記してある場合は、該当するコーディングが、文法的には COBOL コンパイラに受け入れられるが、実行用プログラムを生成する際には無視されることを意味する。

16 進数は、小文字の x の後に引用符で囲んで示される（例：x"9D"）。

第 1 章 : COBOL 言語の紹介

この章では、COBOL 言語の概要を解説する。

1.1 COBOL 言語

COBOL (Common Business Oriented Language) は、事務処理および管理分野のデータ処理に 最も広く使用されているプログラミング言語である。

この COBOL 言語は、*ANSI X3.23-1974 (ISO 1989-1978)*で規定された ANSI および ISO COBOL 標準、さらに *ANSI X3.23-1985 (IS 1989:1985)*、*ANSI X3.23a-1989 (IS 1989:1985/AM1)*、および *ANSI X3.23b-1993 (IS 1989:1985/AM2)*で規定された現行の ANSI および ISO COBOL 標準の上位セットである。拡張された言語は、以下のとおり。

- IBM OS/VS COBOL (リリース 2.3 以降) の拡張機能のうち、広く一般的に使用されているものの大部分
- IBM VS COBOL II (全リリース) 、IBM DOS/VS COBOL および IBM SAA AD/Cycle COBOL/370 の言語構造の大部分
- X/Open に準拠する拡張機能
- 本システムに固有の拡張機能

上記の拡張機能を組み合わせて、原始プログラム中で使用できる。

全機能を備えた ANSI COBOL 1974 および 1985 は下記の機能単位から構成される。

- 中核 (Nucleus)
- 表操作 (table handling) - ANSI '85 では中核に入る
- 順ファイル (sequential I/O)
- 相対ファイル (relative I/O)
- 索引ファイル (indexed I/O)
- 整列併合 (sort-merge)
- 区分化 (segmentation)
- 登録集 (library) /原文操作 (source text manipulation)
- プログラム間通信 (inter-program communication)
- デバッグ (debug)
- 報告書作成 (report writer)
- 通信 (communication) -ANSI '74 の全機能および ANSI 85 の構文

- ANSI85 組込み関数 (intrinsic function)

この COBOL システムは、上記の機能単位に関する米国商務省の全米標準･技術研究所 (National Institute of Standards and Technology) の COBOL コンパイラ検査システム (CCVS) の 合格基準を満たしている。

MFこの他にも、いろいろな面から COBOL 言語の機能拡張がなされている。

- 画面操作機能単位。これは、画面節と追加の ACCEPT 文および DISPLAY 文から構成される。この機能を利用することによって、画面上で項目の位置を正確に指定し、指定した位置から入力されたデータを受け取り、指定した位置に定数字句を表示し、画面属性を定義し、操作卓特性を制御できる。.
- 拡張ファイル入出力機能。原文ファイルを効率よく処理するための追加のファイル編成、原始プログラム内で (データ変数または定数) またはオペレーティングシステムの環境変数から実行時にファイル名を定義する任意選択機能、多数の利用者がファイルを共有する機能、など。
- 言語間の呼出しおよび再帰的 COBOL プログラムに関するシステム･プログラミング機能の拡張。

1.2 正書法

原始(ソース)文は

MF XOPEN自由形式か

固定形式で記述する。

1.2.1 固定形式

固定形式では、 COBOL 正書法(source format)により、COBOL 原始(ソース)レコードを 72 カラム毎に分ける。これらのカラムは以下のように使われる。

カラム 1-6	一連番号
カラム 7	標識領域
カラム 8-11	A 領域
カラム 12-72	B 領域

MFCOBOL 原始(ソース)レコードは 80 カラムまで使用できる。COBOL のシステムは、73 カラムから 80 カラムまでを無視する。

1.2.1.1 一連番号

原始プログラムの各行を識別するために、6 桁の一連番号 (sequence number) を使用する。

MF一連番号の 1 桁目に星印 (*) または印字されない制御文字 (ASCII 文字の照合順序で空白文字よりも小さい文字) を書くと、その行は注記行として扱われ、リスト用のファイルまたは装置には出力されない。この機能を利用して出力したリスト･ファイルを入力用として使用して、コンパイルを行うことができる。

MFこの機能は MFCOMMENT コンパイラ指令に影響される。

(ANS85)一連番号には計算機の文字集合の中の任意の文字を記述できる。

1.2.1.2 標識領域

標識領域 (indicator area) に星印 (*) を書くと、その行は注記行となる。注記行は、見出し部の見出しより後ろならば、どこにでも置ける。注記行の A 領域および B 領域には、ASCII 文字集合の中の 任意の文字を記すことができる。

(OSVS)(VSC2)(MF)注記行は、見出し部の見出しより前に置くことができる。

標識領域に斜線 (/) を書くと、その行が注記行となり、改ページされてから印刷が行われる。

標識領域に"D"または"d"を書くと、その行はデバッグ行となる。デバッグ行の A 領域および B 領域には有効な任意の COBOL 文を記述できる。

標識領域にハイフン"-"を書くと、その行は前の行からの継続を表す。その場合、前の行の後続の空白は、文字定数の場合はその一部とみなされ、文字定数でない場合はないものとみなされる (詳細については、***COBOL の概念***の *行の継続*を参照) 。

(MF)標識領域に"$"を書くと、その行は特殊行となり、指令やコンパイル対象行の選択条件文を記述できる。

1.2.1.3 A 領域と B 領域

節名および段落名は A 領域から書き始め、終止符 (.) と続く空白で止める。 レベル指示語 FD、SD、CD 及び RD は、A 領域から書き始め、 B 領域

(ANS85)または A 領域

から対応する記述を書き始める。レベル番号 01 及び 77 は、A 領域から書き始め、B 領域

(ANS85)または A 領域

から対応するデータ記述を書き始める。レベル番号 02 から 49、66 及び 88 は、B 領域から書き始める。

(MF)レベル番号 78 は、A 領域または B 領域から書き始めてよい。

(MF)見出し部中の注記項以外には、A 領域と B 領域に関して規則をいっさい設けない。

(MF)(XOPEN)自由形式の原始コードが使う場合は、A 領域と B 領域の区別はない。自由形式の原始コードに関する詳細は、この章の次のセクションを参照。

1 行のコーディング中に複数の文を記しても構わない。

1.2.1.4 サンプルプログラム

典型的なプログラムの正書法を図 1-1 に示す。

```
000010 identification division.
000020 program-id. stock-file-set-up.
000030 author. MicroFocus.
000040 environment division.
000050 configuration section.
000060 source-computer. mds-800.
000070 object-computer. mds-800.
000075 special-names. console is crt.
000080 input-output section.
000090 file-control.
000100  select stock-file assign "stock.it"
000110  organization indexed
000120  access dynamic
000130  record key stock-code.
000140 data division.
000150 file section.
000160 fd   stock-file record 32.
000170 01   stock-item.
000180  02  stock-code       pic x(4).
000190  02  product-desc     pic x(24).
000200  02  unit-size        pic 9(4).
000210 working-storage section.
000220 01   screen-headings.
000230  02  ask-code pic x(21) value "stock code    <    >".
000240  02  filler   pic x(59).
000250  02  ask-desc pic x(16) value "description   <".
000260  02  si-desc  pic x(25) value "                >".
000270  02  filler   pic x(39).
000280  02  ask-size pic x(21) value "unit size <    >".
000290 01   enter-it redefines screen-headings.
000300  02      filler       pic x(12).
000310  02  crt-stock-code   pic x(4).
000320  02      filler       pic x(80).
000330  02  crt-prod-desc    pic x(24).
000340  02      filler       pic x(51).
000350  02  crt-unit-size    pic 9(4).
000360  02  filler           pic x.
000370 procedure division.
000380 sr1.
000390  display space.
000400  open i-o stock-file.
000410  display screen-headings.
000420 normal-input.
000430  move space to enter-it.
```

```
000440  display enter-it.
000450 correct-error.
000460  accept enter-it.
000470  if crt-stock-code = space go to end-it.
000480  if crt-unit-size not numeric go to correct-error.
000490  move crt-prod-desc to product-desc.
000500  move crt-unit-size to unit-size.
000510  move crt-stock-code to stock-code.
000520  write stock-item invalid key go to correct-error.
000530  go to normal-input. 000540 end-it.
000550  close stock-file.
000560  display space.
000570  display "end of program".
000580  stop run.
<---->|<-><-------------------------------------------------------->
  |   | |     |
  |   | |     +-- カラム 12-72 - B 領域
  |   | +-- カラム 8-11 - A 領域
  |   +-- カラム 7 - 標識領域
  +-- カラム 1-6 - 一連番号
```

図 1-1 正書法を示すプログラム･コーディング例

1.2.2 自由形式

自由形式は、SOURCEFORMAT"FREE"指令によって選択される。

始めの 6 文字は一般行の一部で、COBOL 原始文に含まれるものとして扱われる。1 カラム目は、以下のような標識の役目をする。

*	注記行
/	リストファイルで改ページしてから始まる注記行
D または d	空白が後続する。デバッグ行
$	特殊行 (指令、条件付きコンパイル等)
その他の文字	一般の行

継続行はない。英数字定数の継続は、"A" & "B"のように連結して行う。

A 領域と B 領域の区別はない。

固定の右マージンはないが、実際上は、行の長さは 1 バイト文字では、最大 250 文字、また 2 バイト文字では、最大 125 文字までの制限がある。

注記は段落見出しとして同じ行内におさめ、次の行に続けられない。

第 2 章 : COBOL 言語の概念

この章では、COBOL 言語の概念を説明する。

2.1 文字集合

COBOL 言語の最も基本的でそれ以上分割できない単位は、文字である。COBOL の文字列および分離符として 使用できる文字集合は、英字、数字、特殊文字から構成される。文字集合を構成する文字を以下に示す。

文字	意味
0 から 9	数字
A から Z	大文字の英字
a から z	ANS85 小文字の英字
	空白
+	正号
-	負号またはハイフン
*	星印
/	斜線
=	等号
\	通貨記号
.	終止符または小数点
,	コンマまたは小数点
;	セミコロン
"	引用符
'	ISO2002 SVS VSC2 MF アポストロフィ
(	左かっこ
)	右かっこ
>	より大きい記号
<	より小さい記号
:	ANS85 コロン
&	MF XOPEN アンパサンド
_	ISO2002 下線

ANS85小文字を文字列や原文語に使用できる。ただし、文字定数や絵記号として使用した場合は例外である。各小文字は、対応する大文字と等しいものとする。

この COBOL システムは、上記の文字集合に制限されている。しかし、文字定数、注記行、注記項、データの内容には、COBOL 翻訳集団の文字を入れることができる。(付録の*文字集合と文字の照合順序*を参照。)

2.2 言語の構造

個々の文字をつないで、文字列または分離符とする。分離符には他の分離符または文字列をつなぐことができる。文字列には分離符だけをつなぐことができる。文字列と分離符をつないでいくことにより、翻訳集団の本文ができる。

2.2.1 分離符

分離符 (separator) は、1 つ以上の句読文字をつないだものである。分離符の構成に関する規則を、以下に記す。

1. 句読文字の空白は、分離符となる。空白を分離符または分離符の一部として使用する場合、1 つ以上の空白を続けてもよい。分離符のコンマ、セミコロン、終止符の直後に続く空白はすべて、その分離符の一部であるとみなされ、独立の分離符とはみなされない。
2. 句読文字のコンマまたはセミコロンの直後に空白を続けたものは分離符となる。この分離符は、分離符の空白を使用できる箇所であれば、どこでも使用できる。ただし、PICTURE 文字列の中に使用されているコンマは分離符ではない。
3. 句読文字の終止符の直後に空白を続けたものは、分離符となる。この分離符は、完結文の末尾を示すため、または形式に示された場合にだけ使用する。
4. 句読文字の左かっこおよび右かっこは、分離符とする。左かっこと右かっこは一対として、添字、

   ANS85関数の引数リスト、部分参照、算術式、条件を囲むときだけ使用する。ただし、仮原文の中で使用するときは、例外である。

5. 句読文字の左右の引用符は、分離符となる。

   ISO/MG GWS VSC2 MF左右の引用符において、アポストロフィまたは引用符を引用記号文字として使用できる。

   句読文字の左側の引用符とは、以下のいずれかである。

   - 引用符記号
   - MF次の引用符付き文字のいずれか：G"、G'、H"、H'、N"、N'、X"、X'

   句読文字の右側の引用符とは、以下のいずれかである。

- 引用符記号（左側の原文区切り記号が引用符記号 の場合）
- アポストロフィ（左側の原文区切り記号がアポストロフィの場合）

左側の引用符の直前は、空白、左かっこ、仮原文区切り記号のいずれかとする。右側の引用符の直後は、分離符の空白、コンマ、セミコロン、終止符、右かっこ、右側の仮原文区切り記号のどれかとする。左側の引用符の直前、および右側の引用符の直後のこれらの分離符は、分離符としての引用符の一部を構成するものではない。

6. 仮原文区切り記号は分離符となる。左側の仮原文区切り記号の直前は空白とする。右側の仮原文区切り記号の直後は、分離符の空白、コンマ、セミコロン、 終止符のいずれかとする。

   (MF)左側の仮原文区切り記号の直前の空白は、省略可能。

   仮原文区切り記号は一対として、仮原文

   (MF)および動詞記号

   を囲むときだけ使用する。（*翻訳指示文の原始文操作*および *Micro Focus COBOL OO 言語拡張のメソッドインターフェイス定義*の章を参照。）

7. COBOL キャラクタのコロンは分離符となり、一般形式で表示される場合は必要となる。ただし、起動演算子の一部となる場合を除く。
8. 分離符の空白は、すべての分離符の直前に置くことができる。ただし、以下の場合を除く。
   a. 正書法に指定されている場合。（*COBOL 翻訳集団の概念*の章の*正書法*の節を参照。）
   b. 分離符としての右側の引用符。この場合、引用符の直前の空白は文字列定数の一部と解釈され、分離符とはみなされない。
   c. 左側の仮原文区切り記号の直前。この場合、分離符の空白を必ず置かなければならない。
9. 分離符の直後には、分離符の空白を置いても置かなくてもよい。ただし、左側の引用符の直後は例外とする。この場合、引用符の直後の空白は文字列定数の一部と解釈され、分離符とはみなされない。

PICTURE 文字列（*データ部 - データ記述*の *PICTURE(形式)句*の章を参照。）または数字定数中の句読文字は、PICTURE 句の文字列または数字定数を指定するための記号であり、句読文字とはみなされない。PICTURE 句の文字列は、分離符の空白、コンマ、セミコロン、終止符によってのみ区切られる。

分離符の構成に関する規則は、文字定数、注記項、注記行の中の文字には適用しない。

2.2.2 文字列

文字列（character-string）はひとつながりの文字であって、COBOL の語、定数、PICTURE 文字列、注記項を形成する。文字列は分離符で区切る。

2.2.2.1 COBOL の語

ANS85COBOL の語 (word) は 30 字以内の文字列であって、翻訳指示語、コンテキストセンシティブ語、利用者語、システム名、予約語、関数名に使用する。特殊文字を含まない COBOL の語は文字と数字とハイフン

ISO2002と下線から構成される。

非特殊文字語においては、ハイフン

ISO2002または下線

を先頭または末尾に置いてはならない。小文字は対応する大文字と等しいものとみなされる。. .

ISO2002文字列は 31 文字含むことができる。

ANS85原始要素内では、下記の規則が適用される。

1. LENGTH, RANDOM, SUM を除くすべての COBOL の語に関して、
    a. 予約語の集合は利用者語、システム名、関数名の集合と重なり合ってはならない。
    b. 利用者語、システム名、関数名のそれぞれの集合は重なり合ってもよい。つまり、同じ語を利用者語とシステム名と関数名に使用してもよい。その場合、使用された語がどれに属するかは、文脈によって判定される。
2. COBOL の語 LENGTH, RANDOM, SUM に関して、
    a. LENGTH, RANDOM, SUM は予約語にも関数名にも含まれる。同じ COBOL の語である LENGTH, RANDOM, SUM を予約語としても関数名としても使用してよい。その場合、使用された語がどちらに属するかは、文脈によって判定される。
    b. COBOL の語である LENGTH, RANDOM, SUM は利用者語およびシステム名とは別の集合に属する。したがって、どのような文脈においても、COBOL の語の LENGTH, RANDOM, SUM は利用者語としてもシステム名としても使用してはならない。

利用者語：利用者語 (user-defined word) は COBOL の語であって、句や文の書き方を満足するように利用者が指定するものである。利用者語の各文字は下記の一連の文字の中から選択する。

実際の利用者語には以下の種類がある。

- 符号系名 (alphabet-name)
- ISO2002MFオブジェクト指向用の字類名 (class-name)
- ANS85真値提案用の字類名 (class-name)
- 条件名 (condition-name)
- MF定数名 (constant-name)

- データ名 (data-name)
- ファイル名 (file-name)
- (ISO2002)関数プロトタイプ名 (function-prototype-name)
- 指標名 (index-name)
- (ISO2002)(MF)インターフェイス名(interface-name)
- レベル番号 (level-number)
- 登録集名 (library-name)
- (ISO2002)(MF)メソッド名(method-name)
- 呼び名 (mnemonic-name)
- 段落名 (paragraph-name)
- (ISO2002)パラメータ名(parameter-name)
- プログラム名 (program-name)
- (ISO2002)プログラムプロトタイプ名 (program-prototype-name)
- (ISO2002)プロパティ名(property-name)
- レコード名 (record-name)
- 報告書名(report-name)
- (MF)手順名 (routine-name)
- (ISO2002)(MF)(OPEN)画面名 (screen-name)
- 節名 (section-name)
- 区分番号 (segment-number)
- (ISO2002)(MF)分割キー名 (split-key-name)
- (ANSI85)記号文字(symbolic-character)r
- 原文名 (text-name)
- (MF)型定義名(typedef-name)
- (ISO2002)利用者関数名 (user-function-name)

1 つの原始要素の中では、利用者語が次に示すように互いに重ならない種類に分かれる

- 符号系名
- (ISO2002)(MF)オブジェクト指向用の字類名
- (ANSI85)真値提案用の字類名 (class-name)
- 条件名、

  (MF)定数名、

  データ名、

  (ISO2002)プロパティ名、

  レコード名、

  (ISO2002)(MF)分割キー名、

  (MF)型定義名

- ファイル名

- (ISO2000)関数プロトタイプ名
- 指標名
- (ISO2000)インターフェイス名
- 登録集名
- (ISO2000)(MF)メソッド名
- 呼び名
- 段落名
- (ISO2000)パラメータ名
- プログラム名
- (ISO2000)(MF)プログラムプロトタイプ名
- 報告書名
- (MF)手順名
- (ISO2000)(MF)(OPEN)画面名
- 節名
- (ANSI85)記号文字
- 原文名
- (ISO2000)利用者関数名

区分番号およびレベル番号を除く利用者語はすべて、上記の互いに重ならない種類のうちのただ 1 つに属する。更に、同じ種類の利用者語は、*一意参照*の節で指定されたものを除き、すべて一意でなければならない。

段落名、節名、レベル番号、区分番号以外の利用者語はすべて、最低限 1 文字の英字、

(MF)またはハイフンを 1 つ含まなければならない。

区分番号およびレベル番号は一意である必要はない。区分番号およびレベル番号は他の区分番号やレベル番号と同じであってもよいし、段落名や節名とも同じであってもよい。

下記の利用者語は操作環境に対して外部化される。

1. 他のプログラムに含まれないプログラムのプログラム名、

   (ISO2000)(MF)クラス名、

   (ISO2000)関数プロトタイプ名、

   (ISO2000)インターフェイス名、

   (ISO2000)(MF)メソッド名、

   (ISO2000)プログラムプロトタイプ名、

   (ISO2000)利用者関数名

2. (ANS85)EXTERNAL 属性を 付して記述された項目のデータ名、ファイル名、およびレコード名

これらの名前のいずれかの代わりに、またはそれに加えて、定数を指定した場合、その定数の内容が操作環境に対する外部名となる。そのさい、大文字と小文字は区別される。定数を指定しなかった場合は、小文字を大文字に変換して外部名が作成される。コンパイラ指令の FOLD-CALL-NAME を使用すると、外部名化したクラス名、インターフェイス名、プログラム名を大文字にするか小文字にするかを制御できる。コンパイラ指令の OOCTRL を使用すると、外部名化したメソッド名を大文字にするか小文字にするかを制御できる。+F と-U を指定すると、メソッド名は小文字にされるこれが省略時の解釈である。

条件名:

条件名 (condition-name) とは、データ項目がとるすべての値の組の中の、特定の値または値の組または値の範囲に付けた名前である。条件名を付けたデータ項目を条件変数 (conditional variable) という。条件名は、データ部または環境部の特殊名段落の中で定義できる。特殊名段落では、実行時スイッチのオンの状態、オフの状態、または両方の状態に命名する。

条件名を使用するのは、次の場合だけである。

1. RERUN (再開) 句。
2. ここで条件名を使用することによって、比較条件を簡略に書き表わすことができる。比較条件の中で条件名を使用して、条件名が割り当てられている値の組の 1 つに対応する条件変数が等しいかどうかを表わす。
3. 対応する値が条件変数に移されることを示す SET 文。

(MF)定数名:

定数名 (constant-name) とは固定的な値に付けた名前である。

呼び名:

呼び名 (mnemonic-name) は、作成者語 (implementor-name) に利用者語を割り当てるものである。この指定は環境部の特殊名段落の中で行う。(*環境部*の章の*特殊名段落*の節を参照。)

段落名:

段落名 (paragraph-name) とは、手続き部の段落に付けた名前である。段落名が等しいと判断されるのは、同じ数の数字または文字が同じ順序で並んでいる場合だけである。

節名:

節名 (section-name) とは、手続き部の節に付けた名前である。節名が等しいと判断されるのは、同じ数の数字または文字が同じ順序で並んでいる場合だけである。

その他の利用者語

その他の利用者語の定義については、*用語集*を参照。

システム名: システム名 (system-name) とは、操作環境との連絡に使用する COBOL の語である。

システム名には、最低 1 つの英字を入れるか、

(MF)または 1 つのハイフンを入れる。

システム名には、以下の 3 種類がある。

1. 計算機名 (computer-name)
2. 作成者語 (implementor-name)
3. 言語名 (language-name)

1 つの処理系の中では、システム名のそれぞれの種類は互いに重なり合わない。1 つのシステム名はただ 1 つの種類に属する。

上記のシステム名は、それぞれ*用語集*に定義を記載してある。

(ANS85)**関数名:** 関数名 (function-name) は、COBOL 原始 要素の中で使用できる語のリストの1つである。関数名と同じ語を利用者語またはシステム名として、原始要素中の別々の文脈において使用できる。ただし、LENGTH と RANDOM と SUM は例外で、利用者語またはシステム名としては使用できない。(*手続き部 - 組み込み関数*の章の*関数の定義*の節を参照。)

予約語: 予約語 (reserved word) とは、COBOL 翻訳集団の中で用いる語のうちの、予め予約されていて、利用者語またはシステム名としては翻訳集団中で使用できないものである。予約後は、一般形式での指定に従ってのみ使用される (付録の*予約語*を参照。)

予約語には、下記の種類がある。

1. 必要語 (key word)
2. 補助語 (optional word)
3. 特殊レジスタ (special register)
4. 表意定数 (figurative constant)
5. 特殊文字語 (special-character word)
6. (SOURCE)(MF)定義済みオブジェクト一意名

必要語:

必要語 (key word) とは、翻訳集団の中である書き方をするときに、必ず書かなければならない語である。それぞれの一般形式の中で、必要語は大文字で記し下線を引いてある。

必要語には、下記の 3 種類がある。

1. ADD (加算) 、READ (読み込み) 、ENTER (導入) などの動詞
2. 文や記述項の書き方に現れる必要な語

3. NEGATIVE (負)、SECTION (節) などの機能的な意味をもつ語

補助語: 補助語 (optional word) とは、それぞれの一般形式の中で、大文字で記してあるが下線を引いてない語である。補助語は書いても書かなくてもよい。補助語を書いたか書かなかったかによって、COBOL 原始要素の意味が変わることはない。

特殊レジスタ: 特殊レジスタ (special register) とは、COBOL の特殊な機能に関連する情報を記憶しておくために、COBOL コンパイラによって作成される記憶領域である。特殊レジスタには名前を付け、その名前によって参照する。これらについては、この節の*特殊レジスタ*の項で後述する。

表意定数: 表意定数 (figurative constant) とは、特殊な定数値に予め付けた名前である。この名前によって、値を参照する。具体的な表意定数については、この節の*表意定数の値*の項で後述する。

特殊文字語: 特殊文字語 (special-character word) とは、特殊な文字の予約語である。

定義済みオブジェクト一意名: あらかじめ定義されたオブジェクト一意名などの予約語である。以下のものがある。

1. (SO2002)(MF)SELF
2. (SO2002)(MF)SUPER
3. (MF)SELFCLASS

(SO2002)**文脈依存語:** 文脈依存語は、一般形式に指定されているようにだけ予約される COBOL の語である。同じ語を、関数名、利用者語、システム名として使用できる。文脈依存語および予約された文脈については、付録の*文脈依存語*を参照。

2.2.3 名前の適用範囲

(ANS85)

ある原始要素の中に別の原始要素が、直接的または間接的に、含まれることがある。各原始要素は、他の原始要素によってまったく同じに名前を付けられた利用者語を使用することができる。(*COBOL の語*の節の利用者語の解説を参照。) このような場合、ある原始要素で参

照した名前は、違う種類の利用者語であってもその原始要素に記述されているものを指すのであって、他の原始要素中に記述されている同じ名前のものを指すのではない。

利用者語のうちの下記の種類に属するものは、利用者語を宣言した原始要素中の文または記述項においてだけ、参照できる。

- 段落名
- 節名

利用者語のうちの下記の種類に属するものは、どの COBOL 原始要素からも参照できる。

- 登録集名
- 原文名

利用者語のうちの下記の種類に属するものは、それらが構成節の中に宣言されている場合、その構成節を含む原始要素またはその原始要素に含まれる原始要素中の文または記述項からだけ、参照できる。

- 符号系名
- 字類名(真値提案用)
- 条件名
- 呼び名
- 記号文字

上記の条件に該当しない場合、利用者語のうちの下記の種類に属するものには、宣言および参照に関して、特別の表記法が適用される。

- (ISO2002)(MF)字類名(オブジェクト指向)
- 条件名
- データ名
- ファイル名
- (ISO2002)(MF)関数プロトタイプ名
- 指標名
- (ISO2002)インターフェイス名
- (ISO2002)(MF)メソッド名
- プログラム名
- (ISO2002)(MF)プログラムプロトタイプ名
- レコード名
- 報告書名
- (ISO2002)(MF)(XOPEN)画面名
- (MF)型定義名
- (ISO2002)利用者関数名

2.2.3.1 プログラム名に関する表記法

(ANS85)

プログラムのプログラム名は、プログラムの見出し部のプログラム名段落で宣言される。プログラム名を参照できるのは、CALL (呼ぶ) 文、

(MF)CHAIN 文、

CANCEL 文、

(MF)SET 文、および

プログラム終了見出しから 1 つの実行単位を構成するいくつかのプログラムに付けられたプログラム名は、必ずしも一意であるとは限らない。しかし、1 つの実行単位中の 2 つのプログラム名が同じである場合、少なくとも一方のプログラムは、もう一方のプログラムを含まない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なければならない。

プログラム名の適用範囲を規定する規則は、下記のとおり。

1. 共通属性がなく、他のプログラムに直接的に含まれるプログラムのプログラム名は、親プログラム中の文からだけ参照できる。

   (OS/390)(3.0.0)(MF)または、プログラムに再帰的な属性がある場合は、そのプログラム自体の内部から参照できる。

2. 共通属性をもち、他のプログラムに直接的に含まれるプログラムのプログラム名は、親プログラム中の文、および親プログラムに直接的または間接的に含まれるプログラム中の文からだけ参照できる。

   (OS/390)(3.0.0)(MF)ただし、共通属性をもつプログラムおよびそのプログラムに含まれるプログラムからは、そのプログラムが再帰的な属性を持つ場合を除き、そのプログラム名を参照できない。

3. 他のプログラムに含まれないプログラムのプログラム名は、実行単位中の他のどのプログラム中の文からも参照できる。ただし、このプログラムに直接的または間接的に含まれるプログラムからは例外とする。

2.2.3.2 条件名、データ名、ファイル名、レコード名、報告書名、(MF)型定義名に関する表記法

(ANS85)

原始要素の中に条件名、データ名、ファイル名、レコード名、報告書名、

(MF)および型定義名

が宣言されている場合、これらの名前はその原始要素の中でだけ参照できる。ただし、これらの名前のうちのいくつかが大域的に使用され、その原始要素中に他の原始要素が含まれる場合は、含まれる他の原始要素からも参照できる。

単一の原始要素の条件名、データ名、ファイル名、レコード名、報告書名、

(MF)および型定義名

に付けた名前の一意性に関する必要条件については、前述 *COBOL の語*の節の利用者語の解説を参照。

ある原始要素の中で、その中に含まれる原始要素中に宣言されている条件名、データ名、ファイル名、レコード名、報告書名、

(MF)および型定義名

は参照できない。

大域名は、それが宣言されている原始要素の中、あるいはその原始要素に直接的または間接的に含まれている原始要素の中から参照できる。

原始要素 A に原始要素 B が直接的に含まれている場合、両方の原始要素で条件名、データ名、ファイル名、レコード名、報告書名、

(MF)および型定義名

をそれぞれ同じ利用者語を使用して設定できる。両方の原始要素中に存在する名前を原始要素 B 中で参照した場合、下記の規則に従って、どちらのものを指すかが判定される。

1. 原始要素 A 中に定義されているすべての大域名、および原始要素 A 及び原始要素 B 中に定義されているすべての名前が、参照された名前がどちらの原始要素に属するものかを判定するために使用される。そして、通常の修飾規則および参照の一意性に関するその他の規則が適用されて、該当するものがいくつか見つけ出される。
2. 該当するものが 1 つしか見つけ出されなかった場合、それが参照されたものである。
3. 該当するものが 2 つ以上見つけ出された場合、原始要素 B 用の名前は 2 つ以上はあり得ない。原始要素 B 用の名前をもつものがないかまたは 1 つある場合、下記の規則を適用する。
    a. 参照された名前が原始要素 B の中で宣言されているならば、原始要素 B 中のものが参照されたとする。
    b. これ以外の場合、原始要素 A が他の原始要素に含まれているならば、下記のように判定する。
        1. 参照された名前が原始要素 A の中で宣言されているならば、原始要素 A 中のものが参照されたとする。
        2. 参照された名前が原始要素 A の中ではなく、原始要素 A を含む原始要素中で宣言されているならば、原始要素 A を含む原始要素中のものが参照されたとする。更に上位の原始要素があるならば、単一の有効な名前が見つかるまで、この規則を順次上に適用する。

2.2.3.3 指標名に関する適用規則

(ANS85)

大域的属性をもつデータ項目が指標名を記述した表を含む場合、その指標名も大域的属性をもつことになる。したがって、指標名の範囲は、その指標名によって名付けられる指標を持つ表に名前を付けるデータ名のものと同じであり、データ名用の名前の範囲規則が適用される。

指標名を修飾することはできない。

(OSVS)指標名は修飾することができる。

2.2.3.4 クラス名（オブジェクト指向の場合）およびインターフェイス名の適用規則

(ISO10613)(MF)

ソース要素内で参照されるクラスのクラス名は、それを含むクラス定義の名前であるか、またはそのソース要素またはそれを含むソース要素の

(MF)またはクラス管理段落

内に宣言されていなければならない。

該当のクラス名に関して翻訳集団内で定義できるクラス定義は1つだけである。その翻訳集団内に定義がなくともかまわない。

ソース要素内で参照されるインターフェイスのインターフェイス名は、それを含むインターフェイス定義の名前であるか、またはそのソース要素またはそれを含むソース要素のリポジトリ段落内に宣言されていなければならない。

該当のインターフェイス名に関して翻訳集団内で定義できるインターフェイス定義は1つだけである。その翻訳集団内に定義がなくともかまわない。

ソース要素のクラス管理段落またはリポジトリ段落内で宣言されたクラス名またはインターフェイス名は、そのソース要素およびそれにネストされるソース要素の中で使用できる。

(MF)ソース要素のクラス管理段落で宣言されたクラス名またはインターフェイス名は、そのソース要素および内包されるソース要素の中で使用できる。

2.2.3.5 メソッド名の適用規則

(ISO10613)(MF)

メソッドのメソッド名はメソッド名段落中に宣言する。メソッド名を参照できるのは、INVOKE 文、メソッド終了見出し、および行中のメソッド呼出しだけである。

クラス定義中に宣言されるメソッドのメソッド名は、そのクラス定義内で一意であるものとする。子クラス内で宣言されるメソッドの名前が親クラスのものと同じであってもかまわない。 ただし、メソッド名段落の条件に従うものとする。

インターフェイス定義中に宣言されるメソッドのメソッド名は、そのインターフェイス定義内で一意であるものとする。継承を受けるインターフェイス内で宣言されるメソッドの名前が継承元のインターフェイスのものと同じであってもかまわない。ただし、メソッド名段落に記述されている条件に従うものとする。

2.2.3.6 関数プロトタイプ名の適用規則

(SO2002) (MF)

ソース要素中で参照される関数プロトタイプ名は、それを含む関数定義であるか、または、レポジトリ段落とソース要素のいずれかで宣言されていなければならない。

関数プロトタイプ名がレポジトリ段落で指定され、かつ、その関数プロトタイプ名を宣言する関数プロトタイプが同じ翻訳集団中で指定されている場合は、指定された関数プロトタイプ名が使用され、外部レポジトリ中のこのプロトタイプの情報は無視される。

2.2.3.7 プログラムプロトタイプ名の適用規則

(SO2002) (MF)

ソース要素中で参照されるプログラムプロトタイプ名は、それを含むプログラム定義のプログラム名、またはレポジトリ段落で宣言されたプログラムプロトタイプ名でなければならない。

プログラムプロトタイプがレポジトリ段落で指定され、かつ、同じプログラムプロトタイプ名を宣言するプログラムプロトタイプが同じ翻訳集団中で指定されている場合は、指定されたプログラムプロトタイプが使用され、このプロトタイプの外部レポジトリ中のこのプロトタイプの情報は無視される。

2.2.4 定数

定数 (literal) とは、以下のいずれかである。

- 文字を並べて値を設定した文字列
- 表意定数を表わす予約語
- (MF) 定数値を参照する利用者語

各定数は、文字定数、数字定数、および各国語型定数のいずれかに属する。

2.2.4.1 文字定数

文字定数 (nonnumeric literal) とは、計算機の文字集合中で利用できる任意の文字を並べて、両端を引用符

(SOM3)(OSV3)(VSC2)(MF)またはアポストロフィ

で区切ったものである。文字定数の長さは 1 文字以上 160 文字以内である。引用符

(SOM3)(OSV3)(VSC2)(MF)またはアポストロフィ

を区切り文字として使用した場合、文字定数内にその区切り文字を含めるには、その文字を 2 つ並べて書く。区切り文字ではない文字を文字定数内に含めるには、その文字を 1 つだけ書く。ランタイム要素中での文字定数の値は文字の列そのものである。ただし、次の条件がある。

- 両端の引用符は含まれない
- 中に含まれる引用符は 2 つ並んだものが 1 つの文字を表わす

その他の句読文字はすべて、分離符ではなく、文字定数の一部を構成する。すべての文字定数の項類 (category) は英数字 (alphanumeric) である。(*データ部 - データ記述*の章の *PICTURE(形式)句*の節を参照。)

(VSC2)(MF)さらに、16 進数を文字定数として扱うことができる。このためには、X"*nn*" という形式で文字定数を書き表わす。ここで、*n* は 0-9 と A-F の 16 進文字を表わす。*nn* は 160 回まで繰り返すことができる。ただし、n の個数は偶数とする。

(MF)16 進桁の個数は奇数でもよい。

2.2.4.2 数字定数

数字定数 (numeric literal) は、固定小数点数でも浮動小数点数でもよい。

2.2.4.2.1 固定小数点数の数字定数

数字定数は、"0"から"9"までの数字と正号、負号、小数点からなる文字列である。この処理系では、数字定数の長さは 1 文字以上 18 文字以内である。数字定数を作る際には、下記の規則が適用される。

- 数字定数は、最低 1 文字を含まなければならない。
- 符号文字を複数書くことはできない。符号を付けるときは、左端に置く。符号を付けないと、正の数となる。
- 小数点を複数書くことはできない。小数点は仮想小数点として扱われる。小数点を置く位置は右端以外のどこでもよい。小数点を付けないと、整数となる。
- 数字定数の値は数字定数中の文字によって表される代数値である。数字定数の項類は数字 (numeric) である。(*データ部 - データ記述*の章の *PICTURE(形式)句*の節を参照。)

上記の規則に適合するが、引用符で囲まれている定数は、文字定数として扱われる。

標準データ形式の文字で表した数字定数の大きさは、利用者が指定した数字の桁数に等しいものとする。

(MF)さらに、16 進数を数字定数として扱うことができる。このためには、H"*nn*" という形式で文字定数を書き表わす。ここで、*n* は 0-9 と A-F の 16 進文字を表わす。*nn* は 8 回まで繰り返すことができる。ただし、n の個数は偶数とする。

2.2.4.2.2 浮動小数点数の数字定数

(OSVS)(VSC2)(MF)

浮動小数点数定数の書き方は下記のとおり。

仮数部と指数部の符号は、付けても付けなくてもよい。符号を省略すると、正の数として扱われる。

仮数部の長さは、1 文字以上 16 文字以内である。仮数部には小数点を含めなければならない。

指数部は、E に続けて符号を記した後に、1 文字または 2 文字で指定する。ただし、符号は付けても付けなくてもよい。

浮動小数点定数の値の範囲は、0.54E-78 から 0.72E+76 までである。この範囲から外れる値に対しては診断メッセージが出され、値はそれぞれ 0 または 0.72E+76 で置き換えられる。整数の数値定数が必要なときに、浮動小数点数の数値定数を使用しないように、注意すること。

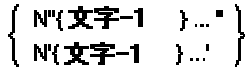
2.2.4.3 各国語型定数

(ISO2002)(MF)

各国語型定数は、計算機内の保存場所で同一サイズの文字で表示される、一連の各国語文字である。詳しくは、使用している COBOL システムの各国語データ (Unicode) に関する文書を参照。

一般形式

形式 1



形式 2

```
{ NX"{16進文字列-1    }..."  }
{ NX'{16進文字列-1    }...'  }
```

構文規則

全形式共通

1. 各国語型定数の長さは、分離符を除き、0 より大きく 160 以上の文字数でなければならない。
2. 文字、1 は、各国語文字の間に対応関係のあるいずれかの文字コードである。

形式 1

3. 左側の区切り記号で使用される引用記号に符合する、2 個の連続した引用符号は、定数では、1 個の引用符号と見なされる。この引用符号は、左側の引用符号と同じ文字コードセットに属するものでなければならない。

形式 2

4. 各 16 進文字列-1 は、上位のバイトから順に符号化された 4 桁の 16 進数 (2 バイト) である。

一般規則

全形式共通

1. 各国語型定数を区切る分離符は、各国語型定数の値には含まれない。
2. 各国語型定数は、クラスおよび種別としての国を表す。

形式 1

3. 翻訳時の定数の値は、計算機の翻訳時にコーディングされた文字セットに含まれる文字-1 の個数である。

   実行時の定数の値は、定数の翻訳時の値の、対応する実行時の値への変換の結果である各国語文字である。

形式 2

4. 実行時の定数の値は、各国語文字である。各文字は、1 個だけ含まれる 16 進文字列-1 により指定されたビット構成を含む。

2.2.4.4 表意定数の値

表意定数の値は、COBOL システムによって作り出される。その値を、下記の予約語を使用して参照する。表意定数として使用するときは、この予約語を引用符で囲んではならない。表意定数の単数形と複数形の値は同じであるので、どちらを使用してもよい。

表意定数の値、およびそれらを参照するために使用する予約語を表 2-1 に示す。

表 2-1: 表意定数の値と対応する予約語

定数	内容
ZERO ZEROS ZEROES	値"0" を表わす。文脈によっては 1 つ以上の文字"0" を表わす
SPACE SPACES	計算機の文字集合から 1 つ以上の空白文字を表わす
HIGH-VALUE HIGH-VALUES	文字の照合順序の最も高い文字を 1 つ以上表わす(拡張 ASCII 文字集合では x"FF")
LOW-VALUE LOW-VALUES	文字の照合順序の最も低い文字を 1 つ以上表わす(拡張 ASCII 文字集合では x"00")
QUOTE QUOTES	1 つ以上の文字「" 」を表わす。数値定数をくくるために、原始プログラム中で引用符の代りに QUOTE または QOUTES を使用することはできない。したがって、"ABD" を表わすために、QUOTE ABD QUOTE と記すのは誤りである。
ALL literal	指定された文字が何文字か含まれる文字列を表わす。定数は、文字定数または ISO2002 MF 各国語型定数、または 表意定数でなければならない。ただし、定数として ALL を指定することはできない。 ISO2002 MF XOPEN この文字定数または各国語型定数は、連結式の場合もある。 定数として表意定数を指定した場合、ALL は読みやすくするためだけに用いられる。
VSC2 MF NULL NULLS	1 つ以上の未設定ポインタ値 MF DB2V8 または手続きポインタ値 を表わす。USAGE POINTER

	MF DB370または PROCEDURE-POINTER を持つデータ項目、および値が NULL であるポインタ変数はどのデータ項目 MF DB370または手続き も指さないことが保証される。 NULL 値は環境間で変化し、通常、各環境用の COBOL 以外の言語で使用される等しい値と一致する。

表意定数が何文字かの文字列を表わす場合、その長さは文脈に応じて、COBOL システムによって決定される。その際、以下の規則が順に適用される。

1. SOLARIS MF XOPEN表意定数が連結式として指定されているときは、文字列の長さは 1 文字となる。
2. 表意定数を VALUE (値) 句内で指定した場合、または表意定数を他のデータ項目と関係付けた (たとえば、表意定数を他のデータ項目に転記したり、他のデータ項目と比較したりした) 場合、表意定数に指定した文字が 1 文字ずつ右方向に継ぎ足され、その長さが対応するデータ項目と等しいかそれより大きくなったところで止められる。次いで、得られた文字列が右側から順次切り詰められ、残っている文字数が 1 または対応するデータ項目の長さのどちらか大きい方と等しくなったところで止められる。JUSTIFY (けたよせ) 句が指定されている場合、この処理はそれよりも先に独立して行われる。
3. 「ALL 定数」以外の表意定数のときは、文字列の長さは 1 文字となる。
4. 文字列の長さは定数の長さとなる。

   MF DISPLAY 文の形式 3 における表意定数の使用は、その一般規則で説明する通り、特別な効果を持つ。

形式に定数が示されているところでは、どこでも表意定数を使用できる。ただし、数字定数に限定されている箇所では、表意定数は ZERO (ZEROS, ZEROES) だけを使用できる。

表意定数の HIGH-VALUE (S) または LOW-VALUE (S) を使用した場合、, 実際に表意定数が表わす文字は、指定されている文字の照合順序によって決まる。(*環境部*の章の*実行用計算機段落*および*特殊名段落*の節を参照。)

表意定数を表わす予約語は、それぞれが独立した文字列である。ただし、「ALL 定数」は例外で、2 つの別々の文字列から構成される。

OSVS VSC2 MF表意定数の QUOTE/QUOTES の値は、指令の APOST および QUOTE の影響を受ける。

(ANS85)定数の長さが 2 桁以上ある表意定数の「ALL 定数」を数字項目または数字編集項目に関係付けることは、ANSI '85 標準では廃要素に分類される。これは ANSI 標準の次の全面改訂時に削除される予定である。

(MF)この COBOL 処理系に組み込まれている方言は、この構文を全面的に使用できる。FLAGSTD 指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。

(XOPEN)標準 COBOL 定義の一環であるにもかかわらず、この構文は X/Open の COBOL 言語定義からは明示的に除外されている。したがって、X/Open に準拠する COBOL 原始プログラムの中では、この構文を使用すべきではない。

2.2.4.5 定数名

(MF)

定数名 (constant-name) とは、データ部の 78 レベルのデータ記述項 (data description entry) に指定した利用者語である。形式に定数が示されているところでは、どこにでも定数名を使用できる。定数名の働きは、そのデータ記述項に値として指定した定数を書いたのと同じ結果が得られることにある。形式に整数の定数が示されているところでは、整数の値をもつ定数名を使用することができる。例としては、レベル番号、区分番号、PICTURE 文字列が挙げられる。

定数名は、定義した後でだけ使用できる。定数名は前方参照の対象とはならない。

2.2.4.6 連結式

(ISO2002)(MF)(XOPEN)

連結式は、連結演算子で分離された 2 個の作用対象から成る。

一般形式

```
{定数-1   }
{連結式-1 }& 定数-2
```

構文規則

1. 両方の作用対象は同じ字類とする。ただし、表意定数は1つまたは両方の作用対象を構成してもよい。作用対象は数字にはできない。定数-1 も定数-2 も、語 ALL で始まる表意定数にはできない。
2. (ISO2002)連結の結果である値のサイズは、160 文字以下でなければならない。

一般規則

1. 連結演算の結果としての連結式の字類は、以下のいずれかとなる。

a. 作用対象の1つが表意定数の場合、他の作用対象を構成する連結式または定数の字類
b. 作用対象の両方が表意定数の場合、字類は英数字である
c. 作用対象と同じ字類

2. 連結式の値は、それを構成する定数、表意定数、連結式の値を連結したものである。
3. 連結式は、同じ字類および値をもつ定数とまったく等しい。その字類の定数を使えるところには、どこにでも連結式を使用できる。

2.2.5 特殊レジスタ

特殊レジスタ (special register) は、COBOL システムによって作成されるデータ項目または一時的な値である。特殊レジスタを参照するには、対応する名前または式を使用する(表 2-2 参照)。 その際、以下に示す特別な規則が適用される。また、特殊レジスタには暗黙のデータ記述 (PICTURE) が想定される。

表 2-2: 特殊レジスタ

特殊レジスタ名または式	暗黙のデータ記述 PICTURE	使用法
VSC2 MF ADDRESS OF データ名 1	USAGE IS POINTER	データ名-1 の番地を示すポインタ値を生成する。この式は、使用する文で一般形式として明示的に指定する。データ名-1 は、連絡節で 01 レベルおよび 77 レベルのデータ項目として 宣言される。 MF または、データ部の各所で各レベル番号で宣言する。
OSVS CURRENT-DATE[1]	X(8)	現在の日付が記録される。(この日付けは、COBOL 実行環境から提供される。) 次の形式をしている。 *MM/DD/YY* *MM* は月を、 *DD* は日を、*YY* は年 (1900 年からの下 2 桁) を表わす数字である。 CURRENT-DATE は MOVE 文の送出し側としてだけ使用できる。
DEBUG-ITEM	可変長のグループ項目	デバッギング節が実行された原因となった理由に関する情報を表す。詳しくは、***言語リファレンス - 追加トピック***の*デバッグモジュール*の章を参照。
VSC2 MF LENGTH OF データ名-2[2]	9(9)	データ名-2 によって使用される、記憶領域の現在の バイト数を示す値を生成する。この式は、数字データ項目を使用できるところであればどこでも使用できる。ただし、添字付けまたは部

		分参照は例外とする。 MF レベル 78 項目の値を設定して使用することもできる。
LINAGE-COUNTER		レコード順ファイル記述の中に LINAGE 句が存在する場合に生成される。整数 1 または、LINAGE 句のデータ名 1 が参照するデータ項目と同じサイズの、符号のつかない整数を説明しているとされる。
OSVS VSC2 MF RETURN-CODE[3]	S9(4) COMP OPEN S9(9) COMP	以下が可能となる。 • プログラムによって値を設定する。STOP RUN 文、EXIT PROGRAM 文、または GOBACK 文を実行する前に値を設定することで、呼出し元のランタイム要素 (または実行環境) に値を渡すことができる。 • 他の COBOL プログラムを CALL した後で読み出して、呼ばれたプログラムによって設定された RETURN-CODE の値を入手する。 プログラムの実行を最初に開始するときには、そのプログラムの RETURN-CODE はゼロに設定される。手続き部の文の中で基本データ項目を参照できるところならば、どこでも RETURN-CODE をデータ名として使用できる。
VSC2 SHIFT-IN	X(1)	文字の表現形式を 2 バイト文字から 1 バイト文字に戻すために使用する。該当する環境において用いる。
VSC2 SHIFT-OUT	X(1)	文字の表現形式を 1 バイト文字から 2 バイト文字に切り替えるために使用する。該当する環境において用いる。
VSC2 SORT-CONTROL OSVS VSC2 SORT-CORE-SIZE SORT-FILE-SIZE SORT-MESSAGE SORT-MODE-SIZE	X(8) S9(8) COMP S9(8) COMP X(8) S9(5) COMP	これらの特殊レジスタを手続き部の中で参照できる。ただし、その値はゼロ (数字レジスタの場合) または空白 (英数字レジスタの場合) である。
OSVS VSC2 MF SORT-	S9(4) COMP	SORT 手続きを異常終了させるために使用で

RETURN		きる。このレジスタに値 16 を入れると、次の RELEASE または RETURN の後で SORT 処理は停止される。
OSVS VSC2 TALLY	9(5) COMP	EXAMINE...TALLYING 文によって作成される情報が記録される。手続き部の文の中で基本データ項目を参照できるところならば、どこでも TALLY をデータ名として使用できる。
OSVS TIME-OF-DAY	9(6) DISPLAY	現在の時刻 (24 時間制) が記録される。(この時刻は COBOL 実行環境から提供される)次の形式をしている。 *hhmmss* *hh* は時間を、*mm* は分を、*ss* は秒を表わす数字である。TIME-OF-DAY は、MOVE 文の送出し側としてだけ使用できる。
OSVS WHEN-COMPILED	X(20)	COBOL 翻訳集団が COBOL システムに投入された時刻と日付が記録される。次の形式をしている。 *hh.mm.ssMMM DD, YYYY* *hh* は時間 (24 時間制) を、*mm* は分を、*ss* は秒を、*MMM* は月の名前 (頭の 3 文字) 、*DD* は日を、*YYYY* は年を表わす。 WHEN-COMPILED は MOVE 文の送出し側としてだけ使用できる。
VSC2 WHEN-COMPILED	X(20)	COBOL 翻訳集団が COBOL システムに投入された時刻と日付が記録される。次の形式をしている。 *MM/DD/YYhh.mm.ss* *DD*、*hh*、*mm*、*ss* は上記の通り。*YY* は年の下 2 桁、*MM* は月を表わす。 WHEN-COMPILED は MOVE 文の送出し側としてだけ使用できる。
OS/390 MF XML-CODE	S9(9) COMP	XML パーサと、XML PARSE 文に記述された処理手順との間で状況を伝達するために使用される。XML パーサは、各イベントの XML-CODE を解析の終了時に設定する。利用者は、通常のイベントの後に、処理手順の XML-CODE を-1 にリセットすることができ、これにより、XML パーサが利用者の操作による例外として終了したことが示される。これは、返された XML-CODE の値、-1 で示される EXCEPTION XML イベントとは区別される。

OS/390 MF XML-EVENT[4]	X(30)	XML パーサからのイベント情報を、XML PARSE 文に記述された処理手順に伝達するために使用される。XML パーサは、制御を処理手順に渡す前に、XML-EVENT 特殊レジスタを XML イベントの名前に設定する。XML-EVENT はデータを受領する項目として使用することはできない。
OS/390 MF XML-NTEXT[4]		XML 解析中に定義され、USAGE NATIONAL である文書片を含む。XML-NTEXT は、含まれている XML 文書片のサイズの、初歩的な各国語データ項目である。XML-NTEXT のサイズは、実行時に動的に変化する。 XML PARSE 文の演算数が各国語データ項目である場合、ATTRIBUTE-NATIONAL-CHARACTER および CONTENT-NATIONAL-CHARACTER イベントに関しては、XML パーサは、XML-NTEXT をイベントに関連づけられた文書片に設定した後で、制御を処理手順に渡す。 XML-NTEXT が設定されている場合、XML-TEXT 特殊レジスタのサイズは 0 となる。どの時点でも、サイズが 0 でないのは、XML-NTEXT と XML-TEXT 特殊レジスタのいずれかのみである。 XML-NTEXT に含まれる各国語文字の数を決定するには、LENGTH 関数を使用すること。 XML-NTEXT は、受領する項目としては使用できない。
OS/390 MF XML-TEXT[4]		クラス文字である文書片を内包するための XML 解析中に定義される。XML-TEXT は、内包される XML 文書片のサイズの、初歩的な文字データ項目である。XML-TEXT のサイズは、実行時に動的に変化する。 XML PARSE 文の演算数が文字データ項目である場合、ATTRIBUTE-NATIONAL-CHARACTER および CONTENT-NATIONAL-CHARACTER イベントを除き、XML パーサは、XML-TEXT をイベントに関連づけられた文書片に設定した後で、制御を処理手順に渡

		す。 XML-TEXT が設定されている場合、XML-NTEXT 特殊レジスタのサイズは 0 となる。どの時点でも、サイズが 0 でないのは、XML-NTEXT と XML-TEXT 特殊レジスタのいずれかのみである。 XML-TEXT に含まれるバイト数を決定するには、LENGTH 関数または、XML-TEXT 用 LENGTH OF 特殊レジスタを使用すること。 XML-TEXT は、受領する項目としては使用できない。

[1] CURRENT-DATE 特殊レジスタの内容の形式は、CURRENT-DATE 指令の影響を受ける。

[2] LENGTH OF 特殊レジスタは、Micro Focus 方言では英数字定数で続けても構わない。

[3] RETURN-CODE 特殊レジスタのサイズは、XOPEN 指令および RTNCODE-SIZE 指令の影響を受ける。

[4] XML-TEXT および XML-NTEXT の内容は、XML-EVENT の内容により変化する。詳しくは、表 2-3 を参照。

表 2-3：XML-EVENT および XML-NTEXT または XML-TEXT 特殊レジスタの内容

XML-EVENT の内容	XML-TEXT または XML-NTEXT の内容
ATTRIBUTE-CHARACTER	属性値の中の、定義済みの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に囲まれた値。値に参照が含まれる場合は、属性値に含まれる文字列の一部でもあり得る。
ATTRIBUTE-NAME	属性名で、=の左にある文字列。

ATTRIBUTE-NATIONAL-CHARACTER	XML PARSE 文中の一意名-1で指定されたXML 文書の種類にかかわらず、XML-TEXTは空 (から) で、XML-NTEXTは、数字参照に対応する単一の各国語文字を含む。
COMMENT	左側の文字列、"<!--"と右側の文字列、"-->"の間の注釈テキスト。
CONTENT-CHARACTER	要素内容の中の、定義済みの参照に対応する単一の文字。
CONTENT-CHARACTERS	開始タグと終了タグの間の要素内容。ここに他の要素への参照が含まれる場合は、要素内容中の文字列の一部でもあり得る。
CONTENT-NATIONAL-CHARACTER	XML PARSE 文中の一意名-1で指定されたXML 文書の種類にかかわらず、XML-TEXTは空 (から) で、XML-NTEXTは、数字参照に対応する単一の各国語文字を含む(1)。
DOCUMENT-TYPE-DECLARATION	左右の文字

	列、"<!DOCTYPE"および">"を含む、文書種別宣言全体。
ENCODING-DECLARATION	引用符またはアポストロフィに囲まれたXML 宣言中の符号化宣言の値。
END-OF-CDATA-SECTION	つねに文字列、"]]>"を含む。
END-OF-DOCUMENT	空で、サイズは0。
END-OF-ELEMENT	終了要素タグまたは空要素タグの名。
EXCEPTION	走査が完了した文書の一部で、例外が検出された場所までの部分(2)。特殊レジスタ、XML-CODE は、例外を示す一意のエラー符号を含む(3)。
PROCESSING-INSTRUCTION-DATA	右側の文字列、"?>"を除く、処理指示の残りの部分。ただし、直後の空白文字は含まれるが、直前の空白文字は含まれない。
PROCESSING-INSTRUCTION-TARGET	処理指示の対象の名で、処理指示の左側の文字列、"<?"の直後に置かれる。.

STANDALONE-DECLARATION	引用符またはアポストロフィに囲まれた、XML 宣言中の独立した宣言の値。
START-OF-CDATA-SECTION	つねに、文字列、"<![CDATA["を含む。
START-OF-DOCUMENT	文書全体。
START-OF-ELEMENT	開始要素タグまたは終了要素タグの名。要素種別としても知られる。
UNKNOWN-REFERENCE-IN-CONTENT	参照名。ただし、区切り記号、"&"および";"を除く。
UNKNOWN-REFERENCE-IN-ATTRIBUTE	参照名。ただし、区切り記号、"&"および";"を除く。
VERSION-INFORMATION	引用符またはアポストロフィに囲まれた、XML 宣言中の版宣言の値。これはつねに"1.0"。

(1) 65,535 (NX"FFFF")より大きいスカラ値を持つ各国語文字は、2 つの符号化単位 (代理組) を使用して表記される。この符号化単位が、XML-NTEXT の内容についての操作により分離されないよう注意する必要がある。2 つの符号化単位で 1 つの図形文字を形成するため、分離すると無効なデータとなる。

(2) 符号化の競合の例外は、解析が開始される前に報告される。これらの例外がある場合は、XML-TEXT は、サイズが 0 であるか、文書からの符号化宣言の値のみを含む。

(3) XML 例外符号について詳しくは、*IBM Enterprise COBOL Programming Guide* を参照。このマニュアルに記載されていない例外には必ず、201 という値が返される。

2.2.6 定義済みオブジェクト一意名

定義済みオブジェクト一意名は、以下の通り。

定義済みオブジェクト一意名	使用方法
(2002)(MF)SELF	現在のメソッドの実行対象となっているオブジェクトを参照する。メソッドの手続き部で使用できる。SELF が使用されているメソッドを呼び出すために使用されたオブジェクトを参照する。メソッド呼出しに SELF を指定すると、該当のオブジェクトに関して宣言されているすべてのメソッドが検索の対象となる。
(MF)SELFCLASS	現在のオブジェクト (SELF) のクラス・オブジェクトであるオブジェクトを参照する。SELF 自体がクラス・オブジェクトである場合は、SELFCLASS はシステム・クラスの BEHAVIOR となる。クラス・オブジェクトの BEHAVIOR は自己参照を終了させる働きをする。(つまり、SELF がクラスの BEHAVIOR であるならば、SELFCLASS もそうである。)
(2002)(MF)SUPER	現在のメソッドの実行対象となっているオブジェクトを参照する。メソッドの手続き部で使用できる。INVOKE 文でメソッドを呼び出すのに使用されたオブジェクトでありうる。SELF が使用されているメソッドを呼び出すために使用されたオブジェクトを参照する。メソッド呼出しに SUPER を指定すると、実行中のメソッドと同じクラス内に定義されているすべてのメソッドが検索の対象外とされる。
(2002)(MF)NULL	空のオブジェクト参照値を参照する。それはオブジェクトを絶対に参照しないことが保証された固有の値である。NULL は暗黙的にクラス・オブジェクトおよび項類オブジェクト参照として記述されており、一般的オブジェクト参照ではない。受取り側の作用対象に NULL を指定してはならない。

2.2.7 PICTURE 文字列

PICTURE 文字列は、記号として使用される COBOL の文字集合の中の、ある種の文字を組み合わせたものである。PICTURE 文字列の構成および使用上の規則の詳細については、***データ部 - データ記述***の章の *PICTURE(形式)句*節を参照。

PICTURE 文字列の指定の中に現れる句読文字は、句読文字としては解釈されない。それらはその PICTURE 文字列の指定の中で使われている記号として解釈される。

2.2.8 注記項

注記項 (comment-entry) は見出し部の記述項であり、計算機の文字集合中の任意の文字の組合せからなる。注記項は注記を書く目的でのみ使用する。1行以上書くことができ、行の A 領域で、予約語である部名、節名、段落名のいずれかが出たところ、

(OSVS)または任意の文字が出たところ

で終了される。標識領域にハイフンを書いて注記項を続けることは許されない。

2.3 形式と規則

2.3.1 一般形式

一般形式 (general format) とは句や文の要素の並べ方を指す。このマニュアルでは、句または文を定義する記述のすぐ後ろに一般形式を示す。2 通り以上の並べ方がある場合には、一般形式をいくつかに分けて番号を付けて示す。句は一般形式に示された順番どおりに書かなければならない。任意に指定する句でも、指定する場合には所定の位置に書かなければならない。場合によっては、示された以外の順序で句を記してもよいことがある。その場合は、関連する規則にその旨を明記する。適用範囲、必要条件、制限事項は規則の項に説明する。

2.3.2 構文規則

構文規則 (syntax rule) とは、語や要素を並べて句や文などの大きな要素にまとめる際の規則を指す。構文規則によって、個々の語や要素に制限が課される場合もある。

構文規則は文の形式を定義したり、明確に規定するものである。つまり、文の要素を並べる順序や各要素が表現するものに関する制限を規定する。

2.3.3 一般規則

一般規則 (general rule) とは、単一の要素または一連の要素の意味または意味の関連を定義したり明確に規定したりする規則を指す。この一般規則によって、文の意味や、実行または中間コードの生成に文が与える影響が明確にされる。

2.3.4 要素

句や文を構成する要素 (element) には、大文字で記した語、小文字で記した語、レベル番号、角かっこ、中かっこ、連結語および特殊文字がある。

2.4 機種に依存しないデータ記述の概念

データを、できるかぎり機種に依存しないものとするために、データの性質や特性は装置向けの形式ではなく、標準データ形式で記述する。この標準データ形式は一般のデータ処理応用に向いている。10 進数は、計算機の基数系にかかわらず、数値を表現するために使用する。COBOL 文字集合中のその他の文字は、文字データ項目を表現するために使用する。

2.5 レベルの概念

レコードは、レベル構造の概念に従って構成される。この概念は、データを参照するためにレコードを細分化する必要があることから生ずる。いったん細分化したものをさらに細分して、もっと細かいデータとして参照できる。

レコードを最も基本的な単位に細分したもの、つまりそれ以上細分できないものを、基本項目 (elementary item) という。したがって、レコードは一連の基本項目から構成されるといえる。または、レコード自体が 1 つの基本項目である場合もある。

一組の基本項目を参照するために、それらをまとめて集団にする。各集団はいくつかの基本項目を並べて名前を付けたものである。さらに、集団をいくつかまとめて集団とすることもできる。したがって、ある基本項目は何階層もの集団に属することがある。

2.5.1 レベル番号

レベル番号 (level-number) は、基本項目と集団項目 (group item) の構成を表わす。レコードはデータ項目のうちの最も包括的なものであるので、レコードのレベル番号は 01 から始まる。レコードに含まれるデータ項目には、01 より大きいレベル番号を付ける。ただし、レベル番号は連続している必要はなく、49 を超えないものとする。1 レコードに含められるレベル数の最大は 49 である。この規則の例外である特殊なレベル番号として、66, 77,

MF 78

および 88 がある (下記を参照) 。各レベル番号を用いるごとに、それぞれ記述項を書く。

ある集団は、それに続く集団項目か基本項目のレベル番号に、その集団のレベル番号に等しいか、小さいものが出てくるまでのすべてを含む。ある集団項目に直接属する項目はすべてその集団項目のレベル番号よりも大きな同一のレベル番号を使用しなければならない。

OSVS VSC2 この規則は強制されない。

例:

正しい	OSVS VSC2 MF 正しくないが、許される

01 A.
  05 C-1.
  10 D PICTURE X.
  10 E PICTURE X.
  05 C-2.
	01 A.
	  05 C-1.
	  10 D PICTURE X.
	  10 E PICTURE X.
	  04 C-2.

レベルの概念が当てはまらない記述項が、以下の 4 種類存在する。

1. RENAMES (再命名) 句で作った基本項目または集団項目の記述項
2. 作業場所節および連絡節で独立の項目を指定する記述項
3. 条件名を指定する記述項
4. MF 定数名を指定する記述項

RENAMES 句を用いてデータ項目を再編成するための記述項には、特別のレベル番号 66 を使用する。

他の項目を細分したのではなく、それ自体も細分されない、独立のデータ項目用の記述項には、特別のレベル番号 77 を使用する。

条件変数の特定の値に関連付ける条件名を指定するための記述項には、特別のレベル番号 88 を使用する。

(MF)特定の定数の値に関連付ける定数名を指定するための記述項には、特別のレベル番号 78 を使用する。

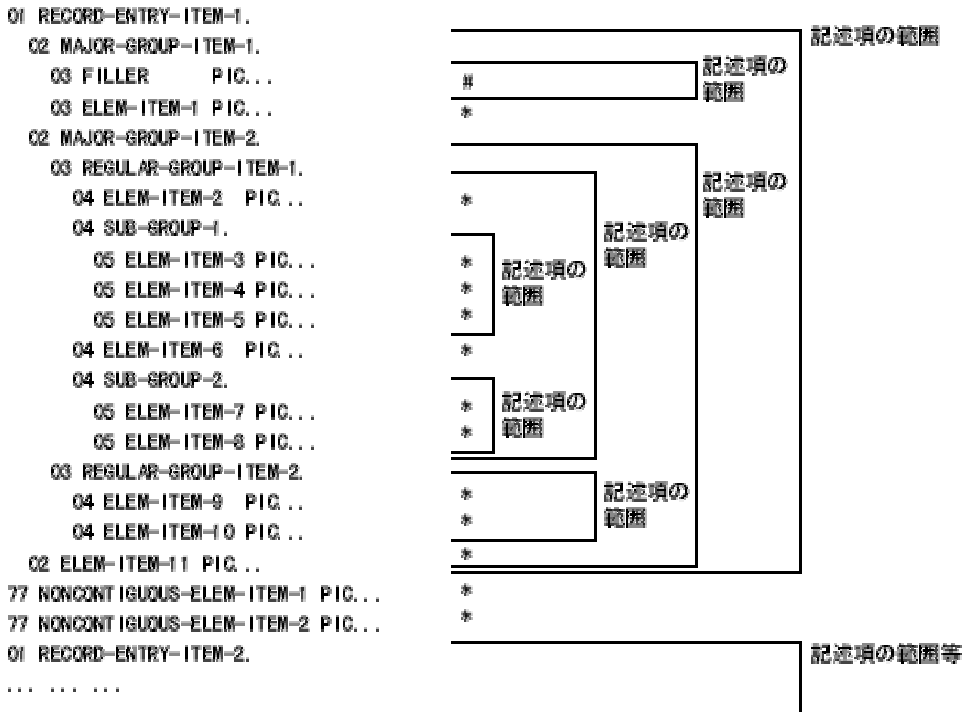


図 2-1: 階層構造をとるレベル番号の例

COBOL 原始コードを段落付けで書くのは読みやすくするためであって、文法的に必要なわけではない。

基本項目は、定義上、下位レベルの (レベル番号の大きい) 記述項が後ろに続かない項目である。基本項目には、記憶域を定義しなければならない。(*データ部 - データ記述*の章の *PICTURE(形式)句*および *USAGE(用途)句*の各節を参照。)

基本項目(上記で"*"を付けたもの)および FILLER (無名) 項目(上記で"#"を付けたもの) だけに記憶域が明示的に取られることに注意すること (これは PICTURE 句の働きによる) 。集団項目の記憶域は、それに属する下位項目の大きさと桁詰めに必要なバイト数に基づいて、暗黙的に取られる。(*データ部 - データ記述*の章の *SYNCHRONIZED(桁詰め)句*節を参照。)

では、分かりやすくするために、レベル番号を連続的に付けた。しかし実際には、レベル番号を連続的に付ける必要はない。したがって、01 の次の下位レベルのレベル番号が 05、さらにその下位レベルのレベル番号が 10 といった具合いにしてもよい。

図のデータ・レコード用に取られる記憶域は以下のように構成される。

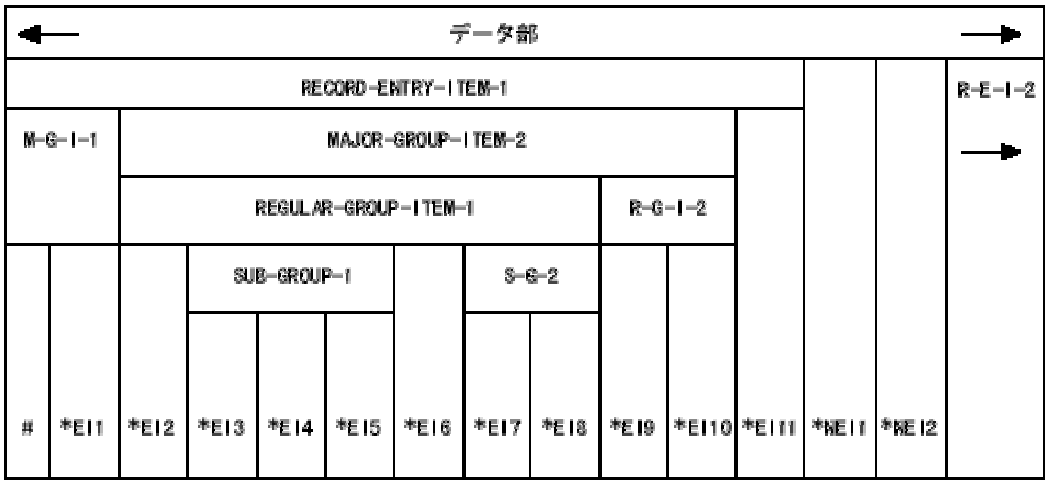


図 2-2: データ・レコード記憶域の割り当て

ここで、

R-E-I	はレコード・エントリ項目(Record-Entry-Item)
M-G-I	は主集団項目(Major-Group-Item)
R-G-I	は一般集団項目(Regular-Group-Item)
S-G	は従集団(Sub-Group)
EI	は基本項目(Elementary-Item)
NEI	は独立基本項目(Noncontiguous Elementary-Item)

2.6 データの字類と項類

基本データ項目、定数、関数は、それぞれ字類と項類を持つ。データ項目の字類と項類は、その PICTURE 文字列か、BLANK WHEN ZERO 句か、用途によって定義する。定数の字類と項類については、前述の*定数*節を参照。

(ANS85)また、組み込み関数の字類と項類は、組み込み関数の定義によって記述する。(*手続き部 - 組み込み関数*の章を参照。

集団項目の項類は英数字である。

基本項目の字類と項類の関係を、表にして下に示す。

図 2-3: 基本項目の字類と項類の関係	
字類	項類
英字	英字
英数字	集 英数字 ISC2 MF 2 バイト文字
指標	指標
各国語	各国語」
数字	数字 OSVS ISC2 MF 内部浮動小数点数 OSVS ISC2 MF 外部浮動小数点数
オブジェクト	SO2000 MF オブジェクト参照
ポインタ	ISC2 MF SO2000 ポインタ MF OB37 プログラムポインタ

2.6.1 算術符号

算術符号は、次の 2 つの項類に分類される。

1. 演算符号 (operational sign) は、符号付き数字データ項目または符号付き数字定数に付けて、その代数的特性を示すものである。
2. 編集用符号 (editing sign) は、編集された報告書上で、項目の符号を表示するために使用する。

SIGN (符号) 句を使用して、演算符号の位置を明示的に指定できる。この句は指定してもしなくてもよい。演算符号については、後述の*文字の表現と基数の選定*節を参照。

編集用符号は、PICTURE 句の符号編集用文字を用いて、データ項目に挿入する。

2.6.2 標準桁寄せ規則

基本項目内にデータを収めるときの標準規則は、受取り側データ項目の項類によって決まる。その規則は以下のとおり。

1. 受取り側データ項目が数字である場合
   a. データは小数点の位置を合わせて、受取り側に収められる。このとき、必要に応じて、端にゼロが補われたり、端が切り捨てられたりする。
   b. 想定小数点 (assumed decimal point) を明示的に指定していない場合、データ項目の右端に想定小数点があるものとみなされ、後は上記 a.と同様に扱われる。
2. 受取り側データ項目が数字編集データ項目である場合、データは小数点の位置を合わせて、受取り側に収められる。このとき、必要に応じて、端にゼロが補われたり、端

が切り捨てられたりする。ただし、編集操作によって先行ゼロ列 (leading zeros) が置き換えられた場合は、ゼロは補われない。

3. 受取り側データ項目が英数字 (英数字編集を除く) 、英数字編集、英字のデータ項目である場合、送出し側データは左側を揃えて受取り側データに転記される。このとき、必要に応じて、右端に空白が補われる場合や、右端が切り捨てられる場合がある。
4. OSVS VSC2 MF 受取り側データ項目が外部浮動小数点数である場合、データは左端の文字位置をそろえられる。それに応じて、指数部が調整される。

受取り側に JUSTIFIED (桁寄せ) 句を指定した場合、*データ部 - データ記述*の章の *JUSTIFIED(桁寄せ)句*節の記述に従って、桁寄せ規則が修正される。

2.6.3 実行用プログラムの効率を高めるための項目の桁詰め

ある種の計算機の記憶装置は、番地付けのための境界 (語の境界、半語の境界、バイトの境界) をもつように構成されている。データの格納では、 これらの境界を意識する必要はない。

しかし、データのある種の使い方 (算術演算や添字など) では、データが番地付けのための境界に合わせて記憶されている方が効率のよい場合がある。特に、複数個のデータ項目の部分が隣り合う 2 つの境界の間に含まれていたり、1 つのデータ項目が境界によって分断されていたりすると、これらのデータを呼び出したり記憶したりするための、余分な機械命令がランタイム要素の中で必要になる。

余分な機械命令を必要としないように、データ項目を境界に合わせて配置することを、桁詰め (synchronization) という。桁詰めされた項目はその形式に合わせて処理される。桁詰めされた形式への変換は、項目へデータを収める手続き (READ と WRITE を除く) を実行するときにだけ行われる。

桁詰めを行う方法には、下記の 2 通りがある。

1. SYNCHRONIZED (桁詰め) 句を使用する。
2. SYNCHRONIZED 句を使わずに、適切な境界に合わせてデータを構成する。

SYNCHRONIZED 句を使用して集団内で特別な桁詰めを行うと、作用対象にその集団を指定している文の実行結果に影響を及ぼすことがある。 暗黙の FILLER の効果、それらの集団を参照する文の意味については、この章で後に詳しく説明する。

2.6.4 文字の表現と基数の選定

数字項目 (PICTURE によって数字と定義されたもの。*データ部 - データ記述*の章の *PICTURE(形式)句*節を参照) の値は、記憶装置内では、2 進法や 10 進法などの形式で表現される。どの方式を取るかは USAGE (用途) 句を用いて宣言する(*データ部 - データ記述*の章の *USAGE(用途)句*節を参照。) 指定できる数字の形式は以下の通り。

- DISPLAY
- COMPUTATIONAL, COMP,

ANS85 BINARY,

OSVS VSC2 MF COMPUTATIONAL-4 または COMP-4

- OSVS VSC2 MF XOPEN COMPUTATIONAL-3, COMP-3 または

ANS85 PACKED-DECIMAL

- OS/390 MF XOPEN COMPUTATIONAL-5, COMP-5,
- MF COMPUTATIONAL-X または COMP-X
- OSVS VSC2 MF COMPUTATIONAL-1, COMPUTATIONAL-2, ISO2002 FLOAT-SHORT または FLOAT-LONG
- VSC2 MF POINTER
- MF DB2V7 PROCEDURE-POINTER

ANS85 英数字関数は常に標準データ形式で表示される。標準データ形式の英数字関数の大きさは、その関数の定義によって定まる。

整数関数および数字関数の表現形式は、作成者が下記のように指定する。

- 関数が三角関数か、または複数の関数の引数が COMP-2 で、関数が整数を戻さないように定義されている場合は、 COMP-2 とする。
- それ以外は PIC SV9(39)ES9(4)とする。指数は S9(4) COMP で、範囲はおおよそ+/-32k となる。

整数関数および数字関数は、算術式の中でだけ使用できる。整数関数および数字関数は関数を評価した結果の値を表わすが、その関数の作用対象の構成あるいは受取り側データ項目に制約はない。

ある計算機にデータ表現形式がいくつも備わっているときは、データ記述に指定しないと標準データ形式が使用される。

ANS85 ただし、整数関数および数字関数は例外とする。

2.6.4.1 DISPLAY 形式

数値を表わす 0 から 9 の COBOL の数字は、計算機の記憶域 1 バイトあたり 1 文字で、基数 10 を持つ。 これが COBOL 言語の標準データ形式である。符号付きのデータ項目に符号を SEPARATE と指定しないと、符号は数字の上に付けられる。(*データ部 - データ記述*の章の *SIGN(符号)句*節を参照。NUMERIC SIGN 句については、*環境部*の章の*特殊名段落*節を参照。) SIGN 句に LEADING と指定すれば、符号は左端の数字位置に付けられ、TRAILING と指定すれば符号は右端の数字位置に付けられる。符号付きデータにどのように符号が組み込まれるかを表 2-4 に示す。(数値が負である場合、6 番目のビット (値 "40") が 0 から 1 に設定される。) 符号付きのデータ項目に符号を SEPARATE と指定すると、数値を表わす数字の他に 1 文字が符号として付加される。この符号文字は、正か負かに応じて、"+"または "-"となる。符号付きのデータ項目に SIGN 句を指定しないと、符号文字は特殊名段落に

NUMERIC SIGN 句が指定されないかぎり、 右端の数字位置に付けられる。データ記述項に SIGN 句を指定すると、NUMERIC SIGN 句が指定されていれば、その項目用には無視される。

次の表では、かっこ内の数字は COBOL 文字を示す 16 進数である。これはシステムによっては、CHARSET コンパイラ指令または SIGN コンパイラ指令の指定により変わる。

表 2-4: DISPLAY 形式の SEPARATE でない符号付き数字

符号を付ける前の左端または右端の値	符号付きの値を表わす文字					
	正の値			負の値		
	文字集合(ASCII)		文字集合(EBCDIC)	文字集合(ASCII)		文字集合(EBCDIC)
	符号(ASCII)	符号(EBCDIC)	符号(EBCDIC)	符号(ASCII)	符号(EBCDIC)	符号(EBCDIC)
0	0(30)	{(7B)	{(C0)	p(70)	}(7D)	}(D0)
1	1(31)	A(41)	A(C1)	q(71)	J(4A)	J(D1)
2	2(32)	B(42)	B(C2)	r(72)	K(4B)	K(D2)
3	3(33)	C(43)	C(C3)	s(73)	L(4C)	L(D3)
4	4(34)	D(44)	D(C4)	t(74)	M(4D)	M(D4)
5	5(35)	E(45)	E(C5)	u(75)	N(4E)	N(D5)
6	6(36)	F(46)	F(C6)	v(76)	O(4F)	O(D6)
7	7(37)	G(47)	G(C7)	w(77)	P(50)	P(D7)
8	8(38)	H(48)	H(C8)	x(78)	Q(51)	Q(D8)
9	9(39)	I(49)	I(C9)	y(79)	R(52)	R(D9)

SIGN 句に SEPARATE と指定したときに、DISPLAY データ項目を記憶するのに必要な文字数は、PICTURE 句の中の "9"の数に 1 を加えた数となる。DISPLAY 用と宣言したデータ項目に対しては、SYNCHRONIZED 句は効果をもたない。

2.6.4.2 COMPUTATIONAL, ANS85 BINARY,または OSVS VSC2 MF COMPUTATIONAL-4 形式

これらの数字データ項目は、計算機の記憶域では、純粋な 2 進数の形で保持される。この形式では、数値は 2 を基数として表わされ、計算機に保持されている各ビットは右端を最下位の桁として位取りされ、各位の 2 のべき乗値の有無を表わす。 負の数は絶対値の等しい正の数の補数を取り (すべてのビットの値を逆転する) 、その結果に 1 を加えることによって表される。データ項目を記憶するのに必要な文字数は、PICTURE 句の中の"9" の数と符号が付けられているかいないか によって決まる。(*データ部 - データ記述*の章の *PICTURE(形式)句*、*SIGN(符号)句*、*USAGE(用途)句*を参照。) COBOL システムは COMPUTATIONAL データ項目に記憶域を割り当てる方法を、バイト記憶方式と語記憶方式の 2 通り用意している。この COBOL 処理系では特に指定しないと、バイト記憶方式が採られる。

計算機の記憶域の境界：現在の計算機の記憶域の基本的な境界は、通常、8 ビットからなる文字を基礎としている。これをバイトという。この基本的な枠組みの中で、計算機はさらに 2 種類に分類される。1 つはバイト以外の境界をもたないものであり、もう 1 つは複数バイトを単位とする境界をもつものである。ここでは、前者をバイト単位計算機、後者を語単位計算機と呼ぶ。

バイト単位計算機では、COBOL コンパイラは数字データ項目に対して占有するバイト数が最小になるように記憶域を割り当てる。(前述の*文字の表現と基数の選定*節を参照。) この場合、SYNCHRONIZED 句は意味をもたず、効果もない。

語単位計算機の語長には、2 バイト、4 バイト、8 バイトがある。COBOL 言語は COMPUTATIONAL 句または SYNCHRONIZED 句が指定されたときに、それに応じてデータ項目に記憶域を割り当て桁詰めできるようになっている。COMPUTATIONAL 形式のデータに対する語の割り当て方は、COBOL システム指令の IBMCOMP を用いて制御する。

表 2-5：COMP(UTATIONAL) 形式のデータ項目に対する記憶域の割り当て

PICTURE 句の中の数字 (9) の数		割り当てられる記憶域のバイト数	
符号付き	符号なし	バイト単位方式	語単位方式
1-2	1-2	1	2
3-4	3-4	2	2
5-6	5-7	3	4
7-9	8-9	4	4
10-11	10-12	5	8
12-14	13-14	6	8
15-16	15-16	7	8
17-18	17-19	8	8
19-21	20-21	9	16
22-23	22-24	10	16
24-26	25-26	11	16
27-28	27-28	12	16
29-31	29-31	13	16
32-33	23-33	14	16
34-35	34-36	15	16
36-38	37-38	16	16

桁詰め: データ項目の記述に SYNCHRONIZED 句を指定すると、語単位の記憶機能が働いて、そのデータ項目の右端 (最下位) が計算機の記憶域の境界に合わせられる。語長に満たない左側の部分は詰めものまたは暗黙の FILLER として残される。この部分を直接呼び出すことはできないが、集団項目の一部として呼び出すことはできる。

SYNCHRONIZED 句を指定された基本データ項目は、必要なバイト数の語 (表 2-5 を参照) の境界に合わせて配置される。たとえば、語単位の記憶方式では、PICTURE に S9 (5) と記述されている数字データ項目は 4 バイト (データ分 3 バイトと 1 バイトの詰めもの) の記憶域を割り当てられる。そして、SYNCHRONIZED 句が指定されていれば、次に最も近い 4 バイトの境

界に合わせて (レコードの先頭からこのデータ項目の直前のデータ項目の末尾までの長さが 4 の倍数であったかのように) 配置される。直前の項目の末尾が 4 バイトの境界に達しない場合は、暗黙の FILLER が補われる。

集団項目中の OCCURS (反復) 句(*データ部 - データ記述*の章の *OCCURS(反復)句*節を参照)を含むデータ記述に SYNCHRONIZED 句を指定した場合には、暗黙の FILLER が補われることがある。これは、最初の要素が計算機の記憶域の境界に合わせて配置されるのと同様に、反復される 2 番目以降の要素についても境界に位置を合わせるために、余分のバイトが必要になることがあるためである。

暗黙の桁詰め：語単位の記憶方式を採った場合、レコード・レベルのデータ記述はすべて、8 バイトの語長の境界に合わせて自動的に桁詰めされる。

(MF) 自動的な桁詰め機能は ALIGN 指令の影響を受ける。

例 (暗黙の FILLER)： 下に示す COBOL データ記述によって割り当てられる計算機記憶域を図 2-3 に示す。記号の意味は図の下に示してある。

```
01 UNSYNCHRONIZED-RECORD.
    02 UNSYNCHRONIZED-DATA-1        PIC 9(3) DISPLAY.
    02 UNSYNCHRONIZED-DATA-2        PIC X(2).
01 COMPOUND-REPEATED-RECORD.
    02 ELEMENTARY-ITEM-1            PIC X(2).
    02 GROUP-ITEM OCCURS 3 TIMES.
        03 ELEMENTARY-ITEM-2        PIC X.
        03 ELEMENTARY-ITEM-3        PIC S9(2) COMP SYNC.
        03 ELEMENTARY-ITEM-4        PIC S9(4)V9(2) COMP SYNC.
        03 ELEMENTARY-ITEM-5        PIC X(5).
```

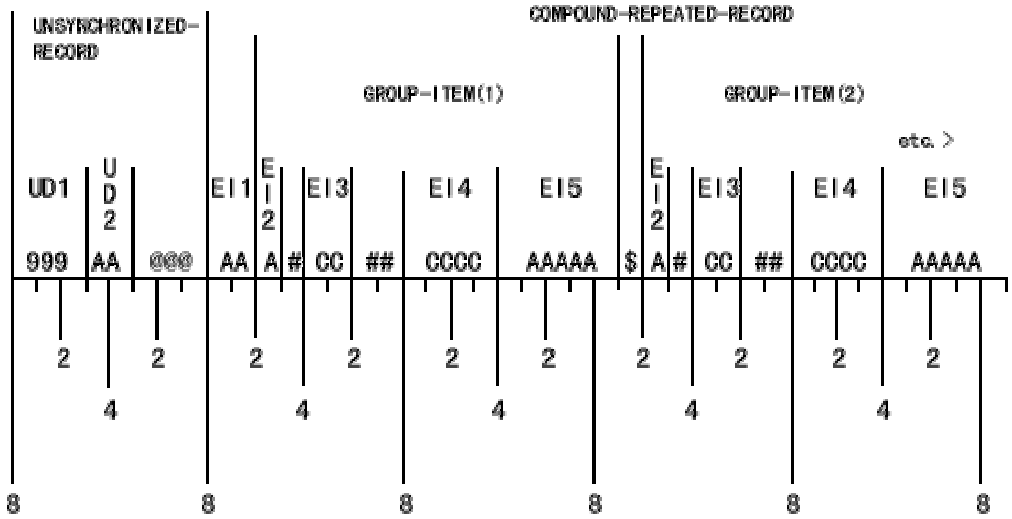


図 2-3: 計算機の記憶域の割当ての例

ここで、

@　レコード (01 レベル) が自動的に桁詰めされたために割り当てられた、暗黙の FILLER バイトを表わす。

#　後続のデータ項目が明示的に桁詰めされたために割り当てられた、暗黙の FILLER バイトを表わす。

$　集団項目に OCCURS 句が指定されたために割り当てられた、暗黙の FILLER バイトを表わす。

9　数字 DISPLAY 文字用に割り当てられたバイトを表わす。

A　英数字 DISPLAY 文字用に割り当てられたバイトを表わす。

C　COMPUTATIONAL データ記憶域用に割り当てられたバイトを表わす。

切捨て：前に述べたように、データ項目に USAGE COMP 句を指定すると、データは 2 進数の形式で保持される。データ項目に割り当てられる記憶域は、PICTURE 句に指定した数値に必要な分よりも大きくなることがある。たとえば、PIC 99 COMP と指定したデータ項目には通常、1 バイトが割り当てられるが、1 バイトは 255 までの数を保持できる。

ANSI COBOL の規則に準拠するために、数字はその形式にかかわらず、10 進数として処理される。算術文が実行された結果、得られた数値が受取り側のデータ項目の PICTURE で可能なよりも大きくなると、桁あふれが発生する。このとき、ON SIZE ERROR (桁あふれ) 句が指定してあると、結果は受取り側の項目に入れられない。算術文以外の文では、受取り側の項目の方が大きさが小さい場合、切捨てが発生する。その場合、該当する数値は、受取り側の PICTURE 句の指定よりも大きい左側の部分が切捨てられる。

OSVS VSC2 MF しかし、USAGE COMP 句を指定したデータを 2 進数として処理されるようにできる。この場合、割り当てられた記憶域にデータを収めるために必要があるときにだけ、切捨てが発生する。USAGE COMP 句を指定したデータ項目の処理の仕方は、COBOL システム指令の TRUNC を用いて制御する。この指令を通じて、PICTURE 句の指定に合わせて 10 進値を切り捨てるか、利用可能な記憶域に合わせて 2 進値を切り捨てるかを選択できる。その切捨て処理が行われる際、算術文の結果とそれ以外の文による転記とは区別される。

指令にどのような設定がされているかにかかわらず、算術文が実行された結果、得られた数値の 10 進の桁数が受取り側のデータ項目の PICTURE に指定されているよりも大きくなると、桁あふれが発生する。

OSVS VSC2 MF **例 (切捨て)**：操作の中には、TRUNC 指令によって結果が変わるものがある。その様子を次に示す。この例では、項目 A の PIC は 99 COMP と記述されている。

操作	結果		
	TRUNC	NOTRUNC	TRUNC"ANSI"
MOVE 163 TO A	63	163	63
MOVE 263 TO A	63	7	63

MOVE 13 TO A, ADD 150 TO A	63	163	結果は保証されない
MOVE 13 TO A, ADD 250 TO A	63	7	結果は保証されない

注:

1. この指令は、非整数データの下位の数字の切捨てには効力をもたない。この操作は常に ANSI COBOL の規定に準拠して行われる。
2. IBMCOMP 指令を設定すると、COMP 項目の上位に余分のバイトが割り当てられることがある。これらのバイトは割り当てられた記憶域に含められる。IBMCOMP がオンであると、SYNC 句を伴う COMP 項目の前に充てん文字が埋められることがある。この充てん文字はその項目の一部ではなく、その項目に記録されるデータに影響されることはない。
3. 符号付き項目に収められる値の桁数が PICTURE 句によって制限されているとき、符号ビットまで上書きする桁数を取ることはできない。ただし、NOTRUNC 指令が設定されている場合は例外で、値が大きければ、符号ビットを上書きする。

2.6.4.3 OSVS VSC2 MF COMPUTATIONAL-1, COMPUTATIONAL-2, ISO2002 FLOAT-SHORT および FLOAT-LONG

形式

この形式は、内部浮動小数点データ項目用に使用する。内部浮動小数点データ項目を使用できる場所は、文法的に数字データ項目が使用でき、かつ COBOL 言語定義の ANSI'74, ANSI'85, ISO2002, OSVS, VSC2 のどれかに構文的に含まれる場所である。内部浮動小数点データ項目は、整数データ項目が必要なところでは使用できない。ただし、特定の COBOL 動詞用の規則に明示的に許されている場合は例外とする。それ以外の構文では、内部浮動小数点データ項目は使用できない。ただし、特別の規則によって許されている場合は例外である。

内部記憶形式はオペレーティングシステムによって異なっていることがある。どのような記憶形式が採られていても、浮動小数点数には 4 つの要素がある。

1. 指数部　　- 仮数部の値に乗除する 10 のべき乗値。
2. 指数部の符号 - 仮数部の値に、1, 10, 100 などの値を掛けるのか、その値で割るのかを示す。
3. 仮数部　　- 具体的な値。これに指数部の値を掛けるか、指数部の値で割ったものが、該当データ項目の数値となる。
4. 仮数部の符号 - 結果として得られたデータ項目の値が正か負かを示す。

ISO2002 USAGE FLOAT-SHORT は USAGE COMPUTATIONAL-1 と同義である。USAGE FLOAT-LONG は USAGE COMPUTATIONAL-2 と同義である。

一般に、USAGE COMPUTATIONAL-1 (COMP-1) のデータ項目を単精度浮動小数点数といい、USAGE COMPUTATIONAL-2 (COMP-2) のデータ項目を倍精度浮動小数点数という。オペレーティングシステムや COBOL で利用できる数学登録集によっては、単精度浮動小数点数と倍精度浮動小数点数で制約条件が異なっている場合がある。たとえば、指数部の最大の大きさや仮数部の最大の大きさに制限がある。(詳細については、使用しているオペレーティングシステムまたは数学登録集の浮動小数点数に関する資料を参照。)

ANSI/IEEE 標準 754-1985、*倍精度浮動小数点数用 IEEE 標準*に従うオペレーティングシステムでは、COMPUTATIONAL-1 および COMPUTATIONAL-2 はそれぞれ Single Format および Double Format と同義である。

内部浮動小数点数表現は、連続した数値を表わすものではないことを理解することが重要である。内部浮動小数点数表現はいろいろなオペレーティングシステムを通じた標準とはなっていない。たとえば、ある内部浮動小数点数表現では、10 進数との対応関係は下記のようになっている。

内部 (16 進) 表現	10 進値
x"AD17E148"	-0.12345673E-23
x"AD17E149"	-0.12345810E-23

したがって、10 進値-.12345678E-23 と正確に等しい内部浮動小数点数を求めても、その値は決して得られない。-方、別の内部浮動小数点数表現では、この値は得られるが、上に記した値は得られないということもある。このため、内部浮動小数点数項目と他の数値 (外部浮動小数点数項目および浮動小数点数定数を含む) が正確に一致することを求める応用は移植性がなく、同じ入力データを使用しても処理手順の流れを変えなければならないことがある。

内部浮動小数点数項目の記憶域の大きさは、USAGE 句によって決まる。USAGE COMPUTATIONAL-1 は項目用に 4 バイトを予約し、USAGE COMPUTATIONAL-2 は記憶域用に 8 バイトを予約する。

IBMCOMP 指令がオンであると、SYNC 句を伴う内部浮動小数点数項目の前に充填文字が埋められることがある。この充填文字はその項目の一部ではなく、その項目に記録されるデータに影響されることはない。

COBOL システムでは、COMP-1 項目は小数点以下 7 桁、COMP-2 項目は小数点以下 16 桁になっている。しかし、メインフレームとの互換性により、DISPLAY 文は、COMP-1 には 8 桁、COMP-2 には 18 桁の小数を表示する。浮動小数点項目を使用するどの操作もこの制限を考慮する必要がある。この制限を超える小数部については無視するようにすること。

2.6.4.4 COMPUTATIONAL-3 (ANS85) または PACKED-DECIMAL 形式

OSVS VSC2 MF OPEN

この形式は一般に 2 進化 10 進形式という。この形式では、数字データ項目は 10 を基数として表現される。数値を表わす各数字は 1 バイトの半分に収められる。その様子を表 2-6 に示す。符号は独立の半バイトとして末尾、つまり右端または最下位の位置に付けられる。

(MF)使用されない半バイトがあれば、その値はゼロに設定される。

表 2-6：COMPUTATIONAL-3 の数字の表現

数字の値	16 進法による数字の表現	
	左の半バイト(偶数桁の数字)	右の半バイト(奇数桁の数字)
0	x"00"	x"00"
1	x"10"	x"01"
2	x"20"	x"02"
3	x"30"	x"03"
4	x"40"	x"04"
5	x"50"	x"05"
6	x"60"	x"06"
7	x"70"	x"07"
8	x"80"	x"08"
9	x"90"	x"09"

注：偶数桁か奇数桁かは、右側から数える。

shoCOMPUTATIONAL-3 用に使用する符号用の桁を表 2-7 に示す。この形式に必要な記憶域は、該当データ項目の PICTURE 句の中の"9s" の数だけによって決まる。その様子を表 2-8 に示す。

表 2-7：COMPUTATIONAL-3 の符号の表現

PICTURE 句内の符号の表現	データ項目の値の符号	16 進法による符号用半バイト
符号なし	n/a	x"0F"
符号付き	+	x"0C"
符号付き	-	x"0D"

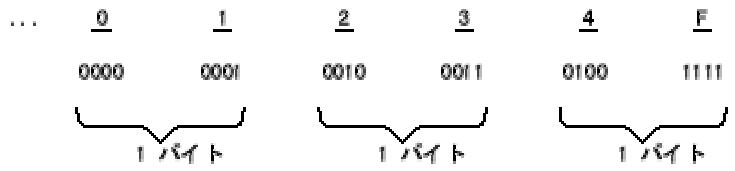
表 2-8：COMP(UTATIONAL)-3
(ANSI85)または PACKED-DECIMAL
PICTURE 句の数字データの桁数と必要記憶域量

必要バイト数	桁数 (符号の有無を問わず)

1	1
2	2-3
3	4-5
4	6-7
5	8-9
6	10-11
7	12-13
8	14-15
9	16-17
10	18-19
11	20-21
12	22-23
13	24-25
14	26-27
15	28-29
16	30-31
17	32-33
18	34-35
19	36-37
20	38

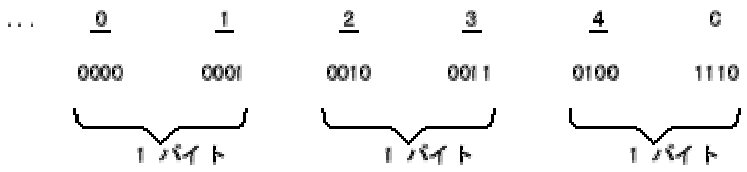
例:

a. COMPUTATIONAL-3 で PICTURE 9999 のデータ項目に値+1234 を収めると、次のようになる。



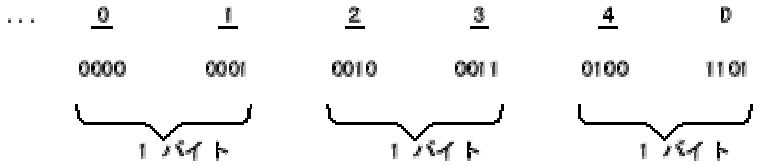
F は正号を表わす (印刷不能文字) 。

b. COMPUTATIONAL-3 で PICTURE S9999 のデータ項目に値+1234 を収めると、次のようになる。



C は正号を表わす。

c. COMPUTATIONAL-3 で PICTURE S9999 のデータ項目に値-1234 を収めると、次のようになる。



D は負号を表わす。

SYNCHRONIZED 句は (LEFT または RIGHT 句があってもなくても) COMPUTATIONAL-3 と宣言されたデータには影響を及ぼさない。

2.6.4.5 (MF) COMPUTATIONAL-X および (OS/390) (MF) (XOPEN) COMPUTATIONAL-5 形式

この形式は COMPUTATIONAL 形式と基本的には同じである。詳細については、前述の *COMPUTATIONAL、BINARY、または COMPUTATIONAL-4 形式*節を参照。

ただし、この形式は下記の点が COMPUTATIONAL 形式と異なる。

- PICTURE 文字列はすべて"x"であってもよい。この場合、"x"の数がそのデータ項目のバイト数を表わす。
- PICTURE 文字列が"x" または"9" のどちらで構成されていても、記憶できる値の大きさは割り当てられた記憶域に収まる 2 進数の最大値が限度となる。このデータ項目はシステム指令の TRUNC, COMP, ALIGN の影響は受けない。しかし、指定できる文字数は最大 18 までである。(PICTURE 句内で"9" を 18 個まで、または"x" を 8 個まで。) コンパイラ指令 INTLEVEL(4)が指定されている場合、指定できる文字数は最大 38 までである。("9"を 38 個まで、または"x"を 16 個まで。)
- 該当項目が算術演算の宛先フィールドになっていて、その文の中に ON SIZE ERROR 句または NOT ON SIZE ERROR 句が含まれている場合、PICTURE 文字列内の"9"の数はランタイム要素に影響を与える。このとき、"9" の数によって、桁あふれ条件が発生したかどうかが判定される。このどちらかの句が指定されているとき、算術演算の結果が "9"の数の範囲内に収まる場合、つまり桁あふれが発生しない場合にだけ、結果がその項目に収められる。

- COMPUTATIONAL-X データ項目と COMPUTATIONAL-5 データ項目の違いは、下記の点だけである。
    - COMPUTATIONAL-5 データ項目には符号を付けることができるが、COMPUTATIONAL-X データ項目には符号を付けてはならない。COMP-5 データ項目用の PICTURE 句の中に"x" を指定すると、そのデータ項目は符号なしの扱いとなり、符号は付けられない。
    - COMPUTATIONAL-X データ項目は、常に BINARY データ項目と同じ記憶方式に従って記憶される。つまり、最上位のバイトが最下位の番地に収められ、以降順次下位のバイトが上位の番地に収められる。

      COMPUTATIONAL-5 データ項目を記憶する方式は、オペレーティングシステムによって異なる。オペレーティングシステムの中には、COMPUTATIONAL-5 データ項目の最下位のバイトを最下位の番地に収め、以降順次上位のバイトを上位の番地に収めるものがある。たとえば、数字項目を逆の順序で記憶するオペレーティングシステムでは、

      h "12 34 56 78 9A"

      という内部値を持つ PIC X(5) COMPUTATIONAL-5 項目は、

      9A 78 56 34 12

      というように記憶される。 一方、COMPUTATIONAL-X 形のデータ項目 (あるいは数字項目の記憶順序を逆転しないオペレーティングシステムのもとの COMPUTATIONAL-5 形のデータ項目) は次のように記憶される。

      12 34 56 78 9A

    - COMPUTATIONAL-5 形式のデータ項目は IBMCOMP システム指令と SYNCHRONIZED 句の影響を受けるが、COMPUTATIONAL-X 形式のデータ項目はその影響を受けない。
- 非算術文によって負の値が COMPUTATIONAL-X 形式のデータ項目または符号なし COMPUTATIONAL-5 項目に収められようとしたときは、絶対値が収められる。

  符号なしの COMPUTATIONAL-5 形式のデータ項目に負の値を収めようとする文の働きは、COMP-5 コンパイラ指令の影響を受ける。

2.6.4.6 ポインタ形式

VSC2 MF

ポインタ形式は利用可能なデータ項目のメモリー番地を表す値を保持する。データ項目が利用できなくなった (たとえば、データ項目が含まれているプログラムがキャンセルされた) 場合には、ポインタ形式には形式が合致しない値が保持されているとみなされる。

ポインタ形式に割り当てられる省略時の記憶領域の量は操作環境によって異なりうる。しかし、少なくとも4バイトはある。メモリー番地を表現する方法は環境によって異なるが、一般的にはCOBOL 以外の言語で使用されている表現と一貫性がある。

システム指令の IBMCOPM を設定した場合、SYNC 指定を伴うポインタ･データの前に充てん文字が生成されることがある。それらの文字はデータ項目の一部ではなく、その項目に収められるデータに影響されることはけっしてない。

2.6.4.7 手続きポインタ形式

MF COB370

手続きポインタ形式は利用可能な手続きのメモリー番地を表す値を保持する。手続きが利用できなくなった (たとえば、手続きが含まれているプログラムがキャンセルされた) 場合には、手続きポインタ形式には形式が合致しない値が保持されているとみなされる。

手続きポインタ形式に割り当てられる省略時の記憶領域の量は操作環境によって異なりうる。しかし、少なくとも4バイトはある。COBOL370 指令を指定すると、8バイトが割り当てられる。メモリー番地を表現する方法は環境によって異なるが、一般的には COBOL 以外の言語で使用されている表現と一貫性がある。

システム指令の IBMCOPM を設定した場合、SYNC 指定を伴う手続きポインタ･データの前に充てん文字が生成されることがある。それらの文字はデータ項目の一部ではなく、その項目に収められるデータに影響されることはけっしてない。

2.6.5 一意参照

2.6.5.1 修飾

COBOL 原始要素中の要素を定義する、利用者が指定したそれぞれの名前、

ANS85 およびその原始要素中で参照する名前は、

すべて一意とする。 名前は、すべて一意とする。そのためには、同じつづりの名前が他にないか、または名前の階層系列の上位にあるいくつかの名前を付け加えて一意にすることができなければならない。このとき、上位の名前を修飾語 (qualifier) といい、一意にする手順を修飾 (qualification) という。 名前を一意にするには、必要な修飾語を付け加えればよく、上位の名前をすべて書く必要はない。

データ部では、修飾に用いるデータ名はすべて、レベル指示語 (level indicator) またはレベル番号 (level-number) の後ろに書いた名前とする。したがって、修飾によって一意にできる場合、

ANS85 またはそれらがまったく参照されない場合

を除いて、2 つの同じデータ名が同じ集団項目に属する記述項として現れてはならない。 手続き部では、2 つの同じ段落名が同じ節にあってはならない。

修飾の階層系列では、レベル指示語の後ろに書いた名前を最高位とし、次にレベル番号 01 に書いた名前、それに続いてレベル番号 02 以降 49 の後ろに書いた名前の順とする。段落名については、節名だけを最高位の修飾語として用いることができる。したがって、階層系列の最高位の名前は、一意でなければならず、修飾することはできない。添字または指標の付いたデータ名や条件変数も、修飾によって一意にすることができる。条件変数の名前は、その条件名の修飾に使用できる。修飾の有無にかかわらず、同じつづりの名前をデータ名と手続き名の両方に使用してはならない。

修飾を行うには、データ名または条件名、段落名、原文名の後ろに IN または OF に続けて修飾語を書く。修飾語をさらに修飾することもできる。IN と OF は、論理的には同義語である。

(ANS85)関数を指定する場合関数定義に従って、その関数を参照する記述の中に、いくつかのパラメータに対する値または値の組を指定する必要がある。そのパラメータの値を用いて、参照した関数の値が算出される。パラメータに実際の値を指定することを引数を与えるという。詳細については、この節の*関数一意名*節で解説する。

修飾の一般書式は、以下の通り。

形式 1

```
{ データ名-1 }
{ 条件名-1   }
{ { IN } データ名-2 } ... [ { IN } { ファイル名-1 } ] ]
{ { OF }            }     { OF } { 通信記述名-1 } ]
{ { IN } { ファイル名-1 }                          }
{ { OF } { 通信記述名-1 }                          }
```

形式 2

```
段落名 [ { IN } 節名 ]
       { OF }
```

形式 3

```
原文名 [ { IN } 登録集名 ]
         { OF }
```

形式 4

```
画面名 [ { OF } 画面名 ] ...
         { IN }
```

(NF)
(NOPEN)

修飾の規則は、以下の通り。

1. 各修飾語は、それが修飾する名前と同じ階層系列に属さなければならない。
2. 複数段階にわたって修飾する場合、低いレベルの修飾語から順に書く。

   (ANS85)原始単位の中で明示的に参照される場合、同じ階層系列中の 2 つのレベルに同じ名前があってはならない。

3. 原始要素中の複数のデータ項目に同じデータ名または条件名を付けた場合、そのデータ名や条件名を手続き部、環境部、データ部で参照するときは、必ず修飾する。ただし、REDEFINES (再定義) 句では修飾できない。
4. 1 つの節の中に同じ段落名を重複して書いてはならない。段落名を節名で修飾するときは、SECTION という語を書かない。同じ節内で参照するときは、段落名を修飾する必要はない。

   (ANS85)段落名も節名も明示的に参照されないかぎり、一意であるかまたは一意にする必要はない。

5. データ名を修飾語として使用するときは、添字を付けることはできない。
6. 修飾する必要のない語を修飾しても構わない。一意にする修飾語の組合わせがいくつもある場合、どの組合わせを使用してもよい。あるデータに対する完全な修飾語の系列が、他のデータ名に対する修飾語の系列の一部と同じであってはならない。

   データ名に付けることのできる修飾語の数は、5 つまでである。

   (ANS85)50 個までの修飾語を付けることができる。

7. 翻訳時に 2 つ以上の登録集を使えるときには、原文名を参照するたびに修飾しなければならない。

   (OSVS)(VSC2)(MF)この規則は強制しない。

2.6.5.2 添字付け

添字 (subscript) は、個別にデータ名を付けていない同種の要素のリストまたは表の中の、個々の要素を参照するために使用する。(***データ部 - データ記述***の章の *OCCURS(反復)句*節を参照。)

添字として使用できるものは、整数である数字定数、データ名、データ名の後ろに"+" または "-" の演算子を付け、さらに符号なしの整数数字定数を続けたものがある。データ名を使用する場合、このデータは数字基本項目で整数とする。添字全体は、左かっこと右かっこが対になった分離符で区切る。

添字に使用するデータ名には、符号が付いていてもよい。この場合、正でなければならない。添字の最小値は 1 である。この値は表の最初の要素を指す。表の要素の 2 番目以降を指す

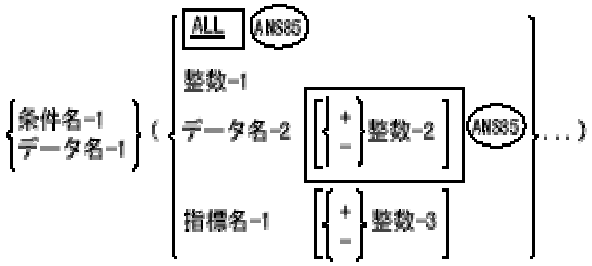
添字の値は、2、3、...という具合になる。添字が取る最大値は、OCCURS 句で指定した項目の反復回数の最大値である。

表の要素を一意に識別する添字または添字の組は、対象とする表要素のデータ名の後ろに、一対の左かっこと右かっこで囲んで付ける。表要素のデータ名に添字を付けたものを、添字付きデータ名または一意名という。複数個の添字を指定するときは、データ構造の階層の上位のものから書く。添字は 3 つまで指定できる。

(AN585)添字は 7 つまで指定できる。

(MF)添字は 16 個まで指定できる。

一般形式



構文規則

1. データ名-1 または条件名-1 に関連するデータ名を含むデータ記述項には、OCCURS 句が含まれていなければならない。または、それらのデータ記述項は、OCCURS 句を含むデータ記述項の下位に属さなければならない。
2. 表要素を参照する場合、指定する添字の数は、参照対象の表要素の記述中の OCCURS 句の数と等しくする。ただし、構文規則 7 に規定された場合は例外とする。複数個の添字を指定する必要があるときは、表の次元の上位のものから書く。
3. (AN585)添字、ALL を使えるのは、関数の引数として添字付きの一意名を使用する場合だけである。条件名-1 を指定したときは、添字 ALL は使用できない。(***手続き部 - 組み込み関数***の章の*引数*節を参照。)
4. 整数-1 は符号を付けてもよい。符号を付けた場合は正でなければならない。
5. データ名-2 は修飾してもよい。データ名-2 は、整数を表わす数字基本項目とする。
6. 指標名-1 は、参照対象の表の階層系列中にあって、その指標名を定義する INDEXED BY (指標付き) 句を含むデータ記述項と対応させる。

   (OSVS)(VSC2)(MF)代わりに、他の表を記述する指標を使用してもよい。ただし、2 つの表に含まれる要素の数が同じであることが条件である。

7. 各表要素を参照する場合は、以下の場合を除き、添字を指定しなければならない。
   a. USE FOR DEBUGGING (デバッグ用使用) 文の中
   b. SEARCH (表引き) 文または SORT (整列) 文の対象として

c. REDEFINES (再定義) 句の中
d. OCCURS (反復) 句の KEY IS (キーは) 句の中

8. (OSVS)同じ一意名の中で、添字と指標を混在させることはできない。

(ANS85)同じ一意名の中で、添字と指標を混在させることができる。

一般規則

(MF)NOBOUND 指令を指定した場合は、この一般規則は実行時に適用されない。

1. 添字の値は、正の整数とする。添字に指定できる最小の値は 1 である。表のどの次元でも、最初の要素は、添字の値 1 によって参照される。その表のその次元の 2 番目以降の要素を指す添字の値は、2、3、...という具合になる。表のどの次元でも、添字が取る最大値は、対応する OCCURS 句で指定した項目の反復回数の最大値である。

   (MF)添字が浮動小数点項目の場合、その値は最も近い整数に丸められるか、または、端数が切り捨てられる。

2. 指標名-1 を用いて参照する指標の値は、対応する表中の要素の出現番号を表わす。
3. 指標名-1 を用いて参照する指標の値は、添字として使用する前に初期化しておく。指標に初期値を与える方法は 3 通りある。具体的には、PERFORM (実行) 文に VARYING (変更) 句を指定するか、SEARCH (表引き) 文に ALL (全部) 句を指定するか SET (設定) 文を指定するかである。指標の値を変更できる文は PERFORM と SEARCH と SET だけである。
4. (ANS85)整数-2 または

   整数-3 を指定した場合、添字の値は次のようにして決定される。演算子に"+" を指定した場合は、

   (ANS85)整数-2 または

   整数-3 の値が追加される。演算子に"-" を指定した場合は、

   (ANS85)整数-2 または

   整数-3 の値が差し引かれる。

   これらは、指標名-1 を用いて指定した値を用いて指定した値に対して行われる。

   (ANS85)または、データ名-2 を用いて指定した値に対して行われる場合もあり得る。

2.6.5.3 指標付け

指標付け (indexing) を指定して、同種の要素からなる表の中の個々の要素を参照できる。指標 (index) を付けるには、表を定義する際に、該当するレベルに INDEXED BY (指標付き) 句

を指定する。INDEXED BY 句を用いて付けた名前を指標名 (index-name) といい、付けた指標を参照するために使用できる。指標の値は、対応する表または別の表の中の要素の出現番号を表わす。指標名は、表の参照に使用する前に初期化しておく。指標名に初期値を与えるには、SET (設定) 文を指定する。

添字は単なる数字データ項目または定数である。これに対し指標は、表中要素の出現番号を表わす特別な型の項目である。その表現形式は何通りかある。指標の内容は数値とはみなされない。

直接指標付け (direct indexing) は、添字と同じ形式で指標名を使用する方法である。相対指標付け (relative indexing) は、指標名の後ろに演算子"+" または "-" と符号の付かない整数を書き、その全体を一対の左かっこと右かっこで囲んで、表要素のデータ名の後ろに続けた形式で指標名を使用する方法である。相対指標付けによる出現番号は次のようにして決定される。演算子に"+" を指定した場合は、指標の値によって表される出現番号に定数の値が追加される。演算子に"-" を指定した場合は、指標の値によって表される出現番号から定数の値が差し引かれる。複数個の指標を用いるときは、データ構造の階層の上位のものから順に書く。

INDEXED BY 句を指定することによってだけ、表に指標を付けることができる。

OSVS OSC2 MF A ある表用に指定した指標を、他の表に適用できる。ただし、どちらの表も要素の数が同じであることが条件である。

指標付きの表の要素を参照する文を実行するとき、その表要素の指標名で参照される指標の値は、1 から表要素の反復回数の最大値の範囲を外れてはならない。この制限は、相対指標付けの結果として得られる値にも当てはまる。

MF NOBOUND 指令を指定した場合は、この一般規則は実行時に適用されない。

データ名には、指標名を 3 つまで指定できる。

ANS85 データ名には指標名を 7 つまで指定できる。

MF データ名には指標名を 16 個まで指定できる。

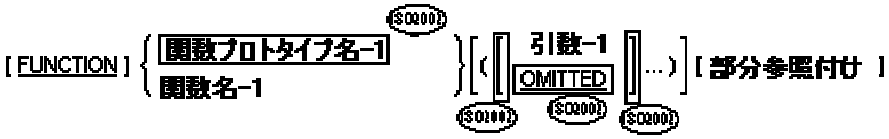
指標付けの一般形式は、*添字付け*節に記した一般形式に含まれている。

2.6.5.4 関数一意名

ANS85

関数一意名とは、関数を評価した結果として得られるデータ項目を一意に参照する名前である。この名前は、文字列と分離符を構文的に正しく組み合わせて付ける。

一般形式



構文規則

1. 関数一意名は、受取り側作用対象として指定される。
2. FUNCTION という単語が含まれる。

   (ISO2002)ただし、以下の場合を除く。

   a. 組み込み関数名-1 がリポジトリ段落で指定されている。
   b. 関数プロトタイプ名-1 が指定されている。
3. (ISO2002)関数プロトタイプ名-1 は、リポジトリ段落で指定された関数プロトタイプである。
4. (ISO2002)組み込み関数名-1 が指定されている場合は、OMITTED という単語は指定されない。
5. 引数-1 は、一意名、定数、または算術式とする。組み込み関数に対する引数-1 の数、字類、および項類についての詳しい規則は、***手続き部 - 組み込み関数***の章の*関数の定義*節にある、組み込み関数の定義で解説する。

   (ISO2002)また、利用者定義関数に対する同規則は、***手続き部***の章の*パラメータおよび返却する項目への準拠性*で解説する。

6. (ISO2002)単語、OMITTED が指定されている場合は、対応する仮パラメータ用に OPTIONAL 句が指定されるものとする。
7. (ISO2002)関数プロトタイプ名-1 が指定され、 かつ引数-1 に対応する仮パラメータが BY VALUE 句とともに指定されている場合は、引数-1 は、字類が数字、オブジェクト、またはポインタである。
8. 整数の作用対象が必要な箇所に数時間数を指定することはできない。ただし、特定の数字関数の参照が整数値を生成する場合もある。
9. 符号のつかない整数が必要な箇所に、ABS 関数の整数型以外の整数関数を指定することはできない。
10. 整数または数時間数を参照する関数一意名は、算術式でのみ使用される。
11. (ISO2002)関数プロトタイプ名-1 が指定されている場合は、***手続き部***の章の*パラメータおよび返された項目への準拠性*節で解説されている準拠規則が適用される。
12. 部分参照付け (reference-modifier) は、英数字項類の関数に対してのみ指定される。

    (ISO2002)(MF)各国語の関数に対しても指定できる。

13. 関数の定義により、関数プロトタイプ名-1 または組み込み関数名-1 の直後に引数と左かっこが続くことが許される場合は、この左かっこはつねに、この関数の引数の左かっこと見なされる。

注:引数の有無にかかわらず、参照される関数 (RANDOM 関数など) については、正しく解釈されるよう、符号化に注意が必要である。以下に関数の例を示す。

```
FUNCTION MAX (FUNCTION RANDOM (A) B)
```

A は、、RANDOM 関数の引数と見なされる。このような関数で、A を MAX 関数の 2 番めの引数とする場合は、以下のいずれかのように書かれなければならない。

```
FUNCTION MAX (FUNCTION RANDOM () A B)

                    または
FUNCTION MAX ( (FUNCTION RANDOM) (A) B)

                    または
FUNCTION MAX (FUNCTION RANDOM A B)
```

一般規則

1. 関数一意名は、実行時に関数が参照されたときにその値が決定される、一時データ項目を参照する。

   組み込み関数名-1 が指定されている場合は、一時データ項目は、基本データ項目であり、その記述項と項類は組み込み関数の定義により指定される。この定義については、手続き部 - 組み込み関数の章の*関数の定義*で解説する。

   (ISO2002)関数プロトタイプ名-1 が指定されている場合は、一時データ項目の記述、字類、および項類は、関数プロトタイプ名-1 で指定された関数プロトタイプの手続き部見出しの RETURNING 句で指定された項目の連絡節の記述により指定されたものである。

2. 関数が参照されたとき、その引数は、左から右へ置かれている順番に個別に読み込まれる。読み込まれる引数は、それ自体が関数一意名、または関数一意名を含む式である場合もある。引数の読み込により参照される関数が、その引数が指定された関数自体であってはならないという規則はない。つまり、再帰的な引数の指定が可能である。組み込み関数については、詳しい解説が*手続き部 - 組み込み関数*の章にある。

   (ISO2002)利用者定義関数については、***手続き部***の章の*手続き部見出し*および*パラメータおよび返却する項目の準拠性*にある。

   (MF)整数の引数をとる関数では、浮動点引数が与えられた場合は、もっとも近い整数に値が丸められる。または、小数点以下は切り捨てられる。

3. (ISO2002)関数プロトタイプ名-1 が指定されている場合は、***環境部***の章の*リポジトリ段落*節にある規則に従って、起動される関数が指定され、その特徴の決定に関数プロトタイプ名-1 が使用される。

4. (SO2002)関数プロトタイプ名-1 が指定されている場合に各引数が渡される方法は以下の通り。
   a. 対応する仮パラメータに対して BY REFERENCE 句が指定または暗黙的に指定されているときは、BY REFERENCE。このとき、引数-1 は、オブジェクトプロパティ、オブジェクトデータ項目、またはファクトリオブジェクトデータ項目以外の、受取り側の作用対象となる。
   b. 対応する仮パラメータに対して BY CONTENT が指定または暗黙的に指定されているときは、BY REFERENCE。このとき、引数-1 は、定数、算術式、オブジェクトプロパティ、オブジェクトデータ項目、ファクトリオブジェクトデータ項目、または、その他の、受取り側の作用対象として許可されない一意名となる。
   c. 対応する仮パラメータに対して BY VALUE 句が指定されているときは、BY VALUE
5. 関数一意名は、以下の手順で評価される。
   a. 各引数-1 が最初に確認される。起動された関数に処理が渡されたときに、その関数が引数-1 の値を使用できるようになる。
   b. 実行システムが、起動される関数を検索する。

      (SO2002)関数プロトタイプ名-1 が指定されている場合は、*関数プロトタイプ名の適用規則*で解説されている規則が指定される。その他に、***環境部***の章の*リポジトリ段落*節で解説する規則も指定される。

   c. 関数が見つかったが、その実行に必要な資源が使用できない場合は、その関数は起動されない。関数の実行に必要な実行資源が何であるかは、作成者により定義される。
   d. 関数一意名により指定された関数が実行可能になり、その関数により指定された呼出し規則に従って、起動された関数に処理が渡される。

      (SO2002)関数プロトタイプ名-1 が指定されており、起動される関数は COBOL 関数である場合は、***手続き部***の章の*手続き部見出し*節の解説に従って実行される。

      組み込み関数名-1 が指定されている場合は、*手続き部 - 組み込み関数*の章の解説に従って実行される。

      (SO2002)関数プロトタイプ名-1 が指定されており、起動される関数は COBOL 関数である場合は、使用している COBOL システムのマニュアルのインターフェイスに関する記述における定義に従って実行される。

6. OMITTED という語が指定されているか、または後に続く引数が省略されている場合は、そのパラメータの省略引数条件が、起動された関数の中で、TRUE であると見なされる（***手続き部***の章の*省略引数条件*節を参照）。
7. 省略引数条件が真であるパラメータが、起動された関数により参照されている場合は、実行時エラー 203 が返される。ただし、引数として参照される場合、および省略引数条件内で参照される場合を除く。

2.6.5.5 部分参照

部分参照 (reference modification) とは、データ項目の一部を切り出してデータ項目を定義することである。切り出す範囲はデータ項目中の起点となる文字位置 (切り出す文字の左端文字位置) と長さによって指定する。

一般形式

一意名-1 (左端文字位置 : [長さ])

構文規則

1. 一意名-1 は、以下のいずれかであるデータ項目を参照しなければならない。
    - 各国語、または字類が英数字である基本項目
    - 用途が DISPLAY である数字項目
    - 集団項目
2. 左端文字位置と長さは算術式とする。
3. 別途指定がないかぎり、各国語、または字類が英数字のデータ項目を参照する一意名が使えるところならば、どこでも部分参照ができる。
4. 一意名-1 は修飾してもよいし、添字付けしてもよい。
5. 一意名-1 が関数一意名である場合は、英数字または各国語関数を参照するものとする。

一般規則

1. 左端文字位置とは、一意名-1 が英数字 (または各国語) データ項目を参照するときの英数字の位置 (または各国語の位置) を示す。
2. 一意名-1 によって参照されるデータ項目の各文字には、左端を 1 として、右方向に 1 ずつ繰り上げた順番を付けられる。一意名-1 のデータ記述項に SIGN IS SEPARATE (独立符号) 句が指定してあると、その符号にも順番が付けられる。
3. 一意名-1 によって参照されるデータ項目の項類が、数字、

   OSVS VSC2 MF 外部浮動小数点数、

   数字編集、英字、英数字編集のどれかであるとき、部分参照の目的上、そのデータ項目は同じ大きさの英数字データ項目として再定義されたものとして操作される。

4. 作用対象に対する部分参照は、下記のように評価される。
    a. 作用対象に添字を指定してある場合、添字が評価された直後に部分参照が評価される。作用対象に添字、ALL が指定してある場合、暗黙的に指定されたすべての表要素に部分参照が適用される。
    b. 作用対象に添字を指定していない場合、添字が指定してあったならば添字が評価される時点で部分参照が評価される。
    c. 関数参照の中に部分参照が指定してある場合、関数が評価された直後に部分参照が評価される。
    d. オブジェクトプロパティ参照の中に部分参照が指定してある場合、オブジェクトプロパティが評価された直後に部分参照が評価される。

5. 部分参照は、一意名-1 によって参照されるデータ項目の部分集合である、一意のデータ項目を作り出す。この一意のデータ項目は下記のようにして、参照対象データ項目から取り出される。
   a. 左端文字位置を評価した結果、数値が得られる。この値は、一意名-1 によって参照されるデータ項目の、左端から振られた文字位置を表わす順番を指す。したがって、左端文字位置の値は、参照対象となるデータ項目の、文字数以下の正の整数とする。

      VSC2 MF 浮動小数点数形式の式では、結果は丸められる。固定小数点数形式の式では、結果は切捨てられる。

   b. 長さを評価した結果、作用対象として使用するデータ項目のバイト数が得られる。この長さは正の数とする。また、左端文字位置と長さの和から 1 を引いたものは、一意名-1 によって参照されるデータ項目の文字数以下とする。長さを指定しないと、部分参照として取り出されるデータ項目は、参照対象データ項目の中の再左端文字位置から末尾までとなる。

      VSC2 MF 浮動小数点数形式の式では、結果は丸められる。固定小数点数形式の式では、結果は切捨てられる。

注: 左端文字位置と長さの評価により生成された値が正しいかどうかの確認は行われない。値が、この規則で述べられた制限に適合しない場合は、未定義の結果が生じ、他のデータ項目が破壊される可能性がある。

---

6. 部分参照として取り出されたデータ項目は、JUSTIFIED (桁寄せ) 句のない基本データ項目とみなされる。関数を参照した場合は、部分参照として取り出されたデータ項目の字類と項類は英数字となる。一意名-1 を指定した場合は、部分参照として取り出されたデータ項目の字類と項類は一意名-1 と同じになる。ただし、
   - 項類が数字、数字編集、英数字編集、

     OSVS VSC2 MF 外部浮動小数点数

     のものは、字類も項類も英数字とみなされる。

   - ISO/IEC MF 項類が各国語編集のものは、字類および項類が各国語とみなされる。
7. OSVS OSVS コンパイラ指令を設定した場合は、条件式の中では部分参照は指定できない (詳細については、***手続き部***の章の*条件式*節を参照) 。

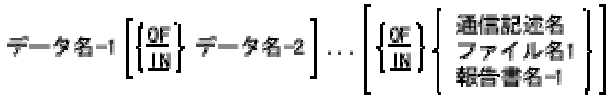
2.6.5.6 一意名

一意名 (identifier) とは、データを一意に参照するために、文字列と分離符をいくつか 並べたものである。

一般形式

形式 1

形式 2



ISO2002 MF **形式 3**

プロパティ名-1 OF 一意名-1

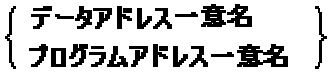
ISO2002 MF **形式 4**

行内呼出し-1

ISO2002 MF **形式 5**

一意名-2 オブジェクトビュー-1

ISO2002 MF **形式 6**



形式 2 の規則

添字-1 は添字付け (*添字付け*節を参照) または指標付け (*指標付け*節を参照) を表す。

1. **語 IN と OF とは同義語である。**

2. 添字付けまたは指標付けの対象とするデータ名は、それ自体が添字付けあるいは指標付けされていてはならない。
3. 添字付けが許されていない箇所では、指標付けも許されない。
4. 指標は、SET (設定) 文、SEARCH (表引き) 文、PERFORM (実行) 文によってだけ変更できる。USAGE IS INDEX (用途は指標) 句を指定したデータ項目には、指標名に対応する値をデータとして収めることができる。このような項目を指標データ項目 (index data item) という。
5. 添字として使用する定数は、正の整数とする。相対添字付けおよび相対指標付けに使用する定数は、符号なしの整数とする。

(S0200)形式 3 の規則

オブジェクト属性はオブジェクトからの情報の読出しおよびオブジェクトへの情報の書戻しを行うための特別な構文を形成する働きをする。オブジェクト属性にアクセスする仕組みとして、オブジェクト読出しメソッドとオブジェクト設定メソッドがある。オブジェクト読出しメソッドには、GET PROPERTY 句を用いて明示的に定義したメソッドと、PROPERTY 句を用いて記述されたデータ項目のために暗黙的に生成されるメソッドとがある。オブジェクト設定メソッドには、SET PROPERTY 句を用いて明示的に定義したメソッドと、PROPERTY 句を用いて記述されたデータ項目のために暗黙的に生成されるメソッドとがある。

1. 属性名-1 はリポジトリ段落中に指定されたオブジェクト属性でなければならない。
2. 一意名-1 はオブジェクト参照でなければならない。一般的オブジェクト参照も定義済オブジェクト参照の NULL も指定してはならない。
3. オブジェクト属性を送出し側項目として使用した場合、一意名-1 によって参照されるオブジェクト内に属性名-1 の属性読出しメソッドが存在しなければならない。
4. オブジェクト属性を受取り側項目として使用した場合、一意名-1 によって参照されるオブジェクト内に属性名-1 の属性設定メソッドが存在しなければならない。
5. 送出し側項目として使用されるオブジェクト属性の記述は属性読出しメソッドの返却する項目の記述と同じである。そのように記述されたデータ項目が送出し側項目として有効なときはいつでも、このオブジェクト属性を指定できる。
6. 受取り側項目として使用されるオブジェクト属性の記述は属性設定メソッドの USING パラメータの記述と同じである。そのように記述されたデータ項目が受取り側項目として有効なときはいつでも、このオブジェクト属性を指定できる。
7. 属性読出しメソッドの RETURNING 句内に指定されているデータ項目の記述は、属性設定メソッドの USING パラメータとして指定されているデータ項目の記述と同じでなければならない。
8. オブジェクト属性が送出し側項目としてのみ使用される場合、概念的な一時データ項目の一時データ-1 が代りに使用される。その属性の値は、INVOKE 文の規則に従って関連する属性読出しメソッドが呼び出され、返された値が一時データ-1 に入れられたかのようにして決定される。一時データ-1 のデータ記述は、属性読出しメソッドの RETURNING 句に指定されているデータ項目の記述と同じである。
9. オブジェクト属性が受取り側項目としてのみ使用される場合、概念的な一時データ項目の一時データ-2 が代りに使用される。その属性の値は、INVOKE 文の規則に従って関連する属性設定メソッドが呼び出され、一時データ-2 の内容がパラメータとして渡されたかのようにして決定される。一時データ-2 のデータ記述は、属性設定メソッドの USING パラメータとして指定されているデータ項目の記述と同じである。

10. オブジェクト属性が送出し側項目と受取り側項目の両方として使用される場合、概念的な一時データ項目の一時データ-1 と一時データ-2 が代りに使用される。一時データ-1 と一時データ-2 は同じ一時データ項目であり、一時データ-2 は一時データ-1 を再定義したものである。送出し処理に関しては、その属性の値は一般規則1の中の送出し側項目と同様に決定される。受取り処理に関しては、その属性の値は一般規則2の中の受取り側項目と同様に決定される。一時データ-1 および一時データ-2 のデータ記述は、属性読出しメソッドの RETURNING 句に指定されているデータ項目の記述と同じである。

形式 4 の規則

行内呼出し-1 は*行内メソッド呼出し*節で定義される。

形式 5 の規則

オブジェクトビュー-1 は*オブジェクトビュー*節で定義される。

形式 6 の規則

番地指定子は*データ番地一意名*節および*プログラム番地一意名*節で定義される。

.

2.6.5.7 条件名

各条件名は一意であるか、または修飾、指標付け、添字付けによって一意にしなければならない。条件名を一意にするために修飾する場合は、関連する条件変数を最初の修飾語として使用できる。修飾する場合、条件変数に関連する名前の階層系列または条件変数自体を使用して、条件名を一意にする。

条件変数を参照するのに指標付けまたは添字付けが必要な場合は、その条件名を参照するときにも、同じ組合わせの指標付けまたは添字付けを使用する。

条件名の修飾、指標付け、添字付けの組合わせの形式と制限は、一意名の場合のデータ名-1 を条件名-1 と置き換えたものと同じである。

一般形式において条件名というときは、必要に応じて修飾、指標付け、添字付けしたものを含む。

2.6.5.8 行内メソッド呼出し

ISO2002 MF

行内メソッド呼出しは、メソッドの呼出しにより返された一時データ項目を参照する。

一般形式

```
{ クラス名-1 }
{ 一意名-1   } :: 定数-1 [ ( { 算術式-1          } ... ) ]
                           { 一意名-2
                           { 定数-2
                           { OMITTED
```

構文規則

1. 行内メソッド呼出しは、受取り側の作用対象として指定される。
2. 一意名-1 は字類がオブジェクトでなければならない。定義済みオブジェクト参照 NULL と一般的オブジェクト参照のいずれも、指定してはならない。
3. 一般規則 1 で指定された INVOKE 文のいずれかは、INVOKE 文の構文規則に従って、有効でなければならない。
4. 呼び出されたメソッドの手続き部見出しの中の RETURNING 指定で参照されたデータ項目は、ANY LENGTH 句で記述されなければならない。

一般規則

1. 行内メソッド呼出しは、適切な形式で実行された INVOKE 文により返された一時一意名と同じ字類、項類、および内容の一時データ項目を参照するものとする。INVOKE 文の例を以下に示す。

   INVOKE 一意名-1 定数-1 USING 引数 RETURNING 一時一意名
   INVOKE 一意名-1 定数-1 RETURNING 一時一意名
   INVOKE 字類名-1 定数-1 USING 引数 RETURNING 一時一意名
   INVOKE 字類名-1 定数-1 RETURNING 一時一意名

   ここで、

   a. 引数 (存在する場合) は、行内メソッド呼出しのかっこ内に指定された作用対象である。
   b. 一時一意名の記述項、字類、および項類は、定数-1 および、一意名-1 (または字類名-1) により識別されたメソッドの指定に含まれる RETURNING パラメータのものと同じである。
   c. 一時一意名は、ここで解説する行内呼出しの機能のためにのみ存在する一時項目で、その他の目的にはいっさい使用されない。

2.6.5.9 オブジェクトビュー

ISO2002 MF

オブジェクトビューにより、オブジェクト参照が指定された記述を持っているように取り扱われるようになる。実行時に、この記述が有効かどうか、オブジェクトが検査される。

一般形式

```
                    [ [FACTORY OF] クラス名-1   [ONLY] ]
一意名-1 AS {       [ インターフェイス名-1              ]
                    [ UNIVERSAL                         ]
```

構文規則

1. 一意名-1 は字類がオブジェクトとする。定義済みオブジェクト参照、SUPER および NULL は指定してはならない。
2. オブジェクトビューは、受取り側作用対象として指定してはならない

一般規則

1. この一意名-1 の参照は、翻訳時に、AS 指定により指定された記述項を持っていると見なされる。
2. 字類名-1 が、 指定可能な句のどちらも使用せずに指定されている場合は、一意名-1 が、USAGE IS OBJECT REFERENCE 字類名-1 で記述されているとして処理される。一意名-1 により参照されるオブジェクトが、字類名-1 のオブジェクトでも字類名-1 に含まれる字類のオブジェクトでもない場合は、EC-OO-CONFORMANCE 例外条件が設定される。
3. FACTORY 指定が存在し、かつ ONLY 指定が存在しない場合は、一意名-1 が、USAGE OBJECT REFERENCE FACTORY OF 字類-1 として記述されているとして処理される。一意名-1 により参照されるオブジェクトが、字類名-1 のファクトリオブジェクトでも字類名-1 に含まれる字類のファクトリオブジェクトでもない場合は、EC-OO-CONFORMANCE 例外条件が設定される。
4. ONLY 指定が存在し、かつ FACTORY 指定が存在しない場合は、一意名-1 が、USAGE OBJECT REFERENCE 字類名-1 ONLY として記述されているとして処理される。一意名-1 により参照されるオブジェクトが、字類名-1 のファクトリオブジェクトでない場合は、EC-OO-CONFORMANCE 例外条件が設定される。
5. FACTORY および ONLY 指定の両方が存在する場合は、一意名-1 が、USAGE OBJECT REFERENCE FACTORY OF 字類名-1 ONLY として記述されているとして処理される。一意名-1 が字類名-1 のファクトリオブジェクトでない場合は、EC-OO-CONFORMANCE 例外条件が設定される。
6. インターフェイス名-1 が指定されている場合は、一意名-1 が、USAGE OBJECT REFERENCE インターフェイス名-1 として記述されているとして処理される。一意名-1 により参照されるオブジェクトが、インターフェイス名-1 を使用しない場合は、EC-OO-CONFORMANCE 例外条件が設定される。
7. UNIVERSAL が指定されている場合は、一意名-1 が、一意名-1 が参照するオブジェクト用の字類またはインターフェイスを表す指定可能な句をまったく使用していない USAGE OBJECT REFERENCE として記述されているとして処理される。この場合は、EC-OO-CONFORMANCE 例外条件が設定される。

2.6.5.10 データ番地一意名

ISO2002 MF

データ番地一意名は、データ項目の番地を含む一意のデータ項目を参照する。

一般形式

ADDRESS of 一意名-1

構文規則

1. 一意名-1 は、ファイル節、作業記憶節、局所記憶節、または連絡節で定義されたデータ項目を参照するものとする。一意名-1 は、インスタンスオブジェクトまたはファクトリオブジェクトの作業記憶節またはファイル節では定義されない。
2. 一意名-1 は、オブジェクト参照を参照してはならない。
3. この一意名形式を、受取り側作用対象として指定してはならない。

一般規則

1. データ番地一意名は、一意名-1 の番地を含む、字類ポインタおよび項類データポインタの、一意のデータ項目を作成する。

2.6.5.11 プログラム番地一意名

ISO2002 MF **一般形式**

ADDRESS OF PROGRAM { 一意名-1 | 定数-1 | プログラムプロトタイプ名-1 }

構文規則

1. 一意名-1 は、項類が英数字または各国語とする。
2. 定数-1 は、英数字または各国語型定数とする。
3. プログラムプロトタイプ名-1 は、リポジトリ段落で指定されたプログラムプロトタイプとする。
4. この一意名形式を、受取り側作用対象として指定してはならない。

一般規則

1. プログラム番地一意名は、以下のいずれかにより識別されるプログラムの番地を含む、字類ポインタおよび項類データポインタの、一意のデータ項目を作成する。
    a. 一意名-1 により参照されるデータ項目の内容

b. 定数-1 の値
c. プログラムプロトタイプ名-1.

一意名-1 または定数-1 が指定される場合は、*COBOL 言語の概念*の章の *COBOL の語*節の解説に従って、この値を使用して参照先プログラムが識別される。

2. プログラムは COBOL で書かれていても、他の言語で書かれていてもよい。COBOL プログラムでは、番地は、もっとも外部にあるプログラムの番地である。このプログラムは、そのプログラム名段落にある外部化されたプログラム名により識別される。
3. プログラムプロトタイプ名-1 が指定されている場合は、プログラム番地一意名は、プログラムプロトタイプ名-1 に制限されたプログラムポインタの性質を持つ。
4. 実行時システムがプログラムを見つけられない場合は、番地一意名の値は定義済みの番地、NULL となる。

2.6.6 明示指定と暗黙指定

COBOL の原始要素には、下記の 4 種類の明示指定と暗黙指定がある。

1. 手続き部の明示指定と暗黙指定
2. 制御の明示移行と暗黙移行
3. 明示属性と暗黙属性
4. ANS85 明示範囲符と暗黙範囲符

2.6.6.1 手続き部の明示参照と暗黙参照

COBOL 原始要素プログラム中では、手続き部の文で、データ項目を明示的にも暗黙的にも参照できる。 明示参照 (explicit reference) とは、手続き部の文に参照対象項目の名前を書いた場合、または COPY (複写) 文、

ANS85 REPLACE 文、

OSVS VSC2 INC 文、または++INCLUDE 文

を実行して参照対象項目の名前を手続き部に複写してきた場合の参照をいう。 暗黙参照 (implicit reference) とは、手続き部の文に参照対象項目の名前を書かないでデータ項目を参照することをいう。

PERFORM (実行) 文を実行する際にも、暗黙参照が発生する。これは、PERFORM 文に関連する制御機構が VARYING 句、AFTER 句、UNTIL 句に指定された指標名または一意名によって参照される、指標またはデータ項目を初期化したり変更したり評価したりした場合である。このような暗黙参照は、データ項目が命令の実行に係わるときだけ行われる。

2.6.6.2 制御の明示移行と暗黙移行

制御の流れでは、書かれている順に文から文へと制御が移行する。ただし、この順序より優先される制御の明示移行が存在する場合、または制御を渡すことのできる次の実行可能な

文が存在しない場合を除く。文から文への制御の移行は、手続き部文がとくに書かれていなくても行われる。したがって、制御の暗黙移行と呼ばれる。

COBOL には、暗黙的な制御移行の機構を明示的にも暗黙的にも変更する手段が備わっている。

制御の暗黙移行は、連続する完結文および連続する文の間で起こるほか、手続き分岐文を実行することなく通常の制御の流れを変えるときにも起こる。文から文への制御の流れを暗黙のうちに変更する方法には、以下のものがある。

1. 他の COBOL 文（PERFORM（実行）、USE（使用）、SORT（整列）、MERGE（併合）など）の制御のもとで、その文の制御範囲の最後の段落が実行されたとき、その最後の文から元の文の制御機構に、暗黙的に制御が移される。さらに、繰り返し実行を起す PERFORM 文では、実行の繰り返しが起こるたびに、その制御機構と制御範囲の最初の段落の最初の文との間で、制御の暗黙移行が起こる。
2. SORT 文または MERGE 文が実行されるとき、関連する入出力手続きに暗黙のうちに制御が移る。
3. ある COBOL 文を実行することが 宣言（declarative）の節（section）の実行につながると、その宣言部分の節に暗黙に制御が移る。

注：その宣言部分の節に暗黙に制御が移る。この宣言部分の節が実行された後で、暗黙のうちに制御が戻る（上記 1.を参照）。

---

4. MF 何らかのファイル操作（OPEN および CLOSE を含む）を行った際に、そのファイルに FILE STATUS データ項目が宣言されてなく、USE 文も明示的に指定されていない場合、暗黙の USE 文によって例外処理が行われる。この暗黙の USE 手続きの内容を下記に示す。

```
5.   USE AFTER ERROR PROCEDURE ON ファイル名.
6.   IF 状態キー-1 >= 3
7.     DISPLAY エラーメッセージ UPON CONSOLE
     STOP RUN.
```

エラーメッセージの定義については、***エラーメッセージ***を参照。

制御の明示移行とは、手続き分岐文または条件文を実行することによって、制御の暗黙移行の機構を変えることをいう。文の間の制御の明示移行は、手続き分岐文または条件文を実行することによってだけ引き起こされる。完結文の間の制御の明示移行は、IF（判断）文の NEXT SENTENCE（次の完結文）句または SEARCH（表引き）文を実行することによってだけ引き起こされる。

手続き分岐文の ALTER（変更）文を実行することは、直接的に制御の明示移行を引き起こす訳ではなく、関連する GO TO（飛び越し）文を実行するときの制御の明示移行に影響する。手

続き分岐文の EXIT PROGRAM (プログラムの出口) 文は、呼ばれたプログラムの中で実行されると、制御の明示移行を引き起こす。

「次の実行完結文」(next executable sentence) という用語は、上記の規則に従って制御が暗黙のうちに移されるか、または NEXT SENTENCE 句によって明示的に制御を移される、次の COBOL 完結文を指すために用いる。次の実行完結文は、現在の完結文を終了させる分離符の終止符の直後の完結文である。

「次の実行文」(next executable statement) という用語は、上記の規則および手続き部の各言語要素に関連する規則に従って制御が移される、次の COBOL 文を指すために用いる。

以下の場合には、次の実行文は存在しない。

- 宣言部分の最後の文で、それを含む段落が他の COBOL 文の制御のもとで実行されないとき。
- 原始要素の最後の文で、それを含む段落が他の COBOL 文の制御のもとで実行されないとき。
- 宣言部分の最後の文で、それを含む PERFORM 文の出口の手続き文がこの宣言部分の最後の文ではないとき。
- COBOL 原始要素の外に制御を移す STOP RUN (実行停止) 文、

  (ISO/IEC 1989:2002) (SVS) (VSC2) (MF) GOBACK(復帰)文、

  (ISO/IEC 1989:2002) (MF) EXIT METHOD(メソッド出口)文、

  (ISO/IEC 1989:2002) EXIT FUNCTION(関数出口)文、

  または EXIT PROGRAM(プログラム出口)文。

- プログラム終了見出し

2.6.6.3 明示属性と暗黙属性

属性は、暗黙的にも明示的にも指定できる。明示的に指定した属性を、明示属性 (explicit attribute) という。属性を明示的に指定しないと、省略時の解釈が取られる。このような属性を、暗黙属性 (implicit attribute) という。

たとえば、データ項目の用途 (usage) は指定しなくてよい。指定のないデータ項目の用途は DISPLAY (表示用) と解釈される。

(VSC2) (MF) (COB31) PICTURE 文字列が"G"または"N"を持たないを除く。この場合、データ項目の用途は DISPLAY-1 である。

2.6.6.4 明示範囲符と暗示範囲符

(ANS85)

範囲符は、手続き部のある種の文 (範囲明示文) の範囲を区切る。これには、明示範囲符と暗黙範囲符の 2 種類がある。

有効な明示範囲符には以下のものがある。

END-ADD
END-PERFORM

MF XOPEN END-ACCEPT

END-CALL

MF END-CHAIN

END-COMPUTE
END-DELETE

MF XOPEN END-DISPLAY

END-DIVIDE
END-EVALUATE
END-IF
END-MULTIPLY
END-READ
END-RETURN
END-REWRITE
END-SEARCH
END-START
END-STRING
END-SUBTRACT
END-UNSTRING
END-WRITE

注：COBOL 言語の方言が異なると、範囲符と対になる範囲明示文が異なることがあるので注意。

暗黙範囲符は下記の場合に発生する。

- 完結文の末尾で、それまで続いている前の文のすべての範囲が分離符の終止符によって終わりとされるとき。

- 他の文を含む文の中で、含まれる文のそれまで続いている文の範囲が、含まれる文に続く含む方の文の次の句（たとえば、ELSE, WHEN, AT END）によって終わりとされるとき。

第 3 章：翻訳集団の概念

3.1 指定しなくてもよい部、節、段落の見出し

MF

この COBOL システムでは、ANSI 標準 COBOL で必要とされる"red-tape"文のうち、指定してもしなくてもよいものがある。しかし、FLAG 指令を使用すると、それらの文が抜けているときに、COBOL コンパイラに警告メッセージを出させるようにすることができる。このマニュアルでは、指定しても指定しなくてもよい文を示すために、それらの文の前後を角かっこ[ ]で囲み、その回りを四角でを囲んでいる。その隣に示した記号は、その機能が任意指定となっている方言を表わす。

3.2 予約語

すべての予約語 (reserved word) を、付録、*予約語*に列挙してある。

3.3 外部リポジトリ

外部リポジトリには、プログラム定義、クラス定義、インターフェイス定義に指定された情報が格納される。

これらのソース単位について格納された情報には、起動および妥当性のチェックに必要なすべての情報が含まれていなければならない。具体的には、下記の情報が必要である。

- ソース単位の外部名
- ソース単位のタイプ - プログラム、クラス、インターフェイス
- ソース単位のパラメータ (存在する場合)
- ソース単位の返却する項目 (存在する場合)
- ソース単位のオブジェクトプロパティ (存在する場合)
- ソース単位中のメソッド (存在する場合) およびメソッドの外部名、パラメータ、返却する項目の詳細
- ソース単位中に DECIMAL-POINT IS COMMA 句が指定されているか否か。クラスに関しては、この詳細情報はファクトリとオブジェクトの両方のために格納されなければならない。
- ソース単位中に CURRENCY 句が指定されているか否か。クラスに関しては、この詳細情報はファクトリとオブジェクトの両方のために格納されなければならない。

ソース単位に関するこれらの情報のうち、外部名を除いたものをシグネチャと呼ぶ。

REPOSITORY 指令が ON 指定と共に指定された場合、コンパイラは各翻訳単位をコンパイルするさいにその情報を外部リポジトリに入れる。

REPOSITORY が明示的にまたは省略値として CHECKING 指定と共に指定された場合、定義およびプロトタイプのそれらの特性が指定されていて外部リポジトリ中の対応する定義と合致しないと、コンパイラは警告メッセージを発する。

ソース単位の名前と外部リポジトリ中の情報の関連の詳細は、*環境部*の章の *リポジトリ段落* の節に指定される。

3.4 CALL プロトタイプ

MF

CALL プロトタイプは、呼び出されるサブプログラムの特性を定義するのに役立つ、プログラム宣言ないしプログラムの骨組みである。それらの特性には、必要なパラメータの数、それらのパラメータの型、呼出し方式がある。プロトタイプが存在するサブプログラムを参照する CALL 文がソース要素中に入っている場合、その CALL 文はそのプロトタイプと照合チェックされる。明確な不一致がある場合には、エラーとされる。CALL 文中に定義されていない特性が何かある (たとえば、呼出し方式) 場合、それらプロトタイプから引き出される。

CALL プロトタイプは完全なプログラムとして定義され、その PROGRAM-ID 段落中に EXTERNAL 句が指定される。そのプログラム構造は、プログラム名段落を伴う見出し部、連絡節を伴うデータ部、手続き部の見出し、および ENTRY 文のみから構成されなければならない。ただし、ENTRY 文は指定してもしなくてもよい。これらの CALL プロトタイプは他の型の翻訳単位の前に置かれる。その方法はマルチプログラム原始ファイルの場合と同様である。

CALL 文をチェックさせるすべてのサブプログラムに関して、その CALL 文が含まれる翻訳単位の見出し部の見出しの前に、CALL プロトタイプのコピーを含めなければならないことに注意。

3.5 ファイル

3.5.1 ファイル結合子

ANS85

ファイル結合子はファイルに関する情報が格納されている領域であって、ファイル名と物理ファイルの間およびファイル名とそれに関連するレコード領域との間の連絡に使用される。

3.5.2 順ファイル

この章において、特に「行」または「レコード」と断らずに「順ファイル」または「順編成」という用語を使用している場合、その記述内容は両方のファイルに当てはまる。省略時解釈の順ファイルの動作は、SEQUENTIAL コンパイラ指令の影響を受ける。

3.5.2.1 レコード順ファイル

レコード順ファイルでは、ファイル中のレコードを確立されている順に呼び出すことができる。レコードの順序は、ファイルにレコードを書き込むことによって確立される。

3.5.2.2 行順ファイル

(MF)

行順ファイルでは、テキストファイルのレコードを確立されている順序に従って呼び出すことができる。行順ファイルのフォーマットは、オペレーティングシステムのエディタによって作成されるフォーマットと同じである。格納されるレコードの末尾の空白は削除される。

3.5.2.3 レコード順ファイルおよび(MF)行順ファイルの編成

順ファイルにおいては、各レコードに先行レコードと後行レコードが必ず存在する。ただし、先頭のレコードには先行レコードはなく、末尾のレコードには後行レコードはない。先行/後行関係は、ファイルを作成するときの WRITE 文によって決まる。いったん確立された先行/後行関係は通常は変わらない。ただし、ファイルの末尾にレコードが追加された場合は、例外である。

3.5.2.4 呼出し法

順ファイルに適用できる呼出し法は、順呼出し法だけである。つまり、レコードを呼び出す順序は、 レコードが書き出された順序である。

3.5.3 相対ファイル

相対ファイルを使用すると、プログラマは大記憶ファイル内に書かれているレコードを、乱呼び出しすることも順呼び出しすることもできる。 相対ファイル中の各レコードは、正の整数値によって識別される。この整数値は、ファイル中のレコードの位置の順序番号を表わしている。

3.5.3.1 相対ファイルの編成

相対ファイル編成は、ディスク装置上でだけ使用できる。相対ファイルは、相対レコード番号 (relative record number) によって識別されるレコードから構成される。相対ファイルは、1 論理レコードを保持する領域が連続して並んでいるものと考えることができる。各領域には相対レコード番号が付けられている。たとえば、相対レコード番号 10 番は 10 番目のレコード領域を指す。

3.5.3.2 呼出し法

順呼出し法 (sequential access method) では、ファイル中に現存するレコードを相対レコード番号の昇順に呼び出す。

乱呼出し法 (random access method) では、レコードを呼び出す順序はプログラマが制御する。求 めるレコードを呼び出すには、相対レコード番号を相対キーデータ項目に設定する。

動的呼出し法 (dynamic access method) では、プログラマが順呼出しと乱呼出しを自由に切り替えることができる。このためには、適切なファイル入出力文を使用する。

3.5.4 索引ファイル

索引ファイルを使用すると、プログラマは大記憶ファイル内に書かれているレコードを、乱呼び出し することも順呼び出しすることもできる。索引ファイル中の各レコードは、その中の 1 つ以上のキー項目の値によって識別される。

3.5.4.1 索引ファイルの編成

索引ファイルは大記憶ファイルであり、その中のレコードはキーの値によって呼び出される。キーはレコード記述に定義する。1 つ以上の項目をキーに指定できる。各キー項目には索引が付けられる。各索引はレコード中のキーデータ項目の内容に基づいて、そのデータレコードを呼び出す論理的な経路となる。

ファイルのファイル管理記述項の RECORD KEY 句に指定したデータ項目が、そのファイルの主レコードキー (primary record key) となる。 ファイル中のレコードの挿入、更新、および削除の対象となるレコードは主レコードキーによってのみ識別される。したがって、主レコードキーの値は一意でなければならず、レコードを更新する際に変更されてはならない。

ファイルのファイル管理記述項の ALTERNATE RECORD KEY 句に指定したデータ項目は、そのファイルの副レコードキー (alternate record key) となる。DUPLICATES 指定をしておけば、副レコードキーの値は一意でなくてもよい。副レコードキーを使用することによって、ファイルからレコードを検索する際の代わりの呼出し経路とすることができる。

MF この COBOL システムでは、分割キーを使えるように機能が拡張されている。分割キーとは、いくつかのデータ項目から構成されるキーである。レコード記述の中では、それらのデータ項目は互いに隣接し ていなくても構わないものである。

3.5.4.2 呼出し法

順呼出し法では、レコードキーの値の昇順にレコードを呼び出す。同じレコードキーの値をもつレコードの組の中では、レコードはその組の中で書き出された順に呼び出される。

乱呼出し法では、レコードを呼び出す順序はプログラマが制御する。求めるレコードを呼び出すには、レコードキーデータ項目にレコードキーの値を設定する。

動的呼出し法では、プログラマが順呼出しと乱呼出しを自由に切り替えることができる。そのためには、適切なファイル入出力文を使用する。

3.5.5 共有モード

共有モードとは、ファイルを共有しレコードのロックを行うか否かを示すとともに、ファイルに関して許す共有（または非共有）の度合いを表す。共有モードには、該当の OPEN の期間中に、他のファイル結合子を通じて共有ファイルに加えることのできる処理を指定する。

共有モードの確立に関して、OPEN 文中の SHARING 指定はファイル管理記述項中の SHARING 句よりも優先する。OPEN 文中に SHARING 指定がない場合には、共有モードはファイル管理記述項中の SHARING 句によって完全に決定される。どちらにも指定がない場合には、下記の条件のうちの最初に満足されるものによって、共有モードが決定される。

- OPEN 文中に WITH LOCK 指定が書かれている場合は、共有モードは SHARING WITH NO OTHER となる。
- SELECT 句に LOCK MODE IS EXCLUSIVE 指定が書かれている場合は、共有モードは SHARING WITH NO OTHER となる。
- SELECT 句に LOCK MODE IS MANUAL 指定または LOCK MODE IS AUTOMATIC 指定が書かれている場合は、共有モードは SHARING WITH ALL OTHER となる。
- OPEN モードが OUTPUT, I-O, EXTEND のいずれかである場合、共有モードは SHARING WITH NO OTHER となる。
- OPEN モードが INPUT である場合は、環境オプションの OPENINPUTSHARED に応じて共有モードが決まる。このオプションが ON に設定されていると、共有モードは SHARING WITH ALL OTHER となる。そうでなければ、共有モードは SHARING WITH READ ONLY となる。

共有モードが OPEN 文かファイル管理段落の中に指定されている場合でも、上記のリストから決まる場合でも、標準の共有モードに関する規則は同じである。

COBOL のファイル共有機能とその環境内の他のファイル共有機能との間に相互運用性はない。

共有ファイルはディスク上に存在しなければならない。

OPEN 文を通じて共有ファイルへのアクセスが許可される前に、ファイルに現在関連している他すべてのファイル結合子によって、共有モードとオープンモードが許可される。それに加えて、現在の OPEN 文の共有モードによって、ファイルに現在関連している他すべてのファイル結合子用のすべての共有モードとオープンモードが許可される。（***手続き部 - MERGE - OPEN*** の章の *OPEN(開く)文*、とくに表、***現在は他のファイル結合子によって開かれている利用可能な共有ファイルを開く処理***を参照。）

共有モードでは、ファイルへのアクセスは下記のように制御される。

1. SHARING WITH NO OTHER はファイルを排他的にアクセスすることを指定する。その指定のあるファイルが現在他のファイル結合子を通じて開かれているならば、そのファイル結合子をファイルに関連付けようとしても失敗する。OPEN 文が成功したならば、そのファイル結合子が閉じられてないうちに、他のファイル結合子を通じてそのファイルを開こうとする以降の要求は不成功に終わる。レコードロックは無視される。
2. 読み取り専用モードで共有すると、該当のもの以外のファイル結合子を通じてそのファイルに同時にアクセスするのは、入力モードのみに限定される。入力以外のモード

で開かれているファイルをこのファイル結合に関連付けようとすると、エラーとなる。OPEN 文が成功したならば、そのファイル結合子が閉じられないうちに、入力以外のモードにある他のファイル結合子を通じてそのファイルを開こうとする以降の要求は不成功に終わる。レコードロックは効力を有する。

3. 上記以外のいかなるモードで共有する場合も、INPUT, I-O, EXTEND のどれかを指定した他のファイル結合子を通じて、ファイルに同時アクセスすることができる。ただし、他に制限が適用される場合がある。レコードロックは効力を有する。

同じ実行単位または別の実行単位内のランタイム要素や内包されている要素や別のランタイムモジュールの中に、複数のアクセス経路が存在する可能性がある。開く対象のファイルのファイルロックを保持しているファイル結合子のロケーションに関係なく、ファイル共有に矛盾を来す条件が存在する可能性がある。

ファイルロックを設定することは、入出力文の不可分な処理の一部である。

あるファイル結合子に用に確立されたファイルロックおよびすべてのレコードロックは、そのファイル結合子に関して明示的または暗黙的に実行された CLOSE 文によって解除される。

3.6 オブジェクト指向 COBOL の概念

オブジェクト指向プログラミングについて詳しくは、マニュアル、*オブジェクト指向 COBOL プログラミング* を参照。

3.6.1 オブジェクトとクラス

オブジェクトとは、独自のデータを持ちクラス定義中に定義されているメソッドを共有する、ランタイム要素であるオブジェクトはクラスによって定義される。クラス定義には、データおよびクラスの各オブジェクトに対して呼び出される可能性あるメソッドの特性が記述されている。

各クラスには、ファクトリオブジェクトが1つある。ファクトリオブジェクトには独自の一連のデータとメソッドが備わっている。ファクトリオブジェクトは通常、オブジェクトのインスタンスを生成し、クラスのすべてのインスタンスに共通のデータを維持更新する働きをする。.

3.6.2 オブジェクト参照

オブジェクト参照とは、あるオブジェクトが存在している間そのオブジェクトを一意に参照する値（オブジェクト参照値）を含む、暗黙的（または明示的）に定義されたデータ項目を意味する。暗黙的に定義されたオブジェクト参照とは、予め定義されているオブジェクト参照および、オブジェクトプロパティ、オブジェクトビュー、行中のメソッド呼出しまたは関数から返されるオブジェクト参照である。明示的に定義されたオブジェクト参照とは、USAGE OBJECT REFERENCE 句を定義しているデータ記述項によって定義されているデータ項目である。

2つの別個のオブジェクトが同じオブジェクト参照値を持つことはない。各オブジェクトには少なくとも1つのオブジェクト参照がある。

3.6.3 定義済みオブジェクト参照

定義済オブジェクト参照とは、一意名、NULL、SELF、

(MF)SELFCLASS、

および SUPER のいずれかによって参照される、暗黙的に生成されたデータ項目である。各定義済オブジェクト参照にはそれなりの意味がある。それについては、*COBOL 言語の概念*の章の*定義済みオブジェクト一意名*節に説明がある。

3.6.4 メソッド

ブジェクト中の手続きコードはメソッドに収められる。各メソッドには、独自のメソッド名とデータ部と手続き部がある。メソッドが呼び出されると、その中の手続きコードが実行される。メソッドを呼び出すためには、そのオブジェクトを参照する一意名と該当のメソッド名を指定する。メソッドにパラメータと返却する項目を指定することができる。

3.6.5 メソッド呼出し

INVOKE 文を用いてメソッドを呼び出すと，その中の手続きコードが実行される。その際、INVOKE 文、行中のメソッド呼出し、

(MF)動詞シグネチャの参照、または

オブジェクトプロパティの参照が使用される。メソッド呼出しに対応するメソッド処理系は、メソッド呼出しの対象のオブジェクトの実行時のクラスによって決まる。それは必ずしもオブジェクト参照の定義中に静的に指定されているクラスであるとは限らない。メソッド呼出しを具体的なメソッド処理系に対応付けるために使用されるのは、実行時に実際に参照されるオブジェクトのクラスである。

オブジェクト参照を用いてオブジェクトを指定してメソッドを呼び出す場合は下記のようになる。

1. 識別されたオブジェクトがファクトリオブジェクトである場合は、ファクトリメソッドが呼び出される。
2. そうでない場合は、オブジェクトメソッドが呼び出される。

クラス名を用いてオブジェクトを指定してメソッドを呼び出す場合、指定したクラスのファクトリオブジェクトがメソッドを呼び出す対象のオブジェクトとして使用されて、ファクトリメソッドが呼び出される。

メソッドの解明処理は下記のように進められ、該当する最初のものが適用される。

1. 呼出しに指定されたメソッド名のメソッドがオブジェクトのクラス中に定義されている場合は、そのメソッドが対応付けられる。そうではなく、
2. 呼出しに指定されたメソッド名のメソッドがオブジェクトのクラスによって継承されたクラス中に定義されている場合は、そのメソッドが対応付けられる。継承されるクラスにはクラス階層のさらに上位の任意のクラスを含む。
3. メソッドが見つからない場合は、実行時にエラーとなる。

3.6.6 適合性とインターフェイス

このドキュメントで使用する「適合性」という用語にはいくつかの意味がある。オブジェクト指向の観点からは、「適合性」という用語はオブジェクトインターフェイス間の関連を記述するために使用され、継承やインターフェイス定義や適合性チェックといった基本的な機能の基盤となる。

注：適合性チェックはコンパイル時にのみ行われる。ただし、オブジェクト修飾子を使用する場合、および一般的オブジェクト参照を用いてメソッドを呼び出す場合は、例外である。

3.6.6.1 オブジェクト指向への準拠

オブジェクトが適合性のあるものであると、オブジェクト自体のクラスのインターフェイス以外のインターフェイスに従ってクラスを指定することができる。適合性とは、あるインターフェイスから別のインターフェイスへの、およびあるオブジェクトからあるインターフェイスへの、一方向性の関連である。

3.6.6.1.1 インターフェイス

どのオブジェクトにもインターフェイスがある。オブジェクトのインターフェイスは、名前およびそのオブジェクトでサポートされている各メソッドのパラメータ仕様から構成される。そのメソッドには継承されたメソッドも含む。各クラスにはインターフェイスが2つある。具体的には、ファクトリオブジェクト用のインターフェイスとオブジェクト用のインターフェイスである。

個々のクラスから独立にインターフェイスを定義することもできる。そのためには、インターフェイス定義中に、メソッドプロトタイプを指定する。

3.6.6.1.2 インターフェイス間の適合性

下記の条件に該当する場合にのみ、インターフェイス「インターフェイス-1」はインターフェイス「インターフェイス-2」に準拠する。

1. インターフェイス-2 中の各メソッドに対応するメソッドがインターフェイス-1 中に存在し、その名前が同じで同じ数のパラメータを取り BY REFERENCE および BY VALUE の仕様が一貫している。

2. インターフェイス-2 中のあるメソッドの仮パラメータがオブジェクト参照であるならば、インターフェイス-1 中の対応する仮パラメータは下記の規則に従ったオブジェクト参照である。
    a. インターフェイス-2 中のパラメータが一般的オブジェクト参照であるならば、インターフェイス-1 中の対応するパラメータも一般的オブジェクト参照である。
    b. インターフェイス-2 中のパラメータがインターフェイス名を用いて記述されているならば、インターフェイス-1 中の対応するパラメータも同じインターフェイス名を用いて記述されている。
    c. インターフェイス-2 中のパラメータがクラス名を用いて記述されているならば、インターフェイス-1 中の対応するパラメータも同じクラス名を用いて記述されている。また、FACTORY 指定および ONLY 指定の有無が両方のインターフェイスで同じである。
    d. インターフェイス-2 中のパラメータが ACTIVE-CLASS 句を用いて記述されているならば、インターフェイス-1 中の対応するパラメータも ACTIVE-CLASSACTIVE-CLASS 句を用いて記述され、両方のインターフェイス中での FACTORY 句の有無も同じとなる。
3. インターフェイス-2 中のあるメソッドの仮パラメータがオブジェクト参照でないならば、インターフェイス-1 中の対応する仮パラメータに同じ ANY LENGTH、PICTURE、USAGE、SIGN、SYNCHRONIZED、JUSTIFIED、および BLANK WHEN ZERO 句が指定されている。ただし、下記の例外がある。
    a. 対応する通過文字列が同じである場合に限り、通貨記号は一致する。
    b. DECIMAL-POINT IS COMMA 句が両方のインターフェイス中で有効または無効である場合に限り、終止符は一致する。DECIMAL-POINT IS COMMA 句が両方のインターフェイスで有効または無効である場合に限り、コンマは一致する。
4. 対応するメソッド中では、手続き部内の RETURNING 指定の有無が同じである。
5. インターフェイス-2 のあるメソッド中の返却する項目がオブジェクト参照であるならば、インターフェイス-1 中の対応する返却する項目も下記の規則に従ったオブジェクト参照である。
    a. インターフェイス-2 中の返却する項目が一般的オブジェクト参照であるならば、インターフェイス-1 中の対応する返却する項目はオブジェクト参照である。
    b. インターフェイス-2 中の返却する項目がインターフェイス int-r を識別するインターフェイス名を用いて記述されているならば、インターフェイス-1 中の対応する返却する項目は下記のどちらかである。
        1. int-r または、int-r を参照する INHERITS 句で記述されたインターフェイスを識別するインターフェイス名を用いて記述されているオブジェクト参照。
        2. 下記の規則に従って、クラス名を用いて記述されているオブジェクト参照。
            a. 記述に FACTORY 指定が含まれているならば、指定されたクラスのファクトリオブジェクトは、int-r を参照する IMPLEMENTS 句で記述される。
            b. 記述に FACTORY 指定が含まれていないならば、指定されたクラスのインスタンスオブジェクトは、int-r を参照する IMPLEMENTS 句で記述される。

c. インターフェイス-2 中の返却する項目がクラス名用いて記述されているならば、インターフェイス-1 中の対応する返却する項目はオブジェクト参照である。ただし、下記の規則に従う。
    1. インターフェイス-2 中の返却する項目の記述に ONLY 指定があるならば、インターフェイス-1 中の返却する項目の記述にも ONLY 指定および同じクラス名が指定されている。
    2. インターフェイス-2 中の返却する項目の記述に ONLY 指定がないならば、インターフェイス-1 中の返却する項目の記述には同じクラス名またはそのクラス名のサブクラスが指定されている。
    3. FACTORY 指定の有無が同じである。
d. インターフェイス-2 中の返却する項目の記述に ACTIVE-CLASS 指定が指定されているならば、インターフェイス-1 中の対応する返却する項目の記述にも ACTIVE-CLASS 指定が指定されている。そして、FACTORY 指定の有無が同じである。

インターフェイス-1 中のメソッドの返却する項目の記述においてインターフェイス-2 が直接的または間接的に参照されているならば、インターフェイス-2 中の対応するメソッドの返却する項目の記述においてインターフェイス-1 を直接的にも間接的にも参照しない。

6. インターフェイス-2 のあるメソッド中の返却する項目がオブジェクト参照でない場合、対応する返却する項目の PICTURE, USAGE, SIGN, SYNCHRONIZED, JUSTIFIED, BLANK WHEN ZERO の各句が同じである。ただし、下記の例外がある。
    a. 対応する通過文字列が同じである場合に限り、通貨記号は一致する。
    b. DECIMAL-POINT IS COMMA 句が両方のインターフェイス中で有効または無効である場合に限り、終止符は一致する。DECIMAL-POINT IS COMMA 句が両方のインターフェイスで有効または無効である場合に限り、コンマは一致する。
7. 適合性チェックの観点からは、集団項目は同じ長さの英数字の基本項目であるとみなされる。
8. 対応するパラメータでは、OPTIONAL 句の有無は同じである。.

3.6.6.1.3 パラメータ化されたクラスおよびインターフェイスのための適合性

パラメータ化されたクラスまたはインターフェイスを使用するときは、クラス定義またはインターフェイス定義の全体を通じて、パラメータの代わりに実パラメータのクラスまたはインターフェイスが用いられているかのように、クラスまたはインターフェイスは扱われる。

3.6.7 多相性

多相性とはあるひとつの文で種々の事柄を行えるようにする機能である。COBOL においては、ひとつのデータ項目に種々のオブジェクトまたは異なるクラスを保持できることは、そのデータ項目に対するメソッド呼出しは可能性のある多くのメソッドのうちのどれかひとつに対応することを意味する。時には実行前にそのメソッドを識別できることがある。しかし、一般的には、実行するときまでそのメソッドを識別することはできない。

あるクラスのオブジェクトまたはそのクラスの任意のサブクラスを保持するものとして、データ項目を定義することができる。また、あるインターフェイスに準拠するオブジェクトクラスを保持するものとして、データ項目を定義することもできる。インターフェイスを使用する場合、オブジェクト同士のクラス間に関係がまったくないことがありえる。

3.6.8 クラスの継承

クラスの継承とは、あるクラスのインターフェイスと処理系を、他のクラスの基盤として使用する仕組みである。下位のクラス（サブクラスとも言う）は上位のクラス（スーパークラスとも言う）から継承する。サブクラスは継承元のクラス中に定義されているすべてのメソッドを受け継ぐ。その中には、継承元がさらに上位から継承したメソッドも含まれる。サブクラスは継承元のクラス中に定義されているすべてのデータ定義を受け継ぐ。その中には、継承元がさらに上位から継承したデータ定義も含まれる。

注：これは、データを記述した実際のソースコードがアクセス可能であるという意味ではなく、このソースコードに記述されたデータ項目がサブクラスで直接参照可能であるという意味でもない。サブクラスは、そのソースコードにスーパークラス定義のコピーが含まれているかのように取り扱われる、つまり、継承されたデータ項目がサブクラスで定義されていると見なされるという意味である。

継承されたデータ定義には、継承されたデータが、サブクラスのすべてのオブジェクトおよびそのファクトリのために定義されている。各インスタンスオブジェクトそれぞれ、継承されたクラスのインスタンスオブジェクトが持つコピーとは異なる、継承されたデータのコピーを持つ。各ファクトリオブジェクトはそれぞれ、継承されたクラスのファクトリオブジェクトが持つコピーとは異なる、継承されたデータのコピーを持つ。継承されたデータ項目の名前と属性は、継承されたクラス内では不可視である。継承されたオブジェクトデータは、オブジェクトが作成されるときに初期化される。継承されたファクトリデータは、継承元のひとつまたは複数のクラスのファクトリデータから独立して割り当てられ、サブクラスのファクトリが作成されるときに初期化される。継承されたファクトリデータは、データを記述するクラスのファクトリ定義で指定されたメソッドとプロパティを介してのみ、アクセス可能である。継承されたオブジェクトデータは、データを記述するクラスのオブジェクト定義で指定されたメソッドとプロパティを介してのみ、アクセス可能である。サブクラスは、データ定義と同様に、すべてのファイル定義を継承し、データ定義と同様の条項の対象となる。サブクラスが定義するメソッドには、継承されたメソッドが含まれない場合もある。また、サブクラスはデータ定義およびファイル定義を指定できるが、これには、継承されたデータ定義およびファイル定義が含まれなければならない。

サブクラスのインターフェイスは継承元のクラスのインターフェイスに必ず準拠する。ただし、継承元のクラスのメソッドの一部をサブクラスにおいて修正して、別の処理系を用意することができる。

継承元のクラス中でユーザが定義した語はサブクラスに継承されない。したがって、それが継承元のクラスに定義されていないかのように、サブクラスの中で定義することができる。

3.6.9 インターフェイスの継承

インターフェイスの継承とは、インターフェイスの定義を、他のインターフェイスの基盤として使用する仕組みである。インターフェイスを継承すると、継承元のインターフェイスに定義されているすべてのメソッド仕様が受け継がれる。その中には、継承元がさらに上位から継承したメソッド定義も含まれる。継承を受けるインターフェイスにおいて新しいメソッドを定義して、一連の継承したメソッドを補強することができる。継承を受けるインターフェイスは継承元のインターフェイスに必ず準拠しなければならない。.

3.6.10 インターフェイスの処理系

インターフェイスの処理系とは、クラスの基盤として 1 つ以上のインターフェイス定義を使用する仕組みである。処理するクラスには、処理するインターフェイス定義に指定されたメソッドのすべてを処理しなければならない。これには、処理定義が継承したあらゆるメソッドが含まれる。処理するクラスのファクトリオブジェクトのインターフェイスは、ファクトリオブジェクトで処理されるインターフェイスに準拠していなければならず、処理するクラスのインスタンスオブジェクトのインターフェイスは、インスタンスオブジェクトで処理されるインターフェイスに準拠していなければならない。

3.6.11 パラメータ化されたクラス

パラメータ化されたクラスは一般形をしたクラスないしクラスの骨組みである。その中には仮パラメータが指定されていて、それが後でいくつかのクラス名またはインターフェイス名で置き換えられる。仮パラメータが実パラメータのクラス名またはインターフェイス名で置き換えられて、パラメータ化されたクラスが展開されたときに、パラメータ化されていないクラスとして機能するクラスが生成される。この拡張について詳しくは、ACTUAL-PARAMS コンパイラ指令を参照。

パラメータ化されたクラスのライフサイクルは、この章で後述する、*パラメータ化されたクラスのライフサイクル*中に定義する。.

3.6.12 パラメータ化されたインターフェイス

パラメータ化されたインターフェイスは一般形をしたインターフェイスないしインターフェイスの骨組みである。その中には仮パラメータが指定されていて、それが後でいくつかのクラス名またはインターフェイス名で置き換えられる。仮パラメータが実パラメータのクラス名またはインターフェイス名で置き換えられて、パラメータ化されたクラスが展開されたときに、パラメータ化されていないインターフェイスとして機能するインターフェイスが生成される。この拡張について詳しくは、ACTUAL-PARAMS コンパイラ指令を参照。

パラメータ化されたインターフェイスのライフサイクルは、この章で後述する、*パラメータ化されたインターフェイスのライフサイクル*中に定義する。

3.6.13 オブジェクトのライフサイクル

オブジェクトのライフサイクルは、オブジェクトが生成されたときに始まり、オブジェクトが破棄されたときに終わる。

3.6.13.1 ファクトリオブジェクトのライフサイクル

ファクトリオブジェクトは、実行単位によって最初に参照される前に、生成される。

ファクトリオブジェクトは、実行単位によって最後に参照された後で、破棄される。

3.6.13.2 オブジェクトのライフサイクル

ファクトリオブジェクトに対して NEW メソッドが呼び出された結果として、オブジェクトが生成される。

実行単位が終了したときに、オブジェクトが破棄される。

(MF)実行単位が終了する前に、オブジェクトに対して FINALIZE メソッドが呼び出された場合は、その結果として、オブジェクトが破棄される。

3.6.13.3 パラメータ化されたクラスのライフサイクル

パラメータ化されたクラスが展開されると、あらゆる点で、パラメータ化されていないクラスと同様に扱われるようになる。

REPOSITORY 段落中にパラメータ化されたクラスが指定されたときに、パラメータ化されたクラスの仕様に基づいて、新しいクラス（パラメータ化されたクラスのインスタンス）が生成される。そのクラスは独自のファクトリオブジェクトを持ち、同じパラメータ化されたクラスの他のインスタンスとは完全に別物である。

実行単位内で、パラメータ化された同じクラスを同じ実パラメータで展開して作成された2つのクラスの外部クラス名が同じ場合、その2つは同じクラスインスタンスである。パラメータ化されたクラスを異なる実パラメータで展開して作成された2つのクラスは同じクラスインスタンスではなく、同じ外部クラス名を持たない。

3.6.13.4 パラメータ化されたインターフェイスのライフサイクル

パラメータ化されたインターフェイスが展開されると、あらゆる点で、パラメータ化されていないインターフェイスと同様に扱われるようになる。

REPOSITORY 段落中にパラメータ化されたインターフェイスが指定されたときに、パラメータ化されたインターフェイスの仕様に基づいて、新しいインターフェイス（パラメータ化されたインターフェイスのインスタンス）が生成される。

実行単位内で、パラメータ化された同じインターフェイスを同じ実パラメータで展開して作成された2つのインターフェイスの外部インターフェイス名が同じ場合、その2つは同じインターフェイスインスタンスである。パラメータ化されたインターフェイスを異なる実パラメータで展開して

作成された2つのインターフェイスは同じインターフェイスインスタンスではなく、同じ外部インターフェイス名を持たない。

3.7 実行単位における通信

3.7.1 共通プログラム、初期プログラムおよび再帰プログラム

その初期状態に影響する属性、またはその呼び出し方法を定義する属性を用いて、プログラムを記述することができる。

ANSI85 共通プログラム (common program) とは、他のプログラムの中に直接的に含まれていながら、そのプログラムに直接的または間接的に含まれる他のプログラムからも呼び出せるものをいう (*COBOL 言語の概念*の章の*名前の適用範囲* 節を参照) 。この共通属性 (common attribute) は、プログラムの見出し部内に COMMON 指定を書くことによって付与される。この指定によって、あるプログラムに含まれるすべてのプログラムによって使用される、副プログラムを容易に作成できる。COMMON 指定が記述されていない場合は、再帰的でない、内包されるプログラムは直接内包しているプログラムによってのみ呼び出しが可能となる。
COMMON 指定により、プログラムに内包される全プログラムにより使用可能なサブプログラムの記述が容易になる。

ANSI85 初期プログラム (initial program) とは、呼ばれたときに状態が初期化されるプログラムをいう。初期プログラムの初期化処理の過程で、そのプログラムのデータが初期化される。詳しくは、*メソッド、オブジェクトまたはプログラムの状態*の節で解説する。 初期属性は、プログラムの見出し部内に INITIAL 句を書くことによって付与される。

OS/390 SQL2002 MF 再帰プログラム (recursive program) は、直接または間接的に自信を呼び出すことができるプログラムである。このプログラムの内部データの初期化については、*関数、メソッドまたはプログラムの状態*の節で解説する。再帰属性の獲得については、*見出し部*の章の*プログラム名段落*の節で解説する。

OS/390 SQL2002 MF メソッドはつねに、再帰的である。そのデータは、再帰プログラムと同様に初期化される。

OS/390 SQL2002 MF 初期プログラムでも再帰プログラムでもないプログラムは、プログラムのデータは、最終使用状態 (最初の起動時を除く) である。これについては、*関数、メソッドまたはプログラムの状態*の節で解説する。再帰属性を持つプログラム以外のプログラムは、実行中に起動することはできない。

3.7.2 データの共有

ANSI85

1 つの実行単位の中の 2 つのランタイム要素は、下記の状況において、共通データ (common data) を参照できる。

- 任意のランタイム要素から外部データレコードの内容を参照できる。ただし、参照元のランタイム要素の中に参照対象のデータレコードを記述しておく必要がある。
- あるプログラムが別のプログラムに含まれている場合、両方のプログラムとも大域属性 (global attribute) をもつデータを参照できる。ここで、対象となる大域属性をもつデータは、それを含むプログラム中のもの、またはそのプログラムを直接的または間接的に含むプログラム中のものである。
- パラメータの値を、呼ぶランタイム要素から呼ばれるランタイム要素へ参照によって引き渡す仕組みによって、共通データ項目が設定される。呼ばれるランタイム要素は呼ぶランタイム要素のデータ項目を参照する際に、別の一意名を使用できる。

3.7.3 ファイル結合子の共有

ANS85

1 つの実行単位の中の 2 つのランタイム要素は、下記の状況において、共通ファイル結合子 (common fileconnector) を参照できる。

- 該当のファイル結合子を記述する任意のランタイム要素から外部ファイル結合子を参照できる。
- あるプログラムが別のプログラムに含まれている場合、両方のプログラムとも関連する大域ファイル名を参照することによって、共通のファイル結合子を参照できる。ここで、対象となる大域ファイル名は、それを含むプログラム中のもの、またはそのプログラムを直接的または間接的に含むプログラム中のものである。

3.8 データ部

3.8.1 概要

データ部は下記の節に分割される。

1. ファイル節
2. 作業場所節 n
3. MF スレッド局所記憶節
4. MF オブジェクト記憶節
5. SOMM MF 局所記憶節
6. 連絡節
7. 報告書節
8. SOMM MF OPEN 画面節

ファイル節では、データファイルの構造を定義する。各ファイルを定義するには、ひとつのファイル記述項といくつかのレコード記述を書く。レコード記述はファイル記述項の直後に書く。

作業場所節には、外部データファイルに属するのではなく、内部的に作成され処理されるレコードおよび非連続的なデータ項目を記述する。また、作業場所節には、値が原始文内で割り当てられて実行中には変化しないデータ項目も記述する。

(MF)スレッド局所記憶節には、各スレッドに固有であり、呼出しの間に一貫しているデータを記述する。スレッド局所記憶節はスレッドに固有の作業場所節であると見ることができる。これは再入可能なプログラムの大部分における競合問題を解決するのに役立つ。多くの場合、節の見出しを WORKING-STORAGE から THREAD-LOCAL-STORAGE に変更するだけで、ファイル処理を伴わないプログラムを完全に再入可能にすることができる。

(MF)オブジェクト記憶節には、クラスオブジェクトデータおよびインスタンスオブジェクトデータを記述する。その構造は連絡節および作業場所節と同様である。

(2002)(MF)局所記憶節では、プログラムの再帰可能性を明示する。再帰可能なメソッド中にも、局所記憶節を記述できる。ランタイム要素が起動されるたびに局所記憶節の別立てのコピーが作成される。そのコピーはそのランタイム要素の寿命がある間だけ存在する。

連絡節は別のソース要素によって起動されるソース要素の中に置かれる。連絡節には、起動する側と起動される側の両方の要素によって参照されるデータ項目を記述する。その構造は作業場所節と同様である。

報告書節にはいくつかのレポート記述項 (RD 記述項) が含まれる。各レポート記述項はひとつのレポートを完全に記述したものである。

(2002)(MF)(COBOL)画面節では、画面の属性を定義する。それを通じて、画面上に表示するフィールドの位置を正確に指定したり、ACCEPT や DISPLAY の操作の間にある種のコンソール機能を制御したりすることができる。

(MF)同じ方法で記述したデータレコードとデータ項目の型は同じである。型定義を宣言することによって、そのような項目の記述を便利に操作できる。型定義は作業場所節と連絡節に置くことができる。呼出しプロトタイプ中の型定義をプログラム定義内で参照できる。

3.8.2 データの種類と状態

内部データ項目およびファイル結合子には3つの種類がある。具体的には、自動的、初期的、静的の別がある。項目を自動的、初期的、静的のいずれに指定するかによって、実行単位を実行中のそれらの項目およびその内容の持続性が変わってくる。

データ項目およびファイル結合子には初期状態および最終使用状態がある。データ項目の初期状態は、そのデータ項目が記述されているデータ記述項中の VALUE 句の有無およびそのデータ項目の記述に左右される。ファイル結合子の初期状態はオープンモードにないということである。

最終使用状態は、データ項目またはファイル結合子の内容はそれが最後に変更された時点の内容であることを意味する。

自動的項目は、メソッドまたはプログラムが起動されたときに初期状態に設定される。メソッドまたはプログラムの各インスタンスにその項目のコピーが保持される。自動的項目は局所記憶性に記述されている項目である。

初期項目は初期プログラムが起動されたときに初期状態に設定される。初期プログラム中のデータ項目とファイル結合子はすべて、初期項目である。また、初期プログラム中の画面項目の属性も初期項目として扱われる。

静的項目は、メソッド、オブジェクト、プログラムのいずれかが初期状態に設定されたときに、初期状態に設定される（*メソッド、オブジェクトまたはプログラムの状態*節を参照）。静的項目は、初期プログラムではないソース要素の通信、ファイル、作業場所のどれかの節に記述されている項目である。また、初期プログラムではないソース要素中の画面項目の属性も静的項目として扱われる。

3.8.3 メソッド、オブジェクトまたはプログラムの状態

3.8.3.1 メソッド、オブジェクトまたはプログラムの状態

実行単位中の任意の時点において、メソッドまたはプログラムはアクティブまたは非アクティブのどちらかの状態にある。メソッドまたはプログラムが起動されたときに、その状態は初期状態または最終使用状態にある。

3.8.3.1.1 アクティブ状態

関数、メソッドまたはプログラムは再帰的に起動することができる。したがって、ある関数、メソッドまたはプログラムの複数のインスタンスが同時にアクティブ状態にある場合もある。

メソッドのインスタンスは、正常に起動されたときに、アクティブ状態に置かれる。そして、そのメソッドのインスタンス内で STOP 文が実行されるか、EXIT METHOD 文が明示的または暗黙的に実行されるまで、アクティブ状態を保つ。

プログラムのインスタンスは、オペレーティングシステムによって正常に起動されるかランタイム要素から正常に呼び出されたときに、アクティブ状態に置かれる。そして、下記のどれかひとつが実行されるまで、アクティブ状態を保つ。

- STOP 文
- 呼び出されたプログラム内での明示的または暗黙的な EXIT PROGRAM 文
- 呼び出されたプログラム内での、または呼出し元のランタイム要素の制御下にないプログラム内での、GOBACK 文

メソッドまたはプログラムのインスタンスが起動されたときはいつでも、そのメソッドまたはプログラムのインスタンスに含まれる PERFORM 文の制御機構が初期状態に設定され、ALTER 文によって参照される GO TO 文が初期状態に設定される。

3.8.3.1.2 データの初期状態と最終使用状態

メソッドまたはプログラムが起動されたときに、その中のデータは初期状態または最終使用状態のどちらかにある。

3.8.3.1.2.1 初期状態

自動的データおよび初期データは、それが記述されているメソッドまたはプログラムが起動されるたびに、初期状態に置かれる。

静的データは下記の場合に初期状態に置かれる。

1. その静的データが記述されているメソッドまたはプログラムが実行単位内で最初に起動されたとき。
2. 初期属性を持ち、その静的データが記述されているプログラムを直接的または間接的に含むプログラムを参照する起動文が実行された後で、その静的データが記述されているプログラムが最初に実行されたとき。
3. その静的データが記述されているプログラムまたはそのプログラムを直接的または間接的に含むプログラムを CANCEL する起動文が実行された後で、その静的データが記述されているプログラムが最初に実行されたとき。

メソッドまたはプログラム中のデータが初期状態に置かれると、下記の事柄が起こる。

1. 作業場所節、局所記憶節、通信節に記述されている初期データが、***データ部 - データ記述***の章の *VALUE(値)句*節の記述に従って、初期化される。
2. そのメソッドまたはプログラムの内部ファイル結合子が、いかなるオープンモードにもないように設定されて、初期化される。
3. 画面項目の属性が、画面記述項に指定されているように設定される。

3.8.3.1.2.2 最終使用状態

最終使用状態になりうるのは静的データと外部データだけである。外部データは、実行単位が起動されているとき以外は、常に最終使用状態にある。静的データは、上に定義された初期状態にあるとき以外は、常に最終使用状態にある。

3.8.3.2 オブジェクト初期状態

オブジェクトの初期状態はそのオブジェクトが生成された直後の状態である。内部データ、内部ファイル結合子、画面項目の属性は、メソッドまたはプログラムが初期状態に置かれた場合と同様に、上記の*初期状態*節の記述に従って初期化される。.

3.8.4 大域名と局所名

ANS85

データ名はデータ項目の名前である。ファイル名はファイル結合子の名前である。これらの名前は大域名と局所名のどちらかに分類される。

大域名は、それが宣言されているソース要素内からでも、そのソース要素に含まれる他の任意のソース要素からでも、関連のある項目を参照するために使用することができる。

それと対照的に、局所名は、それが宣言されているソース要素内から、関連のある項目を参照するためにしか使用することができない。名前の中には常に大域的なものがある。常に局

所的な名前もある。その他に、名前が宣言されているソース要素内の仕様に応じて、大域的または局所的となる名前もある。

レコード名が宣言されているレコード記述項内に GLOBAL 句が指定されている場合、そのレコード名は大域的である。あるいは、ファイル節内のレコード記述の場合、そのレコード記述項に関連するファイル名のファイル記述項に GLOBAL 句が指定されている場合、そのレコード名は大域的である。

データ名が宣言されているデータ記述項またはその上位のデータ記述項の中に GLOBAL 句が指定されている場合、レコード名は大域的である。

データ記述項内に宣言されている条件名は、そのデータ記述項の上位のデータ記述項の中に GLOBAL 句が指定されている場合、大域的である。しかし、特定の規則によって、時にはある種のデータ記述項、ファイル記述項、またはレコード記述項に GLOBAL 句を指定することが禁止されることがある。

該当ファイルのファイル記述項中に GLOBAL 句が指定されている場合、ファイル名は大域的である。

データ記述項中に宣言されたデータ名、ファイル名、条件名が大域的ではない場合、その名前は局所的である。

大域名は他のソース要素内に含まれるソース要素間にわたって推移的である。

3.8.5 外部項目と内部項目

呼出し可能なデータ項目には通常は何らかの表現のデータが格納されている必要がある。ファイル結合子にはファイルに関する何らかの情報が格納されている必要がある。データ項目またはファイル結合子に関する記憶場所は、外部または内部でありえる。

外部および内部の項目は大域名または局所名のどちらかを持てる。

作業場所節中に記述されたデータレコードは、そのデータ記述項内に EXTERNAL 句があると、外部属性を持つ。外部レコードを記述しているデータ記述項の下位のデータ記述項によって記述されているデータ項目も外部属性を取得する。レコードまたはデータ項目が外部属性を持たない場合、それは内部データである。

関連するファイル記述項に EXTERNAL 句があると、ファイル結合子は外部属性を得る。外部属性を持たないファイル結合子は内部的である。

EXTERANL 句が含まれないファイル記述項または整列併合ファイル記述項の下位に記述されたデータレコードならびにそれらのレコードのデータ記述項の下位に記述されたデータ項目は、常に内部的である。ファイル記述項に EXTERNAL 句が含まれていると、データレコードおよびデータ項目は外部属性を得る。

局所記憶節、連絡節、通信節、報告書節、画面節に記述されたデータレコード、下位のデータ項目、関連する各種の制御情報は常に内部的である。連絡節に記述されて、記述されたデータレコードと他のランタイム要素から呼出し可能な他のデータ項目との間に関連づけがなされているデータに対しては、特別な考慮が当てはまる。

データ項目またはファイル結合子が外部的である場合、その項目に関連する記憶場所は、実行単位内の個々のランタイムモジュールとではなく、実行単位と関連する。外部項目は、それを記述する実行単位内の任意のランタイムモジュールから参照できる。異なるランタイムモジュールからデータ項目またはファイル結合子の別立ての記述を使用して外部項目を参照した場合、必ず同じ項目が参照される。実行単位内では、外部項目の表現はひとつしか存在しない。

データ項目またはファイル結合が内部的である場合、それに関連する記憶場所はそれを記述するランタイムモジュールとのみ関連する。

3.9 手続き部

3.9.1 実行

実行は手続き部の最初の文から開始される。ただし、宣言部分は例外である。文は原始要素内に現れる順に実行される。ただし、規則により別の順序が適用される場合は別である。

3.9.2 文と完結文

文は、COBOL の動詞で始まる、語と記号の構文的に有効に組み合わせである。

完結文はいくつかの文で構成され、終止符とそれに続く空白で区切られる。

文には下記の4種類がある。

1. 条件文
2. COBOL システム指示文
3. 無条件文
4. (ANS85)範囲明示文

完結文には下記の3種類がある。

1. 条件完結文
2. COBOL システム指示完結文
3. 無条件完結文

3.9.2.1 条件文

条件文 (conditional statement) は、条件の真理値を判定して、その結果に基づいて以降の制御の流れを変えることを指定するものである。

条件文には以下のものがある。

- (ANS85)EVALUATE,

  IF, SEARCH, RETURN の各文

- AT END 句または INVALID KEY 句のある READ 文
- (OSVS)ON 文
- INVALID KEY 句または END-OF-PAGE 句のある WRITE 文
- INVALID KEY 句のある START, REWRITE, DELETE の各文
- SIZE ERROR 句のある算術文 (ADD, COMPUTE, DIVIDE, MULTIPLY, SUBTRACT)

- ON OVERFLOW 句のある STRING 文または UNSTRING 文
- ON OVERFLOW 句

  (ANS85)または ON EXCEPTION 句

  を含む CALL 文

- (MF)(XOPEN)ON EXCEPTION 句のある ACCEPT 文または DISPLAY 文

3.9.2.2 条件完結文

条件文を終止符 (.) に続く空白で止めたものを、条件完結文 (conditional sentence) という。条件文の前に無条件文があってもよい。

3.9.2.3 COBOL システム指示文

COBOL システム指示文 (COBOL system-directing statement) は、翻訳指示動詞とその作用対象 (operand) からなる。翻訳指示動詞については*翻訳指示文*の章で解説する。

COBOL システム指示文は、実行用プログラムコードを生成する際に COBOL コンパイラに指定された動作を取らせるものである。

3.9.2.4 COBOL システム指示完結文

COBOL システム指示完結文は単一の指示文を終止符 (.) に続く空白で止めたものである。

3.9.2.5 コンパイラ指令

コンパイラ指令はコンパイラの動作を制御するための任意選択機能である。コンパイラ指令を使用すると、種々の機能を指定することができる。具体的には、言語の種々の機能を使えるようにする、実行時の動作を選択する、コンパイル時のオプションを選択する、デバッグ情報を生成する、データファイルの形式を制御する、実行用プログラムコードを最適化する、SQL のオプションを選択する、などを行える。

(ISO2002)標準的に用意されているコンパイラ指令については、***翻訳指示文***の章の*コンパイラ指令*節で解説する。

(MF)この処理系に用意されている各コンパイラ指令の構文と説明については、使用している COBOL システムのマニュアルを参照。

3.9.2.6 無条件文

無条件文 (imperative statement) は、実行用プログラムに無条件に取らせる動作を指示するものである。条件文でも COBOL システム指示文でもない文は、無条件文である。無条件文はいくつかの無条件文をつなげたものでもよく、その間を分離符で区切ってもよい。

無条件動詞には以下のものがある。

ACCEPT[1]		RELEASE
ADD[2]	EXIT	REWRITE[3]
ALTER	GO TO	
CALL[4]	INSPECT	SET
CANCEL	MERGE	SORT
CLOSE	MOVE	START[3]
COMPUTE[2]	MULTIPLY[2]	STOP
DELETE[3]	OPEN	STRING[4]
	PERFORM	SUBTRACT[2]
DISPLAY[1]	READ[5]	UNSTRING[4]
DIVIDE[2]		WRITE[6]

1 指定は任意の ON EXCEPTION 句のないもの

2 指定は任意の SIZE ERROR 句のないもの

3 指定は任意の INVALID KEY 句のないもの

4 指定は任意の ON OVERFLOW または ON EXCEPTION 句のないもの

5 指定は任意の AT END 句または INVALID KEY 句のないもの

6 指定は任意の INVALID KEY 句または END-OF-PAGE 句のないもの

(ANS85)ANSI '85 の追加の無条件動詞には以下のものがある。

CONTINUE	INITIALIZE

(ISO2002)ISO2002 の追加の無条件動詞には以下のものがある。

(ISO2002)(MF)INVOKE

OS/VS OS/VS COBOL 無条件動詞には以下のものがある。

EXAMINE　　EXHIBIT　　GOBACK
TRANSFORM

VSC2 VS COBOL II の追加の無条件動詞には以下のものがある。

GOBACK

MF この COBOL システムで利用できる追加の無条件文には、以下のものがある。

CHAIN　　EXHIBIT　　GOBACK
NEXT SENTENCE

文の一般形式の中に「無条件文」が示されている場合、それは一連の無条件文を指す。その一連の無条件文は終止符によって止められるか、またはその"無条件文"が含まれる文に関連する何らかの句によって止められなければならない。

OS/VS 一連の無条件文中の任意の2つの無条件文の間に、連結語の"THEN"または"AND"を入れることもできる。

3.9.2.7 無条件完結文

無条件完結文 (imperative sentence) とは、無条件文を終止符 (.) に続く空白で止めたものである。

3.9.2.8 範囲明示文

ANS85

範囲明示文 (delimited scope statement) とは、対応する明示的な範囲符 (scope terminator) によってその有効範囲を限られる文をいう。たとえば、IF 文は範囲明示文であり、それに対応する範囲符は END-IF である。1 組の範囲明示文と範囲符との間にある文はすべて、その範囲明示文に含まれると みなされる。

範囲明示文は入れ子にすることができる。その場合、明示的な各範囲符は、その手前にある有限な範囲文のうちでまだ対になっていない最も近いものと対をなすものとみなされる。

範囲明示文を暗黙に終了させることもできる。それには次の2通りがある。手続き完結文の末尾では、分離符の終止符によって、まだ終了していない文がすべて終了される。範囲明示文を含む文が終了すると、含まれる範囲明示文も終了する。

注：すべての文がこの方法で範囲を区切れるとは限らないの。範囲を区切ることのできる文のうちで、範囲符で明示的に区切ったものだけを、範囲明示文と呼ぶ。

詳細については、***COBOL 言語の概念***の章で前述した*明示範囲符と暗黙範囲符*を参照。

3.9.2.9 文の分類

分類	動詞
算術	ADD COMPUTE DIVIDE INSPECT (TALLYING) MULTIPLY SUBTRACT OSVSEXAMINE (TALLYING)
条件	ADD (SIZE ERROR) CALL (OVERFLOW) COMPUTE (SIZE ERROR) DELETE (INVALID KEY) DIVIDE (SIZE ERROR) GO TO (DEPENDING) IF MULTIPLY (SIZE ERROR) READ (END or INVALID KEY) RETURN (END) REWRITE (INVALID KEY) SEARCH START (INVALID KEY) STRING (OVERFLOW) UNSTRING (OVERFLOW) WRITE (INVALID KEY または END-OF-PAGE) ANS85EVALUATE OSVSON
データ転記	ACCEPT (DATE, DAY or TIME) INSPECT (REPLACING) MOVE STRING UNSTRING OSVSEXAMINE

	OSVS TRANSFORM ANS85 INITIALIZE ANS85 INSPECT (CONVERTING) ANS85 SET (TO TRUE) MF SET (TO FALSE) VSC2 MF SET (ADDRESS OF) VSC2 MF SET (POINTER) MF SET (object reference)
終了	MF EXIT METHOD EXIT PROGRAM OSVS VSC2 MF GOBACK STOP
入出力	ACCEPT (一意名) CLOSE MF COMMIT DELETE DISPLAY OPEN READ RECEIVE REWRITE MF ROLLBACK START STOP (定数) MF UNLOCK WRITE OSVS MF EXHIBIT SET (TO ON または TO OFF)
プログラム間連絡	MF CALL CANCEL MF CHAIN OSVS VSC2 MF ENTRY MF EXEC SO000 MF INVOKE OSVS VSC2 SERVICE
空操作	EXIT CONTINUE
順序づけ	MERGE

	RELEASE RETURN SORT
手続き分岐	ALTER CALL MF EXIT PERFORM/EXIT PARAGRAPH/EXIT SECTION GO TO MF NEXT SENTENCE PERFORM
範囲明示	MF XOPEN END-ACCEPT END-ADD END-CALL END-DELETE MF XOPEN END-DISPLAY END-DIVIDE END-EVALUATE END-IF MF END-INVOKE END-MULTIPLY END-PERFORM END-READ END-RETURN END-REWRITE END-SEARCH END-START END-STRING END-SUBTRACT END-UNSTRING END-WRITE
COBOL システム指示	OSVS VSC2 BASIS OSVS VSC2 DELETE OSVS VSC2 INSERT MF $DISPLAY MF $ELSE MF $END MF $IF COPY ENTER USE ANSI85 REPLACE OSVS VSC2 MF ENTRY

	OSVS VSC2 MF EJECT OSVS VSC2 MF SKIP1 OSVS VSC2 MF SKIP2 OSVS VSC2 MF SKIP3 VSC2 MF TITLE OSVS NOTE OSVS VSC2 ++INCLUDE OSVS VSC2 -INC
表操作	SEARCH SET

IF と ON は、英語では動詞ではないが、COBOL では動詞として扱う。

3.10 正書法

この節では固定方式のみを取り上げる。この COBOL システムは自由方式で書かれた翻訳集団をも受け付ける。自由方式の原始文の詳細については、*COBOL 言語の紹介* の章を参照。

正書法 (reference format) は、COBOL の原始文を記述するための標準的な方法である。この正書法は入出力媒体上の行を構成する文字位置を基準として定める。COBOL システムは正書法に従って書かれた原始文を受け入れ、正書法に従って出力リストを作り出す。 (原始プログラムの例は、***COBOL 言語の概要***の章の *サンプルプログラム*に掲載。)

正書法に関する空白あけの規則は、他のすべての空白あけの規則に優先する。

3.10.1 正書法の表現

行の正書法は、図 3-1 のとおり。

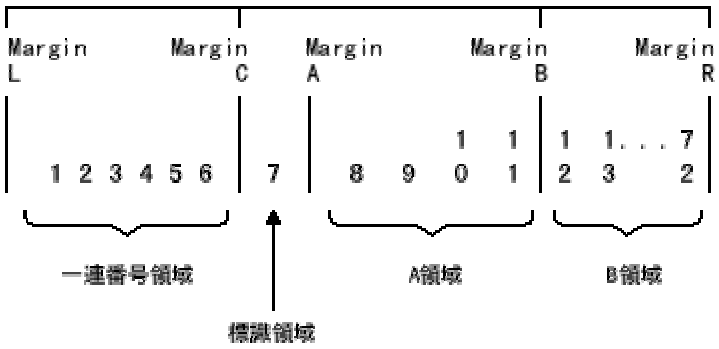


図 3-1：COBOL の原始行の正書法

-
- 境界 L は、1 行の最左端の文字位置のすぐ左とする。
- 境界 C は、1 行の 6 番目と 7 番目の文字位置の間とする。
- 境界 A は、1 行の 7 番目と 8 番目の文字位置の間とする。
- 境界 B は、1 行の 11 番目と 12 番目の文字位置の間とする。
- (MF)境界 R は、1 行の最右端 (72 番目) の文字位置のすぐ右とする。1 行を 80 文字まで使用できる。しかし、73 番目から 80 番目までの文字位置にある文字は COBOL システムによって無視される。

一連番号領域は、境界 L と境界 C との間の 6 文字分である。

標識領域は、1 行の 7 番目の文字位置である。

A 領域は、境界 A と境界 B との間の文字位置 8, 9, 10, 11 である。

B 領域は、境界 B と境界 R との間の、12 番目から 72 番目までの文字位置を占める。

(ISO2002)プログラム原文領域は A 領域と B 領域の両方から構成される。プログラム原文領域のどこから原文語を書き始めてもかまわない。

3.10.1.1 一連番号

一連番号 (sequence number) は、一連番号領域に書く 6 桁の数字であり、原始行を識別するために使用する。一連番号は通常、連続する各原始行に対して、昇順に番号付ける。

(OSVS)(OSC2)一連番号は、行を編集するために BASIS 機構によって使用される。この場合、一連番号は数字で、プログラム全体を通じて昇順になっていなければならない (***翻訳指示文***の章の *BASIS 機構*節を参照) 。

一連番号が昇順になっているかどうか、COBOL システムで検証することができる。検証を行うには、SEQCHK コンパイラ指令を使用する。

(ANS85)一連番号領域の内容は、数字である必要はない。また、一意である必要もない。

(MF)一連番号領域の最初の文字位置に、星印 (*) または印字されない制御文字 (ASC II 文字の照合順序で空白文字より小さいもの) を指定すると、その行は注記として扱われ、印刷用のファイルまたは装置に出力されない。 この機能を利用して、出力印刷用ファイルを次のコンパイルの原始ファイルとして使用できる。この機能は、MFCOMMENT コンパイラ指令の影響を受ける。

3.10.1.2 行のつなぎ

文、記述項、指定、句が 2 行以上にわたるとき、その続きの部分は B 領域に書く。この続きの行を後の行 (continuation line) と呼ぶ。続けられる行を前の行 (continued line) と呼ぶ。この書き方をするときは、任意の語や定数、

(ANS85)または PICTURE 文字列

を途中で分割して後ろの行に書いてもよい。

ある行の標識領域にハイフン (-) を書くと、現在の行の B 領域の空白でない最初の文字が、前の行の空白でない最後の文字の後ろに続くことを示す。この場合、間の空白は存在しないものとみなされる。

(ANS85)注記行または空白行は前の行とは扱われない。

ただし、前の行にまだ引用符で閉じていない文字定数がある場合には、後の行の B 領域の空白でない最初の文字は引用符とする。文字定数の続きは、この引用符のすぐ次の文字位置から書く。この書き方をすると、前の行の終わりまでの空白は、すべてその定数の一部であるとみなされる。後ろの行の A 領域は空白とする。

標識領域にハイフンを書かないと、前の行の最後の文字の後ろに空白があるものとみなされる。分離符 "=="は、2 文字とも同じ行に書く。

(VSC2)(MF)X"および G"は、2 文字とも同じ行に書く。

(MF)H"は、2 文字とも同じ行に書く。

(MF)(COBOL370)N"は、2 文字とも同じ行に書く。

(MF)(XOPEN)*>"は、2 文字とも同じ行に書く。

(ISO1002)>>は、2 文字とも同じ行に書く。

3.10.1.3 空白行

境界 C から境界 R までがすべて空白の行を、空白行 (blank line) という。空白行は、 原始文内のどこにあってもよい。

3.10.1.4 仮原文

仮原文 (pseudo-text) を構成する原文語 (text-word) と区切り文字の空白は、A 領域または B 領域のどちらから書き始めてもよい。ただし、仮原文を開始する区切り文字に続く行の標識領域にハイフンを書いた場合、その行の A 領域は空白とする。その場合、行のつなぎに関する規則が、原文語を書く際にも適用される (*翻訳指示文*のを章参照) 。

3.10.2 部、節、段落の正書法

3.10.2.1 部の見出し

部の見出しは、A 領域から書き始める (図 3-1 を参照) 。

(SO2002)(MF)B 領域から書き始めてもよい。

3.10.2.2 節の見出し

節の見出しは、A 領域から書き始める (図 3-1 を参照) 。

(SO2002)(MF)B 領域から書き始めてもよい。

環境部と手続き部では、節はいくつかの段落から構成される。段落がない場合もありえる。データ部では、節はいくつかの記述項から構成される。記述項がない場合もありえる。

3.10.2.3 段落の見出し、段落名、段落

段落には 2 通りある。1 つは、先頭に段落名を書いて終止符 (.) に続く空白で止め、その後ろに完結文をいくつか書いたものである。もう 1 つは、段落の見出しの後ろに記述項をいくつか書いたものである。段落内に注記項を含めることができる。段落の見出しおよび段落名は、部または節の最初の行よりも後ろの任意の行に、A 領域から書き始める。

(SO2002)(MF)また、B 領域から書き始めることもできる。

段落の最初の完結文や記述項は、段落名または段落の見出しを書いた行か、またはその後ろの注記行や空白行でない行の B 領域から書き始める。以降の完結文や記述項は、前の完結文や記述項と同じ行の B 領域か、またはその後ろの空白行でも注記行でもない行の B 領域から書き始める。

(SO2002)(MF)完結文は、A 領域または B 領域のどこから書き始めてもよい。ただし、AREACHECK 指令を指定したときは別である。

段落中の完結文や記述項が 2 行以上にわたる場合は、*行のつなぎ*の規則に従って書くことができる。

3.10.3 データ部の記述項

データ部の各記述項は、レベル指示語またはレベル番号で始め、その後ろに空白を置いて、記述項の名前を続け

(ANS85) (存在する場合) 、

さらにその記述項について記述する一連の独立な句を続ける。最後の句は常に終止符 (.) に続く空白で止める。

データ部の記述項には 2 種類ある。具体的には、レベル指示語で始まるものと、レベル番号で始まるものである。

レベル指示語 (level indicator) には下記のものがある。

- FD (***データ部 - データ記述***の章の*ファイル記述項の全体的な骨組み*節を参照)
- SD (***データ部 - データ記述***の章の*ファイル記述項の全体的な骨組み*節を参照)
- RD (『言語リファレンス - 追加トピック』の***報告書作成***の章の*報告書記述項*節を参照)

それらのデータ部の記述項がレベル指示語で始まる記述項の場合、レベル指示語は A 領域から書き始め、その後ろに空白を置き、それに続けて、B 領域

(ANS85)または A 領域に

関連する名前と適切な記述情報を書く。

レベル番号で始まる記述項を、データ記述項という。

レベル番号の値は 1 から 49 まで、66, 77,

(MF)78

および 88 のいずれかとする。値が 1 から 9 のレベル番号は 1 文字で書いてもよいし、前にゼロを付けて書いてもよい。レベル番号と後ろに続く語との間は、最低 1 文字の空白で区切る。

レベル番号が 01 または 77 で始まるデータ記述項の場合、レベル番号を A 領域から書き始め、その後ろに空白を置き、それに続けて B 領域

(ANS85)または A 領域

に関連するレコード名または項目名と適切な記述情報を書く。

引き続くデータ記述項は、最初のものと同じ形式にしてもよいし、レベル番号に従って階段状に書いてもよい。階段状に書いても、レベル番号の大きさは左右されない。

レベル番号を階段状に書くとき、新しいレベル番号は境界 A から任意の個数の空白を空けた位置から書き始めてよい。階段の段差の大きさは、物理的な幅によってのみ制限される。

ISO2002 MF レベル番号が 01 と 77 以外のデータ記述項を、A 領域から書き始めてもよい。

3.10.4 宣言部分

手続き部の宣言部分の始めと終わりを示す必要語の DECLARATIVES と END DECLARATIVES は、それぞれ 1 行に単独で書く。これらの語は A 領域から書き始め、終止符 (.) に続く空白で止める (図 3-1 を参照) 。

3.10.5 注記行

注記行とは、標識領域に星印 (*) を付けた行である。見出し部の見出しの後ろならば、注記行はソース要素中のどこに置いてもよい。注記行の A 領域と B 領域には、計算機文字集合の文字を任意に組み合わせて書いてよい (図 3-1 を参照) 。星印および A 領域と B 領域内の文字は出力リストには出されるが、翻訳はされない。

OSVS OSC2 MF 注記行を、見出し部の見出しの前に置くことができる。

星印の代りに斜線 (/) を使用することもできる。この場合、注記行は改ページが行われてから出力リストに出される。それ以外は、星印の場合と同じである。

注記行を 2 行以上にわたって書いてもよい。ただし、後ろの行の標識領域に星印を書かなければならない。

3.10.5.1 行内注記

ISO2002 MF

分離符の空白に続けて"*>"と書くと、それ以降、行末までが行内注記 (in-line comment) となる。この書き方をすると、文字列や分離符と同じ行に自由方式の注を書ける。COBOL 翻訳集団または COBOL 登録集の原文の中で分離符の空白を置けるところには、どこでも行内注記を書くことができる。登録集原文、仮原文、原文を評価する際には、行内注記は 1 つの空白文字とみなされる。行内注記を次の行に続けることはできない。

第 4 章：翻訳集団の定義の概要

4.1 はじめに

この章では COBOL 言語の基本的な定義について説明する。

まず最初に翻訳集団全体の構造を説明する。翻訳集団の各構成要素は4つの部に分かれている。具体的には、見出し部、環境部、データ部、手続き部がある。

したがって、この資料もそれらの部別に構成されている。各部の中では、COBOL の節ごとに見出しを付けて説明する。項は、参照しやすいように、アルファベット順に配列してある。組み込み関数および COBOL の動詞は項のレベルに含める。

追加の言語機能として、DBCS、報告書作成もサポートされている。それらについては、マニュアル *『言語リファレンス』の『追加トピック』*を参照。

4.2 翻訳集団の構造

翻訳集団は、コンパイラに提出されるソース単位すべてである。

ANSI85 翻訳集団には、0 個以上のソース単位が含まれる。翻訳集団は一連のソース単位である。ソース単位の中に他のソース単位を含めることができる。他のソース単位に含まれるソース単位から、親のソース単位の資源の一部を参照できる。

ソース単位は見出し部で始まる。

ANSI85 ソース単位には、別のソース単位が含まれる。

ソース原文操作文、

ANSI85 および終了見出し、

SOM92 MF および翻訳指示を除き、

ソース単位の文、記述項、段落、および節は、次の順序で 4 つの部分にグループ化され、配列される。

1. 見出し部
2. 環境部
3. データ部
4. 手続き部

ソース単位中の部の開始は適切な部の見出しによって示される。

(ISO2000)(MF)見出し部の開始は、見出し部内で許されている段落の見出しのどれかによっても示すことができる。

ソース単位中の部の終了は下記のどれかひとつによって示される。

1. そのソース単位中の次の部の開始
2. (ANSI86)そのソース単位の終了見出し
3. それ以上原始行を記述できない物理位置の後

ソース単位の終了は、翻訳集団内に後続の原始行が存在しないことによって示される。

(ANSI86)ソース単位の終了は、終了見出しによってのみ示される。

(ANSI86)ソース単位 B が他のソース単位 A に含まれるとき、包含関係は直接的でも間接的でもかまわない。A に含まれ B を含む他のソース単位が存在しない場合、ソース単位 B はソース単位 A に直接的に含まれる。A に含まれ B を含む他のソース単位が存在する場合、ソース単位 B はソース単位 A に間接的に含まれる。

他のソース単位に直接的にまたは間接的に含まれるソース単位は、この仕様においては、別のソース単位とみなされる。ただし、入れ子になったソース単位は親のソース単位中に定義されている資源の一部を追加的に参照することができる。

他のソース単位に含まれるソース単位をコンパイルした結果の実行用プログラム･コードは、この仕様においては、親のソース単位をコンパイルした結果の実行用プログラム･コードと不可分であるとみなされる。

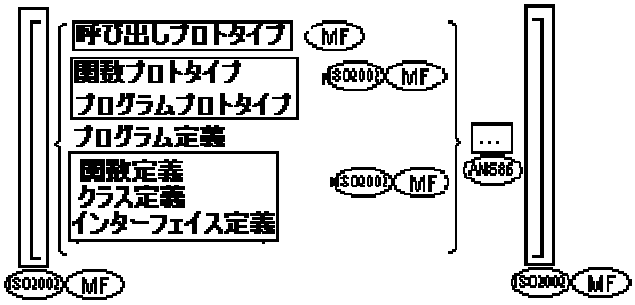
(ANSI86)翻訳単位は他のソース単位に含まれていないソース単位である。翻訳集団内で翻訳単位の前または後ろに他の翻訳単位が来ることもある。

ソース要素はネストされているソース単位を除いたソース単位である。

ランタイム要素はソース要素をコンパイルした結果である。

ランタイム･モジュールは翻訳単位をコンパイルした結果である。

一般形式



ここで 呼び出しプロトタイプ は: (MF)

```
[ { ID            } DIVISION . ]
  { IDENTIFICATION }
PROGRAM-ID .  プログラム名-1  IS EXTERNAL PROGRAM .
[  データ部   ]
[  手続き部   ]
 END PROGRAM  プログラム名-1  .
```

(SO2002) (MF)

ここで 関数プロトタイプ は:

```
[ { ID (MF)       } DIVISION . ]
  { IDENTIFICATION }
FUNCTION-ID. {  関数プロトタイプ名-1  [ AS  定数-1  ]} IS PROTOTYPE.
[  環境部  ]
[  データ部  ]
[ 手続き部  ]
 END  FUNCTION  関数プロトタイプ名-1  .
```

(SO2002) (MF)

ここで プログラムプロトタイプ は:

```
[ { ID (MF)       } DIVISION . ]
  { IDENTIFICATION }
PROGRAM-ID. { プログラムプロトタイプ名-1  [ AS  定数-1  ]} IS PROTOTYPE.
[  環境部  ]
[  データ部  ]
[ 手続き部  ]
 END  PROGRAM プログラムプロトタイプ名-1 .
```

ここで プログラム定義 は

[{ ID (OSVS)(VSC2)(MF) / IDENTIFICATION } DIVISION .] (SO2002)(MF)

[program-id 段落] (MF)

[環境部] (ANS85)

[データ部] (ANS85)

[手続き部 [プログラム定義] ...] (ANS85)

[END PROGRAM プログラム名-1] (ANS85)

(SO2002)(MF)

ここで 関数定義 は:

[{ ID (MF) / IDENTIFICATION } DIVISION .]

FUNCTION-ID. 利用者関数名-1 [AS 定数-1].
[環境部]
[データ部]
[手続き部]
END FUNCTION 利用者関数名-1 .

ここで クラス定義 は:

```
[ { ID (MF)          } DIVISION . ]
  { IDENTIFICATION   }
CLASS-ID. クラス名-1      [ AS 定数-1 ] [ IS FINAL ]
  [ INHERITS FROM クラス名-2 ]
  [ USING { パラメタ名-1 } ... ] .
[ 環境部 ]

[ [ { ID (MF)          } DIVISION . ]
  [ { IDENTIFICATION   }
  FACTORY . [ IMPLEMENTS インターフェイス名-1 } ... . ]
  [ 環境部 ]
  [ データ部 ]
  [ 手続き部 ]
  END FACTORY . ]

[ [ { ID (MF)          } DIVISION . ]
  [ { IDENTIFICATION   }
  OBJECT . [ IMPLEMENTS インターフェイス名-2 } ... . ]
  [ 環境部 ]
  [ データ部 ]
  [ 手続き部 ]
  END OBJECT . ]

END CLASS クラス名-1 .
```

ここでインターフェイス定義は:

```
[ { ID (MF)          } DIVISION . ]
  { IDENTIFICATION   }
INTERFACE-ID .インターフェイス名-1 [ AS 定数-1 ]
  [ INHERITS FROMインターフェイス名-2 ]
  [ USING { パラメタ名-1 } ... ] .
[ 環境部 ]
[ 手続き部 ]
END INTERFACE インターフェイス名-1.
```

```
(ISO/IEC) (MF)
ここで メソッド定義 は:
[ { ID (MF)          } DIVISION . ]
  { IDENTIFICATION   }

                  { メソッド名-1        [ AS 定数-1 ]                  }
METHOD - ID . { { GET } PROPERTY   属性名-1                          } [ OVERRIDE ]
              { { SET }                                               }
[ 環境部 ]
[ データ部 ]
[ 手続き部 ]
END METHOD [ メソッド名-1 ] .
```

構文規則

1. 翻訳集団内では、

   (MF)呼び出しプロトタイプ、および

   (ISO/IEC)(MF)関数プロトタイプおよびプログラムプロトタイプは、

   他のすべての型のソース単位の前に置かれなければならない。

2. (ISO/IEC)(MF)翻訳集団に、同様に外部化された名前を持つ関数定義と関数プロトタイプ定義の両方が含まれる場合は、この両方の翻訳単位の署名は同一とする。
3. (ISO/IEC)(MF)インターフェイス定義中のメソッドのデータ部には連絡節だけを含めることができる。
4. (ISO/IEC)(MF)インターフェイス定義中のメソッドのデータ部には手続き部だけを含めなければならない。
5. (MF)呼出しプロトタイプにおいては、手続き部の見出しは手続き部の見出しの形式 2 の規則に準拠していなければならず、ENTRY 文を書くときは ENRTY 文の形式 2 の規則に従わなければならない。

一般規則

1. (MF)呼出しプロトタイプにおいては、データ部内の節は連絡節以外はすべて無視され、手続き部は手続き部の見出しと ENTRY 文以外は無視される。
2. (ISO/IEC)(MF)関数プロトタイプおよびプログラムプロトタイプでは、データ部の節は、連絡節以外は無視される。また、手続き部は、PROCEDURE DIVISION ヘッダを除き、無視される。

4.3 終了見出し

(ANS85)

終了見出しは定義の終わりを示す。

一般形式

```
END { PROGRAM プログラム名-1 .
      FUNCTION 関数プロトタイプ名-1
      FUNCTION 利用者関数名-1
      PROGRAMプログラムプロトタイプ名-1
      CLASS クラス名-1
      FACTORY
      OBJECT
      METHOD [ メソッド名-1 ]
      INTERFACE インターフェイス名-1 } .
```

ISO2002 MF

構文規則

1. 他のソース単位を含むソース単位、他のソース単位に含まれるソース単位、他のソース単位に先行するソース単位には、終了見出しを付けなければならない。
2. プログラム名-1 は先行するプログラム名段落中に宣言されているプログラム名と同じでなければならない。
3. プログラム名-1 のプログラム名段落とプログラム終了見出しとの間に特定のプログラム名を宣言したプログラム名段落を記述する場合、プログラム名を参照するプログラム終了見出しはプログラム名-1 を参照するプログラム終了見出しより前に来なければならない。.
4. ISO2002 MF 関数プロトタイプ名-1 は、対応する関数名段落で宣言されている関数プロトタイプ名と同じでなければならない。
5. ISO2002 MF 利用者関数名-1 は、対応する関数名段落で宣言されている利用者関数名と同じでなければならない。
6. ISO2002 MF プログラムプロトタイプ名-1 は、対応するプログラム名段落中で宣言されているプログラムプロトタイプ名と同じでなければならない。
7. ISO2002 MF クラス名-1 は対応するクラス名段落中に宣言されているクラス名と同じでなければならない。
8. ISO2002 MF メソッド名-1 は対応するメソッド名段落中に宣言されているメソッド名と同じでなければならない。

   ISO2002 MF メソッド名段落中に PROPERTY 指定を指定した場合は、メソッド名-1 を省く。

9. ISO2002 MF インターフェイス名-1 は対応するインターフェイス名段落中に宣言されているインターフェイス名と同じでなければならない。
10. 終了見出しによって終わりを区切られているソース単位が他のソース単位に含まれている場合、次の文は別のソース単位の最初の文か親のソース単位の終わりを区切る他の終了見出しのどちらかでなければならない。
11. 終了見出しによって終わりを区切られているソース単位が他のソース単位に含まれていない場合、次の文は別の翻訳単位の最初の文でなければならない。

一般規則

1. 終了見出しは指定したソース単位の終了を示す。

第 5 章 : 見出し部

5.1 見出し部

見出し部では、プログラム、

ISO2002 MF クラス、ファクトリオブジェクト、オブジェクト、メソッド、

ISO2002 関数、またはインターフェイス

を識別する。

MF この部全体を (見出しも含めて) プログラム定義中に指定しても指定しなくてもよい。

MF ソース要素を含むファイルのファイル名から得られる基本名からも、そのファイル中に定義されている最初のソース要素が識別される。

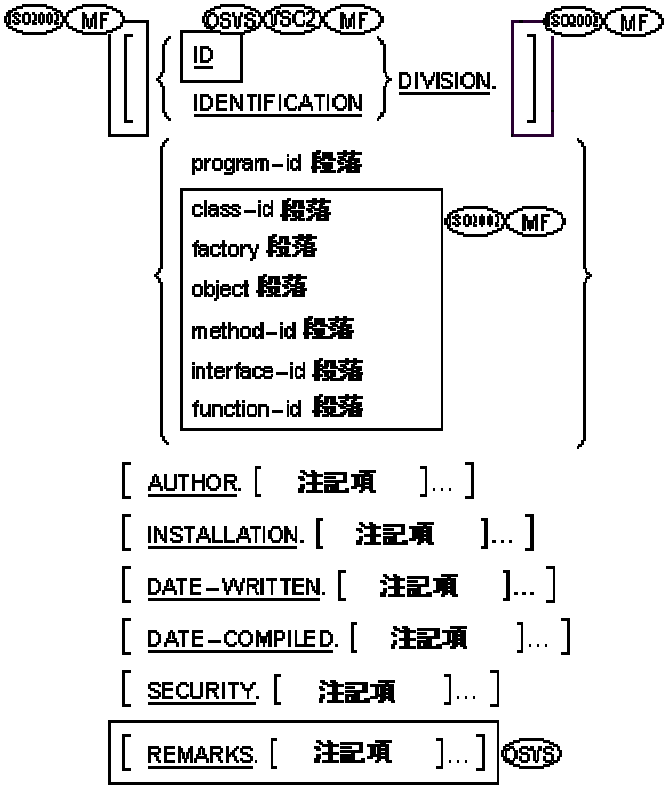
見出し部は、多くの段落から構成される。段落の見出しはその段落に含まれる情報の種類を示す。

ANS85 AUTHOR、INSTALLATION、DATE-WRITTEN、DATE-COMPILED、および SECURITY の各段落は、ANSI'85 標準では廃要素に分類されている。ANSI'85 標準の次の全面改訂の際に削除される予定である。

MF この構文は、ISO2002 を除く Micro Focus COBOL に組み込まれているすべての方言で使用できる。FLAGSTD コンパイラ指令を使用すると、この構文が用いられているすべての箇所を見つけ出すことができる。

XOPEN 標準 COBOL 定義の一部を構成するにもかかわらず、X/Open COBOL 言語定義では、廃要素が明示的に除外されている。したがって、X/Open に準拠する原始プログラムでは、それらを使用するべきではない。

一般形式



構文規則

ANS85

1. OSVS VSC2 MF 段落はどのような順番で書いてもよい。
2. OSVS VSC2 見出し部内の段落名の後ろの終止符は、書いても書かなくてもよい。同様に、プログラム名の後ろの終止符も、書いても書かなくてもよい。
3. 注記項には、計算機の文字集合に属する任意の文字を、組み合わせて書ける。標識領域にハイフンを指定して、注記項を次の行に続けることはできない。しかし、コンパイラ指令の SOURCEFORMAT に FIXED を指定するか省略時の解釈としておくと、注記項を複数行にわたって書くことはできる。この場合、注記項を書ける範囲は B 領域に限定される。次の行を A 領域から書き始めると、注記項ではないものを書き始めたことになる。

   MF コンパイラ指令の SOURCEFORMAT に FREE と指定すると、注記項を継続することはできない。次の行には注記項以外を記述しなければならない。

   VSC2 注記項の中の任意の位置に、SKIP1, SKIP2, SKIP3, EJECT, TITLE の各文を含めることができる。これらの文は、1 行に単独で書いた場合にその機能を発揮する。この場合は、これらの文は、注意項を終わらせることはない。

(OSVS)注記項の中の B 領域の任意の位置に、SKIP1, SKIP2, SKIP3, EJECT の各文を含めることができる。これらの文は、1 行に単独で書いた場合にその機能を発揮する。この場合は、これらの文は注記項を終わらせることはない。

(OSVS)注記項は、注記行の A 領域または B 領域の任意の位置に書ける。ただし、次の行の A 領域に下記の COBOL の語のどれかを書くと、注記項は終わりとされ、次の段落または部が始まるものとみなされる。

PROGRAM-ID
AUTHOR
INSTALLATION
DATE-WRITTEN
DATE-COMPILED
SECURITY
ENVIRONMENT
DATA
PROCEDURE

5.2 プログラム名段落

プログラム名 (PROGRAM-ID) 段落は、プログラムを識別する名前を付けるとともに、指定されたプログラム属性をプログラムに割り当てる。

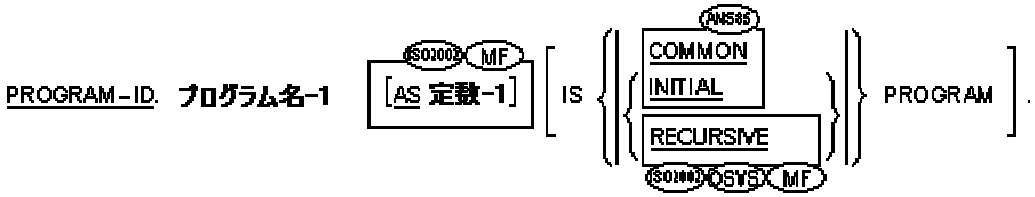
(ISO2002)(MF)プログラム名段落は、プログラムプロトタイプを識別する名前を指定する。

(MF)プログラム名段落は、呼出しプロトタイプを識別する名前を指定する。

一般形式

形式 1

形式 2



形式 3

```
(MF)
PROGRAM-ID. プログラム名-1 [IS EXTERNAL PROGRAM].
```

形式 4

```
(ISO2002)(MF)
PROGRAM-ID. プログラムプロトタイプ名-1 [AS 定数-1] IS PROTOTYPE .
```

指令

1. 予約語リストにフラグを付けたり、その内容を修正する機能を提供するコンパイラ指令に加えて、下記の指令がこの節に記述されている構文または意味内容に影響を与える可能性がある。
    - CASE - 大文字と小文字を同じに扱うか否かを指定する。
    - DEFAULT-BYTE - 作業場所節に定義されたデータ項目に、VALUE 句を指定せずに、値を設定する。
    - MAPNAME - プログラム名の中のアルファベットでない文字の取扱いに影響する。
    - PROGID-COMMENT - 形式 2 ではなく、形式 1 を想定する。

構文規則

すべての形式

1. (OSVS)(VSC2)(MF)プログラム名-1 は文字定数であってもよい。それを引用符で囲んでも囲まなくてもよい。定数の内容はプログラム名の構成規則に従っていなければならない。ただし、文字@,#, $を含んでいてもよい。
2. プログラム名-1 は他の利用者語と同じであってはならない。

   (OSVS)(MF)プログラム名-1 はユーザーが定義した他の語と同じであってもよい。

形式 1 および 2

3. (OSVS)(VSC2)他のプログラムに含まれないプログラムのプログラム名-1 の最初の 8 文字はシステム内で一意でなければならない。先頭の文字はアルファベットでなければならない。そうでない場合は、下記の規則に従って変換される。:
    - 先頭から 9 文字目までが順に A から I に変えられる。
    - それ以外の文字はすべて J に変えられる。

   (OSVS)(VSC2)他のプログラムに含まれないプログラムのプログラム名-1 の 2 文字目から 8 文字目までの間にハイフンがあると、それはゼロに変えられる。

VSVS VSC2 この動作は MAPNAME コンパイラ指令を用いて制御する。

VSC2 他のプログラムに含まれるプログラムに関しては、プログラム名-1 は長さが 30 文字まで任意のユーザー定義の COBOL の語とすることができる。最初の8文字が一意である必要はない。上記のプログラム名の変換は行われない。小文字を使用してもかまわない。ただし、小文字を使用した場合、大文字と小文字は区別される。

4. ANS85 他のプログラムに含まれるプログラムに、その親の翻訳単位内に含まれている他のプログラムと同じ名前を付けてはならない。

形式 2

5. ANS85 任意指定の COMMON 句は、プログラムが他のプログラムに含まれる場合だけ指定できる。
6. ANS85 IS PROGRAM 指定を書く場合は、COMMON 句と INITIAL 句のどちらか一方、または両方、

   OS/390 ISO2002 MF または RECURSIVE 句

   を指定する。

7. ISO2002 定数-1 は英数字の定数でなければならず、表意定数であってはならない。
8. ISO2002 他のプログラム内に含まれるプログラムの中に定数-1 を指定してはならない。
9. ISO2002 MF このプログラムを直接または間接に含むプログラムが初期プログラムである場合は、RECURSIVE 句を指定してはならない。
10. OS/390 他のプログラムに含まれるプログラムの中で、RECURSIVE 句を 指定してはならない。
11. OS/390 他のプログラムを含むプログラムの中で、RECURSIVE 句を 指定してはならない。

形式 4

1. ISO2002 MF Program-prototype-name-1 は、プログラムプロトタイプを識別する。ただし、定数-1 (指定されている場合) は、運用環境の外部のプログラムプロトタイプである。

一般規則

形式 1 および 2

1. プログラム名-1 はこのプログラム定義によって宣言されるプログラムに名前を付けるものであり、このプログラムに関する実行用プログラム・コード・ファイルと関連する。定数-1 を指定した場合、それは運用環境の外部のプログラム名となる。

形式 2

2. (ANS85)COMMON 指定は、プログラムが共通のものであることを指定する。共通のプログラムは他のプログラムの中に含まれるが、親のプログラム以外のプログラムからも呼び出せる。
3. (ANS85)INITIAL 指定は、プログラムが初期処理用のものであることを指定する。初期プログラムが呼ばれると、そのプログラムおよびその中に含まれるすべてのプログラムが初期化される（***翻訳集団の概念***の章で前述した*初期状態*節を参照）。
4. (OS/390)(SO1002)(MF)RECURSIVE 句は、プログラムが再帰的であることを指定する。

   (SO1002)(MF)再帰プログラムに含まれるプログラムはいずれも再帰的である。

   (OS/390)(SO1002)(MF)局所記憶節の指定を含むプログラムは再帰的である。

   再帰的プログラムは、実行中に呼び出すことができる。また、自身を呼び出すこともできる。

5. (OS/390)(SO1002)(MF)プログラム中で RECURSIVE 句が指定されておらず、プログラムに対して暗黙的に指定されてもない場合は、そのプログラムを実行中に呼び出すことはできない。
6. (OS/390)(SO1002)(MF)共通、初期、および再帰プログラムに関する詳しい規則は、***翻訳集団の概念***の章の*共通プログラム、初期プログラム、および再帰プログラム*節で解説する。

形式 3

7. (MF)Program-name-1 は、呼出しプロトタイプの名を示す。このプロトタイプを使用して、構文のチェックが行われ、CALL 定数文の有効性と性質が決定される。

5.3 クラス名段落

(SO1002)(MF)

クラス名段落は、該当の見出し部でクラス定義を行おうとしていることを示し、クラスを識別する名前を指定するとともに、クラスにクラス属性を割り当てる。

一般形式

```
CLASS-ID. クラス名-1 [ AS 定数-1 ] [ IS FINAL ]
        [ INHERITS FROM { クラス名-2 } ... ]
        [ USING { パラメータ名-1 } ... ] .
```

指令

1. 予約語リストにフラグを付けたり、その内容を修正する機能を提供するコンパイラ指令に加えて、下記の指令がこの節に記述されている構文または意味内容に影響を与える可能性がある

- ACTUAL-PARAMS - パラメータ化されたクラスやインタフェイスからのクラスの作成に使用されるパラメータを指定する。

構文規則

1. 定数-1 は英数字の定数であり、表意定数であってはならない。
2. クラス名-2 は、リポジトリ段落

   (MF)またはこのソース要素のクラス管理段落

   の中で指定されたクラスの名前とする。.

3. クラス名-2 は、クラス定義によって定義されたクラスの名前であってはならしない。
4. クラス名-2 は、クラス名-1 を直接的にも間接的にも継承しない。
5. クラス名-2 は、FINAL 句で定義されたクラスの名前であってはならない。
6. パラメータ名-1 は、このクラス定義のリポジトリ段落中のクラス指定子またはインターフェイス指定子に指定された名前とする。
7. クラス名は、INHERITS 句内に2度以上存在してはならない。
8. 同じ名の別のメソッドが2個以上継承されている場合は、そのどちらも FINAL 句で指定してはならない。同じメソッドが、2個以上の中間の上位クラスを介して、その上位クラスから継承される場合は、FINAL 句で指定することができる。
9. メソッド名が2個以上の継承されたクラスから継承される場合で、これらのメソッド用のプロトタイプが、同じインターフェイスでメソッドを定義しており、このクラス内のプロトタイプのどれかのために、このクラスが、継承された全クラスと適合できない場合は、同じメソッド名のメソッドがこのクラスで宣言されるものとする。このメソッドは、この章の*メソッド名段落*節にある構文規則 9 に従う。

注: クラス A が、2 個のクラス、B と C からメソッド M を継承し、M 用のメソッドインターフェイスがクラス B と C では異なっている (たとえば、B の M はクラス X のオブジェクトを返し、C の M は関係のないクラス Y のオブジェクトを返す) とする。この場合は、クラス A で、M による有効なメソッドの優先が定義され、メソッドインターフェイスの適合性の問題が解決されていない限り、継承は無効である。これはつねに可能であるわけではない (つまり、継承は無効となる) が、継承を有効にできる場合もある。たとえば、X と Y を継承するクラスとして Z を定義すると、M によるメソッドの優先をクラス A で指定し、クラス Z のオブジェクトが返されるようにすることができる。

一般規則

1. クラス名-1 は、このクラス定義を宣言するクラスの名とする。ただし、定数-1 (存在する場合) は、運用環境で外部化されたクラスの名である。
2. INHERITS 句は、***クラスの継承***節に従い、クラス名-1 により継承されるクラスの名とする。

3. FINAL 句が指定されている場合は、そのクラスは他のクラスの上位クラスとはなり得ない。
4. USING 指定があれば、パラメータ化されたクラスとなる。パラメータ名-1 は、仮パラメータにつけられた名とする。パラメータ化されたクラスの動作について詳しくは、***翻訳集団の概念***の章の*パラメータ化されたクラス*節を参照。
5. パラメータ名-1 は、このクラス定義内で指定され、クラス名またはインターフェイス名のみが許可される。
6. 同じクラスが2度以上継承されている場合、そのクラス用のデータは1度だけクラス名-1 に追加される。

注: 同じクラスが直接継承されるのは、1度だけであるが、間接的には複数回継承され得る。たとえば、クラス D がクラス B および C から継承し、クラス B および C のどちらもクラス A から継承するとする。この場合、クラス D は、クラス B の上位クラスおよびクラス C の上位クラスとして、クラス A から間接的に2度継承する。

5.4 ファクトリ段落

ファクトリ段落は、この見出し部でファクトリ定義を行おうとしていることを示す。

一般形式

FACTORY. [IMPLEMENTS {インターフェイス名-1} ... .]

構文規則

1. インターフェイス名-1 は、親のクラス定義のリポジトリ段落で指定されたインターフェイスの名前とする。
2. 使用されたインターフェイスの各メソッドプロトタイプは、それにより、このクラスのファクトリインターフェイスが、使用されている全インターフェイスと適合するものとする。

一般規則

1. IMPLEMENTS 句は、***翻訳集団の概念***の章の*インターフェイスの処理系*節に従い、親のクラスのファクトリオブジェクトにより使用されるインターフェイスの名を指定する。
2. ファクトリオブジェクトは、以下の場合にインタフェイス int-1 を指定する。
   a. ファクトリオブジェクトが、int-1 を指定する IMPLEMENTS 句で定義されている。
   b. ファクトリオブジェクトが、int-1 を継承するインターフェイスを使用している。

c. ファクトリオブジェクトを含むクラスが、int-1 を使用するファクトリオブジェクトを含むクラスを継承している。

5.5 オブジェクト段落

ISO10027 MF

オブジェクト段落は、この見出し部でオブジェクト定義を行おうとしていることを示す。

一般形式

OBJECT. [IMPLEMENTS {インターフェイス名-1} ... .]

構文規則

1. インターフェイス名-1 は、親のクラス定義のリポジトリ段落で指定されたインターフェイスの名前とする。
2. 使用されたインターフェイスの各メソッドプロトタイプは、それにより、このクラスのファクトリインターフェイスが、使用されている全インターフェイスと適合するものとする。

一般規則

1. IMPLEMENTS 句は、*翻訳集団の概念*の章の*インターフェイスの処理系*節に従い、親のクラスのインスタンスオブジェクトにより使用されるインターフェイスの名を指定する。
2. インスタンスオブジェクトは、以下の場合にインタフェイス int-1 を指定する。
   a. インスタンスオブジェクトが、int-1 を指定する IMPLEMENTS 句で定義されている。
   b. インスタンスオブジェクトが、int-1 を継承するインターフェイスを使用している。
   c. インスタンスオブジェクトを含むクラスが、int-1 を使用するファクトリオブジェクトを含むクラスを継承している。

5.6 メソッド名段落

ISO10027 MF

メソッド名段落は、この見出し部でメソッド定義を行おうとしていることを示し、メソッドまたはメソッドプロトタイプを識別する名前を指定するとともに、メソッドにメソッド属性を割り当てる。

一般形式

```
                  ┌ メソッド名-1      [AS 定数-1 ]                 ┐
METHOD-ID.        │                                                │ [OVERRIDE] [IS FINAL].
                  └ {GET} PROPERTY 属性名-1                        ┘
                    {SET}
```

構文規則

1. 定数-1 は英数字の定数であり、表意定数であってはならない。
2. 親のオブジェクトまたはファクトリの定義の作業場所節にデータ名としてプロパティ名-1 を指定した場合、そのデータ名のデータ記述項に PROPERTY 指定を書いてはならない。
3. GET 指定を書いた場合、手続き部の見出し中にメソッドの USING 指定パラメータを書いてはならず、メソッドに単一の RETURNING 指定を書かなければならない。
4. SET 指定を書いた場合、手続き部の見出し中にメソッドの USING 指定パラメータを書かなければならず、RETURNING 指定を書いてはならない。
5. OVERRIDE 指定は、メソッドプロトタイプで使用してはならない。
6. OVERRIDE 指定が使用されている場合は、上位クラスで定義されたメソッド定義により宣言されたメソッドと同じ名のメソッドが存在していなければならない。上位クラスにあるメソッドは、FINAL 句で定義されていてはならない。
7. OVERRIDE が使用されていない場合で、
    a. このメソッド定義がクラス定義に含まれる場合は、継承されたメソッドはどれも、このメソッド定義で宣言されたメソッドと同じ名であってはならない。
    b. このメソッド定義がインターフェイス定義に含まれる場合は、継承されたメソッドプロトタイプはどれも、このメソッド定義で宣言されたメソッドプロトタイプと同じ名であってはならない。
8. FINAL 句は、メソッドプロトタイプで指定されていてはならない。
9. メソッド名-1 または定数-1 が、親の定義より継承されたメソッド名と同じである場合は、手続き部見出しの中のパラメータ宣言および返される項目が適合規則に従ったものでなければならない。これにより、このメソッド定義を含む定義が、***手続き部***の章の*パラメータおよび戻り項目の適合*節に従い、継承された定義すべてに適合することが保証される。

一般規則

1. このメソッド定義により宣言されるメソッドの名は、以下のように決定される。
    a. PROPERTY 句を指定した場合、メソッド名は下記のように設定される。

        1. GET 指定を書いた場合、文字"GET"にプロパティ名-1 を大文字化したものが結合される。
        2. SET 指定を書いた場合、文字"SET"にプロパティ名-1 を大文字化したものが結合される。

    b. PROPERTY 句を指定しなかった場合、メソッド名-1 がメソッド名となる。ただし、定数-1 を指定した場合、それは運用環境の外部にあるクラスの名前を指す。
2. OVERRIDE 指定は、このメソッドが継承されたメソッドに優先することを示す。
3. FINAL 句は、どの下位クラスでも、このメソッドが他のメソッドに優先されないことを示す。
4. メソッドの名前は、このメソッドが定義されているクラスのオブジェクトに関するメソッド呼出しの中で参照できる。
5. 指定した利用者語がこのメソッド定義のデータ部および上位のファクトリ定義またはオブジェクト定義のデータ部の中に定義されている場合、このメソッドにおいて使用した利用者語はこのメソッド内の宣言を参照する。このメソッドから上位のファクトリ定義またはオブジェクト定義中の宣言を呼び出すことはできない。
6. GET 指定を指定した場合、このメソッドはプロパティ名-1 のプロパティ読み出しメソッドとなる。
7. SET 指定を指定した場合、このメソッドはプロパティ名-1 のプロパティ設定メソッドとなる。

5.7 インターフェイス名段落

INTERFACE-ID 段落は、この見出し部でインターフェイス定義を行おうとしていることを示し、インターフェイスを識別する名前を指定するとともに、インターフェイスにインターフェイス属性を割り当てる。

一般形式

```
INTERFACE-ID. インターフェイス名-1 [ AS 定数-1 ]
        [ INHERITS FROM { インターフェイス名-2 } ... ]
        [ USING { パラメータ名-1 } ... ] .
```

指令

1. 予約語リストにフラグを付けたり、その内容を修正する機能を提供するコンパイラ指令に加えて、下記の指令がこの節に記述されている構文または意味内容に影響を与える可能性がある
    - ACTUAL-PARAMS - パラメータ化されたクラスやインタフェイスからのインターフェイスの作成に使用されるパラメータを指定する。

構文規則

1. 定数-1 は英数字の定数であり、表意定数であってはならない。
2. インターフェイス-2 このソース要素のリポジトリ段落中に指定されたインターフェイスの名前でなければならない。
3. インターフェイス名-2 はインターフェイス-1 から直接的にも間接的にも継承してはならない。

4. パラメータ名-1 は、このインターフェイス定義のリポジトリ段落中のクラス指定子またはインターフェイス指定子の中に指定された名前でなければならない。
5. メソッド名が2個以上のインターフェイスから継承されている場合は、継承された各インターフェイス内のメソッドプロトタイプにより、このインターフェイスが継承された全インターフェイスに適合するものとする。

一般規則

1. インターフェイス名-1 はこのインターフェイス定義に宣言されたインターフェイスに名前を付ける。しかし、定数-1 を指定した場合、それは運用環境の外部にあるインターフェイスの名前を指す。
2. INHERITS 指定は、***翻訳集団の概念***の章の*インターフェイスの継承*節に基づいてインターフェイス名-1 によって継承される、インターフェイスの名前を指定する。
3. USING 指定は、該当のインターフェイスがパラメータ化されていることを指定する。パラメータ名-1 は仮パラメータに付けられた名前である。
4. パラメータ名-1 は、クラス名またはインターフェイス名が許されている、このインターフェイス定義中のみで指定する。

5.8 関数名段落

(ISO/IEC) (MF)

関数名段落は、関数の識別名を指定し、その関数に選択された属性を割り当てる。

一般形式

形式 1 (定義)

FUNCTION-ID. 利用者関数名-1 [AS 定数-1].

形式 2 (プロトタイプ)

FUNCTION-ID.
　関数プロトタイプ名-1 [AS 定数-1] IS PROTOTYPE.

構文規則

1. 定数-1 は英数字の定数であり、表意定数であってはならない。

一般規則

形式 1

1. 利用者関数名-1 は、この関数定義により宣言される関数の名となる。ただし、定数-1 (指定されている場合) は、運用環境の外部にある関数の名とする。

形式 2

2. 関数プロトタイプ名-1 は、この関数定義により宣言される関数プロトタイプの名となる。ただし、定数-1 (指定されている場合) は、運用環境の外部にある関数プロトタイプの名とする。

5.9 翻訳日付段落

機能

翻訳日付 (date-compiled) 段落は、原始プログラム・リストの見出し部の中に、中間コードが生成された日を挿入する。

一般形式

```
DATE-COMPILED. [注記項]...
```

指令

1. 予約語リストにフラグを付けたり、その内容を修正する機能を提供するコンパイラ指令に加えて、下記の指令がこの節に記述されている構文または意味内容に影響を与える可能性がある。
   - DATE - この段落に日付を挿入する形式を指定する。

構文規則

1. 注記項には、計算機の文字集合に属する任意の文字を組み合わせて書ける。標識領域にハイフンを指定して、注記項を次の行に続けることはできない。しかし、注記項を複数行にわたって書くことはできる。この場合、注記項を書ける範囲は B 領域に限定される。次の行を A 領域から書き始めると、注記項ではない段落を書き始めたことになる。

一般規則

1. 翻訳日付段落が書いてあると (注記項の有無を問わない)、COBOL システムは生成される実行用プログラム・コードに日付注記項を挿入する。プログラム・リスト中では、その内容は下記の形に置き換えられる。

   DATE-COMPILED. 現在の日付と時刻。

   ここで、日付と時刻の形式は、DD-MMM-YY hh:mm である。

DATE 指令を使用すると、注記項を置き換えるために COBOL コンパイラによって使用される文字列を変形できる。

ANSI85 翻訳日付段落に挿入される値は、組込み関数の WHEN-COMPILED で使用される値でもある。ただし、両者の形式に違いがあることもある。.

第 6 章：環境部

6.1 一般説明

環境部では、計算機機種の物理的な特性によるデータ処理上の問題を取り扱う、標準的な方法を指定する。たとえば、翻訳用計算機と実行用計算機の構成を指定できる。それに加えて、入出力管理、ハードウエアの特殊な仕様、管理技法などに関する情報も指定できる。

(ANS85)COBOL 原始要素において、環境部は書いても書かなくてもよい。

一般形式

ENVIRONMENT DIVISION. [構成節] [入出力節]

構文規則

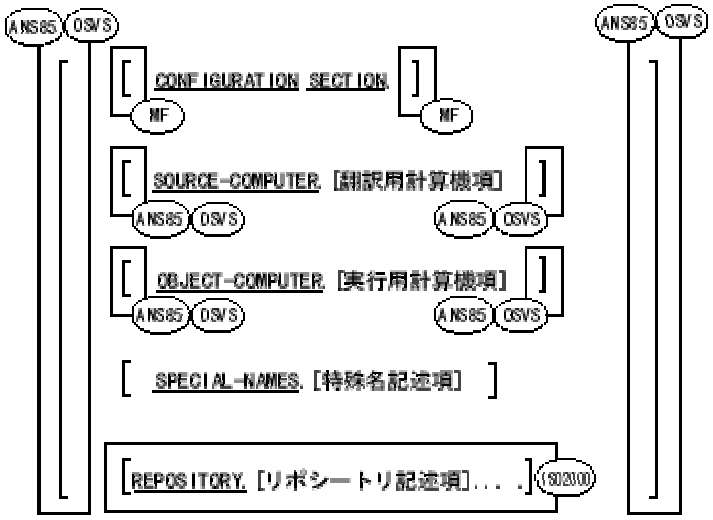
1. (MF)プログラムに環境部が含まれる場合、環境部がプログラムの中の最初の部である場合にのみ、環境部の見出しを書かなくてもよい。

6.2 構成節

構成節 (configuration section) は、環境部を構成する節の 1 つである。構成節で指定する事柄には、特定のシステムに依存するデータ処理システムの様相、特殊な管理技法、局所名を外部名に関連づける方法などがある。この節は翻訳用計算機段落、実行用計算機段落、特殊名段落、リポジトリ段落に分けられる。翻訳用計算機段落では、ソース要素を翻訳する計算機の構成を記述する。実行用計算機段落では、コンパイラによって生成されたランタイム・モジュールを実行する計算機の構成を記述する。特殊名段落では、通貨記号および通貨文字列の指定、小数点の選択、記号文字の指定、作成者語と利用者が指定する呼び名との関連づけ、符号系名と文字集合または文字の照合順序への対応づけ、クラス名と文字集合の関連づけを行う。リポジトリ段落では、局所名を外部資源と関連づけ、どの組み込み関数がこの原始単位用の予約語となるかを指定する。

(ANS85)(OSVS)構成節は書いても書かなくてもよい。.

一般形式



構文規則

1. (MF)構成節がプログラムの中の最初の文である場合にだけ、構成節の見出しを書かなくてもよい。
2. (ANS85)他のプログラムに含まれるプログラムの中には、構成節を記述できない。
3. (SO2000)メソッド定義中に構成節を記述してはならない。
4. (SO2000)ファクトリ定義またはオブジェクト定義の中に翻訳用計算機、実行用計算機、リポジトリの各段落を記述してはならない。

一般規則

1. (ANS85)他の原始単位を含むソース単位の構成節の中に明示的または暗黙的に記述されている事項は、直接的または間接的に含まれる各原始単位にも適用される。

6.2.1 翻訳用計算機段落

機能

翻訳用計算機 (source-computer) 段落は、翻訳単位を翻訳する計算機を識別する。

一般形式

一般規則

1. 翻訳用計算機段落は注記としてだけ使用する。
2. WITH DEBUGGING MODE は、標準 ANSI COBOL に従ったデバッグ用コードを組み入れるために使用する(*言語リファレンス - 追加トピック*中の *デバッグ*の章の *COBOL Debug の環境部*を参照。)
3. (ISO2002)翻訳用計算機段落のすべての句は、それらが明示的または暗黙的に指定されたソース単位およびその原始単位に含まれる任意のソース単位に適用される。

6.2.2 実行用計算機段落

機能

実行用計算機 (object-computer) 段落には、コンパイラによって作成されたランタイムモジュールを実行するのに使う計算機を記述する。

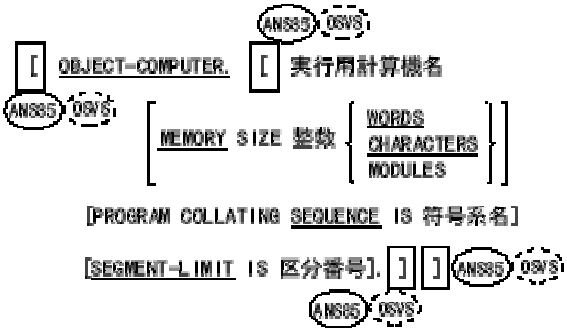
(ANS85)(OSVS)この段落は書いても書かなくてもよい。

(ANS85)MEMORY SIZE (記憶容量) 句と SEGMENT-LIMIT(区分制限)句は、ANSI'85 においては廃要素に分類されており、ANSI 標準を次回に全面改訂する際には削除される予定である。

(MF)この COBOL 処理系に組み込まれているすべての方言では、MEMORY SIZE 句 と SEGMENT-LIMIT 句は注記としてだけ使用する。FLAG STD 指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。

(XOPEN)標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、MEMORY SIZE 句と SEGMENT-LIMIT 句は明示的に除外されている。したがって、X/Open に準拠する COBOL 原始プログラムにおいては、MEMORY SIZE 句と SEGMENT-LIMIT 句を使用するべきではない。

一般形式



構文規則

1. 実行用計算機名は、利用者が定義した 1 語の COBOL の語である。
2. (ANS85)(OSVS)実行用計算機段落は、OBJECT-COMPUTER 見出しだけから成る。

一般規則

1. 翻訳用計算機名は、機器構成を識別する手段となる。この場合、計算機名およびそれが意味する機器構成は、利用者が指定する。構成の定義には記憶容量に関する情報を含む。計算機名と MEMORY SIZE 句は注記としてだけ使用する。
2. PROGRAM COLLATING SEQUENCE プログラムの文字の照合順序) 句を指定しないと、計算機に固有の文字の照合順序が適用される。 付録の*文字集合と文字の照合順序* に、ASCII および EBCDIC のすべての文字の照合順序の一覧を掲げてある。NATIVE コンパイラ指令を使用して、計算機に固有の文字の照合順序を選択することもできる。
3. PROGRAM COLLATING SEQUENCE 句を指定すると、プログラムの文字の照合順序は指定した符号系名のものとなる。
4. 実行用計算機段落に指定したプログラムの文字の照合順序は、下記の文字比較における真理値の決定に用いられる。
   a. 比較条件の中で明示(***手続き部***の章の*比較条件*節を参照)。
   b. 条件名条件の中で明示(***手続き部***の*条件名条件 (条件変数)* 節を参照)。
5. SORT (整列) 文または MERGE (併合) 文に COLLATING SEQUENCE 句を指定しないと、整列または併合用の文字のキーにも PROGRAM COLLATING SEQUENCE 句が使用される。
6. 索引ファイルの順序づけには、PROGRAM COLLATING SEQUENCE 句は何の効力も発揮しない。
7. PROGRAM COLLATING SEQUENCE 句は、それが指定された原始要素の中、

   (ANS85)およびその中に含まれる原始要素の中

   で使用される。

8. SEGMENT-LIMIT (常駐区分の範囲) 句は、注記としてだけ使用する(***言語リファレンス - 追加トピック***の区分化*区分化*節を参照)。

6.2.3 特殊名段落

機能

特殊名 (special-names) 段落は、通貨記号および通貨文字列、小数点,

(ANS85)記号文字、

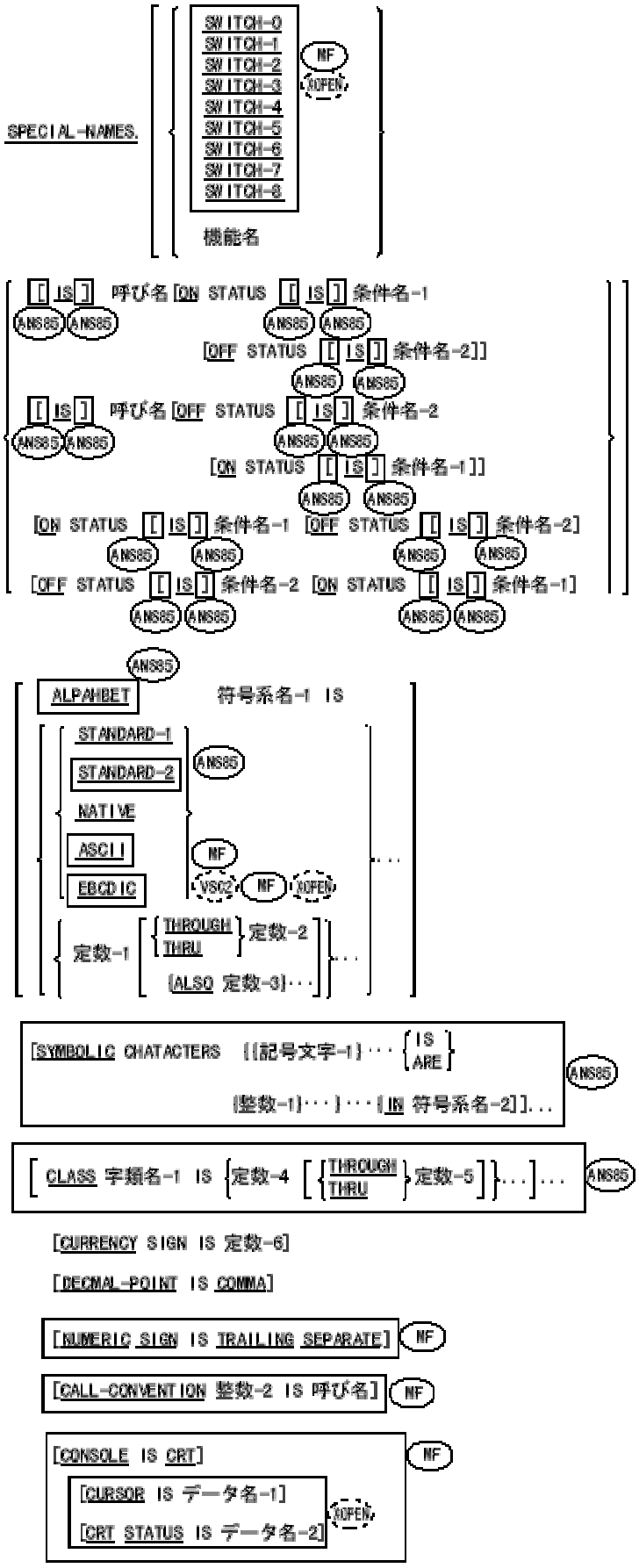
利用者が指定する呼び名に対する作成者語、文字集合や文字の照合順序に対する符号系名、文字の組に対するクラス名などの、指定または選択を行う手段を示す。

(MF)パラメータの引渡し方について予め定義してあるいくつかの方式の中から、1 つを選択できる。CALL (呼ぶ) 文および PROCEDURE DIVISION の見出しの拡張機能によって、これらのパラメータ引渡し方式をプログラム間連絡に使用できるようになった。

例:

1. CRT 状態キーの使用方法については、*言語リファレンス - 追加トピック*中の*例*の章の*特殊名段落の CRT STATUS 句*節を参照。

一般形式


SPECIAL-NAMES.
SWITCH-0
SWITCH-1
SWITCH-2
SWITCH-3
SWITCH-4
SWITCH-5
SWITCH-6
SWITCH-7
SWITCH-8
NF
XOPEN
機能名
[IS] 呼び名 [ON STATUS [IS] 条件名-1
ANS85 ANS85 ANS85 ANS85
[OFF STATUS [IS] 条件名-2]]
ANS85 ANS85
[IS] 呼び名 [OFF STATUS [IS] 条件名-2
ANS85 ANS85 ANS85 ANS85
[ON STATUS [IS] 条件名-1]]
ANS85 ANS85
[ON STATUS [IS] 条件名-1 [OFF STATUS [IS] 条件名-2]
ANS85 ANS85 ANS85 ANS85
[OFF STATUS [IS] 条件名-2 [ON STATUS [IS] 条件名-1]
ANS85 ANS85 ANS85 ANS85
ANS85
ALPAHBET
符号系名-1 IS
STANDARD-1
STANDARD-2
NATIVE
ANS85
ASCII
EBCDIC
NF
VSC2
NF
XOPEN
定数-1
THROUGH
THRU
定数-2
[ALSO 定数-3]...
[SYMBOLIC CHATACTERS {[記号文字-1]... IS ARE
[整数-1]...}...[IN 符号系名-2]]...
ANS85
CLASS 字類名-1 IS 定数-4 [THROUGH THRU 定数-5]...
ANS85
[CURRENCY SIGN IS 定数-6]
[DECMAL-POINT IS COMMA]
[NUMERIC SIGN IS TRAILING SEPARATE]
NF
[CALL-CONVENTION 整数-2 IS 呼び名]
NF
[CONSOLE IS CRT]
NF
[CURSOR IS データ名-1]
[CRT STATUS IS データ名-2]
XOPEN

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - ALPHASTART - ALPHABET 句の中で使用する有効な数値定数が0と1のどちらで始まるかを指定する。
    - CHARSET - どのアルファベットを固有文字集合とみなすかに影響する。環境によっては、この指令は無視される。
    - NATIVE - どのアルファベットを固有文字集合とみなすかに影響する。
    - SYMBSTART - SYMBOLIC CHARACTERS 句の中で使用する有効な数値定数が0と1のどちらで始まるかを指定する。

構文規則

1. OSVS VSC2 MF 特殊名段落中の句は、どのような順序で書いても構わない。
2. ISO2000 クラスを定義する原始単位の外側のレベルでは、CURSOR および CRT STATUS 句を指定してはならない。
3. ISO2000 クラス定義のファクトリまたはオブジェクトにおいては、指定できる句は CURSOR および CRT STATUS だけである。
4. ISO2000 ンターフェイス定義中に特殊名段落を指定した場合、使用できる句は ALPHABET, CURRENCY, DECIMAL-POINT だけである。
5. 呼び名には、任意の COBOL の利用者語を当てることができる。最低限 1 文字は英字とする。
6. 関数名は、COBOL コンパイラによって使用されるシステム装置または機能を参照する。

   OSVS VSC2 MF 関数名が UPSI-0 から UPSI-7 のどれかである場合、関数名は外部スイッチを参照する。

7. 関数名が外部スイッチを参照するか、

   MF XOPEN または SWITCH-*n* オプションが使用されている

   場合は、その呼び名は指定できない。

   MF XOPEN ただし、SET 文の中でだけは指定できる。

   その呼び名には、条件名を最低限 1 つ付ける。

   ANS85 条件名を付ける必要はない。

8. 外部スイッチを参照しない関数名に対応する呼び名を指定できるのは、ACCEPT（受け取り）、DISPLAY（表示）、SEND（送信）、WRITE（書き出し）の各文だけである。このような作成者語に条件名を関係付けることはできない。
9. 符号系名-1 句の定数句の中に指定する定数は、下記のようにする。

a. 数字である場合は、符号の付かない整数とし、その値は 1 以上で、計算機に固有の文字集合の文字の個数以下とする。

   ALPHABET (符号系名) 句の中の数字定数の有効な最小値は、ALPHASTART 指令の影響を受ける。

b. 文字であり、THROUGH (から) 指定または ALSO (もまた) 指定を指定した場合、すべて長さは 1 文字とする。
c. (ISC2)(MF)浮動小数点数または 2 バイト文字は指定できない。

10. 符号系名-1 句の定数句を指定する場合、1 つの符号系名の中に同じ文字を 2 回以上書いてはならない。
11. THRU と THROUGH は同義語であり、どちらを書いてもよい。
12. (MF)STANDARD-D1 と ASCII は同義語であり、どちらを書いてもよい。
13. (ANS85)特殊名段落の中では、予約語の IS はまったく必要ない。
14. (ANS85)定数-4 句の中に指定する定数は、下記のようにする。
    a. 数字である場合は、符号の付かない整数とし、その値は 1 以上で、計算機に固有の文字集合の文字の個数以下とする。
    b. 文字であり THROUGH (から) 指定を指定した場合、すべて長さは 1 文字とする。
    c. (ISC2)(MF)浮動小数点数または 2 バイト文字は指定できない。
15. (ANS85)定数-1 から定数-5 には、記号文字の表意定数を指定してはならない。
16. (ANS85)1つの SYMBOLIC CHARACTERS (記号文字) 句の中には、同じ記号文字-1 は 1 回だけ使用できる。
17. (ANS85)各記号文字-1 とそれに対応する各整数-1 とは、SYMBOLIC CHARACTERS 句の中の位置によって関係付けられる。最初の記号文字-1 が最初の整数-1 と対にされ、2 番目の記号文字-1 が 2 番目の整数-1 と対にされるという具合いに、SYMBOLIC CHARACTERS 句全体を通じて対応付けが行われる。IS または ARE を間に置くのは読みやすくするためだけである。

    整数-1 の有効な最小値は、SYMBSTART コンパイラ指令の影響を受ける。

18. (ANS85)記号文字-1 に指定する一連の文字と整数-1 に指定する一連の数字とは、1 対 1 で対応させる。このことは、各 IS 指定または ARE 指定の内部でも、SYMBOLIC CHARACTERS 句全体でも、当てはまる。
19. (ANS85)整数-1 で指定する文字位置は、計算機固有の文字集合の中に存在させる。IN 指定を書く場合は、符号系名-2 で指定する文字集合の中に、整数-1 で指定する文字位置を存在させる。符号系名-2 は、ALPHABET (符号系) 句に指定してあるものとする。
20. 定数-6 は、英数字の定数でなければならない。

    (COB370)(ISO1000)(MF)または、各国語の定数でなければならない。

21. 定数-6 は表意定数であってはならない。
22. PICTURE SYMBOL 指定が使用されていない場合は、定数-6 は、1 文字から成るものとする。ただし、以下の文字を除く。
    a. 0 から 9 までの数字

b. 英字の A、B、C、D、L、P、R、S、V、X、Z、およびその小文字、および空白

MF C または R は使用可能

ANS85 L は使用可能

ISO2002 OSVS VSC2 MF E は使用不可

COBOL74 ISO2002 MF N は使用不可

VSC2 MF G は使用不可

ANS85 小文字の英字は使用不可

ISO2002 小文字の英字は、ここで使用不可と明記されているものを除き、使用可能

c. 記号、'+' '-' ',' '.' '*' '/' ';' '(' ')' '"' '='

23. OS/390 ISO2002 MF PICTURE SYMBOL が使用されている場合は、定数-6 の長さに制限はなく、以下に従うものとする。
    a. 空白以外の文字を 1 個含む。
    b. 計算機の符号化文字セットの文字のすべてが使用可能。ただし、0 から 9 の数字と記号の '+' '-' ',' '.' '*'を除く。
24. OS/390 ISO2002 MF 定数-7 は、1 字の英数字定数から成るものとし、表意定数であってはならない。同じ定数-7 が複数存在してはならない。
25. OS/390 ISO2002 MF 定数-7 には、計算機の符号化文字セットの文字のすべてが使用可能。ただし、以下を除く。
    a. 0 から 9 の数字
    b. 英字の A、B、C、D、E、N、P、R、S、V、X、Z、およびその小文字、および空白

       MF C および R は使用可能

       MF G は使用不可

    c. 記号、'+' '-' ',' '.' '*' '/' ';' '(' ')' '"' '='
26. 定数-6 の字類が英数字の場合は、関連づけられた通貨記号は、用途が DISPLAY である数字編集項目の定義にのみ使用可能とする。

    OS/390 ISO2002 MF 定数-6 の字類が各国語型の場合は、関連づけられた通貨記号は、用途が NATIONAL である数字編集項目の定義にのみ使用可能とする。

27. MF 整数-2 は、0 から 65535 までの符号のつかない整数とする。
28. MF OPEN CURSOR IS 句のデータ名-1 は、作業場所節 で宣言されなければならない。
29. MF OPEN CRT STATUS 句は、各形式 4 以降の状態値の転記先となるデータ項目を指定する。

(MF)または、各形式 5 以降の状態値の転記先となるデータ項目を指定する。

(MF)(XOPEN)または、各 ACCEPT 文以降の状態値の転記先となるデータ項目を指定する。環境によっては、これは作業場所の最初の 64K でのみ定義可能。

30. (MF)(XOPEN)データ名-2 は作業場所節で記述され、3 バイトの長さでなければならない。
31. (MF)(XOPEN)CURSOR IS 句は、ACCEPT 文によって使用されるカーソルの番地を保持するデータ項目を指定する。

一般規則

1. 外部スイッチは、実行時に操作員によって設定される。その設定は、 関連する条件名を検査することで判定される。
2. 呼び名が外部スイッチと関係付けられている場合、形式 1 の SET (設定) 文を実行することによって、その外部スイッチの状態を変更できる。(*手続き部 - SEARCH - XML PARSE* の章の *SET(設定)文*節を参照。)
3. 符号系名-1 句は、文字符号系 (character code set) または文字の照合順序に名前を付ける働きをする。PROGRAM COLLATING SEQUENCE 句(前述の*実行用計算機段落*節を参照)、または SORT 文か MERGE 文(*手続き部 - SEARCH - XML PARSE* の章の *SORT(整列)文*節および *手続き部 - MERGE - OPEN* の章の *MERGE(併合)文*節を参照)の中の COLLATING SEQUENCE 句の中で符号系名-1 が参照される場合、符号系名-1 は文字の照合順序を示すことになる。ファイル記述項(*データ部 - ファイルおよびデータ記述*の章の*ファイル記述項の全体的な骨組み*節を参照) の CODE-SET (符号系) 句の中で符号系名-1 が参照される場合、符号系名-1 は文字符号系を示すことになる。
    a. STANDARD-1 または

       (MF)ASCII

       指定が含まれる場合は、文字符号系または文字の照合順序は、米国標準規格 X3.4-1968 に規定されている情報交換用米国標準コードとなる。

       (ANS85)STANDARD-2 指定が含まれる場合は、文字符号系は、国際標準 646 の情報処理交換用 7 ビット･コード化文字集合に規定されている、ISO7 ビット･コードの国際参照版となる。

       (ISC2)(MF)(XOPEN)EBCDIC 指定が含まれる場合は、文字符号系および文字の照合順序は EBCDIC となる。

       NATIVE (計算機固有) 句を指定すると、計算機に固有の文字符号系または文字の照合順序が使用される。計算機に固有の文字の照合順序は、NATIVE 指令を使用して、ASCII または EBCDIC のどちらかに設定できる。

       ASCII 符号系および ASCII と EBCDIC の文字の照合順序と両者の対応関係については、付録の*文字符号系と文字の照合順序*の章を参照。

b. 定数句を指定すると、符号系名-1 は CODE-SET 句の中では参照できない (*データ部 - ファイルおよびデータ記述の*章の *CODE-SET(符号系)句*節を参照)。 文字符号系または文字の照合順序は、下記の規則に従って定義される。
   1. 各定数の値は、下記のことを表わす。
      a. 定数が数字であるならば、固有文字集合内の文字の順序番号。この値は、固有文字集合内の文字の数を超えてはならない。
      b. 定数が文字であるならば、固有文字集合内の実際の文字。文字定数の値が複数の文字から構成される場合、その文字定数内の各文字に、左端の文字から開始して連続した昇順に、文字の照合順序が設定される。
   2. ALPHABET 句内に定数が現れる順番によって、文字の照合順序の文字の順序番号を昇順に定める。
   3. 固有の文字の照合順序のうち、この定数句で書かれなかった文字は、書かれたどの文字よりも大きい順序位置に置かれる。書かれていない文字相互間の照合順序は、固有の文字の照合順序に従う。
   4. 文字符号系を指定する際に、固有の文字集合の中の定数-1 句に指定されなかった各文字に関しては、作成者が指定する文字集合内の順序番号を定義する。
   5. THROUGH 指定を書くと、固有文字集合の定数-1 の値によって指定される文字から定数-2 の値によって指定される文字までの一連の文字が、指定される文字の照合順序で昇順に連続した位置を割り当てられる。THROUGH 指定を書く際には、一連の文字を昇順に指定しても降順に指定してもよい。
   6. ALSO 指定を書くと、定数-1 と定数-3 の値によって指定される固有文字集合の文字は、文字の照合順序

      (ANS85)またはデータを表わすために使用される文字符号系の中で、

      同じ順序を割り当てられる。

4. プログラムの文字の照合順序で順序位置が最大のものが、表意定数の HIGH-VALUE となる。

   (ANS85)ただし、表意定数 HIGH-VALUE を特殊名段落の中で定数として指定した場合を除く。

   プログラムの文字の照合順序で順序位置が最大のものが 2 つ以上ある場合は、最後に指定した方が表意定数の HIGH-VALUE となる。

5. プログラムの文字の照合順序で順序位置が最小のものが、表意定数の LOW-VALUE となる。

   (ANS85)ただし、表意定数 LOW-VALUE を特殊名段落の中で定数として指定した場合は例外である。

プログラムの文字の照合順序で順序位置が最小のものが 2 つ以上ある場合は、最後に指定した方が表意定数の LOW-VALUE となる。

6. OSVS VSC2 MF 浮動小数点数定数は、利用者が定義した文字の照合順序の中では使用できない。
7. ANS85 特殊名段落の中で定数として指定した場合、表意定数の HIGH-VALUE と LOW-VALUE はそれぞれ、固有の文字の照合順序の中の順序位置が最大のものと最小のものとなる。
8. ANS85 SYMBOLIC CHARACTERS 句は、表意定数のように使用できる利用者語を定義するために使用する。IN 指定を指定しないと、記号文字-1 は固有の文字集合の中で順序位置が整数-1 によって指定される文字を表わす。IN 指定を書くと、整数-1 は符号系名-2 によって指定される文字集合の中の文字の順序位置を表わす。
9. ANS85 記号文字-1 の内部表現は、固有の文字集合の中の該当する文字の内部表現となる。
10. CURRENCY SIGN 句は、受取り側項目として使用されている数字編集項目中に置かれる通貨文字列の指定に使用され、この編集項目が数字または数字編集項目を受取り側とする送出し項目として使用されるときに、そこから逆編集される。さらに、基本形式の PICTURE 句の文字列に、この通貨文字列の使用を指定するための記号を決定するためにも使用される。この記号は、通貨記号と呼ばれる。

    定数-6 は、この通貨文字列の値を表す。

    OS/390 3.0 MF CURRENCY SIGN 句が PICTURE SYMBOL 指定とともに使用されている場合は、定数-7 が通貨記号として使用される。

    CURRENCY SIGN 句が PICTURE SYMBOL 指定を伴わずに使用されている場合は、定数-6 が通貨記号として使用される。

    通貨記号が小文字である場合は、その大文字として処理される。

    この句を指定しないと、COBOL の文字集合に定義されている通貨記号だけが、PICTURE 句の中で使用できる。詳細については、***COBOL 言語の概念***の章の*文字集合*節を参照。

11. DECIMAL POINT IS COMMA 句は、PICTURE 句の文字列と数字定数においてはコンマ (,) と終止符 (.) の機能を逆転させることを意味する。
12. MF NUMERIC SIGN 句を指定すると、符号付き数字項目の符号の位置を個々に指定しなかった場合、末尾に独立の符号が付けられる。
13. OSVS VSC2 MF 呼び名を使用できるところでは、代わりに機能名を使用できる。
14. 機能名に外部スイッチを指定しない場合、下記のものの中から選ぶことができる。

MF TAB	縦のタブ位置へ飛ぶ (必要に応じて出力レコード中に ASCII X"0B"を挿入)	WRITE ADVANCING 文
MF PRINTER	プリンタ	DISPLAY 文

MF FORMFEED	改ページする (必要に応じて出力レコード中に ASCII X"0C" を挿入)	WRITE ADVANCING 文
MF XOPEN COMMAND-LINE	コマンド転送	ACCEPT および DISPLAY 文
MF XOPEN ARGUMENT-NUMBER	コマンド行の引数番号	ACCEPT および DISPLAY 文
MF XOPEN ARGUMENT-VALUE	コマンド行の引数値	ACCEPT 文
MF XOPEN ENVIRONMENT-NAME	環境変数名	DISPLAY 文
MF XOPEN ENVIRONMENT-VALUE	環境変数値	ACCEPT および DISPLAY 文
MF XOPEN SYSERR	標準エラー装置	DISPLAY 文

15. OSVS VSC2 この COBOL システムに組み込まれている OS/VS COBOL,VS COBOLII および SAA において使用できる機能名を下の表 6-1 に示す。

表 6-1: 使用できる機能名

	OSVS	VSC2 Rel (2)	COBOL/370 または VSC2 Rel (3)/(4)	SAA L1
SYSIN	y	y	y	y
SYSIPT	U	U	y	
SYSOUT	y	y	y	y
SYSLIST			y	
SYSLST	U	U	y	
SYSPCH	U	U	y	
SYSPUNCH	U	y	y	
CONSOLE	y	y	y	y
C01	y	y	y	y
C02	y	y	y	
C03	y	y	y	
C04	y	y	y	
C05	y	y	y	
C06	y	y	y	
C07	y	y	y	
C08	y	y	y	

C09	y	y	y	
C10	y	y	y	
C11	y	y	y	
C12	y	y	y	
S01	y	y	y	
S02	y	y	y	
S03		U	y	
S04		U	y	
S05		U	y	
CSP	y	y	y	y
1 char rw literal	y			
AFP-5A			y	

y　メインフレームのコンパイラおよび COBOL システムに明記されており、受け付けられる。

U　明記されていないが、メインフレームのコンパイラに受け付けられる。

17. (ANS85)CLASS 句は、指定した文字の組に名前を付ける働きをする。クラス名-1 はクラス条件の中でだけ参照できる。クラス名-1 には、この句に指定する定数の値によって表わされる文字の組だけが含まれる。各定数の値は下記の内容を表わす。
    a. 定数が数字の場合、固有の文字集合の中の文字の順序番号。この値は、固有の文字集合の文字の数を表わす値を超えてはならない。
    b. 定数が文字の場合、固有の文字集合の中の実際の文字。文字定数が複数の文字から構成される場合、その文字定数の各文字がクラス名-1 によって表わされる文字の組に含まれる。

    (ANS85)THROUGH 指定を書くと、固有文字集合における定数-4 から定数-5 までのすべての文字がクラス名-1 によって表わされる文字の組に含まれる。THROUGH 指定に書く一連の文字は、固有文字集合の中で昇順であっても降順であってもよい。

18. 原始要素の特殊名段落の中に指定したすべての句は、その原始要素 に含まれる原始要素にも適用される。親の原始要素の特殊名段落の中に指定した条件名、英数字名、クラス名、通貨記号、英字名、および記号文字は、子となる原始要素いずれからも参照できる。
19. (MF)CALL-CONVERSION 句を使用すると、パラメータ受渡しに関して予め定義されているいくつかの方式の中から 1 つを選択できる。これによって、COBOL 以外の言語で書かれ、パラメータ受渡し方式の異なる副プログラムを呼び出せる。
20. (MF)CONSOLE IS CRT 句を指定すると、ACCEPT 文または DISPLAY 文のうち、作用対象が画面でなく特定の形式に固有の句を含まないものが、形式 5 として扱われるようになる。CONSOLE IS CRT 句を指定しないと、それらの文は標準の ANSI ACCEPT 文または DISPLAY 文として扱われる。

21. (MF)(OPEN)CURSOR IS 句は、ACCEPT 文によって使用され、カーソルの番地を保持するデータ項目を指定する。
    a. ACCEPT 文の開始時に、データ名-1 に画面上の文字位置として有効な値が入っていると、その文字位置がカーソルの初期位置として使用される。データ名-1 に有効な値が入っていない場合は、データ名-1 は無視されて、画面上の最初の入力項目の先頭がカーソルの初期位置となる。データ名-1 に入っている文字位置を ACCEPT 文の中で使用すると、ACCEPT 文の終了時にデータ名-1 の値が更新される。その結果、ACCEPT 文が終了した時点でのカーソルの位置がデータ名-1 に入れられる。
    b. CURSOR IS 句は、画面上の項目への位置付けには効力をもたない。
    c. データ名-1 の長さは 4 文字または 6 文字とする。データ名-1 の長さが 4 文字である場合、上 2 文字が行番号として解釈され、下 2 文字がカラム番号として解釈される。データ名-1 の長さが 6 文字である場合、上 3 文字が行番号として解釈され、下 3 文字がカラム番号として解釈される。
    d. データ名-1 に入っている値が有効でない (たとえば、ゼロや文字や画面の下辺から外れる値) 場合は、この句は効力をもたない。
    e. データ名-1 に含まれている値が有効な文字位置ではあるが、実行されようとしている ACCEPT 文の入力項目に対応しない場合、カーソルは次の入力項目に位置付けられる。このとき、次の入力項目が存在しないと、最初の入力項目に位置付けられる。入力項目の順番はデータ部にデータ記述を書いた順番である。
22. (MF)(OPEN)CRT STATUS 句は、形式 4 または形式 5 の ACCEPT 文が実行されるたびに、状態値が転記されるデータ項目を指定する。

    (MF)(OPEN)特殊名段落に CRT STATUS 句を指定すると、形式 4 または形式 5 の ACCEPT 文が実行されるたびに (詳細については後述する) 、その結果を示す値がデータ名-2 に入れられる。データ名-2 は状態キーから構成される。そのそれぞれに、ACCEPT 文を実行した結果の状態が記録される。各状態キーの意味を以下に説明する。

CRT 状態キー

(MF)(OPEN)

状態キー1: データ名-1 の最初のバイトは、CRT 状態キー1 である。このデータ記述には、PICTURE 9 USAGE DISPLAY と指定しておく。この状態キーは ACCEPT 操作の終了を引き起こした条件を表わす。このキーがとる具体的な値は下記のとおり。

"0"　終了キーまたは最後の項目の外に出る自動スキップを表わす

"1"　利用者が定義したファンクション･キーを表わす

"2"　COBOL システムで定義されているファンクション･キーを表わす

"9"　エラーが発生したことを表わす

終了キーとは、ACCEPT 操作を終わらせるためのキーである (たとえば、実行キー) 。
ACCEPT の最後の項目でフィールド･タブ･キーを押すと、終了キーとして働くように設定する構成オプションもある。ファンクション･キーを定義することも構成オプションである。

この状態値に"0" が返されて終了することを、正常終了という。

ACCEPT 文の中に ON EXCEPTION 指定を書くと、CRT 状態キーの値が" 0" 以外であった場合、指定した無条件命令が実行される。

CRT 状態キー2: データ名-1 の 2 番目のバイトは、状態キー2 である。ここには、ACCEPT 操作を終了させた条件のさらに詳細を示すコードが記録される。状態キー2 の形式と値は CRT 状態キー1 の値によって変わる。その具体的な値を下の表 6-2 に示す。

表 6-2: CRT 状態キー1 と 2 の有効な組合せ

キー 1	キー 2		意味
	形式	値	
0	PIC 9 DISPLAY	0	操作員が終了キーを押した
0	PIC 9 DISPLAY	1	最後の項目の外に出る自動スキップ
1	PIC 99 COMP	0-127	ファンクション･キー番号
2	PIC 99 COMP	0-26	ファンクション･キー番号
9	PIC 99 COMP	0	画面内に項目が何もない

ファンクション･キー番号については、COBOL システムのマニュアル中の画面処理およびユーザインターフェイスに関する記述を参照。

CRT 状態キー3: データ名-1 の 3 番目のバイトは、状態キー3 である。これは PICTURE 99 COMP-X または PICTURE 99 COMP として定義する。 (NOIBMCOMP 指令を指定して)。
CRT 状態キー1 と CRT 状態キー2 がともにゼロであるならば、ACCEPT 操作を終わらせたキーそのままのキーボード･コードが CRT 状態キー3 に記録される。これ以外では、CRT 状態キー3 の内容はどうなるかわからない。

単一のキーではなく、いくつかのキーを連続して打つことによって 1 つの機能が実行されるように設定されている場合、最初に打たれたキーに対するコードだけが返される。

6.2.4 リポジトリ段落

リポジトリ段落では、環境部の範囲内で使用できるクラス名、インターフェイス名、属性名、およびプログラムプロトタイプ名を指定できる。さらに、FUNCTION という語を指定せずに使用できる組み込み関数名を宣言することもできる。

一般形式

```
REPOSITORY.
[ { クラス指定
    インターフェイス指定
    プログラム指定
    属性指定
    関数指定 } ... ]
```

ここで クラス指定 は:

```
CLASS クラス名-1 [AS 定数-1 ] [ EXPANDS クラス名-2 USING { クラス名-3
                                                          インターフェイス名-1 } ]
```

ここで インタフェース指定 は:

```
INTERFACE インターフェイス名2 [AS 定数-2 ] [ EXPANDS インターフェイス名-3 USING { クラス名-4
                                                                                  インターフェイス名-4 } ]
```

ここで プログラム指定 は:

```
PROGRAM プログラムプロトタイプ名-1 [AS 定数-3 ]
```

ここで 属性指定 は:

```
PROPERTY 属性名-1 [AS 定数-4 ]
```

ここで 関数指定 は:

書き方1（利用者定義）

```
FUNCTION 関数プロトタイプ名-1 [AS 定数-5 ]
```

書き方2（組み込み）

```
FUNCTION { { 組み込み関数名-1 }...
           ALL                   } INTRINSIC
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
   - RDFPATH - リポジトリ･ファイル用の登録集の所在を指定する。
   - ACTUAL-PARAMS - パラメータ化されたクラスまたはインターフェイスからのクラスまたはインターフェイスの作成に使用されるパラメータを指定する。

構文規則

1. リポジトリ段落中にクラス名-1、インターフェイス名-2、属性名-1、またはプログラムプロトタイプ名のいずれかを2回以上指定する場合、その名前での指定はすべて同じでなければならない。

2. 定数-1、定数-2、定数-3、定数-4、および定数-5 は文字定数でなければならず、表意定数であってはならない。
3. CLASS 指定を書く場合、クラス名-2、クラス名-3、インターフェイス名-1 は下記のどちらかでなければならない。
    a. それを含むクラス定義またはインターフェイス定義のパラメータ
    b. クラス名－1が定義されているのと同じリポジトリ段落中に定義されている
4. このリポジトリ段落を含むクラス定義のクラス名段落の USING 指定またはこのリポジトリ段落を含むインターフェイス定義のインターフェイス名段落の USING 指定の中にクラス名-1 を指定した場合、EXPANDS 指定を書いてはならない。
5. インターフェイス指定を書く場合、クラス名-4、インターフェイス名-3、インターフェイス名-4 は下記のどちらかでなければならない。
    a. それを含むクラス定義またはインターフェイス定義のパラメータ
    b. インターフェイス名-2 が定義されているのと同じリポジトリ段落中に定義されている
6. このリポジトリ段落を含むクラス定義のクラス名段落の USING 指定またはこのリポジトリ段落を含むインターフェイス定義のインターフェイス名段落の USING 指定の中にインターフェイス名-2 指定した場合、EXPANDS 指定を書いてはならない。
7. プログラムプロトタイプ名-1 はいずれも、プログラムプロトタイプの名、この翻訳集団中ですでに指定されたプログラム定義、または以下のいずれかとする。
    a. 定数-3 が指定されていない場合は、プログラム-プロトタイプ名-1 のプログラムの外部リポジトリにその情報があるものとする。
    b. 定数-3 指定されている場合は、定数-3 のプログラムの外部リポジトリにその情報があるものとする。
8. 関数プロトタイプ名-1 はいずれも、関数プロトタイプの名、この翻訳集団中ですでに指定された関数定義、または以下のいずれかとする。
    a. 定数-5 が指定されていない場合は、関数プロトタイプ名-1 の関数の外部リポジトリにその情報があるものとする。
    b. 定数-5 指定されている場合は、定数-5 の関数の外部リポジトリにその情報があるものとする。
9. 組み込み関数名-1 は、このリポジトリ段落の範囲内にある利用者定義語として指定してはならない。
10. ALL が関数指定の組み込み形式に指定されている場合は、組込関数の名はいずれも、このリポジトリ段落の範囲内にある利用者定義語として指定してはならない。
11. 指定したクラス名-1 がこのリポジトリ段落を含むクラス定義の名前である場合、クラス名-1 を参照するとそのクラス定義を参照することになり、このクラス指定子は無視される。
12. 指定したインターフェイス名-2 がこのリポジトリ段落を含むインターフェイス定義の名前である場合、インターフェイス名-2 を参照するとそのインターフェイス定義を参照することになり、このインターフェイス指定子は無視される。
13. 指定したプログラムプロトタイプ名-1 が、このリポジトリ段落の指定を含むプログラム定義の名、またはそれを含むプログラムの名である場合は、プログラムプロトタイプ名-1 は、名づけられたプログラム定義を参照し、このプログラム指定は無視される。
14. 指定した関数-プロトタイプ名-1 が、このリポジトリ段落の指定を含む関数定義の名である場合は、関数-プロトタイプ名-1 は、その関数定義を参照し、この関数指定は無視される。
15. クラス指定を書いた場合、

a. 定数-1 を指定したならば、クラスである定数-1 に関する情報が外部リポジトリ中に存在しなければならない。
b. 定数-1 を指定しなかったならば、クラスであるクラス名-1 に関する情報が外部リポジトリ中に存在しなければならない。

16. インターフェイス指定を書いた場合、
    a. 定数-2 を指定したならば、インターフェイスである定数-2 に関する情報が外部リポジトリ中に存在しなければならない。
    b. 定数-2 を指定しなかったならば、インターフェイスであるインターフェイス名-2 に関する情報が外部リポジトリ中に存在しなければならない。
17. PROPERTY 指定を書いた場合、
    a. 定数-3 を指定したならば、属性である定数-3 に関する情報が外部リポジトリ中に存在しなければならない。定数-3 はこの段落中に宣言されたクラスまたはインターフェイスのどれかの一部である。
    b. 定数-3 を指定しなかったならば、属性である属性名-1 に関する情報が外部リポジトリ中に存在しなければならない。属性名-1 はこの段落中に宣言されたクラスまたはインターフェイスのどれかの一部である。

一般規則

1. クラス名-1 はそれが指定されている環境部の範囲内で使用できるクラスの名前である。
2. AS 指定を書いた場合、定数-1、定数-2、定数-3、または定数-5 はそれぞれ、クラス、インターフェイス、プログラム、または関数が操作環境によって認識される名前である。定数-4 は、指名された属性を実現するメソッドとして、操作環境によって認識される名前である。AS 指定を書く必要があるのは、クラスやインターフェイスや属性の名前が利用者語を形成するための規則に従っていないか、または名前の大文字と小文字を区別する場合である。
3. クラス名-3 とインターフェイス名-1 は、クラス名-2 によって参照されるパラメータ化されたクラスのための、実パラメータである。
4. クラス名-4 とインターフェイス名-4 は、インターフェイス名-3 によって参照されるパラメータ化されたインターフェイスのための、実パラメータである。
5. クラス指定子に EXPANDS 指定を書いた場合、パラメータ化されているクラスであるクラス名-2 から、クラス名-1 がクラスとして作成される。クラス指定子の EXPANDS 指定の USING 指定の中のパラメータの数は、クラス名-2 のクラス名段落の USING 指定の中のパラメータの数と同じであるものとする。クラス名-1 用のインターフェイスは、クラス名-2 用に指定されたインターフェイスのパラメータをクラス指定子に指定されたパラメータで置き換えたものである。

   クラスであるクラス名-1 はパラメータ化されたクラスであるクラス名-2 から作成される。そのさい、仮パラメータの各仕様が対応する実パラメータで置き換えられる。その置換は COPY 文と REPLACE 文の処理が終わった後で行われる。

6. コンパイラはクラス名-1 に指定されている情報を外部リポジトリと一緒に使用して、使用するクラスの詳細を決定する。そのクラスに関するリポジトリ情報は、ファイルに格納されていなければならない。そのファイル名はそのクラスを外部化した名前に拡張子 rdf を付けたものでなければならない。RDFPATH 指令を指定した場合には、その

ファイルは指定されたディレクトリのもとに置かれていなければならない。そうでない場合には、.int ファイルおよび.idy ファイルが作成されるローカル･ディレクトリのもとにそのファイルが置かれていなければならない。

7. インターフェイス名-2 はそれが指定されている環境部の範囲内で使用できるインターフェイスの名前である。
8. インターフェイス指定子に EXPANDS 指定を書いた場合、パラメータ化されているインターフェイスであるインターフェイス名-3 から、インターフェイス名-2 がインターフェイスとして作成される。インターフェイス指定子の EXPANDS 指定の USING 指定の中のパラメータの数は、インターフェイス名-3 のインターフェイス名段落の USING 指定の中のパラメータの数と同じであるものとする。インターフェイス名-2 用のインターフェイスは、インターフェイス名-3 用に指定されたインターフェイスのパラメータをインターフェイス指定子に指定されたパラメータで置き換えたものである。

   インターフェイス interface-名-2 は、指定された各仮パラメータを対応する実パラメータに置き換えることにより、パラメータ化されたインターフェイス interface-3 から作成される。この置き換えは、COPY および REPLACE 文の処理後に実行される。

9. コンパイラはインターフェイス名-2 に指定されている情報を外部リポジトリと一緒に使用して、使用するインターフェイスの詳細を決定する。そのクラスに関するリポジトリ情報は、ファイルに格納されていなければならない。そのファイル名はそのインターフェイスを外部化した名前に拡張子 rdf を付けたものでなければならない。RDFPATH 指令を指定した場合には、そのファイルは指定されたディレクトリのもとに置かれていなければならない。そうでない場合には、.int ファイルおよび.idy ファイルが作成されるローカル･ディレクトリのもとにそのファイルが置かれていなければならない。
10. 属性名-1 はそれが指定されている環境部の範囲内で使用できるオブジェクト属性の名前である。
11. プログラムプロトタイプ名-1 は、これを含む環境部の範囲全体で使用されるプログラムプロトタイプの名とする。
12. プログラムプロトタイプ名-1 のプロトタイプ定義またはプログラム定義が、この翻訳集団中ですでに指定されている場合は、その定義を使用して、プログラムプロトタイプ名-1 が示すプログラムの起動に関する詳細が決定される。そうでない場合は、外部リポジトリを使用して、外部化された名、プログラムプロトタイプ名-1 のプログラムの起動に関する詳細が決定される。この場合、プログラムプロトタイプ名-1 が参照されると、このプログラムが起動する。
13. 関数プロトタイプ名-1 は、これを含む環境部の範囲全体で使用される関数プロトタイプの名とする。
14. 関数-プロトタイプ名-1 のプロトタイプ定義または関数定義が、この翻訳集団中ですでに指定されている場合は、その定義を使用して、関数プロトタイプ名-1 が示す関数の起動に関する詳細が決定される。そうでない場合は、外部リポジトリを使用して、外部化された名、関数プロトタイプ名-1 の関数の起動に関する詳細が決定される。この場合、関数プロトタイプ名-1 が参照されると、この関数が起動する。
15. 関数指定で、組み込み関数名-1 の前に語、FUNCTION をつける必要はない。
16. 関数指定の組み込み形式で ALL を指定した場合は、*手続き部 - 組み込み関数*の章の定義による、各組み込み関数名を指定したものと同様の結果となる。

6.3 入出力節

機能

入出力節では、外部媒体とランタイム要素との間での、データの転送と処理を制御するために必要な情報を取り扱う。

一般形式

INPUT-OUTPUT SECTION.
[ファイル管理段落]
[入出力管理段落]

構文規則

1. (ISO)入出力節はプログラム定義中に指定できる。クラス定義においては、入出力節を指定できる箇所は、ファクトリ定義、オブジェクト定義、メソッド定義の中だけである。インターフェイス定義中に入出力節を指定してはならない。

6.3.1 ファイル管理段落

機能

ファイル管理段落では、各ファイルに名前を付けるとともに、ファイルに関するその他の情報を指定する。

一般形式

6.3.2 ファイル管理記述項

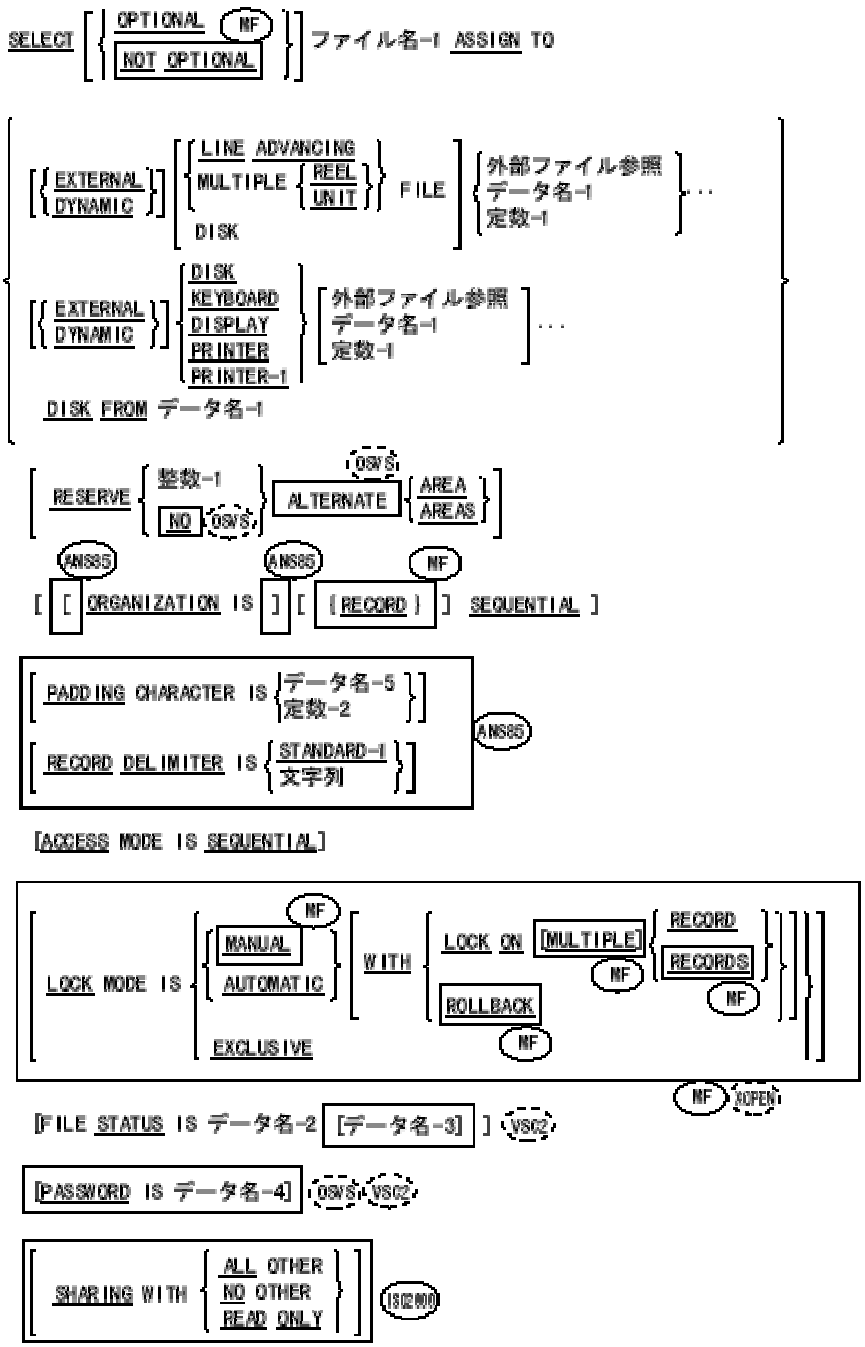
機能

ファイル管理記述項では、ファイルに名前を付けるとともに、ファイルに関するその他の情報を指定する。

(XOPEN)標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、ASSIGN 句中の RECORD DELIMITER 指定と RESERVE 指定と反復記号は明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内では、これらの指定および記号を使用するべきではない。.

一般形式

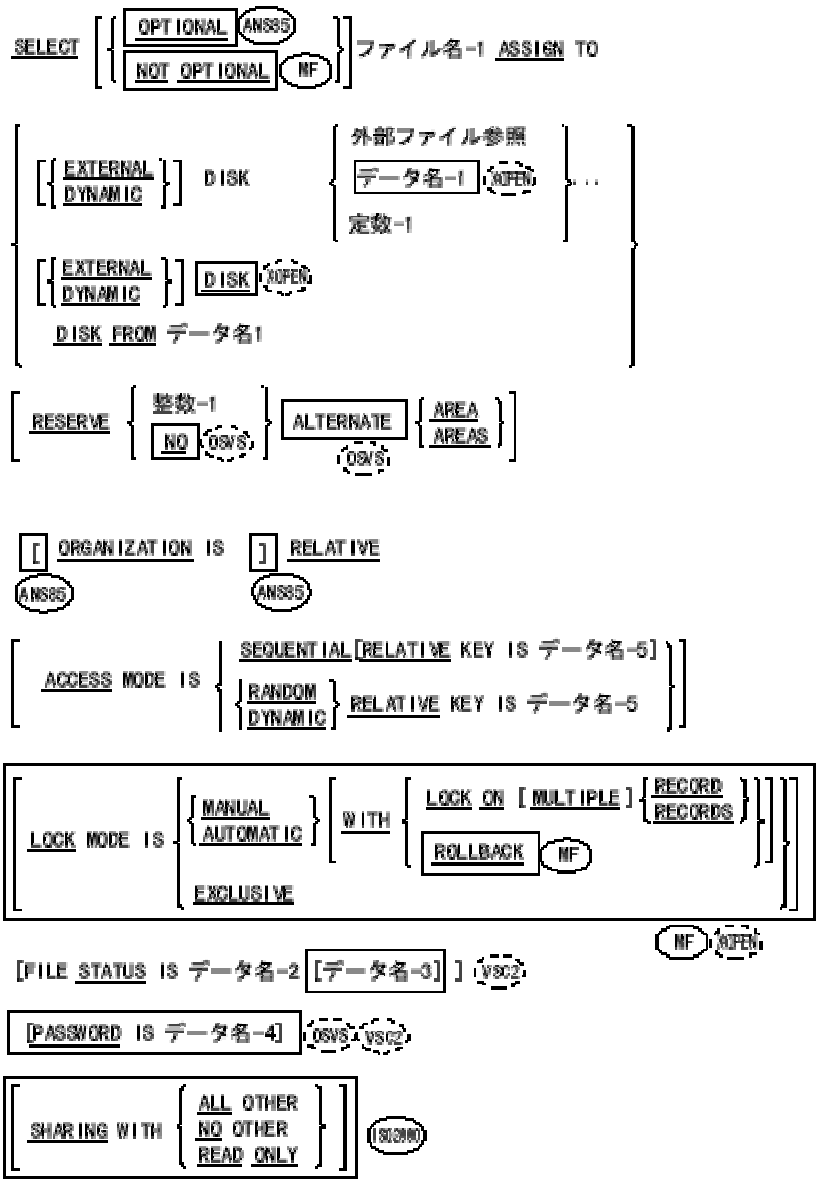
形式 1 (レコード順ファイル)



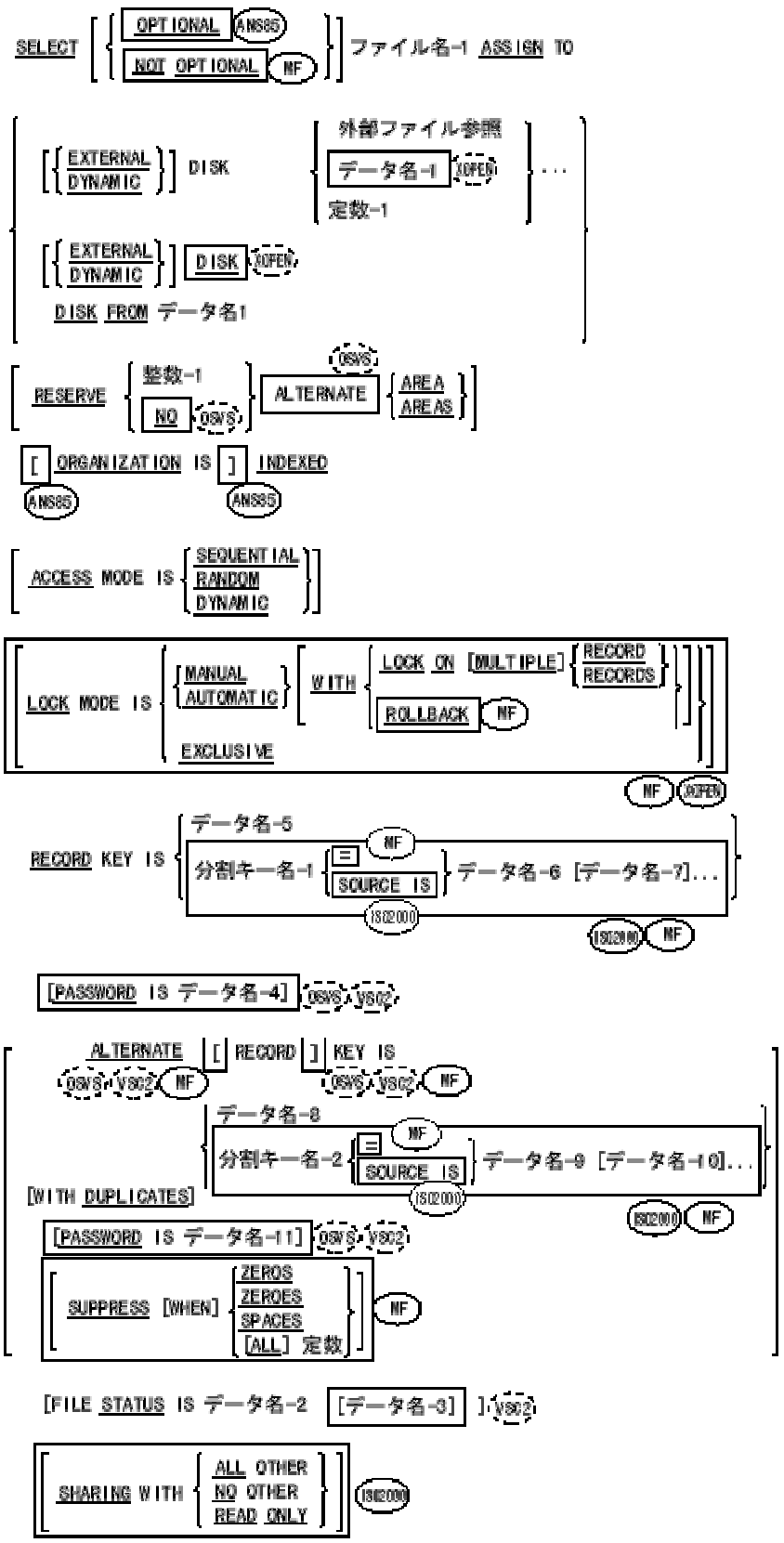
形式 2 (行順ファイル)

```
SELECT [ { OPTIONAL } ] ファイル名-1 ASSIGN TO                                   (NF)
         { NOT OPTIONAL }                                                        (SOPEN)
                                                                                 (ISO2000)
{ [ { EXTERNAL } ] [DISK FILE] { 外部ファイル参照 } ...
    { DYNAMIC  }               { データ名-1       }
                               { 定数-1           }
  [ { EXTERNAL } ] { DISK     } { 外部ファイル参照 } ...
    { DYNAMIC  }   { KEYBOARD } { データ名-1       }
                   { DISPLAY  } { 定数-1           }
                   { PRINTER  }
                   { PRINTER-1 }
  DISK FROM データ名-1                                                           }
[  [ ORGANIZATION IS ] [ LINE ] SEQUENTIAL ]
[ACCESS MODE IS SEQUENTIAL]
                                   (NF)
[ LOCK MODE IS { MANUAL   } [ WITH { LOCK ON [MULTIPLE] { RECORD  } } ] ]
               { AUTOMATIC }       { ROLLBACK           { RECORDS } } ]
               { EXCLUSIVE }                                          ]
[FILE STATUS IS データ名-2]
[PASSWORD IS データ名-4]
[ SHARING WITH { ALL OTHER } ]                                                   (ISO2000)
               { NO OTHER  }
               { READ ONLY }
```

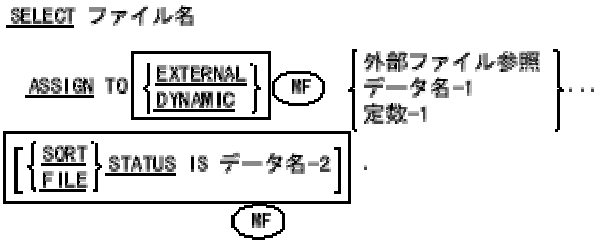
形式 3 (相対ファイル)


SELECT
OPTIONAL
ANS85
NOT OPTIONAL
NF
ファイル名-1 ASSIGN TO
EXTERNAL
DYNAMIC
DISK
外部ファイル参照
データ名-1
NOPEN
定数-1
DISK FROM データ名1
RESERVE
整数-1
NO
OSVS
ALTERNATE
AREA
AREAS
ORGANIZATION IS
RELATIVE
ACCESS MODE IS
SEQUENTIAL
RELATIVE KEY IS データ名-5
RANDOM
LOCK MODE IS
MANUAL
AUTOMATIC
WITH
LOCK ON
MULTIPLE
RECORD
RECORDS
ROLLBACK
EXCLUSIVE
FILE STATUS IS データ名-2
データ名-3
VSC2
PASSWORD IS データ名-4
SHARING WITH
ALL OTHER
NO OTHER
READ ONLY
ISO2001

形式 4 (索引ファイル)



形式 5 (整列併合ファイル)



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - ASSIGN - 定数、データ名、 または外部ファイル参照を評価する方法を決定する。
    - SEQUENTIAL - ORGANIZATION IS SEQUENTIAL として定義されているファイルが LINE SEQUENTIAL または RECORD SEQUENTIAL のどちらであるかを判定する。

構文規則

すべての形式 (すべてのファイル)

1. ファイル管理記述項の冒頭には、SELECT 句を指定する。SELECT 句の後ろに続く他の句は、どんな順序で書いてもよい。
2. データ部に記述する各ファイルには、ファイル管理段落において一度だけファイル名を付ける。ファイル管理記述項に指定した各ファイルに対応して、同じファクトリ、メソッド、オブジェクト、またはプログラムのデータ部のファイル記述項を書かなければならない。
3. ACCESS MODE 句を指定しないと、暗黙的に ACCESS MODE IS SEQUENTIAL を指定したものとみなされる。
4. (MF)データ名-1 は、データ部の中で英数字データ項目または集団項目として宣言できる。この場合、そのファイルの外部名を保持できるだけの十分な長さをとる必要がある。 データ名-1 を原始要素の中で明示的に宣言しないと、COBOL システムが暗黙的に英数字データ項目として宣言する。その長さは、ファイル名の長さとして許される最大の長さがとられる。FROM 指定の中にデータ名-1 を書く場合は、データ部の中で明示的に宣言しておかなければならない。
5. (MF)NOT OPTIONAL 指定は、入出力両用モードでファイルを開く場合にだけ効力をもつ。
6. (MF)外部ファイル参照、データ名-1、定数-1 はファイルの外部名を指定する。EXTERNAL 指定または DYNAMIC 指定を書く場合、定数-1 を指定してはならない DYNAMIC 指定を書く場合、外部ファイル名に指定した語は外部ファイル参照として解釈される。
7. データ名-2 は、データ部の中で、2 文字の英数字データ項目

   (OSVS)(VSC2)(MF)または用途が表示用の 2 文字の数字データ項目

として定義しておく。データ部のファイル節または

MF 局所記憶節

内では定義できない。

8. VSC2 データ名-3 は、作業場所節または連絡節の中で 6 バイトの集団項目として定義しておく。
9. OSVS VSC2 データ名-4 は、作業場所節の中で英数字データ項目として定義しておく。
10. すべてのデータ名は、修飾できる。

形式 1 (レコード順ファイルのみ)

11. ANS85 定数-2 は、1 文字の文字定数とする。
12. ANS85 データ名-5 は修飾してもよい。これはデータ部の中で、1 文字の英数字データ項目として定義しておく。通信節またはファイル節または報告書節の中では定義できない。
13. ANS85 文字列は、予約語または利用者名または定数であってはならない。

形式 1 および 2 (レコード順ファイルおよび行順ファイル)

14. ORGANIZATION 句を指定しないと、順編成であるものと想定される。
15. MF KEYBOARD は、操作卓入力を意味する。
16. MF DISPLAY は、操作卓出力を意味する。
17. MF PRINTER はシステムの主たる印刷装置を指す。
18. MF PRINTER-1 はシステムの 2 番目の印刷装置を指す。

形式 3 (相対ファイル)

19. START 文の作用対象に相対ファイルを指定する場合は、そのファイルに関する RELATIVE KEY を指定しておく。
20. データ名-5 は、該当するファイル名のレコード記述項の中では指定できない。
21. データ名-5 は、符号なしの整数データ項目として定義する。

形式 4 (索引ファイル)

22. ISO/IEC MF 分割キー名は、該当するファイル名のレコード内のいくつかのデータ項目をつなげたものである。これは START 文または READ 文の中でだけ使用できる。
23. データ名-5 とデータ名-8 のデータ項目

ISO/IEC MF および分割キー名-1 と分割キー名-2 によって参照されるデータ名

は、該当するファイルのレコード記述項の中で定義しておく。

OSVS VSC2 MF これらのデータ項目は、どの項類のデータとして定義してもよい。しかし、SELECT 句内で指定したファイル > に対する入出力文に関しては、キーは必ず英数字項目として扱われる。

24. データ名-5 とデータ名-8 のデータ項目

    SOURCE MF および分割キー名-1 と分割キー名-2 によって参照されるデータ名

    に可変長項目を指定することはできない。詳細は、*データ部 - ファイルおよびデータ記述*の章の *OCCURS(反復)句*節を参照。

25. データ名-8 は、左端の文字位置がデータ名-5 により参照される項目、または同じファイルについて指定された他のデータ名-8 により参照される項目の、左端の文字位置に対応する項目を参照してはならない。
26. 索引ファイルが変数長レコードを含む場合は、データ名-5 およびデータ名-8、

    SOURCE MF および分離キー名-1 および分離キー名-2 が参照するデータ名すべて

    は、そのレコードの最初の x バイトに含まれる。この x は、ファイルに指定された最小レコードサイズの値である。*データ部 - ファイルおよびデータ記述* の章の *RECORD(レコードの大きさ)句*節を参照。

27. OSVS VSC2 PASSWORD 句を指定する場合、対応する RECORDKEY 句または ALTERNATE RECORD KEY 句の直後に書く。

形式 5 (整列併合ファイル)

27. データ部に記述する各整列ファイルまたは併合ファイルには、ファイル管理段落で一度だけファイル名を付ける。ファイル管理記述項に記述した各整列ファイルまたは併合ファイルに関して、整列併合ファイル記述項を記述する。
28. ファイル名は整列ファイルまたは併合ファイルを表わすため、ファイル管理段落中のファイル名の後ろには ASSIGN 句だけを記述することができる。

一般規則

すべての形式 (すべてのファイル)

1. ASSIGN 句は、ファイル名-1 が表わすファイルを記憶媒体に対応付ける。最初に記述したファイル割当てだけが効力をもつ。

   OSVS VSC2 MF 1 つの ASSIGN 句内にファイル割当てをいくつか記述した場合、最初のもの以外は注記になる。

2. RESERVE 句を使用すると、利用者は必要な入出力領域の数を指定できる。

(ANS85)使用しているオペレーティングシステムに付随するマニュアルに別途指定されていないかぎり、RESERVE 句は注記になる。

3. ORGANIZATION 句は、ファイルの論理構造を指定する。ファイル編成はファイルが作成されるときに確立され、後から変更することはできない。
4. FILE STATUS 句を指定すると、明示的または暗黙的に該当ファイルを対象とする文が実行されるたびに、ランタイムシステムからデータ名-2 のデータ項目に値が返される。この値はその文を実行した結果の状態を示している。(*手続き部*の章の*入出力状態*節を参照。)

   (MF)データ名-3 は、指定しても注記になる。

5. (MF)PASSWORD 句は注記になる。
6. (MF)ASSIGN 句内の予約語 DYNAMIC は、定数-1 またはデータ名-1 の値が外部ファイル記憶環境におけるファイル名であることを示す。
7. (MF)ASSIGN 句内の予約語 EXTERNAL は、指定したファイルが外部環境においては外部ファイル参照によって識別されることを示す。それを通じて、指定したファイルがさらに外部ファイル記憶環境名に対応付けられることもある。(ある操作環境での外部ファイル名の設定については、ファイル処理に関する COBOL のマニュアルを参照。)

   (MF)外部ファイル参照中に文字"-" が含まれていると、最後の"-" の後ろに続く部分の名前だけが、外部環境においてファイルを識別するために使用される。

8. (MF)外部ファイル参照かデータ名-1 か定数-1 を指定せずに DISK を指定するか、FROM を指定せずに DISK を指定するかすると、そのファイルはディスク・ファイルでその名前はファイル記述中の VALUE OF FILE-ID に指定されることを表わす。このファイル記述中に VALUE OF FILE-ID 句が含まれていない場合は、そのディスク・ファイルの名前は、ファイル名-1 (「内部ファイル名」) と同じであると想定される。
9. (MF)必要語 DISK, KEYBOARD, DISPLAY, PRINTER, PRINTER-1 のどれかの後ろに外部ファイル参照かデータ名-1 か定数-1 が続く場合、その必要語は無視される。
10. (MF)DISK FROM を指定すると、そのファイルはディスク・ファイルでありディスク上のファイル名はデータ名-1 の値であることを表わす。しかし、そのファイルに対して OPEN 文を実行すると、データ名-1 はすべて空白とされて、そのディスク・ファイルの名前はファイル名-1 (「内部ファイル名」) と同じであると想定される。
11. OPTIONAL 指定は、入力モードか入出力両用モードか拡張モードで開くファイルに対してだけ適用できる。ランタイム要素を実際に実行するときに存在するとは限らないファイルに対して、これを指定する必要がある。
12. (SOMM)(MF)SHARING 句では、ファイルに適用する共有モードを指定する。ただし、この句は OPEN 文の SHARING 句によって上書きされることがある。この句ではまた、レコード・ロックが有効であるか否かを指定する。詳しくは、*翻訳集団の概念*の章の*共有モード* 節で指定する。

形式 1、3 および 4 (レコード順ファイル、相対ファイル、および索引ファイル)

13. (MF)LOCK MODE 句は該当するファイルに適用するロック方式を指定するものであり、書いても書かなくてもよい。

    (MF)この句を省略した場合、ファイルを開くと、そのファイルは排他モードにおかれる。ただし、入力モードで開いた場合を除く。

    (MF)LOCK MODE IS EXCLUSIVE を指定すると、ファイルを開いた実行単位はそのファイル全体の鍵を取得する。

    (MF)LOCK MODE IS AUTOMATIC または LOCK MODE IS MANUAL を指定すると、開いたファイルは実行単位の間で共有される。ただし、出力モードで開いたファイルおよび拡張モードで開いた索引ファイルと相対ファイルは、常に排他モードにおかれる。

14. (MF)ROLLBACK 句を指定すると、更新取消し機能を備えた COBOL システムではトランザクションのロギングが行われる。システムでの実現方法については、ファイル処理に関する COBOL のマニュアルを参照。

    (MF)ROLLBACK 句を指定すると、WITH LOCK ON MULTIPLE RECORDS が自動的に発効される。

15. (MF)WITH LOCK ON RECORD 句は、ファイル中の 1 レコードだけをロックすることを指定する。

    (MF)WITH LOCK ON MULTIPLE RECORDS 句は、ファイル中の複数のレコードをロックすることを指定する。複数のレコードをロックする必要があるときは、この句を指定しなければならない。

16. (MF)LOCK MODE IS AUTOMATIC WITH LOCK ON RECORD を指定すると、該当するファイルを対象にした READ 文を実行したときに、該当する 1 レコードがロックされる。その後でそのファイルに入出力操作を行うと、そのレコードはロックを解除される。ただし、START 文は例外である。

    (MF)LOCK MODE IS AUTOMATIC WITH LOCK ON MULTIPLE RECORD を指定すると、該当するファイルを対象にした READ 文を実行したときに、該当する 1 レコードがロックされる。いったんロックされたレコードは、CLOSE, COMMIT, ROLLBACK, UNLOCK のどれかの文を実行するまで、または個々のロックが DELETE 文で削除されるまで解除されない。

17. (MF)LOCK MODE IS MANUAL WITH LOCK ON RECORD を指定すると、該当するファイルを対象にした WITH LOCK 指定を伴う READ 文を実行したときにだけ、該当する 1 レコードがロックされる。その後で、そのファイルに入出力操作を行うと、そのレコードはロックを解除される。ただし、START 文は例外である。

    (MF)LOCK MODE IS MANUAL WITH LOCK ON MULTIPLE RECORDS を指定すると、該当するファイルを対象にした READ WITH KEPT LOCK 文を実行したときに、該当

する 1 レコードがロックされる。いったんロックされたレコードは、CLOSE, COMMIT, ROLLBACK, UNLOCK のいずれかの文を実行するまで解除されない。

18. (MF)ロックモードに MANUAL または AUTOMATIC を指定し WITH LOCK ON MULTIPLE RECORDS を指定しないと、単一レコードをロックするものと想定される。
19. (MF)EXTERNAL と定義したファイルに対して、ASSIGN 指令または必要語を用いて SELECT/ASSIGN 文中でデータ名-1 をファイル名に割り当てるか、あるいはファイル記述項の VALUE OF 句の形式 2 を用いてファイル名を割り当てる場合、下記の規則に従う。
    a. そのファイルを参照するすべての原始要素プログラム中に物理ファイル名を含めるには、同じ名前の一意名を使用する。
    b. 物理ファイル名を保持する一意名の定義にも、EXTERNAL 属性を含める。

    (MF)この規則に反していても、コンパイル時にはそのことは検出されない。しかし、実行単位中のランタイム要素 が実行時にこの規則から外れると、結果は保証されない。

    (MF)レコード･ロックについての詳細は、ファイル処理に関する COBOL のマニュアルを参照。

形式 1 (レコード順ファイル)

20. CLOSE REEL 文または CLOSE UNIT 文を用いてファイルを閉じることができるかまたはそうする意図がある場合、MULTIPLE REEL 指定または MULTIPLE UNIT 指定を書く。
21. ファイル中のレコードは、先行･後行レコード関係によって規定される順序で呼び出される。この関係はファイルを作成または拡張したときに、WRITE 文の実行によって確立されたものである。
22. (MF)LINE ADVANCING FILE を指定すると、印刷装置に適したファイルが作成される。このファイルの冒頭には、復帰文字が置かれる。各レコードを書き出す際の改行指定には、省略時解釈として AFTER ADVANCING 1 LINE が設定される。形式についての詳細は、ファイル処理に関する COBOL のマニュアルを参照。
23. (ANS85)(MF)PADDING CHARACTER 句は注記になる。

    (ANS85)関連するファイル結合子 (file connector) 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する実行単位中の PADDING CHARACTER 句の指定内容は、すべて同じにする。データ名-5 が外部のものである場合、外部データ項目を参照する。

    (ANS85)(MF)RECORD DELIMITER 句は注記になる。

形式 2 (行順ファイル)

24. (MF)(XOPEN)明示的にまたは暗黙的に LINE SEQUENTIAL ORGANIZATION を指定すると、ファイルは固定長のレコードから構成され、その各レコードに 1 行分のデータが記録されるものとされる。このレコードが格納される際には、後行の空白は削除される。1 行の定義はオペレーティングシステムによって異なる。復帰および改行のどちらか

一方または両方によって 1 行の末尾を区切るものもあれば、固定長のレコードとして余白に充てん文字を詰めるものもある。どちらにしても、使用している COBOL システムは、そのオペレーティングシステムのエディタ･ソフトウェアと互換性のあるファイルを必ず作成する。

25. (MF)(OPEN)LOCK MODE IS 句は注記になる。

形式 3 (相対ファイル)

26. 呼出し法が順呼出しのとき、ファイル中のレコードは現存するレコードの相対レコード番号の昇順に呼び出される。
27. 呼出し法が乱呼出しのとき、RELATIVE KEY データ項目の値が呼び出されるレコードを示す。
28. 相対ファイル中に格納されているすべてのレコードは、相対レコード番号によって一意に識別される。この相対レコード番号は、ファイル中のレコードの論理的な順序も示す。つまり、最初の論理レコードの相対レコード番号は 1 であり、以降 2、3、4、...というように続く。
29. データ名-5 のデータ項目は、利用者とオペレーティングシステムとの間で相対レコード番号を情報を取り交わすために使用される。

形式 4 (索引ファイル)

30. 呼出し法が順呼出しのとき、ファイル中のレコードは、指定されたレコードキーの値の昇順に呼び出される。
31. 呼出し法が乱呼出しのとき、レコードキー･データ項目の値が呼び出されるレコードを示す。
32. RECORD KEY 句は、該当するファイルの主レコードキーを指定する。ファイル中のレコードの間で、主レコードキーの値は一意にする。主レコードキーは、索引ファイルからレコードを呼び出す経路を形成する。
33. ファイルが 1 つ以上のレコード記述項をもつ場合、データ名-5

    (SOURCE)(MF)および分割キー名-1 に参照されるどのデータ名

    も、これらのレコード記述項の 1 つにだけ記述されている必要がある。どれかのレコード記述項にあるデータ名-5

    (SOURCE)(MF)および分割キー名-1 に参照されるデータ名のどれか

    によって参照される同一の文字位置は、そのファイルにある他のすべてのレコード記述項のキーとして、暗黙的に参照される。

34. (ANS85)関連するファイル結合子が拡張ファイル結合子である場合、そのファイル結合子に対応する実行単位中のすべてのファイル記述項は、対応するレコード中で、同じ相対的な位置にあるデータ名-1 に対して、同じデータ記述項を記述する。
35. ALTERNATE RECORD KEY 句は、該当するファイルの副レコードキーを指定する。副レコードキーは、索引ファイルからレコードを呼び出す代替経路を形成する。
36. データ名-5

(MF)または分割キー名-1

およびデータ名-8

(MF)または分割キー名-2

のデータ記述ならびにレコード内でのそれらの相対位置は、ファイルが作成されたときと同じにする。また、副レコードキーの数も、ファイルが作成されたときと同じにする。このチェックは環境によっては構成可能である。("ファイル処理のためのプログラマガイド"の"呼出し可能ファイルハンドラ"に記述されている、呼出し可能ファイルハンドラ構成ファイル中の KEYCHECK 属性を参照。)

37. DUPLICATES を指定すると、ファイル中の該当するレコードキーの値がレコード間で重複してもよいことを意味する。DUPLICATES を指定しない場合、該当するレコードキーの値は、ファイル中のすべてのレコードにわたって一意にする。
38. 重複する値をもつレコードを探し出すには、同じ値をもつ最初のレコードから順呼出しで読み込んでいく。
39. (MF)SUPPRESS 句では、副指標 (alternate index) にスパースキー (sparse key) を使用することを要求する。スパースキーは呼出し可能ファイル・ハンドラ (Callable File Handler) を通じてのみ利用可能であり、すべてのシステムで利用できるとはかぎらない (ファイル・ハンドラの詳細については、COBOL システムのマニュアルを参照)。
40. (SOURCE)(MF)分割キー名-1 は、データ名-6 および、発生するデータ名-7 すべての、指定された順での連結から成るレコードキーを定義する。分割キー名-2 は、データ名-9 および、発生するデータ名-10 すべての、指定された順での連結から成るレコードキーを定義する。

形式 5 (整列併合ファイル)

40. ファイル管理記述項において、整列ファイルまたは併合ファイルに名前を付け、それと記憶媒体との関係を指定する。
41. (MF)SORT STATUS 句を指定すると、各整列処理の実行が終わった時点で、データ名-2 によって指定される 2 文字のデータ項目に値が設定される。この値は整列処理の終了結果を示す。以下の状態キーの組み合わせが可能である。状態キー1=0 で状態キー2=0 であれば、正常終了を表わす。状態キー1=3 で状態キー2=0 であれば、パラメータ誤りを表わす。状態キー1=9 であれば、状態キー2 にオペレーティングシステムのエラーメッセージ番号が入れられている。状態キー 1 および 2、および状態の定義について詳しくは、**手続き部**の章の*入出力状態*節を参照。SORT STATUS 句の代わりに FILE STATUS 句を使用できる。しかし、整列ファイルまたは併合ファイルに指定した FILE STATUS 句は、SORT STATUS の同義語として扱われる。
42. ASSIGN 句は注記となる。

6.3.3 入出力管理段落

機能

入出力管理段落では、再開点、ファイル間で共有する記憶領域、複数ファイル･リール中の順編成ファイルの位置を指定する。

(ANS85)入出力管理段落の RERUN 句および MULTIPLE FILE TAPE 句は、ANSI '85 標準では廃要素に分類されており、ANSI 標準の次回の全面改訂の際に、削除される予定である。

(MF)この構文は、この COBOL システムに組み込まれているすべての方言で全面的に使用できる。FLAGSTD 指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。

(XOPEN)標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、RERUN 句および MULTIPLE FILE TAPE 句は明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内ではこれらの句を使用するべきではない。

一般形式

```
I-O-CONTROL.
[ [ RERUN [ON { ファイル名-1 }]
              { 文字列      }
          EVERY { { [END OF] { REEL } } OFファイル名-2
                  { 整数-1 RECORDS   }
                  整数-2 CLOCK-UNITS
                  条件名                 } ] ...
  [ SAME [ RECORD ] AREA FOR ファイル名-3[ファイル名-4]... ] ...
         [ SORT   ]
         [ SORT-MERGE ]
  [MULTIPLE FILE TAPE CONTAINS {ファイル名-5 [POSITION 整数-3]}...] ...
  [APPLY WRITE-ONLY ON{ファイル名-6}...]...   (OSVS)(VSC2)
  [APPLY { CORE-INDEX     } ON ファイル名-7 [ファイル名-8]...]   (OSVS)
         { RECORD-OVERFLOW }
  [APPLY REORG-CRITIERA TO データ名 ON ファイル名-9]...
]
```

構文規則

1. 入出力管理段落は、書いても書かなくてもよい。
2. 整数-1 RECORDS 句または整数-2 CLOCK-UNITS 句を指定するときは、RERUN 句に文字列を指定する。
3. SAME AREA 句においては、SORT と SORT-MERGE は同義語である。
4. SAME SORT AREA 句または SAME SORT-MERGE AREA 句を指定した場合、その中に最低 1 つは整列ファイルまたは併合ファイルであるファイル名を入れなければならない。整列ファイルでも併合ファイルでもないファイルを入れても構わない。

5. SAME 句には 2 つの形 (SAME AREA, SAME RECORD AREA) がある。これらは下記のように使い分ける。

   1 つの原始要素中に複数の SAME 句を指定してもよい。しかし、下記の制約がある。

   a. 1 つのファイル名を複数の SAME AREA (ファイル領域の共用) 句内に指定することはできない。
   b. 1 つのファイル名を複数の SAME RECORD AREA (レコード領域の共用) 句内に指定することはできない。
   c. SAME AREA 句に指定したファイル名を SAME RECORD AREA 句内にも指定する場合、SAME AREA 句内に指定したすべてのファイル名を、SAME RECORD AREA 句内にも指定する。SAME AREA 句内に指定しなかったファイル名を、SAME RECORD AREA 句内に追加指定してもよい。SAME RECORD AREA 句内に指定したすべてのファイルを、任意の時点で開いておくことができる。しかし、1 つの SAME RECORD AREA 句内に指定したファイルは、ある一時点では 1 つしか開くことはできない。
   d. 1 つの整列ファイル名または併合ファイル名を、1 つの SAME AREA 句内に指定してはならない。
   e. 1 つの整列ファイル名または併合ファイル名を、複数の SAME SORT AREA 句または SAME SORT-MERGE AREA 句内に指定してはならない。
   f. SAME AREA 句に指定した整列ファイル名でも併合ファイル名でもないファイル名を、SAME SORT AREA 句または SAME SORT-MERGE AREA 句内にも指定する場合、SAME AREA 句内に指定したすべてのファイル名を SAME SORT AREA 句または SAME SORT-MERGE AREA 句内にも指定する。
6. SAME AREA 句, SAME SORT AREA 句, SAME SORT-MERGE AREA 句, SAME RECORD AREA 句内に指定するファイルは、編成または呼出し法がすべて同じである必要はない。
7. (MF)文字列は、予約語または定数または利用者名であってはならない。
8. END OF REEL/UNIT 句は、ファイル名-2 が順編成ファイルである場合にだけ指定できる。
9. 1 つのファイル名-2 に、複数の RERUN 句を指定してもよい。ただし、その場合は、下記の制約がある。
   a. 整数-1 RECORD 句を複数指定する場合、それらに同じファイル名-2 は指定できない。
   b. END OF REEL 句または END OF UNIT 句を複数指定する場合、それらに同じファイル名-2 は指定できない。
10. CLOCK-UNITS を指定した RERUN 句は、1 つだけ指定できる。
11. (OSVS)(VSC2)入出力管理段落中の各句は、後ろに終止符 (.) を置いてもよい。

一般規則

1. (ANS85)RERUN 句は注記になる。
2. SAME AREA 句は、2 つ以上のファイルの処理用に同じ記憶領域を使用することを指定する。ただし、整列ファイルおよび併合ファイルは対象外である。共有される領域には、対象となるファイルに割り当てられるすべての記憶領域が含まれる。したがって、

SAME AREA 句内に指定したファイルを同時に 2 つ以上開くことは有効ではない。（構文規則 5c を参照）

3. SAME RECORD AREA 句は、現行の論理レコードを処理する間に 2 つ以上のファイルが同じ記憶領域を使用することを指定する。この場合、すべてのファイルを同時に開いておくことができる。SAME RECORD AREA 中の論理レコードは、SAME RECORD AREA 句の中に指定したファイルの内の、出力モードで開いたものの各論理レコード、および最も新しく読み込んだ入力ファイルの論理レコードを表わす。これは、共有領域を暗黙的に再定義していることに相当する。したがって、レコードは左詰めで配置される。
4. (MF) APPLY 句は注記になる。
5. (MF) MULTIPLE FILE 句は注記になる。
6. SAME SORT AREA 句または SAME SORT-MERGE 句を指定した場合、その中に少なくとも 1 つは整列ファイルまたは併合ファイルを入れなければならない。その中に整列ファイルでも併合ファイルでもないファイルを入れても構わない。具体的な記憶領域の共有のされ方は下記のとおり。
    a. SAME SORT AREA 句または SAME SORT-MERGE 句によって、指定した整列ファイルまたは併合ファイルを整列または併合する際に使用される記憶領域が共有される。したがって、ある整列ファイルまたは併合ファイルを整列または併合するために割り当てた任意の記憶領域を、他の整列ファイルまたは併合ファイルを整列または併合するために再利用できる。
    b. 更に、整列ファイルでも併合ファイルでもないファイルに割り当てた記憶領域を、必要に応じて、SAME SORT AREA 句または SAME SORT-MERGE 句の中に指定した整列ファイルまたは併合ファイルを整列または併合するために割り当てることもできる。
    c. 整列ファイルでも併合ファイルでもないファイル同士は、互いに同じ記憶領域を共有するようにはされない。それらのファイル間で記憶領域を共有させたければ、該当原始要素中でそれらのファイルを対象にして SAME AREA 句または SAME RECORD AREA 句を指定する。
    d. この句の中に指定した整列ファイルまたは併合ファイルを使用する SORT 文または MERGE 文を実行する間は、この句の中に指定した整列ファイルでも併合ファイルでもないファイルを開くことはできない。

第 7 章 : データ部

7.1 一般記述

データ部には、実行機能単位 が入力用として受けるデータ、または出力用として計算、作成、生成するデータを記述する。

以下にデータ部内の節の一般形式を示す。ここで、原始要素内で呼ばれる順序が定義される。

一般形式

```
NF                NF
[ [ DATA DIVISION. ]
  NF              NF
  [ FILE SECTION. ]
        [ ファイル記述項 [レコード記述項] ...
          整列併合用ファイル記述項 [レコード記述項] ... ] ...
          報告書ファイル記述項 ]

  [ WORKING-STORAGE SECTION.
        [ レベル77記述項 ]...
          レコード記述項 ] ]

  [ OBJECT-STORAGE SECTION.                  ] NF
        [ レベル77記述項 ]...
          レコード記述項 ] ]

  [ LOCAL-STORAGE SECTION.                   ] NF
        [ レベル77記述項 ]...
          レコード記述項 ] ]

  [ LINKAGE SECTION.
        [ レベル77記述項 ]...
          レコード記述項 ] ]

  [ COMMUNICATION SECTION.
        [通信記述項 [レコード記述項] ...] ... ]

  [ REPORT SECTION.
        [報告書記述項 [報告集団記述項] ...] ... ]

  [ SCREEN SECTION.                          ] NF
        [画面記述項] ...                       ]  NOPEN
```

報告書節については、***言語リファレンス - 追加トピック***の*報告書作成*の章で解説する。他の節は、この章の以下の節で解説する。

7.2 ファイル節

ファイル節の中では、ファイル記述項 (FD) が構造上一番高い位置を占める。FILE SECTION 見出しの後ろにファイル記述項が続く。ファイル記述項は、レベル指示語 (FD) とファイル名と

一連の独立した句から構成される。FD 句には、論理レコードおよび物理レコードの大きさ、ファイルの外部名、ファイルを構成するデータレコードの名前を指定する。項目は終止符を付けて終了する。

SD ファイル記述は、整列ファイルまたは併合ファイルの大きさとデータレコードの名前に関する情報を指定するためのものである。整列ファイルおよび併合ファイルに関しては、利用者が制御できるラベル手続きはない。また、ブロック化および内部記憶のされ方は、SORT 文および MERGE 文に固有である。

ファイル記述項および整列併合ファイル記述項に含まれ得る要素は、*データ部 - ファイルおよびデータ記述*の章で定義されている。

7.2.1 整列併合との関係

整列併合機能単位は、データ部内の表の要素を整列させる機能を備えている。SORT 文および MERGE 文の USING 指定および GIVING 指定の対象とするファイルは、ファイル管理段落中に明示的にまたは暗黙的に、順編成で順呼出し法のファイルとして記述しておく。整列併合ファイル記述中に指定するファイルに対しては、入出力文を実行してはならない。

ANS85 SORT 文および MERGE 文の USING 指定および GIVING 指定の対象とするファイルは、ファイル管理段落中に明示的に、索引編成または相対編成をとるものと記述できる。その呼出し法は、順呼出しにも動的呼出しにもできる。

7.2.2 レコード記述の構造

レコード記述は、個々のレコードの特性を記述するデータ記述項の組から構成される。各データ記述項は、レベル番号と、必要ならばデータ名と、必要に応じて一連の独立した句から構成される。レコード記述は階層構造をとる。したがって、後ろに下位の記述が続くか否かによって、記述する句の形態は大幅に異なることがある。レコード記述の構造は、***COBOL の概念***の章の*レベルの概念*節に定義してある。また、レコード記述に含まれ得る要素については、*データ部 - ファイルおよびデータ記述*の章で定義されている。

7.3 作業場所節

作業場所節 (working-storage section) は、節の見出しで始まる。その後ろに、独立データ項目用のデータ記述項やレコード記述項を書く。データ記述項の構文は、*データ部 - ファイルおよびデータ記述*の章で定義されている。

作業場所節の各レコード名および独立項目名は一意とする。それらは修飾できないためである。レコードに従属するデータ名は、修飾することによって一意にできる場合、一意である必要はない。

ANS85 データ名またはレコード名が参照されることがない場合、修飾して一意にしなくてもよい。

一般形式

```
WORKING-STORAGE SECTION.

    [ レコード記述項
      77レベルデータ記述項 ] ...
```

構文規則

1. 作業場所節は、関数定義またはプログラム定義で指定できる。クラス定義中では、ファクトリ定義またはインスタンス定義で定義しなければならず、メソッド定義で定義してはならない。また、インターフェイス定義で指定してはならない。

7.3.1 独立作業場所 (77 レベル記述項)

作業場所節に置く項目や定数のうち、相互に階層関係がなくさらに細分する必要もないものは、レコードにまとめる必要はない。この場合には、独立基本項目として、別々のデータ記述項に定義する。このレベル番号には特別の 77 という値が割り当てられている。データ記述項の構文は、*データ部 - ファイルおよびデータ記述*の章で定義されている。

各データ記述項には、下記の要素を書く。

- レベル番号 77
- データ名
- 以下のいずれか
    - PICTURE 句
    - USAGE INDEX および以下を指定する USAGE 句

      (VSC2) (MF) USAGE COMPUTATIONAL-1、USAGE COMPUTATIONAL-2、USAGE POINTER、または

      (MF) (COB370) USAGE PROCEDURE-POINTER

      (SO2002) (MF) USAGE OBJECT-REFERENCE

      (MF) USAGE EVENT-POINTER, MONITOR-POINTER、MUTEX-POINTER, SEMAPHORE-POINTER または THREAD-POINTER

- 上記以外のデータ記述句は、必ずしも書く必要はない。必要があれば、項目の記述を完全にするために書けばよい。

7.3.2 作業場所レコード (レコード記述項)

作業場所節に置くデータ項目のうち、相互に階層関係があるものは、レコードにまとめておく。その際、レコード記述の書き方の規則に従う必要がある。ファイル節のレコード記述用のすべての句は、作業場所節のレコード記述にも使用できる。

7.3.3 レコード記述の構造

レコード記述 (record description) は、レコードの特徴を記述する一連のデータ記述項から構成される。各データ記述項 (data description entry) の先頭にはレベル番号を書く。

(ANSI85)必要があれば、

その後ろにデータ名または FILLER 句を書く。さらにその後ろに、必要に応じて、独立の句をいくつか書く。レコード記述は階層構造をとる。したがって、1 つの記述項 (entry) に適用する句は、それに従属する記述項があるかないかによって、かなり違ってくる。レコード記述の構造とレコード記述項内に書ける要素については、***COBOL の概念*** の章の*レベルの概念*節および*データ部 - ファイルおよびデータ記述*の章に説明してある。

7.4 スレッド局所記憶節

(MF)

機能

スレッド局所記憶節は、各スレッドで一意であり、すべての呼出しにおいて持続するデータを記述する。このデータには、このデータの初期値は、VALUE 句 を使用して与え T る。データ記述項の構文は、*データ部 - ファイルおよびデータ記述*の章で定義する。

スレッド局所記憶域は、スレッドごとの作業場所とみなしてもよい。こうすると、ほとんどの再入プログラムの競合問題を解決するときに便利である。多くの場合、ファイル操作を行わないプログラムは、単に作業場所節見出しをスレッド局所記憶節見出しに変更するだけで、完全に再入プログラムにすることができる。

一般形式

```
THREAD-LOCAL-STORAGE SECTION.

    [ record-description          ]
    [ 77-level-description-entry  ]
```

構文規則

1. EXTERNAL 句は使用できない。

一般規則

1. スレッド局所記憶域にある各データ項目はコピーされ、実行中のプログラムに入力される新しいスレッドごとに初期状態に設定される。データ項目を認識できるのは、それ

を実行するスレッドのみである。データ項目は、そのスレッドの実行が終了した地点、またはプログラムに CANCEL 文が実行された時点で、削除される。そうでない場合、そのスレッドで次にそのプログラムが呼ばれた時点では、データ項目は最後に使用されたときの状態である。

2. シングルスレッド環境では、スレッド局所記憶域は作業場所とまったく同じ動作をする。

7.5 オブジェクト記憶節

MF

オブジェクト記憶節は、必要に応じてデータ部中で使用する節で、これにより、クラスオブジェクトデータおよびインスタンスオブジェクトデータを宣言することができる。

形式

```
OBJECT-STORAGE SECTION.
[データ記述項]...
```

構文規則

1. クラスオブジェクトのオブジェクト記憶節で宣言されたデータは、クラスメソッドでのみアクセス可能である。
2. オブジェクトプログラムのオブジェクト記憶節で宣言されたデータは、インスタンスメソッド中でのみアクセス可能である。
3. クラス名段落中で DATA IS RESTRICTED または DATA IS PROTECTED が指定されている場合は、オブジェクト記憶節で宣言されたデータは、下位のクラス中で直接アクセス可能である。そうでない場合、起動するメソッド以外は、このデータにアクセスできない。
4. オブジェクト記憶節で宣言されたデータの構文規則は、連絡節のものと同じである。ただし、ODOSLIDE コンパイラ指令が指定されているときに、OCCURS 句に DEPENDING ON 指令を使用してはならない。

一般規則

1. オブジェクト記憶節で宣言されたデータは、その下位のクラスに継承させることができる。
2. 新しいインスタンスオブジェクトのそれぞれに対し、オブジェクト記憶節で宣言されたデータ項目用の、一意の記憶域が割り当てられる。
3. コンパイラ指令、OOCTRL(+W)が指定されているときは、作業場所節がオブジェクト記憶節として扱われる。

7.6 局所記憶節

OS/390 COBOL MF

機能

プログラムが呼ばれるたびに、局所記憶節の写しが別途作られる。作られた局所記憶節の写しは、そのプログラムが呼ばれている間だけ、存在する。局所記憶節は、再帰型のプログラム呼出しのために特に用意されたものである。

局所記憶節 (local-storage section) は、節の見出しで始まる。その後ろに、独立データ項目用のデータ記述項やレコード記述項を書く。局所記憶節の各レコード名および独立項目名は一意とする。それらは修飾できないためである。レコードに従属するデータ名は、修飾することによって一意にできるならば、一意である必要はない。データ記述項の構文は、*データ部 - ファイルおよびデータ記述*の章で定義されている。

データ名またはレコード名が参照されることがない場合、修飾して一意にしなくてもよい。

一般形式

```
LOCAL-STORAGE SECTION.

    [ レコード記述項
      77レベルデータ記述項 ]
```

構文規則

1. EXTERNAL データは使用できない。
2. GLOBAL データは使用できない。
3. 「CALL ... USING 局所データ項目」という構文は使用できる。
4. 「ENTRY ... USING 局所データ項目」という構文は使用できない。
5. 「PROCEDURE DIVISION ... USING 局所データ項目」という構文は使用できない。
6. 入れ子になったプログラムの中では、LOCAL-STORAGE SECTION 見出しは使用できない。
7. COBOL システムは、プログラムに局所記憶節がある場合にだけプログラムの再帰を行う。局所記憶節内にレコードが存在する必要はない。局所記憶節の見出しがないと、実行時に再帰処理が受け入れられない。詳しくは、***翻訳集団の概念***の章の*共通プログラム、初期プログラム、および再帰プログラム*節を参照。

一般規則

1. DEFAULT-BYTE コンパイラ指令は、局所記憶節内で定義した項目には適用できない。

2. VALUE 句を書くことができるが、注記にとどまる。

   (ISO2002)VALUE 句は、***データ部 - ファイルおよびデータ記述***の章の *VALUE(値)句*節の解説に従い、データ項目を初期化する。

7.7 連絡節

(OSVS)(VSC2)(MF)以下の説明において、手続き部の USING 句に触れている箇所はすべて、ENTRY USING にも当てはまる。

(VSC2)(MF)*手続き部 - SEARCH - WRITE* の章の *SET(設定)文*節に記述されている場合は例外として、

プログラム中の連絡節 (linkage section) が意味をもつのは、呼ぶプログラムから CALL 文に USING 句を指定して、実行用プログラムが呼ばれた場合だけである。

(SOM)(MF)連絡節はメソッドにおいて常に意味を持つ。

呼ばれるプログラムの連絡節に指定した正式なパラメータと戻される項目は、制御が移された時に、その呼ばれるプログラムと呼ぶプログラムの両方から参照される。

(MF)このようなデータは、ファイル節にも作業場所節にも記述できる。

連絡節に記述されているデータ項目に対して、そのプログラムの中では領域は割り当てられない。これらのデータ項目を手続き部で参照すると、呼ぶプログラムの中で実行時に使用している場所を参照することになる。指標名については、そのような対応は付けられない。呼ぶプログラム中の指標名と呼ばれるプログラム中の指標名は、常に別の指標を参照する。

連絡節の中で定義されているデータ項目を、呼ばれるプログラムの手続き部の中で参照できるのは、そのデータ項目が手続き部の見出しの USING 句の作用対象に指定されるか、またはその作用対象に従属するかし、かつ実行用プログラムが USING 句を伴う CALL 文の制御下にある場合だけである。

連絡節の構造は先に述べた作業場所節と同じである。つまり、節の見出しで始まり、その後ろに独立データ項目のデータ記述項やレコード記述項が続く。データ記述項の構文は、*データ部 - ファイルおよびデータ記述*の章で定義されている。

連絡節内の各レコード名および独立項目名は呼ばれるプログラムの中で一意とする。

(ANS85)これらの名は、手続き部の中で参照されていない場合は、一意である必要はない。

連絡節に置くパラメータと戻される項目の指定は、***手続き部***の章の*パラメータと戻り項目の適合*節に記述された内容に従わなければならない。

7.7.1 独立連絡場所

連絡節に置く項目のうち、相互に階層関係のないものは、レコードにまとめる必要はない。この場合には、独立基本項目として、別々のデータ記述項に定義する。データ記述は特別のレベル番号である 77 から始まる。

各データ記述項には、下記の要素を書く。

- レベル番号 77
- データ名
- いかのいずれか。
    - PICTURE 句または
    - USAGE INDEX および以下を指定する USAGE 句

      (VSC2)(MF)USAGE COMPUTATIONAL-1、USAGE COMPUTATIONAL-2、USAGE POINTER,

      (MF)(OB87)USAGE PROCEDURE-POINTER、

      (SQ2003)(MF)USAGE OBJECT-REFERENCE、

      (MF)USAGE EVENT-POINTER、MONITOR-POINTER、MUTEX-POINTER、SEMAPHORE-POINTER、THREAD-POINTER。

- 上記以外のデータ記述句は、必ずしも書く必要はない。必要があれば、項目の記述を完全にするために書けばよい。

7.7.2 連絡レコード

連絡節に置くデータ項目のうち、相互に階層関係があるものは、レコードにまとめる。その際、レコード記述の書き方の規則に従う必要がある。入力レコードまたは出力レコードを記述するために使用する句のすべてを、連絡節でも使用できる。

7.8 画面節

(MF)(XOPEN)

画面節は、ACCEPT 文および DISPLAY 文と併用する画面処理機能を提供する。以下を行うことができる。

- 画面節で定義された画面領域から成る、非スクロール形式の表示。画面節の記述項は、画面記述と呼ばれる。これは、データ記述と似ているが、記憶された領域ではなく、画面項目または画面領域を定義する。
- (MF)非スクロール形式を構成するデータ項目の表示。使用される画面領域の詳細は、関連づけられた ACCEPT 操作および DISPLAY 操作で提供される。

画面節は、形式 4 の ACCEPT 操作または形式 2 の DISPLAY 操作がアクセスする画面の各項目の記述を含む。これらの項目は、画面項目と呼ばれる。画面項目の多くは、画面の一部での項目の配置のみを記述し、明示的に参照されることはない。

画面節は、以下の操作を可能にする構文を含む。

- 項目の厳密な位置の指定
- 指定された位置へのデータのキー入力
- 指定された位置での文字の表示
- 画面属性の定義
- 操作卓機能の制御

この構文は、*データ部 - 画面節*の章で定義する。

データ記述項は、固定方式または自由方式により、数字画面項目または数字編集画面項目となる。利用者インターフェイスについて詳しくは、COBOL システムのマニュアルを参照。

固定方式は、データ記述項を数字画面項目または数字編集画面項目にする際の初期方式である。この方式により、入力されたデータが形式を整えられ、画面に反映される。また、キーが押されるにつれ、項目の仕様の条件に従って、カーソルを動かす。数字以外の文字、"+", "-"、および小数点は拒否される。カーソルが前後に移動する際は、編集項目の挿入記号はとばされる。符号指示子は通常の使用に従って変更される。浮動記号は項目内を左右に移動し、挿入記号は、数字が挿入されれば現れ、削除されれば消える。カーソルが項目の最後の位置に達した後に入力された文字は、最後の文字に上書きされる。

自由方式は、データ記述項を数字画面項目または数字編集画面項目にする際の別の方式である。固定方式が初期方式である。自由法式では、適切な長さの PIC X 項目にデータを入力でき、カーソルが項目を離れたときに初めて、仕様に従ってデータの形式が整えられる。カーソルが項目から離れると、数字、記号、小数点以外の文字はすべて無視される。その後、通常の COBOL での、画面項目または作業場所項目と同じ形式の項目への MOVE の規則に従って、数値の抽出、保管、形式の変更が行われる。この数値は通常、この後で画面に反映される。使用可能な利用者インターフェイスの設定方法の詳細については、COBOL システムのマニュアルを参照。

機能

画面節は、ACCEPT 文および DISPLAY 文と併用する画面処理機能を提供する。

一般形式

```
SCREEN SECTION.
        [画面記述項]...
```

構文規則

1. 画面節は、データ部の最後の節でなければならない。
2. 画面記述項は、画面記述句を含む。画面節で使用できるデータ記述句は、一般形式で示したもののみに制限される。

一般規則

画面節は、形式 4 の ACCEPT 操作または形式 2 の DISPLAY 操作がアクセスする画面の各項目の記述を含む。これらの項目は、画面項目と呼ばれる。画面項目の多くは、画面の一部での項目の配置のみを記述し、明示的に参照されることはない。

第 8 章 : データ部 - ファイルおよびデータ記述

この章では、ファイル記述項およびデータ記述項の全体的な骨組みを解説する。以下に、この 2 つの記述項で使用されるすべての句の説明をアルファベット順に示す。

8.1 ファイル記述項の全体的な骨組み

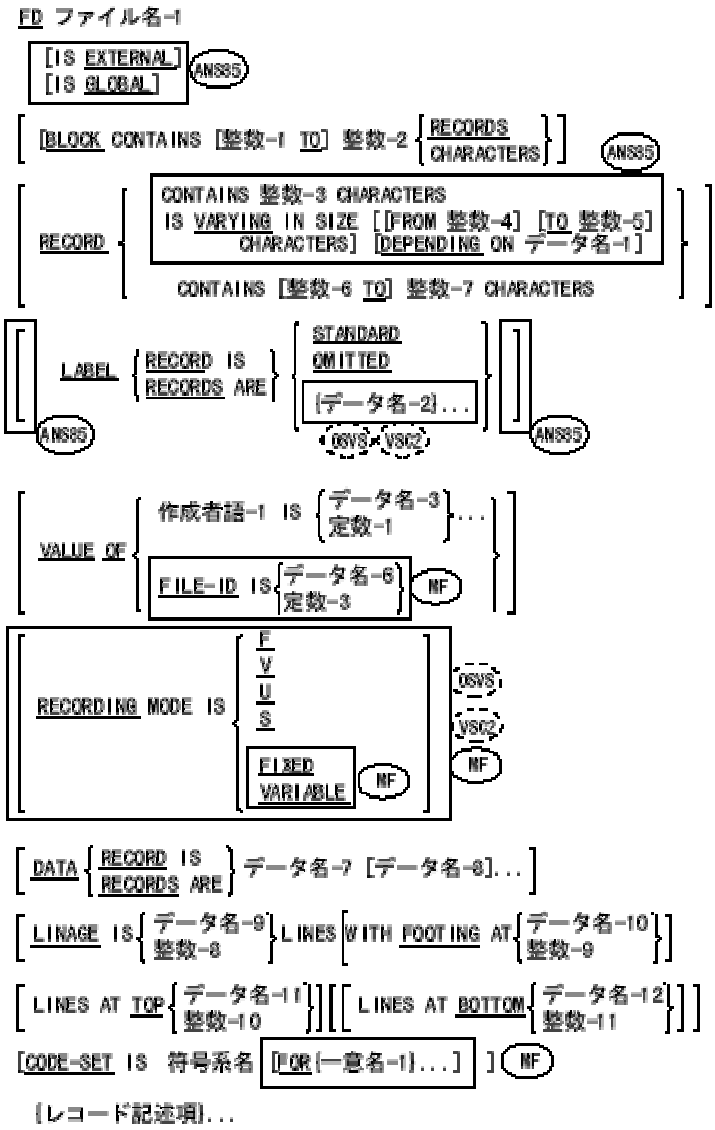
機能

ファイル記述は、当該ファイルに関する物理的構造、識別名、およびレコード名を含む。

(ANS85)ファイル記述は、ファイル結合子、関連づけられたデータレコード、および関連づけられたデータ項目の内部属性および外部属性を決定する。また、ファイル名が局所名と大域名のどちらであるかも決定する。

一般形式

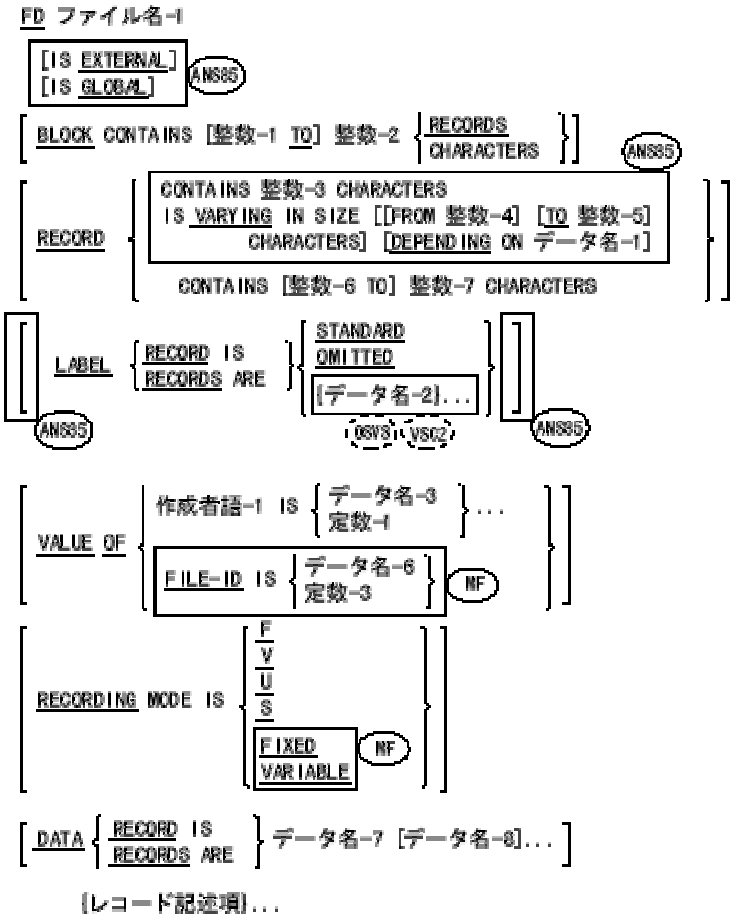
形式 1 (レコード順ファイル)


FD ファイル名-1
[IS EXTERNAL]
[IS GLOBAL]
ANS85
[BLOCK CONTAINS [整数-1 TO] 整数-2 {RECORDS / CHARACTERS}]
ANS85
RECORD
CONTAINS 整数-3 CHARACTERS
IS VARYING IN SIZE [[FROM 整数-4] [TO 整数-5] CHARACTERS] [DEPENDING ON データ名-1]
CONTAINS [整数-6 TO] 整数-7 CHARACTERS
LABEL {RECORD IS / RECORDS ARE}
STANDARD
OMITTED
{データ名-2}...
OSVS VSC2
ANS85
ANS85
VALUE OF
作成者語-1 IS {データ名-3 / 定数-1}...
FILE-ID IS {データ名-6 / 定数-3}
NF
RECORDING MODE IS
F
V
U
S
FIXED
VARIABLE
NF
OSVS
VSC2
NF
[DATA {RECORD IS / RECORDS ARE} データ名-7 [データ名-8]...]
[LINAGE IS {データ名-9 / 整数-8} LINES [WITH FOOTING AT {データ名-10 / 整数-9}]]
[LINES AT TOP {データ名-11 / 整数-10}] [[LINES AT BOTTOM {データ名-12 / 整数-11}]]
[CODE-SET IS 符号系名 [FOR {一意名-1}...]] NF
{レコード記述項}...

形式 2 (行順ファイル)

```
FD ファイル名-1                                                    NF
  [IS EXTERNAL]                                                    NOFEN
  [IS GLOBAL]   ANS85
[ BLOCK CONTAINS [整数-1 TO] 整数-2 { RECORDS }] ANS85
                                    { CHARACTERS }
[ RECORD { CONTAINS 整数-3 CHARACTERS                              }]
         { IS VARYING IN SIZE [[FROM 整数-4] [TO 整数-5]           }
         {        CHARACTERS] [DEPENDING ON データ名-1]            }
         {   CONTAINS [整数-6 TO] 整数-7 CHARACTERS                }
[[ LABEL { RECORD IS } { STANDARD }[ ]]
         { RECORDS ARE } { OMITTED }[ ]
ANS85                   { [データ名-2]... }[ ]  ANS85
                          OSVS  VS02
[ VALUE OF { 作成者語-1 IS { データ名-3 }...                       }]
           {               { 定数-1     }                          }
           { FILE-ID IS { データ名-6 }                             }
           {            { 定数-3     }                             }
[ RECORDING MODE IS { F }                                          }]
                    { V }
                    {   }
                    { FIXED }
                    { VARIABLE }
[ DATA { RECORD IS } データ名-7 [データ名-8]... ]
       { RECORDS ARE }
[CODE-SET IS 符号系名 [FOR {一意名-1}...]]
  {レコード記述項}...
```

形式 3 (相対ファイルおよび索引ファイル)



形式 4 ()整列併合ファイル

```
SD ファイル名-1                                   (OSVS) (VSC2)
[BLOCK CONTAINS [整数-1 TO] 整数-2 {RECORDS   }]
                                   {CHARACTERS}
                                                        (ANS85)
[ RECORD { [ { CONTAINS 整数-3 CHARACTERS                 }  ] }  ]
         {   { IS VARYING IN SIZE [[FROM 整数-4] [TO 整数-5]}     }
         {   {    CHARACTERS] [DEPENDING ON データ名-1]    }     }
         { CONTAINS [整数-6 TO] 整数-7 CHARACTERS                 }
[ LABEL {RECORD IS  } { STANDARD  }  ]  (OSVS)
        {RECORDS ARE} { OMITTED   }  ]  (VSC2)
        {             {[データ名-2]...}
[ DATA {RECORD IS  } [データ名-3]... ]
       {RECORDS ARE}
[ VALUE OF FILE-ID IS {データ名-4} ]  (MF)
                      {定数-1    }
```

構文規則

すべての形式 (すべてのファイル)

1. ファイル名-1 の後の句は、どの順序で書いてもよい。
2. (MF) VALUE OF FILE-ID 句が指定されている場合は、 定数-1 が英数字定数でなければならず、表意定数であってはならない。

形式 1、2 および 3 (レコード順ファイル
(MF) 行順ファイル、
相対ファイル、および索引ファイル)

3. レベル指示語 FD は、ファイル記述の始まりを表しており、FD の後にファイル名が続くものとする。
4. ファイル記述項の後には 1 個以上のレコード記述項が続くものとする。
5. (MF) EXTERNAL 句と THREAD-LOCAL 句を同じファイル記述項に指定することはできない。

形式 4 (整列併合ファイル)

6. レベル指示語 SD は、整列併合ファイル記述の始まりを表しており、SD の後にファイル名が続くものとする。
7. ファイル名の後の句は、どの順序で書いてもよいし、まったく書かなくてもよい。
8. 整列併合ファイル記述項の後には 1 個以上のレコード記述項が続くものとする。 ただし、入出力文はいずれも、このファイルに対して実行できない。

一般規則

形式 1 (レコード順ファイル)

1. 順ファイルに対するファイル記述項に LINAGE 句および EXTERNAL 句が含まれる場合は、LINAGE-COUNTER データ項目が外部データ項目となる。順ファイルに対するファイル記述項に LINAGE 句および GLOBAL 句が含まれる場合は、特殊レジスタ LINAGE-COUNTER が大域名となる。

8.2 データ記述項の全体的な骨組み

機能

データ記述項は、個々のデータ項目の特徴を指定する。

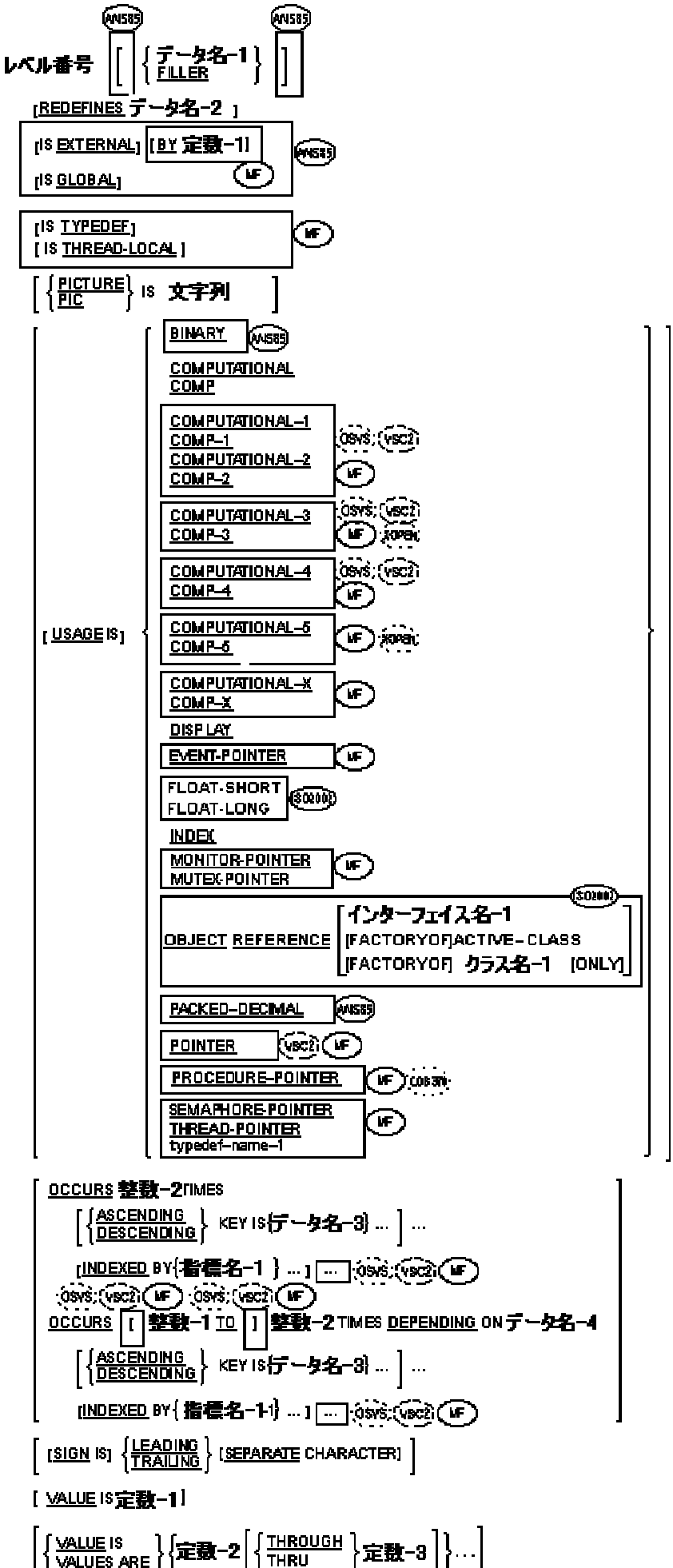
(MF)またはプログラマが定義したデータ型の特徴を記述して、複数のデータ項目の記述を指定するのに使用する。

(ANS85)作業場所節またはファイル節内の 01 レベルのデータ記述項によって、データ･レコードおよびその下位のデータ項目に局所名 (local name) または大域名 (global name) が含まれるか否かが決まる。

(ANS85)作業場所節内の 01 レベルのデータ記述によって、データ･レコードとその下位のデータ項目の内部属性または外部属性が決まる。

一般形式

形式 1


レベル番号
データ名-1
FILLER
ANS85
[REDEFINES データ名-2]
[IS EXTERNAL]
[BY 定数-1]
[IS GLOBAL]
MF
[IS TYPEDEF]
[IS THREAD-LOCAL]
PICTURE
PIC
IS 文字列
[USAGE IS]
BINARY
COMPUTATIONAL
COMP
COMPUTATIONAL-1
COMP-1
COMPUTATIONAL-2
COMP-2
OSVS
VSC2
COMPUTATIONAL-3
COMP-3
XOPEN
COMPUTATIONAL-4
COMP-4
COMPUTATIONAL-5
COMP-5
COMPUTATIONAL-X
COMP-X
DISPLAY
EVENT-POINTER
FLOAT-SHORT
FLOAT-LONG
SO2000
INDEX
MONITOR-POINTER
MUTEX-POINTER
OBJECT REFERENCE
インターフェイス名-1
[FACTORY OF] ACTIVE-CLASS
[FACTORY OF] クラス名-1 [ONLY]
PACKED-DECIMAL
POINTER
PROCEDURE-POINTER
COBOL2002
SEMAPHORE-POINTER
THREAD-POINTER
typedef-name-1
OCCURS 整数-2 TIMES
ASCENDING
DESCENDING
KEY IS データ名-3 ...
[INDEXED BY 指標名-1 ...]
OCCURS [整数-1 TO] 整数-2 TIMES DEPENDING ON データ名-4
[SIGN IS] LEADING TRAILING [SEPARATE CHARACTER]
[VALUE IS 定数-1]
VALUE IS
VALUES ARE
定数-2
THROUGH
THRU
定数-3

形式 2

```
66 データ名-1 RENAMES データ名-2 [ { THROUGH } データ名-3 ] .
                                  { THRU    }
```

形式 3

```
88 条件名 { VALUE  IS } { 定数-2 [ { THROUGH } 定数-3 ] } ...
          { VALUES ARE }         { THRU    }
    [WHEN SET TO FALSE 定数-4] .
                        (MF)
```

形式 4

```
                                                                    (MF)
78  定数名 VALUE IS { 整数-5                    } [ { + } { 整数-1                    } ]
                    { NEXT                      }   { - }   { NEXT                    }
                    { START OF データ名-1       }   { * }   { START OF データ名-3     }
                    { LENGTH OF データ名-2      }   { / }   { LENGTH OF データ名-4    }
                                                    { AND }
                                                    { OR  }
```

構文規則

1. 形式 1 では、レベル番号は 01 から 49 までのどれか、または 77 とする。
2. 原則として、各句をどんな順序で書いてもよい。ただし、例外が 2 つある。データ 1 または FILLER (無名項目) 句が、

   (ANS85)指定されていれば、

   それらをレベル番号の直後に書く。REDEFINES (再定義) 句を指定するときは、データ名-1 または FILLER 句が、

   (ANS85)指定してあれば 直後に書く。指定がなければ、レベル番号の直後に書く。

3. 以下の用途の項目に指定された RENAMES 句の左辺として、PICTURE (形式) 句を指定してはならない。
   - INDEX
   - (ISO2002)(MF)OBJECT-REFERENCE
   - (MF)(OBJ97)PROCEDURE-POINTER
   - (ISO2002)(MF)PROGRAM-POINTER
   - (ISO2002)(VSC2)(MF)POINTER
   - (OSVS)(VSC2)(MF)COMPUTATIONAL-1、COMPUTATIONAL-2、

- ISO2002 MF BINARY-CHAR、BINARY-SHORT、BINARY-LONG、BINARY-DOUBLE、
- ISO2002 MF FLOAT-SHORT、FLOAT-LONG、
- MF EVENT-POINTER、MONITOR-POINTER、MUTEX-POINTER、SEMAPHORE-POINTER、または THREAD-POINTER

.

基本項目を記述する他のどの記述項に対しても、PICTURE 句を指定しなければならない。ただし、次の規則 (構文規則 4) に該当する場合を除く

4. ISO2002 MF 形式 1 の VALUE 句で英数字定数または各国語型定数が指定されている場合は、基本項目に対して PICTURE 句を省略してもよい。以下の場合は、PICTURE 句が暗黙的に指定される。
   a. 定数が英数字の 'PICTURE X(長さ)' である
   b. 定数が各国語型の 'PICTURE N(長さ))'である

   ここで、「長さ」とは、***COBOL 言語の概念***の章の*定数*節で指定された、定数の長さである。

5. THRU と THROUGH は同義語であり、どちらを書いてもよい。
6. ANS85 EXTERNAL 句を指定できるのは、作業場所節内の 01 レベルのデータ記述項の中だけである。
7. ANS85 EXTERNAL 句と REDEFINES 句は、同じデータ記述項の中には指定できない。
8. MF EXTERNAL 句および THREAD-LOCAL 句は、同じデータ記述項の中には指定できない。
9. ANS85 GLOBAL (大域) 句は、01 レベルのデータ記述項の中でだけ指定できる。
10. ANS85 データ名-1 は、GLOBAL 句および EXTERNAL 句を含む記述項に対して、または、GLOBAL 句および EXTERNAL 句を含むファイル記述項に関連づけられたレコード記述に対して指定するものとする。
11. MF TYPEDEF 句は、レベル番号が 01 のデータ記述項の中でのみ指定できる。
12. MF TYPEDEF 句は、データ名-1 が記述されている時にだけ使用できる。他の語では、明示的または暗黙的な FILLER 句のデータ記述項と同じものを使用できない。

    MF TYPEDEF 句を集団項目に対して指定した場合、従属するデータ記述は明示的または暗黙的な FILLER 句で定義できるので注意が必要である。

13. SYNCHRONIZED 句、PICTURE 句、JUSTIFIED 句、BLANK WHEN ZERO の各句は、基本データ項目以外には指定できない。

    OSVS VSC2 SYNCHRONIZED 句は、集団項目に指定できる。

14. MF 定数-6 と整数-1 は、浮動点小数値あるいは負数の値であってはならず、18 桁を超えてはならない。
15. 定数-6 が式の一部として使用されている場合、整数でなければならない。
16. OSVS VSC2 データ名-2 およびデータ名-3 は、暗黙的に修飾できる。

17. (2002)VALUE 句は、字類が指標、オブジェクト、ポインタのデータ項目には指定できない。
18. (2002)(MF)ANY LENGTH 句を指定した場合は、レベル番号、記述項名、PICTURE、および USAGE 以外の句は指定できない。

一般規則

1. 形式 3 は、条件名 (condition-name) に適用する。 各条件名はレベル番号が 88 の記述項に別々に定義する必要がある。ここに、条件名とそれに対応する単独の値またはいくつかの値または値の範囲を指定する。条件変数 (condition variable) に対する条件名記述項は、条件名が関連する項目の記述項の直後に書く。条件名は原則として、レベル番号を含む任意のデータ記述項に関連付けることができる。ただし、下記のものを除く。
    a. 他の条件名
    b. 66 レベルの項目
    c. JUSTIFIED、SYNCHRONIZED、USAGE (USAGE IS DISPLAY を除く) を含む記述項を伴う項目を含む集団項目
    d. 指標データ項目

       (VSC2)(MF)またはポインタ・データ項目

       (*USAGE(用途)句*節を参照。)

    e. (MF)定数名
2. (VSC2)(MF)条件名を、内部浮動小数点数項目と関連付けることができる。

   (MF)条件名を、外部浮動小数点数項目と関連付けることができる。

3. (MF)形式 4 では、定数名を定義する。定数名 (constant-name) とは、原始コードが COBOL システムを経由して渡されるときに割り当てられる定数値を表わす記号名である。COBOL システムは定数名を対応する値で置き換える。
4. (MF)TYPEDEF 句を使用すると、基本データ記述の集団を型定義として宣言できる。そして、USAGE 句内でデータ名-1 を型定義名-1 として使用して、そのデータ記述のインスタンスを宣言できる。型定義自体は記憶領域を割り当てられたデータ項目ではない。
5. (MF)TYPEDEF 句を集団のレベルで指定した場合、その型の任意のデータ項目に関してその型定義の構成要素が暗黙的に宣言される。その構成要素を参照するには、データ項目の修飾に関する COBOL の通常の規則を適用する。

8.3 ANY LENGTH 句

機能

ANY LENGTH 句が指定されている場合は、連絡節項目の長さが実行時に変化し、引数の長さにより決定される。

一般形式

ANY LENGTH

構文規則

1. ANY LENGTH 句は、関数、含まれているプログラム、またはプロパティメソッド以外のメソッドの連絡節中の基本レベル 1 記述項でのみ記述するものとする。

注：ANY LENGTH 句は、もっとも外部にあるプログラムで指定することはできない。これは、もっとも外部のプログラムは、プログラムプロトタイプ形式の CALL 文の有無にかかわらず、呼び出せるためである。プログラムプロトタイプを使用しない呼出では、引数が、ANY LENGTH で記述された仮のパラメータに対応しているかどうかを決定できない。

---

2. ANY LENGTH 句を含む原始要素が含まれるプログラムまたはメソッドである場合、記述項の左辺が、その手続き部見出しで以下のいずれかとして参照されなければならない。
    a. BY REFERENCE 指定を含む仮パラメータ
    b. 戻り値
3. ANY LENGTH 句を含む原始要素が関数の場合、記述項の左辺が、手続き部見出しで BY REFERENCE 指定を含む仮パラメータとして参照されなければならない。
4. 記述項の左辺には PICTURE 句を指定しなければならない。そして、PICTURE 句で指定された文字列は、形式記号'N'または'X'の 1 つのインスタンスでなければならない。

一般規則

1. ANY LENGTH 句を指定した場合、記述項の左辺は以下のいずれかとなる。
    a. 起動するランタイム要素の対応する引数または戻り項目がゼロの長さの項目の場合は、ゼロの長さの項目となる。
    b. その PICTURE 句の文字列中の形式記号が *n* 回繰り返されたものとして扱われる。ここで、*n* は、起動するランタイム要素の対応する引数または戻り項目のながさを表す。

8.4 BLANK WHEN ZERO(ゼロならば空白)句

機能

BLANK WHEN ZERO (ゼロならば空白) 句は、項目の値がゼロのときにそれを空白に変える。

一般形式

```
                 ZERO
BLANK WHEN {                  }
           {  ZEROS  }
           {  ZEROES }  (OSVS) (VSC2)
```

構文規則

1. BLANK WHEN ZERO 句を指定できるのは、PICTURE が数字 (明示的または暗黙的に USAGE IS DISPLAY を伴う) または数字編集であると指定されている基本項目だけである。(後述の *PICTURE(形式)句*を参照。)
2. ゼロ抑制記号としての星印 (*) と BLANK WHEN ZERO は、同じ記述項の中には指定できない。

   (OSVS)両方を同時に指定してもよい。ただし、ゼロ抑制の方が BLANK WHEN ZERO 句に優先する。

一般規則

1. BLANK WHEN ZERO 句を指定すると、その項目の値がゼロのとき、その内容はまったくの空白となる。
2. PICTURE (形式) が数字を表わす項目に BLANK WHEN ZERO 句を指定すると、その項目の項類は数字編集とみなされる。

8.5 BLOCK CONTAINS (ブロックの大きさ) 句

機能

BLOCK CONTAINS (ブロックの大きさ) 句は物理レコードの大きさを指定する。

(XOPEN)標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、この機能は明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内ではこの句を使用するべきではない。

一般形式

```
BLOCK CONTAINS [整数-1 TO] 整数-2 { RECORDS    }
                                  { CHARACTERS }
```

一般規則

1. (MF)BLOCK CONTAINS 句は注記になる。

8.6 CODE-SET(符号系)句

機能

CODE-SET (符号系) 句は外部媒体上のデータを表現するために使用される符号系を指定する。

CODE-SET 句を指定できるのは、レコード順編成

(MF)および行順編成の

ファイルに対してだけである。

(XOPEN)標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、この機能は明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内ではこの句を使用するべきではない。

一般形式

```
CODE-SET IS 符号系名 [FOR {一意名-1}...] (MF)
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - CHARSET - 何が固有の符号系とみなされるかを決定する。

構文規則

1. あるファイルに CODE-SET 句を指定した場合、そのファイル中のすべてのデータ項目の用途を DISPLAY と記述し、符号付き数字データがあれば SIGN IS SEPARATE 句を記述する。

   (MF)上記の制限は適用しない。

2. CODE-SET 句によって参照される符号系名句には、定数を指定できない。

   (MF)上記の制限は適用しない。

3. (MF)一意名-1 は修飾してもよいが、添字は付けられない。
4. (MF)各一意名-1 は、該当するファイルのレコード記述中に記述したデータ項目とする。しかし、それ自体がレコード記述であってはならない。すべての一意名-1 は、同じレコード記述内に含まれていなければならない。

5. (MF)FOR を指定すると、その対象のデータ項目に CODE-SET 句に指定した符号系が適用される。FOR を指定しないと、該当するファイル全体に CODE-SET 句に指定した符号系が適用される。
6. どのようなファイルまたはその中のデータ項目に関しても、固有の文字集合には CODE-SET 句は適用されないものと想定される。

一般規則

1. (MF)レコード領域中のデータは、つねに ASCII コードで書かれる。特殊名段落において符号系名が EBCDIC に設定されていると、CODE-SET 句の影響を受けるデータは、ファイルに書かれるときに ASCII から EBCDIC にコード変換される。特殊名段落において符号系名が STANDARD-1, STANDARD-2, NATIVE, ASCII のどれかに設定されている場合は、コード変換は必要ない。
2. (MF)上記のコード変換に際しては、CODE-SET 句が適用されるデータ項目はどれも英数字として扱われる。そのデータ記述に字類または項類がどのように記述されているかは考慮されない。
3. (MF)一意名-1 に OCCURS 句が指定してある場合、CODE-SET 句はその最初の要素に対してだけ適用される。一意名-1 の下位に属する項目に OCCURS 句が指定してある場合、COSE-SET 句は一意名-1 全体に適用される。

8.7 DATA RECORDS(データレコード)句

機能

DATA RECORDS (データレコード) 句は、ファイルを構成するデータレコードの名前を書き記すだけである。

(ANS85)ファイル記述項の DATA RECORDS 句は、ANSI '85 標準では廃要素に分類されており、ANSI 標準の次回の全面改訂の際に、削除される予定である。

(MF)この構文は、この COBOL 処理系に組み込まれているすべての方言で全面的に使用できる。FLAGSTD 指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。

(XOPEN)標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、この機能は明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内ではこの句を使用するべきではない。

一般形式

```
DATA { RECORD  IS }
     { RECORDS ARE }データ名-1 [データ名-2]...
```

構文規則

1. データ名-1 およびデータ名-2 は、データレコードの名前である。これと同じ名前で、レベル番号が 01 のレコード記述を書く。

   OSVS VSC2 これらのデータ名は、プログラム中のデータ記述項と対応する必要はない。

一般規則

1. データ名を複数指定することは、ファイル中にデータレコードの型が複数あることを意味する。各型のレコードは、大きさや形式などが異なっていてよい。このデータ名を書く順序には意味はない。

   注：FD 中で複数の 01 レベル項目の使用時に、SELECT 文がキー定義を含んでいる場合、キーの大きさは、その FD の最小レコード長の範囲内になければならない。

   ---

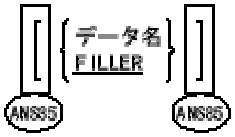
2. 概念的には、1 つのファイル内のデータレコードは、すべて同じ領域を共有する。ファイル内に複数の型のデータレコードが存在する場合でも、そのことは何ら変わらない。

8.8 データ名および FILLER 句

機能

データ名 (data-name) は、記述する対象のデータの名前を表わす。語、FILLER は、レコード中の明示的に参照しない基本項目を指定するために使用する。

一般形式



構文規則

1. データ名または FILLER 句は、

   ANS85 指定するのであれば、

   データ記述項のレベル番号の直後に書かれなければならない。

一般規則

1. 必要語 FILLER は、レコード中の基本項目または集団項目の名前として使用できる。どのような場合でも、FILLER 項目は明示的に参照できない。しかし、必要語 FILLER を条件変数として使用することはできる。この場合は、FILLER 項目を明示的に参照するのではなく、その値を参照するだけだからである。
2. (ANS85)この句を省略すると、記述対象の項目は FILLER を指定したように扱われる。

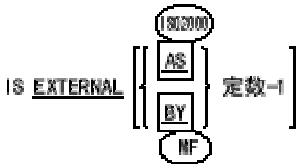
8.9 EXTERNAL(外部)句

(ANS85)

機能

EXTERNAL (外部) 句は、データ項目またはファイル結合子に外部属性を指定する。外部データ・レコードを構成するデータ項目および集団項目は、そのレコードを記述している実行単位に含まれるすべてのランタイム要素 から利用できる。

一般形式



構文規則

1. EXTERNAL 句を指定できる場所は、ファイル記述項と作業場所節内のレコード記述項だけである。
2. 同じ原始要素内で、EXTERNAL 句を含む項目の外部化した名前は、EXTERNAL 句を含む他の項目の外部化された名前とは異なるものにする。
3. EXTERNAL 句が含まれるデータ記述項またはその下位のデータ記述項内では、VALUE 句は指定できない。そのようなデータ記述項と関連する条件名記述項には、VALUE 句を指定できる。
4. (MF)定数-1 は文字または各国語型とし、表意定数としてはならない。
5. (ISO2002)EXTERNAL 句は字類オブジェクトのデータ項目用に指定してはならない。

一般規則

1. データ名句によって名付けられたレコード中に含まれるデータは、外部用である。このデータは、それを記述している実行単位に含まれるすべてのランタイム要素から呼び出して処理できる。このデータを再定義してもよい。ただし、下記の一般規則に従う必要がある。
2. 実行単位に含まれる 2 つ以上の原始要素 において同じ外部データ・レコードを記述する場合、関連するレコード記述項の各レコード名が同じであり、かつ各レコードに含

まれる標準データ形式の文字数も同じでなければならない。しかし、外部レコードを記述する原始要素 に REDEFINES (再定義) 句を含むデータ記述項を記述して、外部レコード全体を再定義することができる。この再定義は、同じ実行単位の中の別々の原始要素で同じにする必要はない。 *REDEFINES(再定義)句*を参照。

3. EXTERNAL 句を指定することは、関連するファイル名またはデータ名が大域名であることを意味するわけではない。後述の *GLOBAL(大域)句*を参照。
4. この記述項に関連するファイル結合子は、外部ファイル結合子である。

   (MF)EXTERNAL 属性を持つファイルの詳細については、ファイル処理に関する COBOL システムのマニュアルを参照。

5. (MF)定数-1 は外部データ項目またはファイル結合子の外部名を定義する。同じ外部データ項目またはファイル結合子を参照するすべての原始要素で、同じ外部名を使用しなければならない。

8.10 GLOBAL(大域)句

(ANS85)

機能

GLOBAL (大域) 句は、データ名やファイル名や報告書名が大域名であることを指定する。大域名は、それを定義しているプログラムに含まれるすべてのプログラムで利用できる。

一般形式

```
IS GLOBAL
```

構文規則

1. GLOBAL 句を指定できる場所は、ファイル節または作業場所節内の 01 レベルのデータ記述項、ファイル記述項、報告書記述項だけである。

   (MF)(OB37)連絡節においても、GLOBAL 句を指定できる。

2. 同じデータ部内では、同じ名前の 2 つのデータ項目のデータ記述項には GLOBAL 句を指定できない。
3. いくつかのファイルに SAME RECORD AREA (レコード領域の共用) 句を指定した場合、それらのレコード記述項またはファイル記述項には GLOBAL 句を指定できない。

一般規則

1. GLOBAL 句を使用して記述したデータ名、ファイル名、報告書名は大域名である。ある大域名の下位に属するデータ名もすべて大域名となる。大域名と関連する条件名もすべて大域名となる。
2. 大域名を定義しているプログラムの中に直接的または間接的に含まれるプログラムからは、その名前を再度定義することなく、参照できる。*COBOL 言語の概念*の章の*名前の適用範囲*節を参照。
3. REDEFINES 句が含まれるデータ記述項の中で GLOBAL 句を使用した場合、その GLOBAL 句は大域属性をもつ REDEFINES だけの対象となる。

8.11 JUSTIFIED(桁寄せ)句

機能

JUSTIFIED (桁寄せ) 句は、受取り側データ項目における標準的でない桁寄せを指定する。

一般形式

```
{JUSTIFIED} RIGHT
{JUST     }
```

構文規則

1. JUSTIFIED 句は、基本項目にだけ指定できる。
2. JUST は、JUSTIFIED の省略形である。
3. 数字項目または編集を指定した項目には、JUSTIFIED 句を指定できない。
4. 指標データ項目(後述の *USAGE(用途)句*節を参照)

   VSC2 MF またはポインタ・データ項目

   には、 JUSTIFIED 句を指定できない。

5. OSVS VSC2 MF 外部浮動小数点数項目または内部浮動小数点数項目には、JUSTIFIED 句を指定できない。

一般規則

1. 受取り側のデータ項目に JUSTIFIED 句が指定してあり、送出し側データ項目の方が受取り側データ項目よりも大きい場合、左端の文字が切り捨てられる。受取り側のデータ項目に JUSTIFIED 句が指定してあり、受取り側データ項目の方が送出し側データ項目よりも大きい場合、データは右端を揃えられ左端には空白が詰められる。

   送出し側データ項目の内容は考慮されない (後部の空白は抑制されない) ので、注意する必要がある。

たとえば、PIC X (4) のデータ項目があり、その値が"A　　"(A の後ろに空白が 3 つ)であるとする。これを、PIC X (6) JUSTIFIED のデータ項目に転記すると、結果は"　A　　"となる。同じデータ項目を PIC X (3) JUSTIFIED のデータ項目に転記すると、左端の文字が切り落とされて、結果は"　　" (空白が 3 つ) となる。

2. JUSTIFIED 句を省略すると、基本データ項目にデータを収める際の標準桁寄せの規則が適用される。(*COBOL 言語の概念*の章の*標準桁寄せ規則*節を参照。)

8.12 LABEL RECORDS(ラベル・レコード)句

機能

LABEL RECORDS は、ラベルの有無を指定する。

ANS85 LABEL RECORDS 句は、ANSI '85 標準では廃要素に分類されており、ANSI 標準の次回の全面改訂の際に、削除される予定である。

MF この構文は、この COBOL 処理系に組み込まれているすべての方言で全面的に使用できる。FLAGSTD 指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。

XOPEN 標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、この機能は明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内ではこの句を使用するべきではない。

一般形式

構文規則

1. 各ファイル記述項に、LABEL RECORDS 句を記述する。

   ANS85 句は必要ない。

2. OSVS VSC2 データ名-1 は、ラベルレコードの名前である。この名前で、01 レベルのレコード記述を用意しておく。
3. OSVS VSC2 データ名-1 は、該当するファイルの DATA RECORDS 句には指定できない。
4. OSVS VSC2 LABEL RECORD IS と LABEL RECORDS ARE はどちらも正しい構文として受け入れられる。

一般規則

1. (MF)LABEL RECORDS 句は注記にとどまる。

8.13 レベル番号

機能

レベル番号 (level-number) は、レコード内のデータの階層を示す。さらに、作業場所項目、連絡項目、条件名、

(MF)定数名、

および RENAMES (再命名) 句の記述項にも、レベル番号を使用する。

一般形式

```
レベル番号
```

構文規則

1. レベル番号は、各データ記述項の最初の要素として必要である。
2. FD または SD 記述項の下位のデータ記述項のレベル番号は、01 から 49、66、

   (MF)78、

   88 のどれかとする。(前述の *ファイル記述*節を参照。)

3. 報告書節および画面節中のデータ記述項のレベル番号は、01 から 49

   (MF)または 78

   とする。

4. 作業場所節、

   (MF)局所記憶節、

   連絡節中のデータ記述項のレベル番号は、01 から 49、66、77、

   (MF)78、

   および 88 のどれかとする。

5. レベル番号は、1 文字または 2 文字の数字とする。

一般規則

1. レベル番号 01 は、各レコード記述の最初の記述項であることを示す。
2. レベル階層の概念が当てはまらない記述項に、特別のレベル番号が割り当てられている。下記のものがある。
   a. レベル番号 77 は、独立作業場所データ項目および独立連絡データ項目を示すために使用する。このレベル番号には、先に示したデータ記述項の骨組みの形式 1 だけが適用できる。
   b. レベル番号 66 は、RENAMES (再命名) 記述項を示すために使用する。このレベル番号には、先に示したデータ記述項の骨組みの形式 2 だけが適用できる。
   c. レベル番号 88 は、条件変数に関連する条件名を定義する記述項を示すために使用する。このレベル番号には、先に示したデータ記述項の骨組みの形式 3 だけが適用できる。
   d. (MF) レベル番号 78 は、定数名を定義する記述項を示すために使用する。このレベル番号には、先に示したデータ記述項の骨組みの書き方 4 だけが適用できる。
3. FD または SD 記述項の下位に属する複数の 01 レベルの記述項は、同じ領域を暗黙的に再定義する。

8.14 LINAGE(行数)句

機能

LINAGE (行数) 句を指定することによって、利用者は論理ページの行数を設定できる。利用者はまた、論理ページ上の上下の余白 (top margin, bottom margin) およびページ本体 (page body) 中の脚書領域 (footing area) の開始行を指定することもできる。

LINAGE 句を指定することができるのは、レコード順編成のファイルに対してだけである。

一般形式

```
LINAGE IS {データ名-1 | 整数-1} LINES [ WITH FOOTING AT {データ名-2 | 整数-2} ]
    [ LINES AT TOP {データ名-3 | 整数-3} ] [ LINES AT BOTTOM {データ名-4 | 整数-4} ]
```

構文規則

1. データ名-1、データ名-2、データ名-3、データ名-4 は、符号なしの基本整数データ項目とする。

2. 整数-1 の値は、ゼロより大きくする。
3. 整数-2 の値は、整数-1 の値を超えてはならない。
4. 整数-3 および整数-4 の値は、ゼロであってもよい。

一般規則

1. LINAGE 句を指定することによって、利用者は論理ページの行数を設定できる。論理ページの大きさは、FOOTING 指定を除く各指定値の和である。FOOTING を指定しないと、脚書機能は組み込まれず、ページあふれ条件 (page overflow condition) とは別のページの終わり条件 (end-of-page condition) は発生しない。

   論理ページと物理ページの大きさの間には、必ずしも関係がある必要はない。

2. 整数-1 またはデータ名-1 のデータ項目の値は、論理ページ上に書いたり改行したりできる行数を表わす。この値はゼロよりも大きくする。論理ページ上の書いたり改行したりできるこの部分を、ページ本体と呼ぶ。
3. 整数-2 またはデータ名-2 のデータ項目の値は、論理ページ上の脚書領域の開始行を表わす。この値はゼロよりも大きく、かつ整数-1 またはデータ名-1 のデータ項目の値を超えてはならない。.

   脚書領域となるのは、整数-2 またはデータ名-2 のデータ項目の値以上、整数-1 またはデータ名-1 のデータ項目の値以下の、論理ページ中の行の部分である。

4. 整数-3 またはデータ名-3 のデータ項目の値は、論理ページ上の上の余白行数を表わす。この値はゼロでもよい。

   整数-4 またはデータ名-4 のデータ項目の値は、論理ページ上の下の余白行数を表わす。この値はゼロでもよい。

5. 整数-1 と整数-3 と整数-4 を指定すると、OPEN 文を用いて該当ファイルを出力モードで開いたときに、これらの値に基づいて論理ページの各構成部分の行数が設定される。整数-2 を指定すると、その値に基づいて脚書領域が決定される。これらの値はすべて、該当ランタイム要素が実行される間、ファイルに書き込まれる論理ページを制御するために使用される。
6. データ名-1、データ名-3、データ名-4 を指定すると、OPEN 文を用いて該当ファイルを出力モードで開いたときに、これらの値に基づいて最初の論理ページの各構成部分の行数が設定される。

   データ名-2 を指定すると、OPEN 文を用いて該当ファイルを出力モードで開いたときに、この値に基づいて最初の論理ページの脚書領域が決定される。

7. LINAGE 句を指定すると、LINAGE-COUNTER が生成される。LINAGE-COUNTER の値は、どの時点でも、現在のページ本体の中で次に書き込まれる行番号を表わしている。LINAGE-COUNTER には下記の規則が適用される。
   a. ファイル節中のファイル記述項に LINAGE 句を指定した各ファイルごとに、別々の LINAGE-COUNTER が設定される。

b. 手続き部の文によって、LINAGE-COUNTER を参照することはできるが、変更することはできない。1 つの原始要素中に複数の LINAGE-COUNTER が存在することがあるので、必要に応じて、ファイル名で LINAGE-COUNTER を修飾しなければならないことがある。

8. 各論理ページは、次の論理ページと間を空けることなく連なっている。
9. ADVANCING PAGE 指定のある WRITE 文を実行したとき、またはページあふれ条件が発生したとき(***手続き部 - SEARCH - WRITE*** の章の *WRITE(書き込み)文*節を参照) 、各データ項目の値に基づいて、次の論理ページを構成する各部分の行数が決定される。
10. データ名-2 を指定すると、ADVANCING PAGE 指定のある WRITE 文を実行したとき、またはページあふれ条件が発生したとき、その値に基づいて、次の論理ページの脚書領域が決定される。
11. 該当するファイルに対して WRITE 文を実行すると、下記の規則に従って、LINAGE-COUNTER は自動的に更新される。
    - WRITE 文に ADVANCING PAGE 指定がある場合は、LINAGE-COUNTER の値は自動的に 1 に再設定される。
    - WRITE 文に ADVANCING 一意名-2 または整数を指定している場合は、LINAGE-COUNTER の値はその整数または一意名-2 の値の分だけ繰り上げられる。
    - WRITE 文に ADVANCING 指定をしていない場合は、LINAGE-COUNTER の値は 1 だけ繰り上げられる。(***手続き部 - SEARCH - WRITE*** の章の *WRITE(書き出し)文*節を参照。)
    - 以降の各論理ページの書き込み可能な最初の行に装置が位置付け直されるたびに、LINAGE-COUNTER の値は自動的に 1 に再設定される。(***手続き部 - SEARCH - WRITE*** の章の *WRITE(書き出し)文*節を参照。)
12. ファイルに対して OPEN 文を実行すると、対応する LINAGE-COUNTER の値は自動的に 1 に設定される。

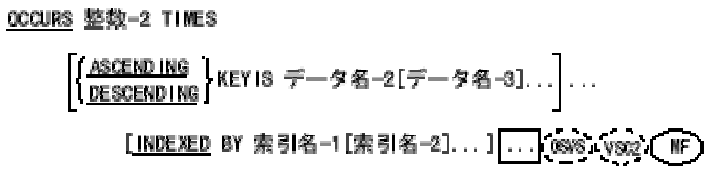
8.15 OCCURS(反復)句

機能

OCCURS (反復) 句は、同じデータ項目が繰り返されるとき、その記述項をいちいち指定する手間を省き、添字 (subscript) や指標 (index) を付けるために必要な情報を与える。

一般形式

形式 1



形式 2

```
OCCURS [整数-1 TO] 整数-2 TIMES DEPENDING ON データ名-1
  (OSVS) (VSC2) (MF) (OSVS) (VSC2) (MF)
  [{ASCENDING / DESCENDING} KEY IS データ名-2[データ名-3]...]...
      [INDEXED BY 索引名-1[索引名-2]...] ... (OSVS) (VSC2) (MF)
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
   - ODOOSVS - OCCURS DEPENDING ON 句について、OS/VS COBOL と互換性のある処理を行うよう要求する。
   - ODOSLIDE - ネストされた OCCURS DEPENDING ON 句、および OCCURS DEPENDING ON 句の後ろに続く固定データの処理を制御する。
   - REENTRANT - プログラムをマルチスレッド環境で使用可能にする。

構文規則

1. 整数-1 と整数-2 を両方指定する場合、整数-1 は 1 以上であるか、

   (ANS85)(OSVS)または 0 に等しく、

   整数-2 は整数-1 よりも大きいか、

   (OSVS)(VSC2)または等しく

   する。

   (OSVS)(VSC2)(MF)形式 2 において、" 整数-1 TO " を省略すると、省略時解釈として 1 がとられる。

2. データ名-1 のデータ記述は、整数とする。
3. データ名-1、データ名-2、データ名-3、...は修飾できる。
4. データ名-2 は、OCCURS 句を含む記述項の名前であるか、またはその下位に属する記述項の名前とする。

   (OSVS)この制限は無視してよい。

5. データ名-3 などは、その下位に属する記述項の名前にする。

   (OSVS)この制限は無視してよい。

6. OCCURS 句の対象またはその下位に属する記述項を指標を用いて参照する場合、INDEXED BY (指標付き) 句を指定する。

   OSVS VSC2 MF ただし、他の表用に定義した指標を使用する場合は、その必要はない。(*COBOL 言語の概念*の章の*指標付け*節を参照。)

   ここに指定する指標名はデータとして扱われず、データの階層にも属さない。

7. レコード記述内で、形式 2 の OCCURS 句が含まれるデータ記述項の後ろに続けられるのは、その下位に属するデータ記述項だけである。

   OSVS VSC2 MF レコード記述内で、形式 2 の OCCURS 句が含まれるデータ記述項の後ろに、その下位に属さないデータ記述項を続けることができる。データ名-1 のデータ項目の値が変更された場合は、表の後に続く (ただし、その下位ではない) データ項目の位置が変更される。これらの項目に含まれるデータは、失われる場合がある。

   MF NOODOSLIDE システム指令を設定した場合は、表を含む集団項目はすべて、データ名-1 の値にかかわらず、常に最大の回数だけ反復されるものとみなされる。したがって、表の後に続くデータ項目の位置は変更されない。

8. OCCURS 句は、下記のデータ記述項には指定できない。
   a. レベル番号が 66 または 88 のもの
   b. 大きさが可変な項目を記述するもの。大きさが可変な項目とは、下位の項目のデータ記述に、形式 2 の OCCURS 句が含まれるものである。

      OSVS VSC2 形式 2 の OCCURS 句を指定したデータ項目の下位に属するデータ記述に、OCCURS 句を指定できる。

9. OCCURS 句は、01 レベルまたは 77 レベルのデータ記述項には指定できない。

   MF この制限は削除された。

10. 形式 2 において、データ名-1 によって定義されるデータ項目は、OCCURS 句を含むデータ記述項によって定義される最初の文字位置から最後の文字位置までの範囲内にあってはならない (OCCURS 句の対象のデータ項目と、別の記憶域を占めるデータ項目にすること)。

    OSVS VSC2 OCCURS DEPENDING ON Slide 指令を設定した場合は、データ名-1 は固定位置であること。

11. データ名-2 がこの記述項の左辺ではないとき、下記のようにする。
    a. KEY IS (キーは) 句に指定するデータ名の項目はすべて、この記述項の左辺である集団項目に含まれているものとする。
    b. KEY IS 句に指定するデータ名の項目は、OCCURS 句を含んではならない。
    c. KEY IS 句に指定するデータ名の項目とこの記述項の左辺との間には、OCCURS 句が含まれる記述項を置いてはならない。

12. 指標名-1、指標名-2 などは、原始要素内で一意の語とする。

    OSVS 指標名-1、指標名-2...は一意である必要はなく、この記述項の左辺のデータ名によって修飾できる。

13. OSVS VSC2 MF 外部浮動小数点数データ項目および内部浮動小数点数データ項目に対して、OCCURS 句を指定できる。
14. ISO/IEC KEY 句は、字類がオブジェクトのデータ項目に対して記述してはならない。

一般規則

1. OCCURS 句は、表または同形のデータ項目の反復を定義するために使用する。OCCURS 句を指定した場合、文の中でこの記述項の左辺であるデータ名を参照するときは、添字または指標を付ける。ただし、SEARCH、

   MF SORT、

   および USE FOR DEBUGGING の各文は例外である。さらに、この記述項の左辺が集団項目の名前である場合、その集団に属するデータ項目を作用対象として使用するときには、すべて添字または指標を付ける。ただし、REDEFINES 句の作用対象として用いる場合は例外である。(*COBOL 言語の概念*の章の*添字付け*、*指標付け* および *一意名* の各節を参照。)

2. OCCURS 句を指定した項目のデータ記述句はすべて、反復される各項目にも使用される。ただし、OCCURS 句自体は反復しては使用されない。
3. データ名-1 は固定位置に置く。したがって、OCCURS DEPENDING ON 句を含む項目の後ろには置けない。
4. 記述項の左辺の反復回数は、下記のように定まる。
   a. 形式 1 では、整数-2 の値が一定の反復回数を表わす。
   b. 形式 2 では、データ名-1 によって参照されるデータ項目の現在の値が反復回数を表わす。

      この形式は、この記述項の左辺の反復回数が可変であることを表わす。整数-2 の値は最大反復回数を表わし、整数-1 の値は最小反復回数を表わす。このことは必ずしもこの記述項の左辺の長さが可変であることを意味するのではなく、反復回数が可変であることを意味する。

      データ名-1 によって参照されるデータ項目の値は、整数-1 から整数-2 の範囲に入らなければならない。データ名-1 のデータ項目の値を小さくすると、それよりも出現番号の大きいデータ項目の内容は保証されない。

5. ANS85 形式 2OCCURS 句で記述された項目を含む集団項目を参照すると、使用される表の領域は下記のように決定される。
   a. データ名-1 によって参照されるデータ項目がその集団項目の外にあるならば、表領域のうちの参照処理が開始される時点で、そのデータ項目の値によって指定される部分が使用される。

b. データ名-1 によって参照されるデータ項目がその集団項目に含まれており、集団データ項目が送出し側としてだけ参照されているならば、表領域のうちの参照処理が開始される時点で、そのデータ項目の値によって指定される部分だけが使用される。その集団項目が受取り側になっている場合は、表の最大領域が使用される。

6. KEY IS 句は、データ名-2、データ名-3 などの値に従って、昇順 (ascending) または降順 (descending) に反復データを並べることを指定する。昇順または降順は、作用対象の比較の規則に従って決められる。(*手続き部の比較条件*節を参照。) これらのデータ名はキーの強さの順に左から右へ指定する。
7. 指標名-1、指標名-2...に割り当てられた記憶域の種類は、プログラムが再帰的である (局所記憶節が定義されている) かどうか、および REENTRANT コンパイラ指令が指定されているかどうかにより決まる。

表 8-1 は、非再帰プログラムに記憶域が割り当てられる方法を示す。表 8-2 は、再帰プログラムに記憶域が割り当てられる方法を示す。

表 8-1: 非再帰プログラムの指標名に割り当てられる記憶域

指標名を定義する節	指標名を割り当てる節		
	NOREENTRANT	REENTRANT (1)	REENTRANT (2)
作業場所	作業場所	作業場所	スレッド局所
スレッド局所	スレッド局所	スレッド局所	スレッド局所
連結	作業場所	局所記憶	局所記憶

表 8-2: 再帰プログラムの指標名に割り当てられる記憶域

指標名を定義する節	指標名を割り当てる節		
	NOREENTRANT	REENTRANT (1)	REENTRANT (2)
作業場所	作業場所	作業場所	スレッド局所
スレッド局所	スレッド局所	スレッド局所	スレッド局所
局所記憶	局所記憶	局所記憶	局所記憶
連結	局所記憶	局所記憶	局所記憶

8.16 PICTURE(形式)句

機能

PICTURE (形式) 句は、基本項目の一般的性質と編集の形式を示す。

一般形式

```
{PICTURE}
{PIC    } IS 文字列
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - INTLEVEL - 方言が MF のときに、数字データ項目に許される最大の長さを決定する。

構文規則

1. PICTURE は、基本項目にだけ指定できる。
2. 文字列は、COBOL 文字集合 (character set) に含まれる文字を記号として用い、一定の規則に従って組み合わせたものである。この組合わせによって、基本項目の項類が決まる。
3. 文字列に含められる文字の数は、最大 30 字である。

   OS/390 ISO1002 MF 文字列に含められる文字の数は、最大 50 字である。

4. PICTURE は、RENAMES (再命名) 句の左辺に指定してはならない。
5. PIC は、PICTURE の省略形である。
6. ゼロ抑制記号としての星印 (*) と BLANK WHEN ZERO 句は、同じ記述項に指定できない。

   OSVS ゼロ抑制記号としての星印 (*) と BLANK WHEN ZERO 句を同じ記述項に指定してもよい。ただし、ゼロ抑制の方が BLANK WHEN ZERO 句に優先する。

7. 1 文字に続けて、整数をかっこで囲んで指定すると、その文字を整数の値分の数だけ並べることを意味する。

一般規則

PICTURE 句に書けるデータは、英字、数字、英数字、英数字編集、数字編集、

OSVS VSC2 MF および外部浮動小数点数

である。以下に、各項類ごとに分けて一般規則を記す。

英字 (alphabetic) 項目に関する規則

1. PICTURE 文字列に書ける記号文字は、"A"と"B"だけである。

   (ANS85)PICTURE 文字列に書ける記号文字は、"A"だけである。

2. 標準データ形式で表わした場合、その内容は 1 つ以上の英字とする。

数字 (numeric) 項目に関する規則

1. 文字列に書ける記号文字は、"9"、"P"、"S"、および"V"だけである。PICTURE 文字列で表わせる桁数は、1 以上 18 以内である。

   (OS/390)(SQL/400)PICTURE 文字列で表わせる桁数は、31 までである。

   (MF)PICTUREPICTURE 文字列で表わせる桁数は、38 までである。

2. 符号を付けない場合、標準データ形式のデータはアラビア数字の"0", "1", "2", "3", "4", "5", "6", "7", "8", "9" を組み合わせたものとする。符号を付ける場合は、上記の数字の他に、"+", "-" およびその他の演算記号を含めることができる。(後述の *SIGN(符号)句*節を参照。) .)

数字データは、標準データ形式以外の形式でも保持できる。(後述の *USAGE(用途)句*節、および ***COBOL 言語の概念***の章の*文字の表現と基数の選定*節を参照。)

英数字 (alphanumeric) 項目に関する規則

1. 文字列に書ける記号文字は、 "A", "X", "9" だけである。結果的には、記号文字をすべて"X" と書いたように扱われる。PICTURE 文字列に含まれる記号文字がすべて"A"または"9"である場合、英数字項目にならない。
2. 標準データ形式で表わした場合、その内容は計算機の文字集合の任意の文字を含んでいてよい。

英数字編集 (alphanumeric edited) 項目に関する規則

1. PICTURE 文字列に書ける記号文字は、 "A", "X", " 9", "B", "0", "/" を一定の規則に従って組み合わせたものに限られる。その規則は下記の通り。
   a. PICTURE 文字列には、最低 1 つの "B"と最低 1 つの"X" 、または最低 1 つの "0" (ゼロ) と最低 1 つの"X" 、または最低 1 つの"/" (斜線) と最低 1 つの"X" を含ませる。または、
   b. PICTURE 文字列には、最低 1 つの"0" (ゼロ) と最低 1 つの"A" 、または最低 1 つの"/" (斜線) と最低 1 つの"A" が含まれるものとする。
2. 標準データ形式で表わした場合、その内容は計算機の文字集合の所定の文字を含むことができる。

数字編集 (numeric edited) 項目に関する規則

1. PICTURE 文字列に書ける記号文字は、"B", "/", "P", "V" , "Z", "0", "9", ",", ".", "*", "+", "-", "CR", "DB", および通貨記号を、一定の規則に従って組み合わせたものに限られる。許容される記号文字の組合わせは、記号の優先順位によって決まる。その規則は下記の通り。
    a. PICTURE 文字列で表わせる桁数は、1 以上 18 以内である。

        OS/390 OS/2000 PICTURE 文字列で表わせる桁数は、31 までである。

        MF PICTURE 文字列で表わせる桁数は、38 までである。

    b. PICTURE 文字列には、最低 1 つの"0" , "B", "/", "Z", "*", "+", ",", ".", "-", "CR", "DB"、または通貨記号が含まれるものとする。
2. 標準データ形式で表わした場合、その内容は 1 つの数値とする。
3. PICTURE 文字列のすべての桁位置が挿入文字によって表わされている場合、少なくとも 1 つの文字を小数点の左側に置く。

外部浮動小数点数

OSVS VSC2 MF

1. PICTURE 文字列は、下記の形式とする。

    { + / − } 仮数部 E { + / − } 指数部

    仮数部 (mantissa) および指数部 (exponent) の前に、符号文字を置く。

    "+" 符号は、出力値が正であれば正号が付けられ、出力値が負であれば負号が付けられることを表わす。

    "-" 符号は、出力値が正であれば符号は空白とされ、出力値が負であれば負号が付けられることを表わす。

    各符号は、1 バイトの記憶域を占める。

    ### 仮数部

    仮数部には、下記の記号を含めることができる。

    9 . V

    実小数点は "."で表わし、想定小数点は "V"で表わす。仮数部には実小数点または想定小数点のどちらかがなければならない。小数点は左端にあっても、中間にあっても、右端にあってもよい。仮数部の長さは 1 桁以上 16 桁以下である。

E

指数を示す。

指数部

指数部は、PIC "99" の形式にする。

2. 外部浮動小数点数項目には、OCCURS、REDEFINES、および RENAMES の各句を指定できる。
3. SIGN 句は、注記としてだけ使用するものであり、符号の表現には影響しない。
4. SYNCHRONIZED 句は、注記としてだけ使用する。
5. 外部浮動小数点数に関しては、下記の句は指定できない。

   BLANK WHEN ZERO
   JUSTIFIED
   VALUE

基本項目の大きさ

基本項目の大きさとは、標準データ形式で表わしたときの、基本項目の文字数をいう。基本項目の大きさは、文字位置を表わすのに使用できる記号の数によって決定される。記号文字 "A", ",", "X" , "9", "P", "Z", "*", "B", "/", "0", "+", "-", または通貨記号の後ろにかっこで囲んで書いた整数は、それらの記号を連続して並べる数を示す。 次の記号文字は PICTURE 文字列内には1つしか書けないことに注意。

"S"、"V"、"."、

OSVS VSC2 MF "E"、

"CR"、および"DB"

使用する記号

基本項目を表わすために使用する記号の働きを、以下に説明する。

A	PICTURE 文字列中の各 "A" は、1 文字の英字または 1 つの空白だけを含むことのできる文字位置を表わす。
B	PICTURE 文字列中の各"B" は、空白文字を挿入できる文字位置を表わす。
OSVS VSC2 MF E	外部浮動小数点数項目の指数部の始まりを示す。指数部は、実行時に 1 バイトの記憶域を占める。
P	PICTURE 文字列中の各"P" は、想定小数点の位置を合わせるための位取りを表わすものである。データ項目中に現れる位の中に小数点が含まれていないときに、想定小数点の位置を指定するために使用する。位取り文字であ

る "P" は、データ項目の大きさには数えられない。数字編集項目または数字項目の最大桁数 (18 桁) を判定する際には、位取り文字は数に入れられる。この"P" は、PICTURE 文字列の左端または右端に連続してだけ書くことができる。これは、"P" は想定小数点の位置を暗示するからである。つまり、PICTURE 文字列の左端に "P" がある場合はその左に想定小数点があることになり、PICTURE 文字列の右端に"P" がある場合はその右に想定小数点があることになる。"P"を含む PICTURE 文字列の左端または右端に想定小数点記号の "V" を書くことは、定義の重複である。

"P" と挿入文字の"." 小数点) を、同一の PICTURE 文字列に含めてはならない。データ項目の内部表現をあるものから別のものへ変換するときに、対象のデータ項目の PICTURE 文字列に "P" が含まれていると、各"P" の位置に値ゼロが入っているものとみなされ、データ項目の大きさにはその桁位置も含まれる。

ANS85 PICTURE 文字列に記号 "P" が含まれるデータ項目を参照する処理で用いられるのは、そのデータ項目の実際の文字表現ではなく、そのデータ項目の代数値である。この代数値は、"P" の配置によって示される位置に小数点を想定し、各"P" をゼロで置き換えたものである。このデータ項目の大きさは、PICTURE 文字列によって表わされる桁数となる。このような処理が行われる事例は下記のとおり。

a. 数字の送出し側作用対象を必要とする処理
b. 送出し側の作用対象が数字であり、その PICTURE 文字列に記号文字 "P"が含まれる場合の、基本 MOVE (転記) 文
c. 送出し側の作用対象が数字編集であり、その PICTURE 文字列に記号文字"P" が含まれ、受取り側の作用対象が数字または数字編集である場合の、MOVE 文
d. 両方の作用対象が数字である場合の比較処理

ANS85 上記以外のすべての処理では、記号文字 "P" は無視され、作用対象の大きさに含められない。

S PICTURE 文字列中の"S" は、演算符号の存在を示す。しかし、演算符号の表現形式や位置までも示すものではない。"S"は PICTURE 文字列の左端に書く。この記号文字は標準データ形式で数えた基本項目の大きさには含まれない。ただし、その記述項に SEPARATE CHARACTER 句を指定した場合は、 "S" は項目の大きさに含められる。(この章の *SIGN(符号)句*節を参照。)

V PICTURE 文字列中の "V" は想定小数点の位置を示す。この記号文字は 1 つの PICTURE 文字列には 1 つだけ書ける。"V" は文字位置を表わすわけではないので、基本項目の大きさには含まれない。想定小数点が PICTURE 文字列の右端の文字の右側にある場合は、"V" を書いても書かなくてもよい。

X PICTURE 文字列中の各"X" は、1 個の文字の位置を示す。そこには、計算機文字集合で許される任意の文字を含めることができる。

Z　PICTURE 文字列中の各 "Z" は、先行ゼロ列のゼロを空白に置き換えることを示す。"Z" は項目の大きさに含められる。

9　PICTURE 文字列中の各 "9" は、数字を含む 1 個の桁位置を示す。"9" は項目の大きさに含められる。

0　PICTURE 文字列中の各 "0" (ゼロ) は、数字のゼロを挿入すべき 1 個の桁位置を示す。"0" は項目の大きさに含められる。

/　PICTURE 文字列中の各"/" (斜線) は、文字"/"を挿入すべき 1 個の桁位置を示す。 "/" は項目の大きさに含められる。

,　PICTURE 文字列中の各"," (コンマ) は、文字"," を挿入すべき 1 個の桁位置を示す。"," は項目の大きさに含められる。PICTURE 文字列の最後には置けない。

ANS85 挿入文字 "," を、PICTURE 文字列の最後に置いてもよい。

.　PICTURE 文字列中に現れる文字"." (終止符) は、編集記号であり、桁合わせの基準とする小数点の位置を表わすとともに、文字"." を挿入する位置を表わす。"." は項目の大きさに含められる。特殊名段落に DECIMAL-POINT IS COMMA を指定すると、原始単位における終止符とコンマの機能が入れ替えられる。この場合、PICTURE 句の中の終止符とコンマに関しても、終止符に関する規則がコンマに適用され、コンマに関する規則が終止符に適用される。挿入文字 " ." は、PICTURE 文字列の最後には置けない。

ANS85 挿入文字 "." を、PICTURE 文字列の最後に置いてもよい。

+, -, CR, DB　これらの記号文字は符号編集用文字であり、符号編集用文字を置く位置を示す。1 つの PICTURE 文字列の中では、これらの記号文字のうちのどれか 1 つだけ使える。これらの文字は項目の大きさに含められる。

*****　PICTURE 文字列中の各"*" (星印) は、先行ゼロ列のゼロを星印に置き換えることを示す。"*" は項目の大きさに含められる。

通貨編集用文字　PICTURE 文字列中の通貨編集用文字は、編集中に通貨編集用文字が収められる文字位置を示す。通貨編集用文字は、通貨記号または特殊名段落に CURRENCY SIGN 句で指定した 1 文字で表わされる。

通貨編集用文字が最初に現れたときに、その文字数が項目の長さに追加される。その後、通貨編集用文字が現れるたびに、1 が項目の長さに追加される。

CURRENCY コンパイラ指令も、使用される通貨編集用文字に影響を与える。

編集規則

PICTURE 句の中で指定できる編集方法は、挿入およびゼロ抑制の 2 つである。挿入による編集には下記の 4 種類がある。

1. 単純挿入 (simple insertion)

2. 特殊挿入 (special insertion)
3. 固定挿入 (fixed insertion)
4. 浮動挿入 (floating insertion)

ゼロ抑制による編集には下記の 2 種類がある。

1. 空白によるゼロ抑制
2. 星印 (*) によるゼロ抑制

項目に行える編集の種類は、その項類によって異なる。表 8-3 にその関係を示す。

表 8-3: データの項類に適用できる編集の種類

項類	編集の種類
英字	"B" による単純挿入だけ [1]
数字	なし
英数字	なし
英数字編集	"0", "B" "/"による単純挿入
数字編集	すべて [2]
2 バイト文字	単純挿入
外部浮動小数点数	特殊挿入

注:

1. ANS85 ANSI '74 標準では、"A" と"B" の両方が含まれる PICTURE 句は英字に対する単純挿入として扱われていた。ANSI '85 標準では英字に対する "B" を認めなくなり、両方が含まれる PICTURE 句は、英数字編集に対する単純挿入として扱われるようになった。
2. 1 つの PICTURE 句の中に、浮動挿入編集とゼロ抑制編集を同時に指定することはできない。1 つの PICTURE 句の中には、1 種類のゼロ抑制だけを指定できる。

単純挿入編集

"," (コンマ), "B" (空白), "0" (ゼロ), " /" (斜線)は、挿入文字として使用される。挿入文字は項目中に文字を挿入すべき位置を表わし、項目の大きさに含められる。

特殊挿入編集

OSVS VSC2 MF この種の編集は、数字編集項目または外部浮動小数点数項目に対して使用できる。

"." (終止符)は、 は挿入文字として使用されるとともに、桁合わせの基準となる小数点を表わすためにも使用される。実小数点用に使用される挿入文字は、項目の大きさに含められる。同じ PICTURE 文字列の中に、記号文字"V" で表わされる想定小数点と、この挿入文字で表わされる実小数点を指定することはできない。特殊挿入編集を施すと、PICTURE 文字列中に指定した位置に、指定した挿入文字が挿入される。

固定挿入編集

通貨編集用文字および符号編集用文字の"+" ,"-", "CR", "DB" は、挿入文字である。1 つの PICTURE 文字列の中では、1 つの通貨編集用文字と、符号編集用文字のどれか 1 つだけを使用できる。符号編集用文字の "CR" または "DB" を使用する場合は、文字列の右端に置く。これらの符号編集用文字は項目の大きさとしては 2 文字分に数えられる。符号編集用文字の"+" または "-" を使用する場合は、文字列の右端または左端に置く。これらの符号編集用文字は項目の大きさとして数えられる。通貨編集用記号は、文字列の左端に置く。

表 8-4：PICTURE 文字列における編集用記号

PICTURE 句の文字列中の編集用記号	編集の結果挿入される文字	
	データ項目の値が正またはゼロ	データ項目の値が負
+	+	-
-	空白1個	-
CR	空白2個	CR
DB	空白2個	DB

浮動挿入編集

通貨編集用文字および符号編集用文字の"+"と"-" は浮動挿入文字である。これらの文字を、1 つの PICTURE 文字列に同時に指定することはできない。

浮動挿入編集を行うには、PICTURE 文字列中に浮動挿入文字を 2 つ以上指定する。浮動挿入文字列の間または右隣に、単純挿入文字を書くことができる。それらの単純挿入文字は、浮動挿入文字列の一部となる。

浮動挿入文字に通貨編集用文字を使用した場合、その浮動挿入文字列の右隣に符号編集用文字の"+", "-", "CR" ,"DB" のどれか 1 つを書くことができる。

浮動挿入文字列の左右両端の文字はそれぞれ、データ項目中で浮動する記号の左右の限界を示す。

左端から 2 番目の浮動文字は、そのデータ項目の中で数字データを収めることができる左側の限界を示す。それ以降右側の浮動文字は、ゼロでない数字で置き換えられる。

PICTURE 文字列の中では、浮動挿入文字を表わす方法は 2 通りである。1 つは、小数点の左側の数字列の先頭の一部または全部を、挿入文字で書く方法である。もう 1 つは、PICTURE 文字列の中のすべての数字位置を、挿入文字で書く方法である。

浮動挿入文字列の左右両端の文字はそれぞれ、データ項目中で浮動する記号の左右の限界を示す。

左端から 2 番目の浮動文字は、そのデータ項目の中で数字データを収めることができる左側の限界を示す。それ以降右側の浮動文字は、ゼロでない数字で置き換えられる。

PICTURE 文字列の中では、浮動挿入文字を表わす方法は 2 通りである。1 つは、小数点の左側の数字列の先頭の一部または全部を、挿入文字で書く方法である。もう 1 つは、PICTURE 文字列の中のすべての数字位置を、挿入文字で書く方法である。

ANS85 PICTURE 文字列の中のすべての数字位置を挿入文字で書いた場合、最低限 1 つの数字位置を、想定小数点または実小数点の左に置く。

切捨てが起こらないようにするためには、受取り側の PICTURE 句の文字列の文字数を、送出し側のデータ項目の文字数に受取り側で編集するための固定挿入文字の数と浮動挿入文字のための 1 字とを加えたもの以上にする。

ゼロ抑制編集

数字位置の先行ゼロ列を抑制するためには、PICTURE 文字列の中で英字 "Z" または "*" (星印) を使用する。1 つの PICTURE 文字列の中では、どちらか一方しか指定してはならない。各抑制文字は、項目の大きさに含められる。抑制文字に "Z" を指定した場合、先行ゼロ列は空白で置き換えられる。抑制文字に"*" を指定した場合、先行ゼロ列は星印で置き換えられる。

ゼロ抑制は、PICTURE 文字列の中に 1 つ以上の抑制文字を並べて示す。この抑制文字列は、先行ゼロ列のゼロの置き換えを行おうとする先行数字位置を示す。抑制文字列の中またはそのすぐ右に書いた単純挿入文字は、その抑制文字の一部となる。

PICTURE 文字列の中では、ゼロ抑制文字を表わす方法は 2 通りである。1 つは、小数点の左側の数字列の先頭の一部または全部を、ゼロ抑制文字で書く方法である。もう 1 つは、PICTURE 文字列の中のすべての数字位置を、ゼロ抑制文字で書く方法である。

PICTURE 文字列中の小数点の左側にだけゼロ抑制文字を書いた場合、PICTURE 文字列中のゼロ抑制文字に対応するデータ中の先行ゼロ列が、置換え文字によって置き換えられる。ゼロ抑制が停止される位置は、小数点またはデータ中のゼロでない最初の数字の、どちらか先に出てきた方となる。

PICTURE 文字列の中のすべての数字位置をゼロ抑制文字で書いた場合、結果はデータの値によって異なる。 値がゼロでない場合は、小数点の左側にだけゼロ抑制文字を書いた場合と同じ結果となる。値がゼロである場合は、ゼロ抑制文字に"Z" を使用していれば、そのデ

ータ項目全体は空白で置き換えられ、ゼロ抑制文字に"*" を使用していれば、そのデータ項目全体は実小数点を除いて星印で置き換えられる。

記号文字"+", "-", "*", " Z" および通貨編集用文字は、浮動置換え文字として使用するときは、1 つの PICTURE 文字列の中ではどれか 1 つだけを書く。

順序規則

PICTURE 文字列の中で記号として使用するときの、文字の順序規則を表 8-5 に示す。"○" 印は、その列の上段に示した文字を、その行の左端に示した文字の前に書いてもよいことを示す。中かっこ{ }で囲んだ文字群は、その中の 1 つしか書けないことを示す。 " 通貨" は、通貨編集用文字を示す。PICTURE 文字列の中には、文字 "A " , "○", "Z", " 9", "*", の少なくともどれか 1 つ、または記号文字 "+" , "-"," 通貨" の少なくともどれか 2 つを書く。

表 8-5 の中で、固定挿入文字の"+"と"-" 、および浮動挿入文字の"Z", "*", " +","-", " 通貨", および記号文字"P" は、それぞれ 2 回ずつ示してある。左の列および上の行は、小数点の左側に書く場合を表わす。右の列および下の行は、小数点の右側に書く場合を表わす。

表 8-5：PICTURE 文字列における順序規則

先行文字		固定挿入文字										浮動挿入文字						その他の文字						
後続文字		B	0	/	,	.	+ -	+ -	CR DB	通貨	E	Z *	Z *	+ -	+ -	通貨	通貨	9	A X	S	V	P	P	G
固定挿入文字	B	x	x	x	x	x	x			x		X	x	x	x	x	x	x	x		x		x	x
	0	x	x	x	x	x	x			x		X	x	x	x	x	x	x	x		x		x	
	/	x	x	x	x	x	x			x		X	x	x	x	x	x	x	x		x		x	
	,	x	x	x	x	x	x			x		X	x	x	x	x	x	x			x		x	
	.	x	x	x	x		x			x		X		x		x		x						
	+ -																							
	+ -	x	x	x	x	x				x	x	X	x			x	x	x			x	x	x	
	CR DB	x	x	x	x	x				x		X	x			x	x	x			x	x	x	
	通貨						x																	
	E				x	x												x			x			
浮動	Z *	x	x	x	x		x			x		X												

挿入文字	Z*	x	x	x	x	x	x			x		X	x								x		x	
	+-	x	x	x	x					x				x										
	+-	x	x	x	x	x				x				x	x						x			
	通貨	x	x	x	x		x									x								
	通貨	x	x	x	x	x	x									x	x				x			
その他の文字	9	x	x	x	x	x	x			x	x	X		x		x		x	x	x	x		x	
	AX	x	x	x														x	x					
	S																							
	V	x	x	x	x		x			x		X		x		x		x		x		x		
	P	x	x	x	x		x			x		X		x		x		x		x		x		
	P						x			x										x	x		x	
	G	x																						x

8.17 PROPERTY(属性)句

機能

(S0100)

PROPERTY (属性) 句はこのデータ項目がオブジェクトの属性であることおよび状況に応じて GET メソッドと SET メソッドのどちらか一方または両方を生成すべきことを指定する。

一般形式

構文規則

1. PROPERTY 句を指定できる場所は、ファクトリ定義またはオブジェクト定義の作業場所節だけである。
2. OCCURS 句の下位に属するデータ項目には PROPERTY 句を指定してはならない。

3. PROPERTY 句を指定できる対象は、参照の一意性を保つために名前を修飾する必要のない、基本データ項目だけである。
4. 記述項の左辺のデータ名はスーパークラスに定義されている属性名と同じであってはならない。

注: スーパークラス内に属性名を定義できる。その方法は2通りある。ひとつは、PROPERTY 指定を記述したひとつまたは一組のメソッドを定義することである。もうひとつは、PROPERTY 句を指定したデータ記述項を記述することである。

一般規則

1. GET 指定を書かないと、PROPERTY 句は該当のデータ項目を含むファクトリまたはオブジェクトに関するメソッドを定義する働きをする。

   その記述項の左辺の字類が指標、オブジェクト、ポインタのいずれかである場合、そのメソッドは暗黙的に下記のように定義される。

```
METHOD-ID. GET PROPERTY データ名.
DATA DIVISION.
LINKAGE SECTION.
01 LS データ名 データ記述.
PROCEDURE DIVISION RETURNING LS データ名.
パラメータ名.
        SET LS データ名 TO データ名
        EXIT METHOD.
END METHOD.
```

   その記述項の左辺の項類が英数字編集、国別編集、数字編集のいずれかである場合、そのメソッドは暗黙的に下記のように定義される。

```
METHOD-ID. GET PROPERTY データ名.
DATA DIVISION.
LINKAGE SECTION.
01 LS データ名 データ記述.
PROCEDURE DIVISION RETURNING LS データ名.
パラメータ名.
        MOVE データ名 TO LS データ名 (1:).
        EXIT METHOD.
END METHOD.
```

注: その記述項の左辺が編集されている場合、受取り側の項目全体を部分参照したときに、そのデータに編集規則が再び適用されることはない。

そうでない場合、このメソッドの暗黙的な定義は下記のようになる。

```
METHOD-ID. GET PROPERTY データ名.
DATA DIVISION.
LINKAGE SECTION.
01 LS データ名 データ記述.
PROCEDURE DIVISION RETURNING LS データ名.
パラメータ名.
        MOVE データ名 TO LS データ名
        EXIT METHOD.
END METHOD.
```

ここで、LS データ名のデータ記述には、その記述項の左辺およびその下位のデータ項目が含まれる。ただし、下記のものは例外である。

- INDEXED BY 指定
- PROPERTY 句
- VALUE 句
- その記述項の左辺のデータ記述内の REDEFINES 句

2. SET 指定を書かないと、PROPERTY 句は該当のデータ項目を含むファクトリまたはオブジェクトに関するメソッドを定義する働きをする。

   その記述項の左辺の字類が指標、オブジェクト、ポインタのいずれかである場合、そのメソッドは暗黙的に下記のように定義される。

```
METHOD-ID. SET PROPERTY データ名.
DATA DIVISION.
LINKAGE SECTION.
01 LS データ名 データ記述.
PROCEDURE DIVISION USING LS-data-name.
パラメータ名.
        SET データ名 TO LS データ名
        EXIT METHOD.
END METHOD.
```

その記述項の左辺の項類が英数字編集、国別編集、数字編集のいずれかである場合、そのメソッドは暗黙的に下記のように定義される。

```
METHOD-ID. SET PROPERTY データ名.
DATA DIVISION.
LINKAGE SECTION.
01 LS データ名 データ記述.
PROCEDURE DIVISION USING LS データ名.
パラメータ名.
        MOVE LS データ名 TO データ名 (1:).
        EXIT METHOD.
END METHOD.
```

注: その記述項の左辺が編集されている場合、受取り側の項目全体を部分参照したときに、そのデータに編集規則が再び適用されることはない。

そうでない場合、このメソッドの暗黙的な定義は下記のようになる。

```
METHOD-ID. SET PROPERTY データ名.
DATA DIVISION.
LINKAGE SECTION.
01 LS データ名 データ記述.
PROCEDURE DIVISION USING LS データ名.
パラメータ名.
        MOVE LS データ名 TO データ名
        EXIT METHOD.
END METHOD.
```

ここで、LS データ名のデータ記述には、その記述項の左辺およびその下位のデータ項目が含まれる。ただし、下記のものは例外である。

- INDEXED BY 指定
- PROPERTY 句
- VALUE 句
- その記述項の左辺のデータ記述内の REDEFINES 句

8.18 RECORD (レコードの大きさ) 句

機能

(ANS85)RECORD (レコードの大きさ) 句は、固定長レコード内の文字数または可変長レコード内の文字数の範囲を指定する。可変長レコードに関しては、最小文字数と最大文字数とを指定する。

(MF)この節全体を通じて使用される固定形式のファイルおよび可変形式のファイルの概念は、直接は行順ファイルに適用されない。使用する COBOL での行順ファイルの使用についての詳細は、COBOL システムのマニュアルを参照。

(XOPEN)標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、RECORD CONTAINS 句は明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内ではこの句を使用するべきではない。

一般形式

形式 1

```
RECORD CONTAINS 整数-1 CHARACTERS
```

形式 2

```
RECORD IS VARYING IN SIZE                                   (ANS85)
      [[FROM 整数-2][TO 整数-3]CHARACTERS]
      [DEPENDING ON データ名-1]
```

形式 3

```
RECORD CONTAINS 整数-4 TO 整数-5 CHARACTERS
```

構文規則

形式 1

1. 該当するファイルのレコード記述項に記述する文字数は、整数-1 と同じにする。

形式 2

2. (ANS85)該当するファイルのレコード記述項に記述する文字数は、整数-2 に指定する文字数より少なくてはならない。また、整数-3 に指定する文字数より多くてはならない。
3. (ANS85)整数-3 は、整数-2 よりも大きくする。

4. (ANS85)データ名-1 は、作業場所節または連絡節中で符号なしの整数の基本項目として記述しておく。

一般規則

すべての形式

1. RECORD 句を指定しないと、レコード記述項に記述する各データの大きさの和がレコードの大きさとなる。
2. (ANS85)対応す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と関連する実行単位中のすべてのファイル記述項において、整数-1、整数-2、整数-3 に同じ値を指定しなければならない。RECORD 句を指定しない場合、そのファイル結合子に関連するレコード記述項は、すべて同じ長さにする。
3. (MF)(XOPEN)行順ファイルは、固定形式ファイルでも可変形式ファイルでもない。このため、RECORDING MODE、RECORD CONTAINS、RECORD VARYING IN SIZE 句のどれかを書いても書かなくても、その性能には影響しない。

形式 1

4. 形式 1 は、固定長レコードを定義するために使用する。整数-1 には、ファイル中の各レコードの文字数を指定する。

形式 2

5. (ANS85)形式 2 は、可変長レコードを定義するために使用する。整数-2 には、ファイル中のレコードの中で最短のものの文字数を指定する。整数-3 には、ファイル中のレコードの中で最長のものの文字数を指定する。
6. (ANS85)レコードの文字数は、その中に含まれる基本データ項目の文字数を合計することによって決まる。ただし、再定義および再命名分を除く。また、桁詰めによって暗黙の FILLER がとられている場合は、それも数に含める。表を指定した場合は、下記のように扱われる。
    a. レコード中に記述した表要素の最小数が、上記の合計に含められて、そのレコードの最短文字数が決定される。
    b. レコード中に記述した表要素の最大数が、上記の合計に含められて、そのレコードの最長文字数が決定される。
7. 整数-2 を指定しないと、該当するファイルのレコードの最短文字数は、そのファイルのレコード用に記述した最小の文字数に等しいものとされる。
8. (ANS85)整数-3 を指定しないと、該当するファイルのレコードの最長文字数は、そのファイルのレコード用に記述した最大の文字数に等しいものとされる。
9. (ANS85)データ名-1 を指定した場合、該当するファイルに対して RELEASE 文、REWRITE 文、WRITE 文を実行する前に、該当するレコードの文字数をデータ名-1 のデータ項目に設定する。
10. (ANS85)データ名-1 を指定した場合、DELETE 文、RELEASE 文、REWRITE 文、START 文、WRITE 文を実行しても、データ名-1 のデータ項目の内容は変わらない。READ 文または RETURN 文の実行が不成功に終わった場合も、データ名-1 のデータ項目の内容は変わらない。

11. ANS85 RELEASE 文、REWRITE 文、WRITE 文を実行する間、該当するレコード中の文字数は、下記の条件によって決定される。
    a. データ名-1 を指定した場合、データ名-1 のデータ項目の内容によって。
    b. データ名-1 を指定せず、レコードに可変反復データ項目が含まれない場合、そのレコードの文字数によって。
    c. データ名-1 を指定せず、レコードに可変反復データ項目が含まれる場合、固定部分と出力文の実行時の反復数に基づく可変な表部分との和によって。

    書き出す論理レコードの文字数が、整数-2 よりも少ないか整数-3 よりも多いと、出力文は不成功に終わる。そして、対応する入出力状態に、その原因となった条件を示す値が設定される。ただし、RELEASE 文の実行中は除く。

12. ANS85 データ名-1 を指定した場合、該当するファイルに対する READ 文または RETURN 文の実行が正常に終了すると、データ名-1 のデータ項目の内容は、読み込んだばかりのレコードの文字数を示すように変更される。
13. ANS85 READ 文または RETURN 文の中に INTO を指定すると、暗黙の MOVE 文の送出し側データ項目となる現在レコード中の文字数は、下記の条件によって決定される。
    a. データ名-1 を指定した場合は、データ名-1 のデータ項目の内容によって。データ名-1 を指定した場合は、データ名-1 のデータ項目の内容によって。
    b. データ名-1 を指定しなかった場合は、データ名-1 を指定していたならばデータ名-1 のデータ項目に転記されるはずの値によって。

形式 3

13. RECORD 句の形式 3 を使用すると、整数-4 が最短のデータレコードの文字数を表わし、整数-5 が最長のデータレコードの文字数を表わす。しかし、この場合は、各データレコードの大きさは、レコード記述項に完全に定義されている。
14. 各データレコードの大きさは、論理レコードを格納するのに必要な文字数で指定する。その際、論理レコード内のデータ項目を表わすために使用される文字の型は問わない。レコードの大きさは、すべての固定長の基本項目の文字数の合計に、可変長項目が含まれればその最大の文字数の合計を加えたものによって、決定される。この合計値が実際のレコードの大きさと違っていてもよい。*COBOL 言語の概念*の章の*文字の表現と基数の選定*節、およびこの章の *SYNCHRONIZED(桁詰め)句*と *USAGE(用途)句*の各節を参照。

8.19 RECORDING MODE 句

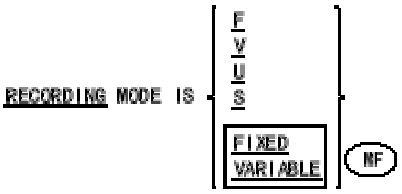
OSVS VSC2 MF

機能

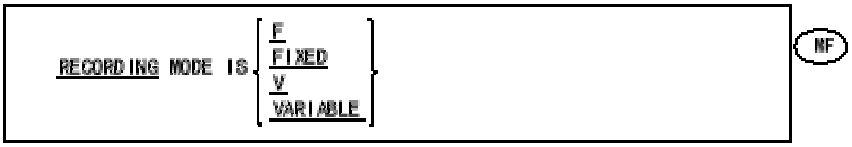
RECORDING MODE 句は、ファイル中の論理レコードの形式を指定する。

一般形式

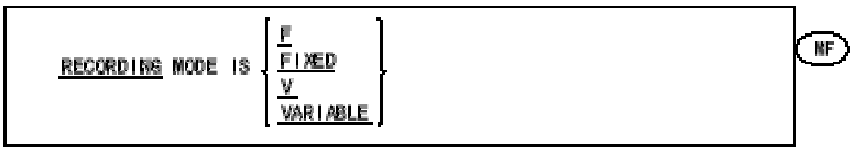
形式 1 (レコード順ファイル)



形式 2 (行順ファイル)



形式 3 (相対ファイルおよび索引ファイル)



構文規則

1. MF F と FIXED は同義語である。
2. MF V と VARIABL は同義語である。

一般規則

すべての形式 (すべてのファイル)

1. RECORDING MODE IS F を指定すると、ファイル中のすべてのレコードの長さが同じとなる。

形式 1 (レコード順ファイル)

2. MF "U" 指定は注記になる。

形式 1 および 3 (レコード順ファイル、相対ファイルおよび索引ファイル)

3. RECORDING MODE IS V 句を指定すると、ファイル中のレコードは固定長でも可変長でもよくなる。各データレコードに、レコード長フィールドがとられる。このレコード長フィールドは、レコード記述の一部とは扱われない。

形式 2

4. (MF)行順ファイルは、固定形式ファイルでも可変形式ファイルでもない。このため、RECORDING MODE、RECORD CONTAINS、RECORD VARYING IN SIZE 句のどれかを書いても書かなくても、その性能には影響しない。

8.20 REDEFINES (再定義) 句

機能

REDEFINES (再定義) 句は、同一の計算機記憶領域を別のデータ記述項によって記述する。

一般形式

```
レベル番号 [ データ名-1 ]
           [ FILLER (ANS85) ] REDEFINES データ名-2
```

上記のレベル番号とデータ名-1

(ANS85)または FILLER

は形式を明確にするために示してある。レベル番号、データ名-1

(ANS85)および FILLER

はいずれも、REDEFINES 句の一部を構成するわけではない。

構文規則

1. REDEFINES 句を指定するときは、データ名-1

   (ANS85)または FILLER

   の直後に書く。

   (MF)あるいは、PICTURE 句または USAGE 句の後ろに REDEFINES 句を書くことができる。

2. データ名-1 とデータ名-2 のレベル番号は、同じにする。ただし、66、

   (MF)78、

   および 88 は、いずれも使用できない。

3. REDEFINES 句は、ファイル節内の 01 レベルの記述項の中では使用できない。レベル指示語の FD または SD の下位に属する複数の 01 レベルの記述項は、同じ領域を暗黙的に再定義するからである。(この章の *DATA RECORDS(データレコード)句*節の一般規則 2 を参照。)

   MF この句を、ファイル節内の 01 レベルの記述項内で使用してもよい。

4. OSVS VSC2 データ名-2 のデータ記述項に、REDEFINES 句が含まれていてもよい。

   データ名-2 は、REDEFINES 句が含まれている記述項の下位に属していてもよい。データ名-2 のデータ記述項には、OCCURS 句が含まれていてはならない。しかし、OCCURS 句が含まれているデータ記述項の下位に、データ名-2 が属していても構わない。この場合、REDEFINES 句の中でデータ名-2 を参照するときに、添字または指標は指定できない。元の定義および再定義のどちらも、OCCURS 句で指定される可変反復データ項目を含んでいてはならない。(この章の *OCCURS(反復)句*節を参照。)

   MF データ名-2 のデータ記述項に、OCCURS 句が含まれていてもよい。

   MF レベル番号が 01 の場合、元の定義と再定義は、OCCURS 句で定義したサイズが可変の項目を含むことができる。

5. データ名-2 とデータ名-1 のデータ記述項の間に、データ 2 およびデータ 1 のレベル番号よりも数値的に値の小さいレベル番号の記述項は置けない。
6. OSVS VSC2 MF 外部浮動小数点数データ項目または内部浮動小数点数データ項目を、REDEFINES 句の左辺または右辺に置いてもよい。
7. ANS85 データ名-2 が一意でない場合でも、データ名-2 を修飾してはならない。この場合は、REDEFINES 句の配置に関する規則上、データ名-2 に曖昧さはないからである。

   OSVS データ名-2 を修飾してもよいが、記述した修飾は無視される。

8. 新しい文字位置の記述を与える項目は、VALUE 句を持ってはならない。ただし、条件名の項目内では例外である。
9. ISO2002 REDEFINES 句は、字類オブジェクトのデータ項目に指定してはならない。

   MF REDEFINES を、字類オブジェクトのデータ項目に指定してもよい。

10. ISO2002 データ名-2 は、字類オブジェクトであってはならない。

    MF データ名-2 は、字類オブジェクトでもよい。

11. ISO2002 MF データ名-2 は、ANY LENGTH 句を使用して記述してはならない。

一般規則

1. 再定義はデータ名-2 から始まり、データ名-2 のレベル番号以下のレベル番号が出てきたところで終わる。

2. データ名-1 のレベル番号が 01 でないときは、データ名-2 によって参照されるデータ項目と同じ文字数を指定する。

   ANSI85 OSVS ただし、データ名-1 の領域が、データ名-2 の領域よりも小さくてもよい場合を除く。

   MF ただし、データ名-1 の領域がデータ名-2 の領域よりも大きくてもよい場合を除く。この場合、再定義する項目と再定義される項目の大きい方を収められるだけの記憶領域がとられる。

   REDEFINES 句は記憶領域を再定義するのであって、記憶されているデータ項目を再定義するのではないことを、認識することが重要である。

3. 同じ文字位置を、複数回、再定義できる。 再定義用の記述項は、再定義される領域を定義する記述項の直後に続ける。その間に、新しい領域を定義する記述項ははさめない。同じ領域を複数回再定義するときは、最初に領域を定義した記述項のデータ名を使用する。

   OSVS VSC2 MF または、先行する再定義記述項のデータ名を使用してもよい。

4. レベル指示語 (FD または SD) の下位に属する複数の 01 レベルの記述項は、同じ領域を暗黙的に再定義する。

8.21 RENAMES(再命名)句

機能

RENAMES (再命名) 句は、基本項目を組み合わせて新たな集団を作る。組合わせは互いに重ね合わせることができる。

一般形式

```
66 データ名-1 RENAMES データ名-2 [{THROUGH} データ名-3].
                                  {THRU}
```

上記のレベル番号 66 とデータ名-1 は書き方を明確にするために示してあるだけで、レベル番号とデータ名-1 は RENAMES 句の一部を構成するわけではない。

構文規則

1. 1 つの論理レコードに対して書ける RENAMES 記述項の数に、制限はない。
2. 1 つの論理レコード中のデータ項目を参照している RENAMES 記述項はすべて、そのレコード記述項の最後のデータ記述項の直後にまとめて書く。

3. データ名-2 およびデータ名-3 は、同一レコード中の基本項目の名前または集団項目の名前とする。しかし、同じデータ名であってはならない。66 レベルの記述項は、77 レベル、

   (MF)78 レベル、

   88 レベル、01 レベルまたは別の 66 レベルの記述項を再命名することはできない。

   (MF)66 レベルの記述項は、別の 66 レベルの記述項、77 レベルまたは 01 レベルの記述項再命名することができる。

4. データ名-1 は、修飾語として使用できない。データ名-1 は対応する 01 レベルまたは FD または SD の記述項の名前によってだけ修飾できる。データ名-2 の記述項にもデータ名-3 の記述項にも、OCCURS 句が含まれていてはならない。また、データ名-2 の記述項もデータ名-3 記述項も、データ記述項に OCCURS 句が含まれる項目の下位に属していてはならない。(前述の *OCCURS(反復)句*節を参照。)
5. データ名-3 によって表わされる領域の始点は、データ名-2 によって表わされる領域の始点よりも左側にあってはならない。データ名-3 によって表わされる領域の終点は、データ名-2 によって表わされる領域の終点よりも右側にあってはならない。したがって、データ名-3 はデータ名-2 の下位に属していてはならない。
6. データ名-2 およびデータ名-3 は修飾できる。
7. THRU と THROUGH は同義語であり、どちらを書いてもよい。
8. データ名-2 からデータ名-3 の範囲内にある項目には、*OCCURS(反復)句*節で定義されているように、可変反復データ項目が含まれていてはならない。

一般規則

1. データ名-3 を指定すると、データ名-1 は集団項目となる。この集団項目には、データ名-2 (データ名-2 が基本項目の場合) またはデータ名-2 の中の最初の基本項目 (データ名-2 が集団項目の場合) から始まり、データ名-3 (データ名-3 が基本項目の場合) またはデータ名-3 の中の最後の基本項目 (データ名-2 が集団項目の場合) で終わる、すべての基本項目が含まれる。
2. データ名-3 を指定しないと、データ名-2 のすべての属性がデータ名-1 のデータの属性となる。
3. (OSVS)データ名-2 およびデータ名-3 を明示的に修飾しなかった場合、参照の曖昧性がなければ、01 レベルの項目による暗黙の修飾がなされる。

8.22 SIGN(符号)句

機能

SIGN (符号) 句は、演算符号の特性を明示的に記述する必要がある場合に、その演算符号の位置と表現形式を指定する。

一般形式

[SIGN IS] { LEADING / TRAILING } [SEPARATE CHARACTER]

構文規則

1. SIGN 句を指定する対象となるのは、PICTURE 文字列に "S" が含まれる数字データ記述項またはそのような数字データ記述項が少なくとも 1 つ含まれる集団項目だけである。
2. SIGN 句を適用する対象の数字データ項目の用途は、明示的または暗黙的に、DISPLAY とする。
3. 数字データ記述項に SIGN 句を記述する場合は、1 つだけとする。

   (ANS85)この規則は削除された。

4. ファイル記述項の中で CODE-SET (符号系) 句を指定するときは、そのファイル記述項の下位に属する符号付きの数字項目に対して、SIGN IS SEPARATE (独立符号) 句を記述する。

   (MF)この規則は強制しない。*CODE-SET(符号系)句*節を参照。

一般規則

1. SIGN 句は、それを指定した数字データ項目またはそれを指定した集団項目の下位に属する数字データ項目の、演算符号の位置と表現形式を指定する。SIGN 句は、PICTURE 文字列に"S" が含まれる数字データ記述項に対してだけ、適用できる。PICTURE 文字列中の "S" は符号が付くことを表わすだけであり、符号の表現形式や位置を表わすわけではない。
2. PICTURE 文字列に "S" が含まれるが SIGN 句は指定されていない数字データ項目は、演算符号をもつが、"S" によって符号の表現形式や位置が指定されているわけではない。このような符号付き数字データ項目には、一般規則の 3 から 5 は当てはまらない。SIGN 句を指定しない場合の演算符号の表現は、*COBOL 言語の概念*の章の*文字の表現と基数の選定*節を参照。
3. SEPARATE CHARACTER 句を指定しないと、符号の位置と表現形式は下記のようになる。
   a. 演算符号は、LEADING または TRAILING に応じて、基本データ項目の左端または右端の桁位置に付く。具体的な符号の付き方は、*COBOL 言語の概念*の章の*文字の表現と基数の選定*節を参照。
   b. PICTURE 文字列中の記号文字"S" は、標準データ形式で表わした項目の大きさには含まれない。
4. SEPARATE CHARACTER 句を指定すると、符号の位置と表現形式は下記のようになる。
   a. 演算符号は、LEADING または TRAILING に応じて、基本データ項目の左端または右端の桁位置に付く。これは桁位置には入らない。

    b. PICTURE 文字列中の記号文字 "S" は、標準データ形式で表わした項目の大きさに含まれる。
    c. 正号および負号は、それぞれ、標準データ形式文字の"+" と "-" である。
5. PICTURE 文字列に記号文字"S" が含まれる数字データ記述項はすべて、符号付数字データ記述項となる。SIGN 句付の項目に対して計算や比較のために変換が必要なときには、その変換は自動的に行われる。
6. (ANS85)SIGN 句付の集団項目の下位に属する項目 (基本数字項目でも集団項目でも) に SIGN 句を指定すると、下位の項目に指定した SIGN 句が優先する。
7. (OSVS)(VSC2)(MF)外部浮動小数点数項目に関しては、SIGN 句は注記としてだけ用いられる。内部浮動小数点数項目に対しては、SIGN 句は無効であり、使用すると診断メッセージが出される。

8.23 SYNCHRONIZED(桁詰め)句

機能

SYNCHRONIZED (桁詰め) 句は、計算機記憶の固有の境界に従い、基本項目の配置を指定する。

(XOPEN)標準 COBOL 定義の一部を構成するにもかかわらず、X/Open COBOL 言語定義では、この機能は明示的に除外されている。したがって、X/Open に準拠する COBOL 原始プログラムにおいてはこの機能を使用するべきではない。

一般形式

```
{SYNCHRONIZED} [LEFT]
{SYNC        } [RIGHT]
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - IBMCOMP - NOIBMCOMP を指定しないと、SYNCHRONIZED 句は説明のためだけの記述として扱われる。

構文規則

1. SYNCHRONIZED 句は、基本項目に対してだけ指定できる。

   (OSVS)(VSC2)集団項目に対しても、SYNCHRONIZED 句を指定できる。

2. SYNC は、SYNCHRONIZED の省略形である。

一般規則

1. SYNCHRONIZED RIGHT は、基本項目を計算機の語の境界の右端の文字位置で終わらせるように割り付ける。

   この句は、IBMCOMP 指令を設定したときだけ、効力を発揮する。

2. SYNCHRONIZED 句の中で LEFT 指定を指定しても、効力はない。
3. SYNCHRONIZED 句の効力は、定義上、作成者がどのように実装したかに左右される。
4. SYNCHRONIZED 句を指定すると、計算機の語の左右の境界の間に、対象とするもの以外はいっさい入り込まないように、データ項目が割り付けられる。

   割付対象のデータ項目を格納するために必要な文字位置の個数が、計算機の語の左右の境界の間の文字位置の個数よりも少ないとき、使用されない文字位置は他のデータ項目のためには使われない。しかし、この使用されない文字位置は、下記のようになる。

   a. その基本項目が属する集団項目の大きさには含まれる
   b. そのデータ項目が REDEFINES 句の右辺であるときには、再定義される文字位置に含まれる

   SYNCHRONIZED 句によって基本項目の大きさが変更されることはないが、余分の文字位置が割り当てられる。

5. SYNCHRONIZED 句に RIGHT も LEFT も指定しないと、実行効率がよくなるように、基本項目が計算機の語の左右の境界の間に割り付けられる。
6. SYNCHRONIZED 句を指定した項目が参照され 、その大きさに依存する操作 (桁寄せ、切捨て、桁あふれなど) が行われるときには、PICTURE 句に指定した元の大きさが用いられる。
7. データ項目に SYNCHRONIZED 句および演算符号を指定すると、そのデータ項目の符号は通常の演算符号の位置に付けられる。この場合、SYNCHRONIZED LEFT または SYNCHRONIZED RIGHT のどちらを指定したかは、演算符号の位置に影響しない。
8. OCCURS 句を伴うデータ項目のデータ記述項、またはその下位に属するデータ項目のデータ記述項に SYNCHRONIZED 句を指定すると、下記のようになる。
   a. 反復される各データ項目が桁詰めされる。
   b. 同じ表の中の他のデータ項目のために暗黙的に生成される FILLER は、反復される各データ項目に対して生成される。
9. OSVS VSC2 集団項目に SYNCHRONIZED 句を指定すると、その集団項目に属するすべての項目に適用される。
10. SYNCHRONIZED 句を使用したときの効果については、*COBOL 言語の概念*の章の*文字の表現と基数の選定*節を参照。

8.24 THREAD-LOCAL(スレッド局所)句

MF

機能

THREAD-LOCAL（スレッド局所）句は、スレッド入力された各プログラムのデータ項目またはファイル結合子に、コピーが作成されることを指定する。

一般形式

IS THREAD-LOCAL

構文規則

1. THREAD-LOCAL 句は、作業場所節のファイル記述項またはレコード記述項でのみ指定できる。

一般規則

1. THREAD-LOCAL 句を指定すると、ファイル結合子またはデータ項目のコピーが作成され、プログラムを入力する新しいスレッドが実行されるたびに、その初期状態に設定される。ファイル結合子またはデータ項目は、それらが作成されたときのスレッドにのみ認識される。そして、そのスレッドの実行が完了したとき、またはプログラムに CANCEL 文が実行されたといに削除される。そうでない場合は、そのスレッド中のプログラムの次回の呼出しの際に、データ項目またはファイル結合子が最後に使用されたときの状態となる。
2. THREAD-LOCAL 句を指定しない場合は、ファイル結合子またはデータ項目がプログラムに入力されるすべてのスレッドにより共有される。
3. シングルスレッド環境では、THREAD-LOCAL 句は無視され、プログラムは、この句が指定されていないときと同様に動作する。

8.25 TYPEDEF 句

MF

機能

TYPEDEF 句は、プログラマによって定義された構造体または使用法としてレコードを定義する。

一般形式

IS TYPEDEF

構文規則

1. TYPEDEF は、レベル番号 01 を持つデータ記述項でのみ、指定可能である。

2. TYPEDEF 句を指定すると、次の句は指定できない。
    - EXTERNAL
    - GLOBAL
    - OCCURS
    - REDEFINES
    - SYNCHRONIZED/SYNC
    - VALUE

    集団項目に TYPEDEF 句が指定されると、従属項目は OCCURS または REDEFINES 句を使用して指定できる。

    VALUE 句は、TYPEDEF 句を指定するデータ記述項目にも、また、TYPEDEF 構造体内の条件名 (88 レベル項目) を除く、どの従属項目にも指定できない。

3. TYPEDEF 句がデータ記述に指定された場合、そのデータ記述にデータ名を含めなければならない。つまり、これを明示または暗示の FILLER 句で指定してはならない。
4. TYPEDEF 句が CALL プロトタイプ (プログラム名段落に EXTERNAL 句を含むプログラム) に指定された場合、この TYPEDEF 句は、コンパイル集団の任意の後続原始コード行で参照することができる。

一般規則

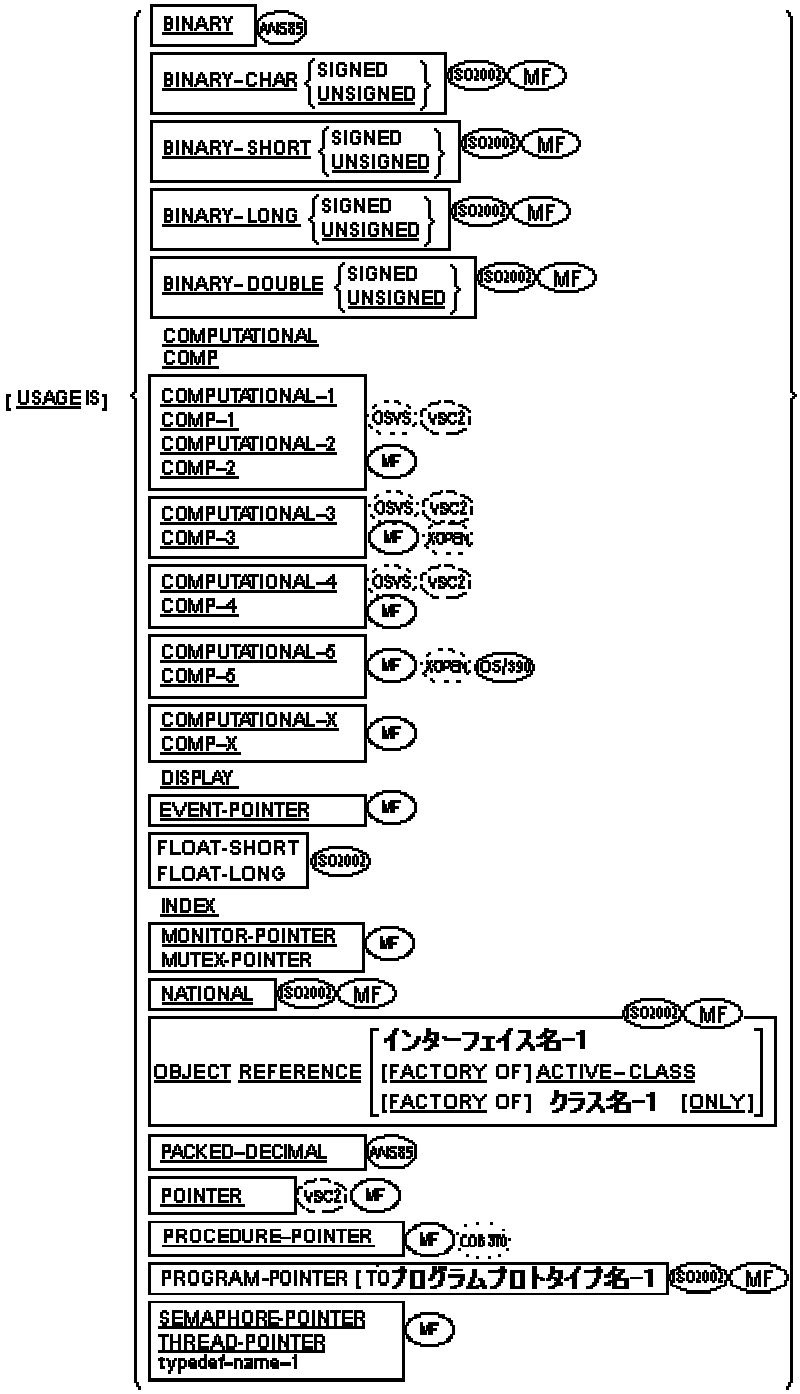
1. TYPEDEF 句の使用目的は、後に USAGE 句で参照できるプログラマ定義の使用法または構造体を作成することにある。
2. TYPEDEF 句を使用して宣言されたレコードは、記憶域を割り当ないが、データ名-1 を型定義名-1 として、後続のデータ記述項に指定できることを宣言する。

8.26 USAGE(用途)句

機能

USAGE (用途) 句は、計算機記憶内のデータ項目の表現形式を指定する。

一般形式



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
   - COBOL370 - USAGE PROCEDURE-POINTER データ項目に割り当てる記憶領域を4バイトとするか8バイトとするかを制御する。
   - COMP-5 - USAGE COMP-5 データ項目に格納するときに、演算符合の取扱いを制御する。

- IBMCOMP - USAGE COMP および USAGE COMP-5 のデータ項目に割り当てる記憶領域を制御する。

構文規則

1. USAGE 句はレベル 66 または 88 のデータ記述項には指定できない。
2. USAGE 句を集団項目の記述項に書いた場合、それに属するすべての基本項目または集団項目の記述項にも書いて構わない。しかし、どちらの項目にも同じ用途を記述しなければならない。
3. 宣言に USAGE 句を持つ基本データ項目、または宣言に USAGE 句を持つ集団項目に従属する基本データ項目は、USAGE 句が COMPUTATIONAL、

   ANS85 BINARY、PACKED-DECIMAL、

   OSVS VSC2 MF OPEN COMPUTATIONAL-3、

   OSVS VSC2 MF COMPUTATIONAL-4、

   OS/390 MF OPEN および COMPUTATIONAL-5

   のどれかを記述している場合、数字項目を定義する PICTURE 文字列で記述しなければならない。(たとえば、PICTURE 文字列が記号 "P", "S", "V", "9" だけを含んでいること。)詳細は、前述の *PICTURE(形式)句*節を参照。

   MF A 宣言に USAGE 句を持たない基本データ項目が、宣言に USAGE 句を持つ集団項目に従属する場合、その USAGE 句は、英字、英数字、英数字編集、数字編集のデータ項目を定義した PICTURE 文字列で記述されたリストの形式で記述してもよい。

4. OS/390 MF OPEN 宣言に USAGE 句を持つ基本データ項目、または宣言に USAGE 句を持つ集団項目に従属する基本データ項目は、USAGE 句が COMPUTATIONAL-5

   MF または COMPUTATIONAL-X

   OS/390 MF OPEN を記述している場合、数字項目を定義する PICTURE 文字列で記述しなければならない。

   MF または、サイズが 1 から 8 バイトの数字項目を定義する PICTURE 文字列で記述しなければならない。サイズが 1, 2, 3, 4, 5, 6, 7, 8 バイトの数字項目の場合、これはそれぞれ 2, 4, 7, 9, 12, 14, 16, 18 の小数点位置である整数項目を記述したことと同じである。

   MF COMPUTATIONAL-X を指定し、PICTURE 文字列が数字項目 定義している場合、その項目には符号を付けてはならない。

5. BLANK WHEN ZERO 句、JUSTIFIED 句、SIGN 句は、用途が明示的にも暗黙的にも DISPLAY でないデータ項目には指定してはならない。
6. SYNCHRONIZED 句および VALUE 句は、用途が INDEX のデータ項目に指定してはならない。

   OSVS VSC2 MF SYNCHRONIZED 句は、用途が INDEX のデータ項目に指定してもよい。

7. COMP は、COMPUTATIONAL の省略形である。

   OSVS VSC2 MF COMP-1 は COMPUTATIONAL-1 の省略形である。
   COMP-2 は COMPUTATIONAL-2 の省略形である。

   OSVS VSC2 MF OPEN COMP-3 は、COMPUTATIONAL-3 の省略形である。

   OSVS VSC2 MF COMP-4 は、COMPUTATIONAL-4 の省略形である。

   OS/390 MF OPEN COMP-5 は COMPUTATIONAL-5 の省略形である。

   MF COMP-X は COMPUTATIONAL-X の省略形である。

8. 指標項目を明示的に参照できるのは、SEARCH (表引き) 文、SET (設定) 文、比較条件、手続き部の見出しの USING (使用) 句、

   OSVS VSC2 MF ENTRY (導入) 文の USING 句、

   ANS85 組み込み関数の引数

   ISO2002 MF INVOKE 文の USING 指定、行内メソッド呼出しの引数、

   ISO2002 MF INITIALIZE 文、

   または USINGCALL (呼ぶ) 文の GIVING 句のみである。

9. ISO2002 VSC2 MF ポインタ

   COB370 ISO2002 MF または手続きポインタ

   VSC2 MF 項目を明示的に参照できるのは、SET (設定) 文、比較条件、CALL (呼ぶ) 文の USING 句、手続き部の見出しの USING (使用) 句、ENTRY (導入) 文の USING 句、

   ISO2002 INVOKE 文、行内メソッド呼出しの引数リスト、関数の引数リスト

   のみである。

10. USAGE IS INDEX、

ISO2002 USAGE IS OBJECT、

VSC2 MF USAGE IS POINTER、

MF COBOL370 または USAGE IS PROCEDURE-POINTER

MF USAGE IS EVENT-POINTER、USAGE IS MONITOR-POINTER、USAGE IS MUTEX-POINTER、USAGE IS PROCEDURE-POINTER、USAGE IS SEMAPHORE-POINTER、または USAGE IS THREAD-POINTER

句のどれかで定義された基本データ項目は、条件変数であってはならない。

11. MF 型定義名-1 は、TYPEDEF 句のレコードと同じ原始ファイル内で前に定義しなければならない。
12. MF USAGE 型定義名-1 を指定した場合、以下の句は指定できない。
    - BLANK
    - JUSTIFIED
    - PICTURE
    - SIGN
    - SYNCHRONIZED
    - VALUE
13. MF 同じ階層の高位レベルに明示的な USAGE 句がある場合、USAGE 型定義名-1 の指定はエラーとなる。
14. MF USAGE 型定義名-1 を指定した項目の直後に、従属する項目(78 以外の高位レベル番号の項目)を記述した場合、エラーとなる。
15. ISO2002 USAGE OBJECT REFERENCE 句は、集団項目用のデータ記述項に指定してはならないが、従属する基本データ項目には指定してもよい。
16. ISO2002 USAGE OBJECT REFERENCE 句をファイル節に記述してはならない。

    MF USAGE OBJECT REFERENCE をファイル節に記述してもよい。

注：オブジェクト参照がまだ、アクティブなオブジェクトを参照しているかどうかは、アプリケーションのロジックにより決まる。

---

17. ISO2002 ACTIVE-CLASS 指定は、ファクトリ定義、オブジェクト定義、メソッド定義にだけ記述できる。
18. MF 用途が EVENT-POINTER のデータ項目は、referenced explicitly onlyCLOSE 文、OPEN 文、 SET 文、WAIT 文、または USING 指定でのみ、明示的に参照できる。
19. MF 用途が MONITOR-POINTER、MUTEX-POINTER、または SEMAPHORE-POINTER のデータ項目は、CLOSE 文、OPEN 文、 SET 文、または USING 指定でのみ、明示的に参照できる。

20. (MF)用途が THREAD-POINTER のデータ項目は、START 文、WAIT 文、または USING 指定でのみ、明示的に参照できる。
21. (2002)(MF)記述に USAGE NATIONAL 句を含む基本データ項目、または記述に USAGE NATIONAL 句を含む集団項目の下位にある基本データ項目は、各国語型または各国語型編集データ項目を記述する PICTURE 文字列で宣言されなければならない。
22. (2002)(MF)基本データ項目、およびそれが属する集団のどれに対しても USAGE 句を指定しない場合は、以下が適用される。
    a. 明示的または暗黙的な PICTURE 文字列が符号、'N'を含む場合、USAGE NATIONAL 句が暗黙的に指定される。
    b. PICTURE 文字列が符号、'N'を含まない場合は、USAGE DISPLAY 句が暗黙的に指定される。
23. (2002)(MF)OBJECT REFERENCE 指定、POINTER 指定、または PROGRAM-POINTER 指定を伴う USAGE 句は、レベル 1 の基本データ項目以外に指定してはならない。

    (MF)この制限は強制しない。

24. (2002)(MF)プログラムプロトタイプ名-1 を指定する場合、記述項の左辺に TYPEDEF 句を指定しなければならない。

    (MF)この制限は強制しない。

25. (2002)(MF)明示的または暗黙的な PICTURE 文字列が符号、'N'を含む基本データ項目に関連づけられた USAGE 句には、NATIONAL を指定しなければならない。

一般規則

1. USAGE 句を集団のレベルに書くと、その USAGE 句はその集団項目に属するすべての基本項目に適用される。

   (MF)ただし、PICTURE 句があり、データ項目の定義が文字である場合を除く。

2. USAGE 句は、計算機記憶内でのデータ項目の表現形式を指定する。USAGE 句は、データ項目の使い方には影響を与えない。ただし、手続き部の文の中には、作用対象を特定の USAGE 句を伴う基本項目に限定するものがある。USAGE 句は項目の基数または文字表現の種類に影響することがある。用途の形式については、***COBOL 言語の概念***の章の*文字表現と基数の選定*節を参照。
3. USAGE IS BINARY 句は、計算機記憶内で数字項目を表現するのに基数 2 を使用することを定義する。USAGE IS BINARY 句は USAGE IS COMPUTATIONAL 句と同義である。
4. USAGE IS PACKED-DECIMAL 句は、計算機記憶内で数字項目を表現するのに基数 10 を使用することを定義する。またこの句は、各桁位置が計算機記憶内で最小限の構成を占めるように定義する。

OSVS VSC2 MF OPEN USAGE IS PACKED-DECIMAL 句は、USAGE IS COMPUTATIONAL-3 と同義語である。

5. USAGE IS DISPLAY 句は、計算機記憶内でデータ項目を表現するのに(明示的または暗示的に) 標準のデータ形式を使用することを定義する。データ項目は文字境界上で整列させられる。
6. OSVS VSC2 MF USAGE IS DISPLAY 句は、下記の種類の項目に適用できる。
   - 英字
   - 英数字
   - 英数字編集
   - 数字編集
   - 外部浮動小数点数
   - 外部 10 進数 (数字)
7. USAGE IS INDEX 句は、データ項目が指標データ項目であることを指定し、その値は表要素の出現回数と対応するものとする。
8. MOVE (転記) 文または入出力文の中で参照される集団の一部に指標データ項目、

   VSC2 MF ポインタ･データ項目、

   MF DB370 手続きポインタ･データ項目

   が含まれていてもよい。この場合、指標データ項目、

   VSC2 MF ポインタ･データ項目、

   MF DB370 手続きポインタ･データ項目

   の変換は行われない。

9. OSVS VSC2 MF USAGE IS COMPUTATIONAL-4 は、USAGE IS COMPUTATIONAL と同義語である。
10. VSC2 MF USAGE IS POINTER 句を指定すると、対象データ項目の番地を記憶できる。(***手続き部 - SEARCH - XML PARSE*** の章の *SET(設定)文*節を参照。)
11. MF DB370 USAGE IS PROCEDURE-POINTER 句は、データ項目が手続きの番地を記憶する手続きポインタ･データ項目であることを指定する。(***手続き部 - SEARCH - XML PARSE*** の章の *SET(設定)文*節を参照。) 参照する対象の手続きはどんな言語で書いてあってもよい。COBOL の場合、それは入れ子になっていないポインタの手続き部を表わす。下記のどちらかによって識別する。
    - プログラム名段落のプログラム名
    - ENTRY (入口) 文の入口名
12. MF 型定義名-1 を基本項目として指定した場合、「USAGE 型定義名-1」句は、型定義名-1 によって参照されるプログラマによって定義された用途と同じ属性を、基本項目に指定することになる。
13. MF 型定義名-1 を集団項目として指定した場合、「USAGE 型定義名-1」句は同一の構造をした集団項目を指定する働きをする。その効果は、型定義名-1 によって識別されるデータ記述項の下位に属する複数のデータ宣言が、「USAGE 型定義名-1」句

で宣言された項目の下位に、まったく同様に指定されたかのようになる。下位のデータ項目のデータ名は、型定義名-1 によって参照されるプログラマによって定義された構造体の中に宣言されているものと同じになる。それらを一意に参照するには、データ名の修飾を行う。.

14. (ISO/IEC)(MF)USAGE OBJECT REFERENCE 句を用いて記述したデータ項目をオブジェクト参照と呼ぶ。オブジェクト参照は字類がオブジェクトであり項類がオブジェクト参照であるデータ項目である。オブジェクト参照の内容はオブジェクトの参照先または空である。ただし、下記の規則に従う。
    a. オブジェクト参照データ項目に割り当てられる記憶領域のサイズは4バイトである。
    b. 選択指定を何も書かないと、そのデータ項目は一般的オブジェクト参照となる。その内容は任意のオブジェクトの参照先であってよい。
    c. インターフェイス名-1 を指定した場合、そのデータ項目によって参照されるオブジェクトは、インターフェイス名-1 によって参照されるインターフェイスに適合しなければならない。
    d. クラス名-1 を指定した場合、そのデータ項目によって参照されるオブジェクトはクラス名-1 のオブジェクトであるかまたはクラス名-1 のサブクラスでなければならない。ただし、下記の規則に従う。
        1. ONLY 指定を書かなかった場合：
            a. FACTORY 指定を書いた場合、そのデータ項目によって参照されるオブジェクトは、指定したクラスのファクトリ・オブジェクトまたはサブクラスでなければならない。
            b. FACTORY 指定を書かなかった場合、そのデータ項目によって参照されるオブジェクトは、指定したクラスのオブジェクトまたはサブクラスでなければならない。
        2. ONLY 指定を書いた場合：
            a. FACTORY 指定を書いた場合、そのデータ項目によって参照されるオブジェクトは、指定したクラスのファクトリ・オブジェクトでなければならない。
            b. FACTORY 指定を書かなかった場合、そのデータ項目によって参照されるオブジェクトは、指定したクラスのオブジェクトでなければならない。
    e. ACTIVE-CLASS 指定を書いた場合、そのデータ項目によって参照されるオブジェクトは、そのオブジェクト参照が指定されているメソッドを呼び出すのに使用されたオブジェクトが属するクラスのオブジェクトでなければならない。
        1. FACTORY 指定を書いた場合、そのデータ項目によって参照されるオブジェクトは、そのクラスのファクトリ・オブジェクトでなければならない。
        2. FACTORY 指定を書かなかった場合、そのデータ項目によって参照されるオブジェクトは、そのクラスのオブジェクトでなければならない。
15. (MF)USAGE IS EVENT-POINTER 句は、ソフトウェアイベントの発生を知らせる桁詰めデータ項目を識別する。イベントデータ項目の値は、TRUE または FALSE である。
16. (MF)USAGE IS MONITOR-POINTER 句は、モニタと呼ばれる桁詰めデータ項目を識別する。このデータ項目について、読み込みロック、表示ロック、書き込みロックを要求することができる。

注：モニタは通常、複数のスレッドが読み込みまたは書き込みを行う可能性のあるデータ構造を保護するために使用される。

---

17. (MF)USAGE IS MUTEX-POINTER 句は、ミューテックスと呼ばれる桁詰めデータ項目を識別する。このデータ項目を使用して、マルチスレッドアプリケーションの重要な節を保護することができる。ミューテックスデータ項目の値は、ON または OFF である。ON 状態のミューテックスデータ項目を持つことができるのは、一度に 1 つのスレッドのみである。
18. (MF)USAGE IS SEMAPHORE-POINTER 句は、数を数える桁詰めデータ項目を識別する。このデータ項目はマイナスの値を持つことはない。
19. (MF)USAGE IS THREAD-POINTER 句は、スレッドのハンドルを保管するデータ項目を識別する。
20. (ISO/IEC)(MF)明示的または暗黙的な USAGE IS NATIONAL 句は、各国語符号化文字で計算機内に保管され、文字境界に並べられたデータ項目を識別する。各国語文字は、計算機の記憶域で同じ大きさの文字として表示される。各国語型データ (UNICODE) について詳しくは、COBOL システムのマニュアルを参照。
21. (ISO/IEC)(MF)USAGE IS BINARY-CHAR、USAGE IS BINARY-SHORT、USAGE IS BINARY-LONG、および USAGE IS BINARY-DOUBLE の各句は、計算機の記憶域で 2 の基数を持つデータ項目を識別する。これらの用途は、以下に示す PICTURE 文字列で USAGE IS COMP-5 を指定することと同じである。USAGE 句で UNSIGNED が指定されている場合を除き、同等の PICTURE 文字列には S が含まれる。

用途	同等の PICTURE 文字列
BINARY-CHAR	[S]9(2)
BINARY-SHORT	[S]9(4)
BINARY-LONG	[S]9(9)
BINARY-DOUBLE	[S] 9(18)

22. (ISO/IEC)(MF)USAGE IS FLOAT-SHORT 句は、USAGE IS COMPUTATIONAL-1 の指定と同じである。
23. (ISO/IEC)(MF)USAGE IS FLOAT-LONG 句は、USAGE IS COMPUTATIONAL-2 の指定と同じである。
24. (ISO/IEC)(MF)USAGE PROGRAM-POINTER 句を指定するデータ記述項は、手続きポインタデータ項目を指定する。このデータ項目は、プログラムポインタおよび手続きポインタとも呼ばれ、プログラムの位置を含む。（*手続き部 - SEARCH - XML PARSE* の章の *SET(設定)文*節を参照。）手続きは、どの言語で書かれていてもよい。COBOL で書かれている場合は、プログラムの手続き部を表す。これは入れ子になっておらず、以下のいずれかで識別される。
    - プログラム名段落のプログラム名
    - ENTRY 文の記述項名
25. (ISO/IEC)(MF)プログラムプロトタイプ名-1 を指定する場合は、このデータ項目は制限されたプログラムポインタとなる。制限されたプログラムポインタは、定義済みの位置、

NULL または、指定されたプログラムプロトタイプ名で識別されたものと同じシグニチャのプログラムの位置のみを含む。

8.27 VALUE(値)句

機能

VALUE (値) 句は、定数の値、作業場書項目および局所記憶項目の初期値、条件名に関連する値を定義する。

例

1. VALUE 句の NEXT 指定の使用例は、*言語リファレンス - 追加トピックの例*の章の*定数名の NEXT 指定*節に掲載されている。

一般形式

形式 1

```
VALUE IS 定数-1
```

形式 2

```
{VALUE IS  }{ 定数-2 [{THROUGH}定数-3]}...
{VALUES ARE}
    [WHEN SET TO FALSE IS 定数-4]  (NF)
```

形式 3

```
                                                   (NF)
                                                   ┌─┐
                                                   │+│
                                                   │-│
                                                   │*│
                                                   │/│
                                                   │AND│
                                                   │OR│
                                                   └─┘
                                                   ┌─────────────────────┐
                                                   │ 整数-1              │
                                                   │ NEXT                │
                                                   │ LENGTH OF 一意名-3  │
                                                   │ START OF 一意名-4   │
                                                   └─────────────────────┘
                                                   ...
┌──────────────────────────────────┐
│ 定数-5                           │
│ NEXT                             │
│ LENGTH OF 一意名-1               │
│ START OF 一意名-2                │
└──────────────────────────────────┘
VALUE IS
```

形式 4

```
{ VALUE IS   } { {定数-6}... FROM ( { 添え字-1 }... ) [ TO ( { 添え字-2 }... ) ] }...
{ VALUES ARE }
```

(ISO2002) (MF)

構文規則

すべての形式

1. 符号付き数字定数を指定する場合は、記述項の左辺は符号付き数字データ項目でなければならない。
2. 記述項の左辺の項類が数字である場合は、VALUE 句注の定数はすべて数字でなければならず、その値は、PICTURE 句または USAGE 句が示す値の範囲内でなければならない用途が FLOAT-SHORT および FLOAT-LONG 以外である場合は、この値が、切り捨てやゼロ以外の数字を要求するものであってはならない。この値は、PICTURE 文字符号 'P'に対応する各位置では数字のゼロでなければならない。
3. 項目の字類が英字または英数字の場合は、VALUE 句中の定数は英数字定数でなければならない。基本項目の VALUE 句中の英数字定数は、明示的な PICTURE 句で示されたサイズを越えてはならない。英数字集団項目の VALUE 句中の英数字定数は、その集団項目のサイズを越えてはならない。
4. 項目の字類が各国語型の場合は、VALUE 句中の定数は各国語型定数でなければならない。基本項目の VALUE 句中の各国語型定数は、明示的な PICTURE 句で示されたサイズを越えてはならない。各国語型集団項目の VALUE 句中の各国語型定数は、その集団項目のサイズを越えてはならない。

形式 1 および 4

5. VALUE 句は、REDEFINES 句を含むデータ記述項、および REDEFINES 句を含む記述項の下位の記述項のいずれでも指定してはならない。
6. 集団レベルで VALUE 句を指定した場合は、定数がその集団項目と同じ項類でなければならない。または、その項類の受取り側項目への MOVE 文で許可された表意定数でなければならない。この集団内の下位のレベルでは、VALUE 句を指定できない。
7. 集団レベルで VALUE 句を指定した場合は、その集団内の下位のレベルを JUSTIFIED 句または SYNCHRONIZED 句で記述してはならない。また、英数字集団項目の下位にあるすべてのデータ項目は、明示的または暗黙的に USAGE DISPLAY で記述しなければならない。

形式 1

6. (OSVS)形式 1 で VALUES ARE を使用できる。
7. (OSVS)(VSC2)(MF)外部浮動小数点数データ項目には、VALUE 句を指定できない。

   (MF)外部浮動小数点数データ項目に、VALUE 句を指定できる。

8. OSVS VSC2 MF USAGE COMP-1 または USAGE COMP-2 を指定した内部浮動小数点数データ項目に、VALUE 句を指定できる。この場合の定数-1 は、浮動小数点定数、表意定数 ZERO、値ゼロを表す数字定数のどれかとする。

形式 2

7. THRU と THROUGH は同義語であり、どちらを書いてもよい。
8. 1 つのデータ項目に、形式 2 の記述項をいくつでも書ける。
9. 形式 2 は、条件名に関してだけ使用できる。
10. 条件名記述項では、VALUE 句を必ず書く。条件名記述項に書けるのは、VALUE 句と条件名自体である。
11. 定数-2 は、定数-3 より小さくする。
12. OSVS VSC2 MF 形式 2 は、内部浮動小数点数データ項目に対応する条件変数を定義するのに使用する。外部浮動小数点数データ項目には使用できない。この場合、定数-2 および定数-3 は、浮動小数点定数、表意定数 ZERO、値ゼロを表す数字定数のどれかとする。

    MF 形式 2 を使用して、外部浮動小数点数データ項目に対応する条件変数を定義できる。

13. ISO2002 MF 定数-4 は、定数-2 から定数-3 の範囲内のどの値とも等しくてはならない。つまり、定数-4 は定数-2 以上かつ定数-3 以下であってはならない。

形式 3

MF

14. 一意名は、78 レベル項目の宣言の前に、すべて定義しておく。一意名-1 または一意名-3 (LENGTH パラメータ) のどちらかが集団項目である場合、その集団の定義は、78 レベル項目の宣言の前に、レベル番号がそれ以下の別のデータ項目が次にくることによって完結されていること。

形式 4

ISO2002 MF

14. VALUE 句を含むデータ記述項は、OCCURS 句を含んでいるか、または OCCURS 句を含むデータ記述項の下位でなければならない。
15. 添え字-1 および添え字-2 は、整数の数字定数でなければならない。
16. 1 つの FROM 指定には、1 個の添え字-1 が含まれなければならない。この添え字-1 は、その記述項またはその上位の記述項の左辺にある各 OCCURS に対して指定される。ここで OCCURS 句は、記述項の左辺に指定される添え字付き参照と同じ順序で指定されるものとする。添え字-1 の各インスタンスは、それに関連づけられた OCCURS 句の中での出現回数の最大値を越えてはならない。
17. TO を指定した場合が、各 OCCURS 句に対し、1 個の添え字-2 が含まれなければならない。この添え字-2 は、その記述項またはその上位の記述項の左辺にある各

OCCURS に対して指定される。ここで OCCURS 句は、記述項の左辺に指定される添え字付き参照と同じ順序で指定されるものとする。添え字-2 の各インスタンスは、それに関連づけられた OCCURS 句の中での出現回数の最大値を越えてはならない。1 個の TO 指定中の添え字-2 は、添え字-2 の表要素は同じ反復であるか、対応する添え字-1 の表要素の次の反復であるという仕様でなければならない。

一般規則

形式 1、2 および 4

1. VALUE 句は、それを指定した対象の項目またはその項目を含む集団項目のデータ記述に書いた他の句と矛盾してはならない。下記の規則が適用される。
   a. 定数がデータ項目の値を定義する場合、その定数は、標準桁寄せ規則に従ってデータ項目に収められる。(*COBOL 言語の概念*の章の*標準桁寄せ規則*節を参照。)
   b. 項目の項類が英字、英数字、英数字編集、数字編集である場合、VALUE 句内の定数はすべて文字とする。これらの定数をデータ項目中に収めるときの扱いは、そのデータ項目が英数字として定義されている場合と同様である。(*COBOL 言語の概念*の章の*標準桁寄せ規則*節を参照。) PICTURE 句内の編集文字は、データ項目の大きさに含められる。(前述の *PICTURE(形式)句*節を参照。) しかし、そのデータ項目の初期値を定めるものではない。したがって、編集済みの形式での編集データ項目に対する定数はの値を指定するのは、プログラマーの責任である。

      (MF)項目の項類が数字編集である場合、VALUE 句内の定数は数字定数でも文字定数でもよい。VALUE 句内の定数が数字定数である場合、項目に収められる値は数字定数を数字編集項目に転記したのと同じとなる。

   c. 初期化は、BLANK WHEN ZERO または JUSTIFIED の指定とは無関係に行われる。

データ項目(形式 1)

2. VALUE 句の使用に関する規則は、データ部のどの節で使用するかによって異なる。
   a. ファイル節内では、VALUE 句は条件名記述項内でだけ使用する。
   b. 作業場書節

      (ISO2002)(MF)および局所記憶節

      内では、条件名記述項の中に VALUE 句を指定する。 また、データ項目の初期値を指定するために、VALUE 句を使用できる。この場合、 ランタイム要素が初期状態になった時に、データ項目の値は VALUE 句で指定された値に設定される。データ記述項内で VALUE 句を指定しないと、そのデータの初期値はどうなるかわからない。

   c. 連絡節内では、VALUE 句は条件名記述項内でだけ使用する。

3. OSVS VSC2 MF ファイル節、連絡節 、

   OS/390 および局所記憶節内

   OSVS VSC2 MF のデータ記述項の中で、VALUE 句を使用できる。ただし、注記にとどまる。

4. 集団レベルの記述項で VALUE 句を指定すると、集団領域は、その集団に含まれる個々の基本項目および集団項目に対して配慮することなく初期化される。
5. VSC2 MF 表意定数の NULL は、USAGE POINTER または USAGE PROCEDURE-POINTER を指定したデータ項目の VALUE 句内にだけ指定できる。この種のデータ項目の VALUE 句内に指定できる値は、この NULL だけである。NULL を指定すると、ポインタは他のデータ項目を指さないことが保証される。
6. ANS85 OCCURS 句が含まれるデータ記述項またはその下位に属するデータ記述項の中で VALUE 句を使用すると、反復される各データ項目に指定した値が割り当てられる。
7. ANS85 可変反復データ項目に関連するデータ項目のデータ記述項の中で、VALUE 句を指定すると、DEPENDING ON 指定によって参照されるデータ項目の値が OCCURS 句によって指定される反復回数の最大値と等しいものとして、そのデータ項目が初期化される。下記の場合に、データ項目は可変反復データ項目と関連する。
   a. VALUE 句を指定したデータ項目が集団項目であり、その中に可変反復データ項目が含まれる。
   b. VALUE 句を指定したデータ項目自体が、可変反復データ項目である。
   c. VALUE 句を指定したデータ項目が、可変反復データ項目の下位に属する。

   DEPENDING ON 指定によって参照されるデータ項目に VALUE 句が関連する場合、可変反復データ項目が初期化された後で、VALUE 句で指定された値がそのデータ項目に入られるものとみなされる。

8. ISO2002 字類がオブジェクトのデータ項目は、ヌルに初期化される。初期値は、VALUE 句が効力を持つ時、およびデータ項目の記憶域が割り当てられた時に効力を持つ。
9. ISO2002 字類がポインタのデータ項目は、ヌルに初期化される。初期値は、VALUE 句が効力を持つ時、およびデータ項目の記憶域が割り当てられた時に効力を持つ。

条件名に関する規則 (形式 2)

10. 条件名記述項には、VALUE 句を必ず指定する。条件名記述項に書けるのは、VALUE 句と条件名自体だけである。条件名の性質は、対応する条件変数の性質によって暗黙的に決まる。
11. 形式 2 は、条件名に関連してだけ使用できる。THRU 指定を書くときは、定数-2 は定数-3 よりも小さくする。
12. ISO2002 MF FALSE 句は、対応する条件名が SET 条件名 TO FALSE 文を参照する場合にだけ意味を持つ。(*手続き部 - SEARCH - XML PARSE* の章の *SET(設定)文*節を参照。)

定数名に関する規則 (形式 3)

MF

13. 形式 3 は、定数名記述項にだけ使用できる。
14. 定数-5 を指定して演算子が続かない場合、 定数名の性質は定数-5 と同様となる。そうでない場合、定数名の性質は整数と同様となる。
15. 任意の数の算術演算子または論理演算子を使用できる。式は左から右へ整数演算として評価される。式の中にかっこは使用できない。中間結果にゼロ未満のものがあると、最終結果はどうなるかわからない。整数-1 の代わりに定数名を使用できる。
16. 論理演算の AND と OR は、2 進表現のビットごとに作用する。
17. 一意名-1 または一意名-3 の LENGTH は、一意名-1 または一意名-3 に割り当てられた記憶領域の大きさである。一意名が集団項目である場合、その長さは下位のデータ項目をすべて含んだものである。
18. NEXT から返される値は、前のデータ宣言の後にある記憶域の次のバイトの相対番地である。このデータ宣言が OCCURS 句によって定義された表のものであると、NEXT から返される値は、表の最初の要素の後に起こる記憶域の次のバイトの相対番地である。
19. 一意名-2 または一意名-4 の START はそれぞれ、一意名-2 または一意名-4 が始まる相対番地である。
20. 規則 18 と 19 に関して、相対番地を下記のように定義する。
    - 一意名が EXTERNAL レコードまたは LINKAGE レコードの一部であるならば、相対番地は対応する 01 レベルの始点から数える。
    - 一意名が LOCAL-STORAGE 中に定義されている場合、相対番地は LOCAL-STORAGE SECTION の始点から数える。
    - それ以外の場合、相対番地は DATA DIVISION の始点から数える。
21. 相対番地は別の COBOL 製品との間で移植性がない。相対番地は翻訳単位を超えた特定の値に依存すべきではない。

表に関する規則 (形式 4)

ISO2002 MF

22. 形式 2 の VALUE 句は、定数-6 の値となる表要素を初期化する。初期化された表要素は、添え字-1 で識別される。それに続く表要素は順番に初期化され、続いて現れる定数-6 の値として初期化される。これらの表要素は、1 つずつ増えていく番号をつけた添え字により参照される。この添え字は表の最小次元を表す。番号をつける前の添え字への参照が、対応する OCCURS 句により指定された最大反復回数と同じであると、添え字の番号は 1 に設定され、それに次いで次元の小さい表の添え字が次の番号となる。
23. TO が指定されている場合は、出現する定数-6 が、指定された順に、規則 22 で記述された初期化中の原文として再利用される。これは、添え字-2 の表要素が初期化されるまで繰り返される。
24. TO を指定しない場合は、TO 指定が、対応する各添え字-1 の表要素の最大出現回数として、各添え字-2 で指定された場合と同様の動作となる。
25. 複数の FROM 指定が同じ表要素を参照している場合は、VALUE 句中の最後の FROM 指定で定義された値が表要素に割り当てられる。

8.28 VALUE OF(ラベル項目の値)句

機能

VALUE OF (ラベル項目の値) 句はファイルのラベル・レコード中の項目の内容を指定する。

(ANS85)ファイル記述項の VALUE OF 句は、ANSI '85 標準では廃要素に分類されており、ANSI 標準の次回の全面改訂の際に、削除される予定である。

(MF)この構文は、この COBOL 処理系に組み込まれているすべての方言で全面的に使用することができる。FLAGSTD 指令を使用すると、この構文が使われているすべての箇所を洗い出すことができる。

(XOPEN)標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、この機能は明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内ではこの句を使用するべきではない。

一般形式

形式 1 (すべてのファイル)

```
VALUE OF { 作成者語-1 IS { データ名-1 } } ...
         {               { 定数-1   }
```

形式 2 (すべてのファイル)

```
VALUE OF FILE-ID IS { データ名-2 }   (MF)
                    { 定数-2   }
```

構文規則

形式 1 (すべてのファイル)

1. データ名-1 は、必要ならば修飾してもよいが、添字や指標を付けることはできない。
   また、データ名-1 には USAGE IS INDEX 句を記述してあってはならない。

形式 2 (すべてのファイル)

2. (MF)データ名-2 は、作業場所節内に記述しておく。
3. (MF)定数-2 は文字定数とする。表意定数であってはならない。

4. (MF)ファイル管理記述項の ASSIGN 句の中で、外部ファイル参照、データ名-1、定数-1 のどれかを指定した場合、VALUE OF FILE-ID 句は使用できない。(*環境部*の章の *ファイル管理記述項*節を参照。)

一般規則

形式 1 (すべてのファイル)

1. (MF)VALUE OF 句は注記になる。

形式 2 (すべてのファイル)

2. (MF)定数-2 またはデータ名-2 に指定した文字列は、外部ファイル名と解釈される。

第 9 章 : データ部 - 画面節

MF XOPEN

この章では、画面記述項の全体的な骨組みを解説する。この種の記述項に使用する句の説明をアルファベット順に掲載する。

9.1 画面節の概要

画面記述項は、実行時に受け取り用または表示用に参照される画面項目の属性、動き、大きさ、位置などを指定する。

利用できる画面属性は、使用している端末ハードウェアやオペレーティングシステムや COBOL の実行時支援機能に依存する。

MF 下記のものは、X/Open の画面操作には含まれない。

ACCEPT (形式 3 および 5)
AUTO-SKIP (AUTO と同義語)
BEEP (BELL と同義語)
COL (COLUMN と同義語)
CONTROL
DISPLAY (形式 3)
EMPTY-CHECK (REQUIRED と同義語)
GRID
LEFTLINE
LENGTH-CHECK (FULL と同義語)
NO-ECHO (SECURE と同義語)
OCCURS
OVERLINE
PROMPT
SIZE
ZERO-FILL

一般形式

レベル番号 [画面名 / FILLER]

[BLANK {SCREEN / LINE}]

[{BELL / BEEP} (NF)]

[BLINK]

[ERASE {EOL / EOS}]

[HIGHLIGHT / LOWLIGHT]

[GRID] (NF)

[LEFTLINE]

[OVERLINE]

[REVERSE-VIDEO]

[UNDERLINE]

[SIZE IS {一意名-1 / 整数-1}] (NF)

[LINE [NUMBER IS [{PLUS / + / -} (NF)] {一意名-2 / 整数-2}]]

[{COLUMN / COL} (NF) [NUMBER IS [{PLUS / + / -} (NF)] {一意名-3 / 整数-3}]]

[{FOREGROUND-COLOR / FOREGROUND-COLOUR} (NF) IS {一意名-4 / 整数-4}]

[{BACKGROUND-COLOR / BACKGROUND-COLOUR} (NF) IS {一意名-5 / 整数-5}]

[CONTROL IS 一意名-6] (NF)

[[VALUE IS] 定数-1]

[{PICTURE / PIC} IS 文字列]

[{[FROM {一意名-6 / 整数-2}] [TO 一意名-7] / USING 一意名-8}]

[[USAGE IS] {DISPLAY / DISPLAY-1} (NF)]

[BLANK WHEN ZERO]

[{JUSTIFIED / JUST} RIGHT]

[[SIGN IS] {LEADING / TRAILING} [SEPARATE CHARACTER]]

[{AUTO / AUTO-SKIP} (NF)]

[{SECURE / NO-ECHO} (NF)]

[{REQUIRED / EMPTY-CHECK} (NF)]

構文規則

1. 各画面記述項の最初にレベル番号を書く。レベル番号は 01 から 49 までとする。
2. 01 レベルの項目には、画面名を指定する。
3. 画面名はレベル番号の直後に書く。画面名は、利用者語の要件を満たす必要がある。
4. 画面項目は、形式 4 の ACCEPT 文または形式 2 の DISPLAY 文によってだけ参照できる。
5. 各基本画面項目には、次のものの中から少なくとも 1 つを指定する: BELL, BLANK LINE, BLANK SCREEN, COLUMN, LINE,PICTURE, VALUE。
6. FROM, TO, USING の各指定の中のデータ項目は、画面項目に「関連付け」られる。USING 指定は、FROM 指定と TO 指定を対にして同じ項目を指定したものと同義である。
7. ACCEPT を実行できるのは、FROM 指定または VALUE 句を伴う画面項目、および TO 指定または USING 指定を伴う画面項目が共に含まれる集団画面項目に対してだけである。
8. 画面名の後ろに続ける句は、どのような順序で書いてもよい。
9. 集団画面項目の画面記述に書いた句は、その集団画面項目に属するすべての基本画面項目のうち、その句が当てはまるものに適用される。
10. 01 レベル以外の画面項目には、データ名または FILLER を指定してもよいし、あるいは何も指定しないでもよい。名前を指定しないと、FILLER を指定したものとみなされる（したがって、その項目は明示的に参照できない）。
11. 同じ画面項目に同じ句を 2 回以上指定すると、階層系列内で最も低いレベルに現れるものが効力を発揮する。
12. 表-1 は、画面節で様々な種類の項目とともに使用可能な画面記述項の句の一覧である。

    (MF)これらの句はさらに、ACCEPT 文および DISPLAY 文を伴う WITH 指定とともに使用可能である。

表 9-1: 使用可能なオプション

画面記述句/ WITH 指定	画面節				(MF)WITH 指定	
	入力項目	出力項目	更新項目	定数項目	ACCEPT	DISPLAY
AUTO	X		X		X	
BACKGROUND-COLOR	X	X	X	X	X	X
BELL	X	X	X	X	X	X
BLANK	X	X	X	X		X
BLANK WHEN ZERO	X	X	X			
BLINK	X	X	X	X	X	X
COLUMN	X	X	X	X		
ERASE	X	X	X	X	X	
FOREGROUND-COLOR	X	X	X	X	X	X

FULL	X		X		X	
GRID	X	X	X	X	X	X
HIGHLIGHT	X	X	X	X	X	X
JUSTIFIED	X	X	X			
LEFT-JUSTIFY					X	
LEFTLINE	X	X	X	X	X	X
LINE	X	X	X	X		
LOWLIGHT	X	X	X	X	X	X
OCCURS	X	X	X			
OVERLINE	X	X	X	X	X	X
PROMPT	X		X		X	
REQUIRED	X		X		X	
REVERSE-VIDEO	X	X	X	X	X	X
RIGHT-JUSTIFY					X	
SECURE	X		X		X	
SIGN	X	X	X			
SIZE	X	X	X	X	X	X
SPACE-FILL					X	
TRAILING-SIGN					X	
UNDERLINE	X	X	X	X	X	X
UPDATE					X	

説明:

X	句またはオプションを使用できる
入力項目	画面記述に TO 指定が含まれる画面項目
出力項目	画面記述に FROM 指定が含まれる画面項目
更新項目	画面記述に USING 指定が含まれる画面項目
定数項目	画面記述に PICTURE 句が含まれない基本画面項目

一般規則

1. 画面名は、画面記述中に記述された画面項目に名前を付ける働きをする。
2. 画面記述は、画面上の領域を定義する。各記述項は、レベル番号といくつかの句で構成する。画面名をはじめとする各種の句は、指定しても指定しなくてもよい。これらの句を通じて、項目の位置や操作卓機能を記述する。

3. 画面項目が表示されるとき、データは、関連する FROM 指定または USING 指定内に指名されている定数またはデータ項目から取り出される。TO 指定の中にだけ指定されている項目は、FROM SPACE または FROM ZERO が指定されているものと扱われる。どちらに扱われるかは、その画面項目の種類による。
4. 画面項目が受け取られると、データは、画面から TO 指定または USING 指定内に指定されているデータ項目に転記される。その項目の項類に応じて、必要があれば、変換および編集が解除される。
5. 入力項目は、画面記述に TO 指定が含まれる画面項目である。
6. 出力項目は、画面記述に FROM 指定が含まれる画面項目である。
7. 更新項目は、画面記述に USING 指定が含まれる画面項目である。
8. 定数項目は、画面記述に PICTURE 句が含まれる基本画面項目である。
9. 集団画面項目を ACCEPT することは、その集団に属する基本項目のうちの入力項目または更新項目を受け取ることである。これらの項目は、画面節内に記述されている順番に、画面記述によって示される画面位置から受け取られる。CURSOR IS 句(*環境部*の章の*特殊名段落*節を参照。) の中で別途指定されていないかぎり、カーソルは、当初最初の項目の始点に位置付けられる。各項目の ACCEPT 処理が終わるにつれて、カーソルは次の項目の始点へと移動される。
10. 集団画面項目を DISPLAY することは、その集団に属する基本項目のうちの出力項目または更新項目と定数項目を表示することである。これらの項目は、画面記述によって示される画面位置に同時に表示される。
11. ACCEPT または DISPLAY の対象の画面項目の長さが現在の行を超えると、次の行の頭に戻って続けられる。
12. 画面項目が長すぎて物理画面に収まりきらない場合、切捨てが発生する。出力項目および英数字の入力項目と更新項目の場合は、画面からはみ出す最初の文字以降が切り落とされる。数字および数字編集の入力項目と更新項目の場合は、画面からはみ出す最初の項目以降が切り落とされる。

9.2 AUTO 句

AUTO (自動) 句は、最後の文字位置に文字が入力されたときに、画面項目の ACCEPT 処理を自動的に終了させる。終了キーを明示的に入力する必要はない。

一般形式

構文規則

1. AUTO 句は、入力項目および更新項目に対してだけ指定できる。
2. この句を集団レベルに指定すると、それに属するすべての基本項目に適用される。
3. (MF) AUTO と AUTO-SKIP は同義であり、どちらを書いてもよい。

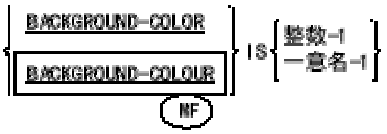
一般規則

1. REQUIRED 句または FULL 句が満たされるならば、カーソルは次の画面項目に位置付けられる。あるいは、ACCEPT する最後の画面項目のところでは、ACCEPT 処理そのものが終了される。
2. この句は、ACCEPT 文の自動飛越しおよび自動終了用の既存の構成オプションに優先する。(構成オプションの詳細については、ユーザインターフェイスに関する COBOL システムのマニュアルを参照。)
3. 固定方式の数字編集画面項目に AUTO 句を指定した場合、整数部分の桁位置がすべて入力されたときに、小数点位置が自動的に飛び越される。固定方式を選択することは、構成オプションである。(構成オプションの詳細については、ユーザインターフェイスに関する COBOL システムのマニュアルを参照。)

9.3 BACKGROUND-COLOR(背景色)句

BACKGROUND-COLOR (背景色) 句は、画面項目の背景色を指定する。

一般形式



構文規則

1. MF BACKGROUND-COLOR と BACKGROUND-COLOUR は同義であり、どちらを書いてもよい。
2. BACKGROUND-COLOR 句は、任意の画面項目に指定できる。
3. BACKGROUND-COLOR 句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
4. 整数-1 の値は、0 から 7 までとする。

一般規則

1. BACKGROUND-COLOR 句は、カラー画面に対してだけ適用できる。
2. 整数-1 または一意名-1 は、画面項目の背景色を指定する。使用できる色と対応する値は下記のとおり。

0	黒	8	灰色
1	青	9	明るい青
2	緑	10	明るい緑
3	藍	11	明るい藍

4	赤	12	明るい赤
5	赤紫	13	明るい赤紫
6	茶または黄	14	黄
7	白	15	高輝度の白

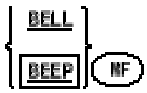
3. カラー画面では、8 から 15 の値の場合、整数 0 から 7 を指定して BLINK 句を指定することと同じである。モノクロ画面では、これは単に BLINK 句を指定することと同じである。

4. BACKGROUND-COLOR 句を指定しないと、背景色は省略時解釈の黒となる。
5. 画面項目に BLANK SCREEN 句と BACKGROUND-COLOR 句を共に指定するか、または BLANK SCREEN 句を指定した画面項目が BACKGROUND-COLOR 句を指定した画面項目の下位に属する場合、DISPLAY 文を用いてその画面項目を表示すると、指定した色が省略時解釈の背景色となる。この省略時解釈の背景色は、別の指定をするまで有効である。別の指定とは、上記と同じ組合わせの指定内容をもつ別の画面項目を表示するか (同じ DISPLAY 文の中でも別の DISPLAY 文の中でもよい) 、または上記の 2 つの句を含む形式 3 の DISPLAY 文を実行することである。
6. 一意名-1 は、符号の付かない 0 から 7 までの整数とする。8 以上の値を指定すると、その値を 8 で割った余りが指定値としてとられる。
7. 一意名-1 は、OCCURS 句の左辺には指定できない。
8. 整数-1 または一意名-1 の値に 6 を指定したときに、茶色と黄色のどちらが表示されるかは、端末ハードウェアによる。

9.4 BELL(警報)句

BELL (警報) 句は、この句が含まれる画面項目が表示されるたびに、警報音を鳴らす。

一般形式



構文規則

1. BELL 句は、基本項目に対してだけ指定できる。
2. (MF) BELL と BEEP は同義であり、どちらを指定してもよい。

一般規則

1. BELL 句は、この句が指定されている画面項目が受け取られるかまたは表示されるたびに、警報音を鳴らす。
2. この句が含まれる画面項目を受け取ったときに警報音が鳴るようにするためには、形式 5 の ACCEPT 文の中に BELL 句を指定する。画面記述項の中に BELL 句を指定して形式 4 の ACCEPT 文を実行しても、警報音は鳴らない。

9.5 BLANK(空白)句

BLANK (空白) 句は、画面項目が表示される前に画面の行または画面全体を空白にする。

一般形式

```
BLANK { LINE   }
      { SCREEN }
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - OLDBLANKLINE - BLANK LINE 句の効果は ERASE EOL と同じであることを指定する。つまり、カーソルの右にある文字はすべて削除される。

構文規則

1. BLANK SCREEN 句は、任意の画面記述項に指定できる。
2. BLANK LINE 句は、基本項目に対してだけ指定できる。

一般規則

1. BLANK SCREEN 句は、どこに記述されていても、画面データ項目が表示される前に実行される。この句が実行されると画面全体が空白にされ、カーソルは行 1、カラム 1 の位置に移動される。
2. BLANK LINE 句を指定すると、記述がある画面データ項目の行がカラム 1 から行の終端まで空白にされる。カーソルは、BLANK LINE 句が実行された位置と同じところにある。
3. BLANK 句も ERASE 句も指定しないと、画面データ項目を表示したときに、その項目の部分だけが画面上で変更される。それ以外の画面部分は、元のまま変わらない。
4. BLANK SCREEN 句が実行されると、画面の前景色と背景色は省略時解釈の状態に戻される。画面項目に FOREGROUND-COLOR または BACKGROUND-COLOR を指定した場合の効果については、それぞれの節を参照。
5. ACCEPT 文の中では、BLANK 句は無視される。
6. BLANK SCREEN 句を指定した画面には、色に関する句 (FOREGROUND-COLOR と BACKGROUND-COLOR) だけを使用すべきである。他の属性は受け付けられるが、無視される。

9.6 BLANK WHEN ZERO(ゼロならば空白)句 - 画面節中

BLANK WHEN ZERO (ゼロならば空白) 句は、画面項目の値がゼロのとき表示上は空白にする。

一般形式

```
BLANK WHEN ZERO
```

構文規則

1. BLANK WHEN ZERO 句は項類が数字または数字編集の入力項目と出力項目と更新項目にだけ指定できる。

一般規則

1. BLANK WHEN ZERO 句は、画面項目の値がゼロのとき表示上は空白にする。
2. カーソルが項目内にあるときには、この句は効力を発揮しない。つまり、現在の入力項目に関しては、たとえゼロであってもその内容は常に表示される。

9.7 BLINK(点滅)句

BLINK (点滅) 句は、画面上に表示される項目を点滅させる。

一般形式

```
BLINK
```

構文規則

1. BLINK 句は、どのような画面項目にも指定できる。
2. BLINK 句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。

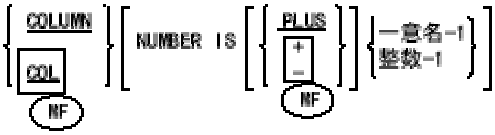
一般規則

1. BLINK 句は、画面上に表示される項目を点滅させる。

9.8 COLUMN (カラム) 句

COLUMN (カラム) 句は、画面項目を画面上に表示する始点となるカラムを指定する。

一般形式



構文規則

1. 一意名-1 は符号の付かない整数であり、その値は 1 以上 255 以下とする。
2. 整数-1 は符号なしで、その値は 1 以上 255 以下とする。
3. MF PLUS と"+" は同義であり、どちらを書いてもよい。
4. NUMBER の指定を省略すると、省略時解釈として値"+1"がとられる。
5. LINE 句を指定して COLUMN 句を指定しないと、カラム 1 が想定される。
6. COLUMN 句は、任意の基本項目に指定できる。
7. MF COL は、COLUMN の省略形である。
8. MF COLUMN 句を SCREEN SECTION 集団項目の階層中のどこかに指定する時、以下の句のうち、最低1つを同じ集団の定義に指定しなければならない。

   BEEP, BELL, BLANK LINE, BLANK SCREEN, ERASE EOL, ERASE EOS, FROM, TO, USING, VALUE.

一般規則

1. COLUMN 句は、ACCEPT 処理または DISPLAY 処理を行うときの、画面上に画面項目を表示するカラムを指定する。
2. COLUMN 句に一意名または整数を指定し、PLUS,

   MF "+" または"-",

   のどれも指定しない場合、その値は絶対カラム番号を表わす。01 レベルはそれぞれ、画面レコードを表わす。AT 指定は、表示画面の先頭に関連付けて、画面レコードの先頭位置を指定する。ACCEPT 文または DISPLAY 文に AT 指定を指定しなかった場合、画面の最初のカラムがカラム 1 となる。

3. COLUMN 句に PLUS,

   MF "+" または"-"

   のどれかを指定すると、一意名または整数に指定した値は、先行する画面項目の末尾からの相対カラム番号を表わす。この場合、先行する項目が実際に画面に表示されるか否かは問わない。先行する画面項目の末尾は、その項目の実行時の有効な長さによって決まる。その長さは、PICTURE 句、VALUE 句、SIZE 句に基づいて算出される。01 レベルの項目が出てくると、カラム番号は 1 に戻る。

4. 01 レベルでない項目の画面記述に LINE 句も COLUMN 句も含めないと、COLUMN+1 が指定されたものとみなされる。つまり、その項目は前の項目の直後に続くことになる。
5. 画面の範囲から外れるカラム位置を指定すると、次（または前）の行の頭に繰り越してカラム位置が数えられる。

9.9 CONTROL(制御)句

(MF)

CONTROL（制御）句は、画面項目に関する属性を実行時に定義できるようにする。

一般形式

```
CONTROL IS 一意名-1
```

構文規則

1. この句は、どの画面項目にも指定できる。
2. この句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. 一意名-1 は、英数字データ項目とする。この項目がとる最大の大きさは、PIC X（65535）である。
4. 一意名-1 に指定した属性は、静的に定義した属性に優先する。
5. CONTROL 句を入れ子にすることはできない。
6. PROTECT 属性は、基本画面記述項にだけ指定できる。

一般規則

1. この項目を明示的または暗黙的に参照する ACCEPT 文または DISPLAY 文が実行される時、一意名-1 によって参照されるデータ項目の内容は、以下に示す画面属性のサブセットを確認しなければならない。これらはどこにでも指定できる。

```
[ [NO] { { BELL }
         { BEEP }
         BLINK
         HIGHLIGHT
         LOWLIGHT
         GRID
         LEFTLINE
         OVERLINE
         REVERSE-VIDEO
         UNDERLINE
         { FOREGROUND-COLOR } IS 整数-3
         { FOREGROUND-COLOUR }
         { BACKGROUND-COLOR } IS 整数-4
         { BACKGROUND-COLOUR }
         { RIGHT-JUSTIFY }
         { JUSTIFY       }
         { JUST          }
         { TRAILING-SIGN }
         { TRAILING      }
         { AUTO      }
         { AUTO-SKIP }
         { SECURE    }
         { NO-ECHO   }
         { FULL          }
         { LENGTH-CHECK  }
         { REQUIRED      }
         { EMPTY-CHECK   }
         PROMPT
         PROTECT
         ZERO-FILL
         BLANK [ { LINE   } ]
               [ { SCREEN } ] } ... ]
```

2. 一意名-1 の内容が空白であると、項目を ACCEPT または DISPLAY する際に、静的に定義されている属性が使用される。
3. CONTROL 一意名-1 内に指定できる各属性の意味は、この参照の以下の部分に定義する。
    a. DATA DIVISION 内の各 SCREEN SECTION CLAUSE
        - AUTO-SKIP/AUTO
        - BACKGROUND-COLOR/BACKGROUND-COLOUR
        - BELL/BEEP
        - BLANK LINE/SCREEN
        - BLINK
        - FOREGROUND-COLOR/FOREGROUND-COLOUR
        - FULL/LENGTH-CHECK
        - GRID

        - HIGHLIGHT
        - JUSTIFY/JUST
        - LEFTLINE
        - LOWLIGHT
        - OVERLINE
        - PROMPT
        - REQUIRED/EMPTY-CHECK
        - REVERSE-VIDEO
        - SECURE/NO-ECH0
        - UNDERLINE
        - ZERO-FILL
    b. PROCEDURE DIVISION 内の ACCEPT 文一般規則
        - RIGHT-JUSTIFY
        - TRAILING-SIGN
        - TRAILING
    c. この CONTROL 句内
        - PROTECT
4. PROTECT 属性は、ACCEPT 文への入力を妨げることを指定する。

9.10 ERASE(消去)句

ERASE (消去) 句は、カーソルが位置している以降の行または画面の一部を空白にする。

一般形式

構文規則

1. ERASE 句は、基本項目に対してだけ指定できる。

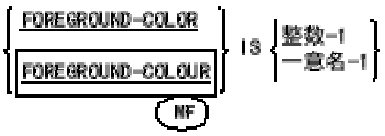
一般規則

1. ERASE EOL を指定すると、その画面データ項目の座標位置 (行とカラム) 以降その行の末尾までが空白にされる。
2. ERASE EOS を指定すると、その画面データ項目の座標位置 (行とカラム) 以降その画面の末尾までが空白にされる。
3. BLANK 句も ERASE 句も指定しないと、画面データ項目を表示する際に、その項目の部分だけが変更される。
4. ACCEPT 処理の際には、この句は無視される。

9.11 FOREGROUND-COLOR(前景色)句

FOREGROUND-COLOR (前景色) 句は、画面項目の前景色を指定する。

一般形式



構文規則

1. OREGROUND-COLOR と FOREGROUND-COLOUR は同義であり、どちらを書いてもよい。
2. FOREGROUND-COLOR 句はどの画面項目にも指定できる。
3. FOREGROUND-COLOR 句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
4. 整数-1 の値は、0 から 15 までとする。

一般規則

1. FOREGROUND-COLOR 句はカラー画面に対してだけ適用できる。
2. 整数-1 または一意名-1 は画面項目の前景色を指定する。使用できる色と対応する値は下記のとおり。

0	黒	8	灰色
1	青	9	明るい青
2	緑	10	明るい緑
3	藍	11	明るい藍
4	赤	12	明るい赤
5	赤紫	13	明るい赤紫
6	茶または黄	14	黄
7	白	15	高輝度の白

3.
4. カラー画面では、8 から 15 の値の場合、整数 0 から 7 を指定して HIGHLIGHT 句を指定することと同じである。モノクロ画面では、これは単に HIGHLIGHT 句を指定することと同じである。
5. この句を指定しないと、前景色は省略時解釈の白となる。
6. 画面項目に BLANK SCREEN 句と FOREGROUND-COLOR 句を共に指定するか、または BLANK SCREEN 句を指定した画面項目が FOREGROUND-COLOR 句を指定した画面項目の下位に属する場合、その画面項目を DISPLAY 文を用いて表示すると、指定した色が省略時解釈の前景色となる。この省略時解釈の前景色は、別の指定を

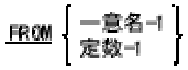
するまで有効である。別の指定とは、上記と同じ組合わせの指定内容をもつ別の画面項目を表示するか（同じ DISPLAY 文の中でも別の DISPLAY 文の中でもよい）、または上記の 2 つの句を含む形式 3 の DISPLAY 文を実行することである。

7. HIGHLIGHT 句も同時に指定すると、前景色の輝きと明るさが増大される。たとえば、ハードウェアによっては、黒が灰色に変えられ茶色が黄色に変えられることがある。しかし、これは BLANK SCREEN 句には適用されない。
8. 一意名-1 は符号の付かない 0 から 7 までの整数とする。8 以上の値を指定すると、その値を 8 で割った余りが指定値としてとられる。
9. 一意名-1 は、OCCURS 句の左辺には指定できない。
10. 整数-1 または一意名-1 の値に 6 を指定したときに、茶色と黄色のどちらが表示されるかは、端末ハードウェアによる。

9.12 FROM(から)句

FROM 句は表示するデータの源を示す。

一般形式



構文規則

1. FROM 句を TO 句と一緒に使用することは、対応して同じ一意名を指定した USING 句を使用することに等しい。FROM 句を USING 句と一緒に使用してはならない。
2. FROM 句内の一意名を修飾してもよい。画面項目に OCCURS 句が適用されていない場合、その一意名は添字付けまたは指標付けされていてもよい。FROM 句を定義できる場所はプログラムのファイル節、作業場所節、

   (MF)局所記憶節、

   連絡節のいずれかだけである。

一般規則

1. FROM 句が記述されている画面項目に対して DISPLAY 文を実行すると、対応するデータ項目から画面項目にデータが転記され、それから画面項目が画面上に表示される。

9.13 FULL(全桁)句

FULL（全桁）句は、画面項目をまったく空白のままにしておくか、全桁にデータを入力するかを指定する。

一般形式

構文規則

1. FULL 句は、入力項目と更新項目と集団項目に対してだけ有効である。
2. FULL 句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. (MF) FULL と LENGTH-CHECK は同義であり、どちらを指定してもよい。

一般規則

1. FULL 句が効力を発揮するのは、カーソルを画面項目の中に位置付けたうえでデータを受け取る ACCEPT 文が実行されるときである。この句の要件が満たされるまで、終了キーを押しても受け付けられない。固定方式の数字編集項目の場合、カーソルは小数点の位置に位置付け直される。
2. 画面項目の項類が英数字の場合、この句の要件が満たされるのは、次の 2 つの場合である。1 つは、その項目全体に空白またはプロンプト文字が入れられた場合。もう 1 つは、その項目の最初の文字位置と最後の文字位置に、空白でもプロンプト文字でもない文字が入れられた場合。
3. 画面項目の項類が自由方式の数字または数字編集の場合、この句の要件が満たされるのは、次の 2 つの場合である。1 つは、その項目の値が結果的にゼロになる場合。もう 1 つは、その項目の最初の文字位置と最後の文字位置に、空白でもプロンプト文字でもない数字が入れられた場合。
4. 画面項目の項類が固定方式の数字編集の場合、この句の要件が満たされるのは、次の 2 つの場合である。1 つは、その項目の値がゼロである場合。もう 1 つは、その項目内でゼロ抑制が起こらなかった場合。
5. 固定方式の数字項目またはゼロ抑制が指定されていない数字編集項目に対しては、FULL 句は効力を発揮しない。
6. 更新項目に関しては、元のデータと入力されるデータの両方が FULL 句の要件を満たす要素となる。
7. ACCEPT 処理を終了させるためにファンクション･キーが使用された場合は、FULL 句は効力を発揮しない。(構成オプションの詳細については、ユーザインターフェイスに関する COBOL システムのマニュアルを参照。)
8. FULL 句の要件が満たされない場合、エラー･メッセージを画面上に表示させることができる。(構成オプションの詳細については、ユーザインターフェイスに関する COBOL システムのマニュアルを参照。)

9.14 GRID (桁取り) 句

(MF)

GRID (桁取り) 句は、画面に表示される画面項目の各文字の左側に、縦線を付ける。各縦線は、文字位置の範囲内に収まる。

一般形式

```
GRID
```

構文規則

1. GRID 句は、どのような画面項目にも指定できる。
2. GRID 句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. 文字の左側に縦線を引く機能のないシステム上で指定しても、この句は効力を発揮しない。

一般規則

1. GRID 句は、画面に表示される画面項目の各文字の左側に、縦線を付ける。各縦線は、文字位置の範囲内に収まる。

9.15 HIGHLIGHT(強輝度)句

HIGHLIGHT (強輝度) 句は、画面上に表示される画面項目の輝度を強くするよう指定する。

一般形式

```
HIGHLIGHT
```

構文規則

1. HIGHLIGHT 句は、入力項目、出力項目、更新項目、定数項目に対して有効である。
2. HIGHLIGHT 句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。

一般規則

1. FOREGROUND-COLOR 句とあわせて HIGHLIGHT 句を指定すると、前景色の輝度と明るさが増大される。たとえば、ハードウエアの中には、黒が灰色に変えられ茶色が黄色に変えられるものがある。HIGHLIGHT 句は、BLANK SCREEN 句には適用されない。

9.16 JUSTIFIED(桁寄せ)句 - 画面節中

JUSTIFIED (桁寄せ) 句は、画面項目にデータを転記するか入力するときの、非標準的なデータの位置付けを指定する。

一般形式

```
{ JUSTIFIED } RIGHT
{ JUST     }
```

構文規則

1. JUST は、JUSTIFIED の省略形である。
2. JUSTIFIED 句は、入力項目、出力項目、更新項目に対してだけ指定できる。

一般規則

1. JUSTIFIED 句を指定した画面項目が出力項目または更新項目であると、送出し側項目から画面項目にデータが転記されたときに、MOVE (転記) に関する通常の規則に従って桁寄せが行われる。
2. JUSTIFIED 句を指定した画面項目が入力項目または更新項目であると、画面項目へのデータの入力が済んだときに、残っているプロンプト文字の数だけ入力されたデータが右にずらされ、空いた左側に空白が埋められる。この操作は、受取り側データ項目にデータが転記される前に行われる。データが入力されなかったときには、この処理は行われない。
3. 画面項目に SECURE (機密) 句が指定してある場合の処理は、SECURE 句が指定しない場合の処理と同じである。ただし、その処理結果を画面上で見ることはできない。

9.17 LEFTLINE(左端位置)句

MF

LEFTLINE (左端位置) 句は、画面に表示される画面項目の左端の文字の左側に、縦線を付ける。その縦線は、文字位置の範囲内に収まる。

一般形式

```
LEFTLINE
```

構文規則

1. LEFTLINE 句は、どのような画面項目にも指定できる。

2. LEFTLINE 句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. 文字の左側に縦線を引く機能のないシステム上で指定しても、この句は効力を発揮しない。

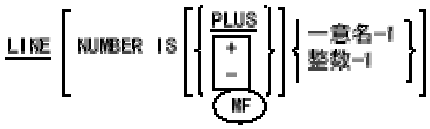
一般規則

1. LEFTLINE 句は、画面に表示される画面項目の左端の文字の左側に、縦線を付ける。その縦線は、文字位置の範囲内に収まる。

9.18 LINE(行)句

LINE (行) 句は、画面項目を画面上に表示する始点となる行を指定する。

一般形式



構文規則

1. 一意名-1 は、符号の付かない整数であり、その値は 1 以上 255 以下とする。
2. 整数-1 は符号なしで、その値は 1 以上 255 以下とする。
3. (MF)PLUS と"+" は同義であり、どちらを書いてもよい。
4. NUMBER の指定を省略すると、省略時解釈として値 PLUS 1 がとられる。
5. LINE 句の指定を省略すると、現在の行が想定される。
6. LINE 句は、任意の基本項目に指定できる。
7. (MF)LINE 句を SCREEN SECTION 集団項目の階層中のどこかに指定する時、以下の句のうち、最低1つを同じ集団の定義に指定しなければならない。

   BEEP, BELL, BLANK LINE, BLANK SCREEN, ERASE EOL, ERASE EOS, FROM, TO, USING, VALUE.

一般規則

1. LINE 句は、ACCEPT 処理または DISPLAY 処理を行うときの、画面上に画面項目を表示する行を指定する。
2. LINE 句に一意名または整数を指定し、PLUS、
3. (MF)"+"または"-"

   のどれかを指定しないと、その値は絶対行番号を表わす。01 レベルはそれぞれ、画面レコードを表わす。AT 指定は、表示画面の先頭に関連付けて、画面レコードの先

頭位置を指定する。ACCEPT 文または DISPLAY 文に AT 指定を指定しなかった場合、画面の最初の行が絶対行 1 となる。

4. LINE 句に PLUS、

   (MF)"+"または"-"

   のどれかを指定すると、一意名または整数に指定した値は先行する画面項目の末尾からの相対行番号を表わす。この場合、先行する項目が実際に画面に表示されるか否かは問わない。先行する画面項目の末尾は、その項目の実行時の有効な長さによって決まる。その長さは、PICTURE 句、VALUE 句、SIZE 句に基づいて算出される。01 レベルの項目が出てくると、行番号は 1 に戻る。

5. 01 レベルでない項目の画面記述に LINE 句も COLUMN 句も含めないと、COLUMN PLUS 1 が指定されたものとみなされる。つまり、その項目は前の項目の直後に続くことになる。
6. 画面の範囲から外れる行位置を指定すると、ACCEPT または DISPLAY において切捨てが発生する。
7. LINE 句に PLUS、

   (MF)"+" または"-",

   のどれかを指定し、画面節の項目に関連する ACCEPT 文または DISPLAY 文で AT LINE NUMBER 句を指定した場合、この項目が示す行番号は、2 つの数の和または差である。

9.19 LOWLIGHT(弱輝度) 句

LOWLIGHT (弱輝度) 句は、画面上に表示される画面項目の輝度を弱くするよう指定する。

一般形式

LOWLIGHT

構文規則

1. LOWLIGHT 句を集団項目レベルで指定すると、その下の基本項目すべてに対して適用される。

一般規則

1. LOWLIGHT 句は、画面上に表示される画面項目の輝度を弱くするよう指定する。

9.20 OCCURS(反復)句 - 画面節中

(MF)

OCCURS (反復) 句は、繰り返される同形の画面項目を個々に記述する手間を省き、添字または指標を使用するのに必要な情報を提供する。

一般形式

```
OCCURS 整数-1 TIMES
```

構文規則

1. OCCURS 句は、01 レベルの項目には指定できない。
2. 項目に USING 句または TO 句を適用する場合、またはある項目が項目に従属する場合、送出し側項目にも同じ出現回数を指定した OCCURS 句を使用するか、または OCCURS 句を使用しないようにする。これらの OCCURS 句に DEPENDING 句を含めてはならない。
3. 項目に FROM 句 を適用する場合、またはある項目が項目に従属する場合、送出し側項目にも同じ出現回数を指定した OCCURS 句を使用するか、または OCCURS 句を使用しないようにする。これらの OCCURS 句に DEPENDING 句を含めてはならない。

一般規則

1. OCCURS 句の左辺にある画面記述では、LINE 句と COLUMN 句は個々の表要素に使用される。したがって、LINE 句と COLUMN 句には相対位置を指定する。これは、この 2 つの句に絶対位置を指定すると、すべての表要素が同じ位置に表示されてしまうためである。
2. OCCURS 句を指定した画面項目が出力項目であり送出し側項目に OCCURS 句が指定されていないと、DISPLAY 処理において送出し側の項目が、画面項目のすべての反復要素に転記される。全面的に添字付けされた基本項目は、OCCURS 句が使われていないものとみなされる。
3. OCCURS 句を指定した画面項目が更新項目であるか、または送出し側項目に OCCURS 句が指定されている出力項目であると、DISPLAY 処理において送出し側の項目の内容が画面項目の対応する要素に転記される。
4. OCCURS 句を指定した画面項目が更新項目または入力項目であると、ACCEPT 処理において画面項目各要素に入力されたデータが、受取り側項目の対応する要素に転記される。

9.21 OVERLINE(上線)句

(MF)

OVERLINE (上線) 句は、画面に表示される画面項目のすべての文字の上側に、水平線を引く。その上線は、文字位置の範囲内に収まる。

一般形式

```
OVERLINE
```

構文規則

1. OVERLINE 句は、どのような画面項目にも指定できる。
2. OVERLINE 句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. 文字の上側に水平線を引く機能のないシステム上で指定しても、この句は効力を発揮しない。

一般規則

1. OVERLINE 句は、画面に表示される画面項目のすべての文字の上側に、水平線を引く。その上線は、文字位置の範囲内に収まる。

9.22 PICTURE(形式)句 - 画面節中

PICTURE (形式) 句は、画面項目の長さ、一般的性質、編集要件を記述する。

一般形式

```
{PICTURE} IS 文字列
{PIC    }
```

構文規則

1. PICTURE 句には、任意の標準編集文字を含めることができる。
2. 各 PICTURE 句には、FROM 指定、TO 指定または USING 指定のどれかを含めなければならない。USING 指定は、FROM 指定または TO 指定と共に使用してはならない。
3. PIC は、PICTURE の省略形である。
4. PICTURE 句は、基本項目だけに指定できる。
5. PICTURE 句は、FROM, TO, USING の各指定の中で参照するデータ項目の PICTURE 句と必ずしも同じである必要はない。しかし、暗黙の MOVE (転記) が正しく行えるようでなければならない。

一般規則

1. 文字列は、画面項目の長さと項類を表わす。これは、データ項目用の PICTURE 句の中の文字列と同じように使用する。

2. 数字画面項目は、すべて数字編集項目とするか、PICTURE 句内に 9 だけを含めるようにすることが望ましい。対応するデータ項目と画面項目との間でデータをやり取りする際に、必要に応じて編集または編集の解除が行われる。

9.23 PROMPT(プロンプト)句

(MF)

PROMPT (プロンプト) 句は、ACCEPT 処理を行う際に、画面上の画面項目の空き文字位置にプロンプトを表示するようにする。このプロンプトは、キー入力されるデータをシステムが受け取って、画面項目に入れられる状態にあることを示す。

一般形式

```
PROMPT [ CHARACTER IS { 一意名-1 } ]
                      { 定数-1   }
```

構文規則

1. PROMPT 句は、入力項目と更新項目と集団項目にだけ指定できる。
2. PROMPT 句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. 一意名-1 は、1 文字の英字または英数字のデータ項目とする。
4. 一意名-1 は、OCCURS 句の左辺には指定できない。
5. 定数-1 は、1 文字の英数字定数または表意定数とする。

一般規則

1. 画面節内の項目には、
   a. PROMPT 句は常にオンである。
   b. PROMPT 句が指定されていない場合、または CHARACTER 指定なしで指定されている場合、省略時解釈のプロンプト文字が使用される。
2. ACCEPT 文で参照される他の項目には、
   a. ACCEPT 文に PROMPT 句が指定されている場合にだけ、PROMPT はオンとなる。
   b. PROMPT 句が CHARACTER 指定なしで指定されている場合、省略時解釈のプロンプト文字が使用される。
3. CHARACTER 句を指定すると、該当する画面項目の空き文字位置にプロンプトが表示されるようになる。プロンプト文字は構成オプションに優先する。(構成オプションの詳細については、ユーザインターフェイスに関する COBOL システムのマニュアルを参照。)
4. PROMPT 句を指定すると、英数字または自由方式の数字の画面項目の中の後部の空白に代わって、プロンプト文字が表示される。また、固定方式の数字編集画面項目の先行部分の抑制された文字位置に、プロンプト文字が表示される。

5. 固定方式の非編集数字画面項目、またはゼロ抑制が指定されていない数字編集画面項目に対しては、PROMPT 句は効力を発揮しない。
6. あるフィールドで PROMPT 句がオンの場合、プロンプト文字が示すフィールドの終端までカーソルを移動することはできない。これを行おうとすると、カーソルは次のフィールドに移動する。
7. あるフィールドで PROMPT 句がオンでない場合、カーソルはフィールドの終わりの後にある空白に移動することができる。
8. SECURE (機密) 句を指定すると、この句は効力を発揮しない。
9. 画面項目中に表示されているプロンプト文字は、ACCEPT 処理が終わると空白に置き換えられる。

9.24 REQUIRED(必須)句

REQUIRED (必須) 句は、画面項目を必ず入力しなければならないことを指定する。

一般形式

構文規則

1. REQUIRED 句は、入力項目と更新項目と集団項目にだけ指定できる。
2. REQUIRED 句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. (MF) REQUIRED と EMPTY-CHECK は同義語であり、どちらを書いてもよい。

一般規則

1. REQUIRED 句が効力を発揮するのは、ACCEPT 文が実行されるときである。このときに、画面項目内にカーソルが位置付けられて、入力されたデータを受け取ることが可能となる。REQUIRED 句を指定しておくと、終了キーを押しても入力を終了できず、カーソルは項目の先頭に位置付け直されるようになる。
2. この句の要件を満たすためには、英数字の画面項目の場合は、空白でもプロンプト文字でもない文字を少なくとも 1 つ入力しなければならない。なお、数字の画面項目の場合は、値がゼロであってはならない。
3. 更新項目に関しては、元のデータと入力されたデータの両方によって、この句の要件が満たされるか否かが決まる。
4. ACCEPT 処理を終了させるためにファンクション･キーを使用した場合は、REQUIRED 句は効力を発揮しない。(構成オプションの詳細については、ユーザインターフェイスに関する COBOL システムのマニュアルを参照。)

5. REQUIRED 句の要件が満たされない場合、エラー･メッセージを画面上に表示させることができる。(構成オプションの詳細については、ユーザインターフェイスに関するCOBOL システムのマニュアルを参照。)

9.25 REVERSE-VIDEO(反転画面)句

REVERSE-VIDEO (反転画面) 句は、画面項目を反転して表示する。

一般形式

構文規則

1. REVERSE-VIDEO 句は、どのような画面項目にも指定できる。
2. REVERSE-VIDEO 句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。

一般規則

1. REVERSE-VIDEO 句は、画面項目を反転して表示する。

9.26 SECURE(機密)句

SECURE (機密) 句は、入力されたデータを画面に表示しないようにする。

一般形式

構文規則

1. SECURE 句は、入力項目と更新項目に対してだけ指定できる。
2. SECURE 句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。

   MF SECURE と NO-ECHO は同義語であり、どちらを書いてもよい。

一般規則

1. 入力項目に SECURE 句を指定すると、画面項目内には空白とカーソルだけが現れる。更新項目に SECURE 句を指定すると、元の内容は表示されるが、それを変更することはできない。

9.27 SIGN(符号)句 - 画面節中

SIGN (符号) 句は、演算符号の位置と表現形式を指定する。

一般形式

```
[SIGN IS] { LEADING } [SEPARATE CHARACTER]
          { TRAILING }
```

構文規則

1. SIGN 句は、PICTURE 文字列に記号文字"S" が含まれる入力項目と出力項目と更新項目にだけ指定できる。
2. SIGN 句は基本項目に対してだけ指定できる。

一般規則

1. 画面記述中に SIGN 句を指定するときは、SEPARATE 句も含めることが望ましい。SEPARATE 句を指定しないと、PICTURE 文字列中の"S" によって表わされる符号は数字と重なって表示される。

9.28 SIZE(大きさ)句

SIZE (大きさ) 句は、画面項目の現在の大きさを指定する。

一般形式

```
SIZE IS { 一意名-1 }
        { 整数-1   }
```

構文規則

1. SIZE 句は、基本画面項目に対してだけ指定できる。
2. 一意名-1 は、符号の付かない整数とする。また、一意名-1 は OCCURS 句の左辺にあってはならない。
3. 整数-1 は、符号なしとする。

一般規則

1. 指定した大きさがゼロのときは、SIZE 句は効力を発揮しない。
2. 数字または数字編集の画面項目に、値がゼロでない SIZE 句を指定すると、画面項目は自由方式のように扱われる。この指定は、構成オプションの設定に優先する。
3. 関連する PICTURE 句または VALUE 句によって暗黙的に示されるよりも値が小さい SIZE 句を指定すると、画面上では画面項目の左側の部分だけが表示される。ACCEPT 文に JUSTIFIED 句がある場合、画面項目の右側の部分だけが表示される。その画面項目の残りの部分は、状況に応じて、空白またはゼロとみなされる。
4. 出力項目または定数項目に、関連する PICTURE 句または VALUE 句によって暗黙的に示されるよりも値が大きい SIZE 句を指定すると、画面項目の右側の部分に空白が埋め込まれる。
5. 一意名-1 の値を変更すると、実行時に画面項目の有効な大きさが変えられる。このため、画面節内でその項目の後ろに記述されている項目の画面上の位置が、変わることがある。(前述の *LINE(行)句*節および *COLUMN(カラム)句*節を参照。)

9.29 TO(へ)句

TO 句は受け取るデータの宛先を指定する。

一般形式

TO 一意名-1

構文規則

1. TO 句を FROM 句と一緒に使用することは、対応して同じ一意名を指定した USING 句を使用することに等しい。TO 句を USING 句と一緒に使用してはならない。
2. TO 句内の一意名を修飾してもよい。画面項目に OCCURS 句が適用されていない場合、その一意名は添字付けまたは指標付けされていてもよい。FROM 句を定義できる場所はプログラムのファイル節、作業場所節、

   (MF)局所記憶節、

   連絡節のいずれかだけである。

一般規則

1. TO 句が記述されている画面項目に対して ACCEPT 文を実行すると、操作員が入力したデータが画面項目に受け取られ、それからそのデータが対応するデータ項目に転記される。

9.30 UNDERLINE(下線)句

UNDERLINE (下線) 句は、画面に表示される画面項目に下線を引く。

一般形式

```
UNDERLINE
```

構文規則

1. UNDERLINE 句は、どのような画面項目にも指定できる。
2. UNDERLINE 句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。

一般規則

1. UNDERLINE (下線) 句は、画面に表示される画面項目に下線を引く。
2. 下線を引く機能のないシステム上で指定しても、この句は効力を発揮しない。

9.31 USAGE(用途)句 - 画面節中

USAGE (用途) 句は計算機の記憶領域内のデータ項目の形式を指定する。

一般形式

```
[USAGE IS] { DISPLAY      }
           { DISPLAY-1    } (NF)
```

構文規則

1. 画面節のデータ項目の用途は、明示的または暗黙的に、USAGE DISPLAY または USAGE DISPLAY-1 と定義しなければならない。

9.32 USING(使用)句

USING 句は表示するデータの源を指定する。

一般形式

```
USING 一意名-3
```

構文規則

1. USING 句は、対応する一意名に同じものを指定した、TO 句と FROM 句に等しい。UNING 句を FROM 句または TO 句と一緒に使用してはならない。
2. USING 句内の一意名を修飾してもよい。画面項目に OCCURS 句が適用されていない場合、その一意名は添字付けまたは指標付けされていてもよい。FROM 句を定義できる場所はプログラムのファイル節、作業場所節、局所記憶節、連絡節のいずれかだけである。

一般規則

1. USING 句が記述されている画面項目に対して DISPLAY 文を実行すると、対応するデータ項目から画面項目にデータが転記され、それから画面項目が画面上に表示される。
2. USING 句が記述されている画面項目に対して ACCEPT 文を実行すると、操作員が入力したデータが画面項目に受け取られ、それからそのデータが対応するデータ項目に転記される。

9.33 VALUE(値)句 - 画面節中

VALUE (値) 句は、画面に表示する定数の情報を指定する。

一般形式

[VALUE IS] 定数-1

構文規則

1. VALUE 句に指定する定数は、文字定数とする。表意定数は指定できない。
2. VALUE 句は、PICTURE 句を伴わない基本項目に対してだけ指定できる。

一般規則

1. VALUE (値) 句は、画面に表示する定数の情報を指定する。

9.34 ZERO-FILL(ゼロ補填)句

(MF)

ZERO-FILL (ゼロ補填) 句は、後部のプロンプト文字を空白ではなく、ゼロで置き換えることを指定する。

一般形式

ZERO-FILL

構文規則

1. ZERO-FILL 句は、英字または英数字の入力項目と更新項目に対してだけ指定できる。

一般規則

1. ZERO-FILL 句は、データが画面項目から受取り側項目に転記されるときに、後部のプロンプト文字を空白ではなく、ゼロで置き換えさせる。これは画面項目にデータが入力されたときにだけ行われる。
2. ZERO-FILL 句を指定した画面項目の受取り側項目に、JUSTIFIED 句が指定されていると、桁寄せによって空いた先行部分にゼロが埋められる。

第 10 章 : 手続き部

10.1 一般説明

(ANS85)手続き部(procedure division)は、宣言しても、宣言しなくてもよい。

プログラム定義、

(ISO2002)(MF)関数定義、

またはメソッド定義の中の手続き部は、実行されるべき手続きを含む。これらの手続きは、宣言文または非宣言文のいずれか、または両方である。

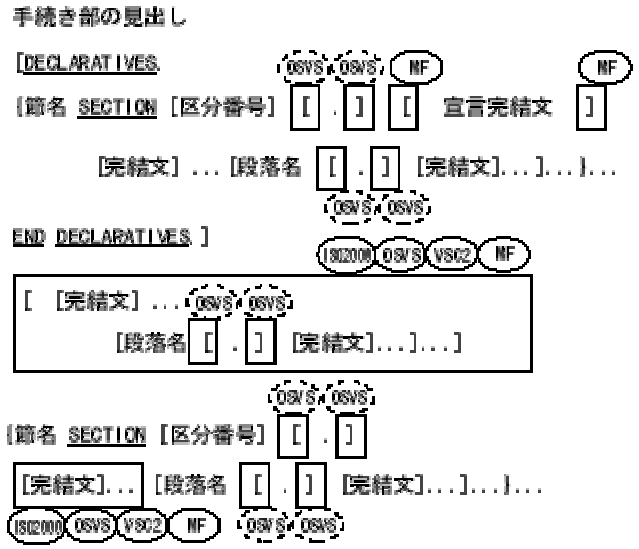
(ISO2002)(MF)オブジェクト定義およびファクトリ定義の中の手続き部は、オブジェクトまたはファクトリオブジェクトにより起動されるメソッドを含む。

(MF)呼出しプロトタイプの中の手続き部は、手続きをまったく含まない。ただし、ENTRY 文を含む場合はある。

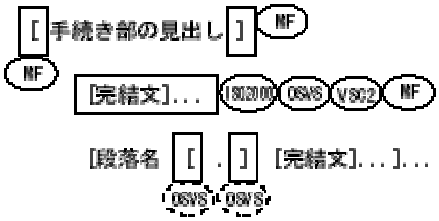
(ISO2002)(MF)関数プロトタイプ、メソッドプロトタイプ、またはプログラムプロトタイプの中の手続き部では、パラメータまたは戻り項目 (もし、あれば) を指定する。

一般形式

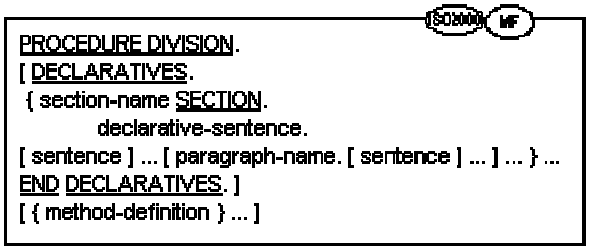
形式 1 (節を含む)



形式 2 (節を含まない)



形式 3 (インターフェイスオブジェクトまたはファクトリ)



用語、「手続き部の見出し」については、次の*手続き部の見出し*節で説明する。

形式 1 および 2 は、プログラム定義中、

ISO2002 MF プログラムプロトタイプ定義、関数定義、関数プロトタイプ定義、

ISO2002 MF メソッド定義、または

MF 呼出しプロトタイプ

でのみ使用できる。

ISO2002 MF 形式 3 は、ファクトリ定義またはインターフェイス定義、

ISO2002 MF またはオブジェクト定義

でのみ使用できる。

上記の宣言完結文 (declarative-sentence) は、USE 文である。(本書の***手続き部 - SEARCH - XML PARSE*** の章の *USE(使用)文*節、および ***言語リファレンス - 追加トピック***の中の ***デバッグモジュール***の章の *USE FOR DEBUGGING 文*節および ***報告書作成***の章の *USE BEFORE REPORTING 文*節を参照) USE 文は、該当する節をいつ実行するかを指定する。

手続きは、手続き部内の段落または段落集団、あるいは節または節集団から成る。手続き名は、それが使用される原始要素中の段落または節を示す語である。手続き名は段落名 (修飾される場合もある) または節名から成る。

ある段落がある節内にある場合、すべての段落がその中になくてはならない。

OSVS VSC2 MF この規則は強制しない。

段落は、段落名の後に続けて終止符(.)と空白を書き、後ろに完結文をいくつか書いたものである。段落の終了は、次の段落名または節名の直前、あるいは、手続き部の終わり、宣言部分の END DECLARATIVES のところまでとなる。

節は、節見出しと、その後に続くいくつかの段落から成る。節の終了は、次の節名、あるいは手続き部の終わり、宣言部分の END DECLARATIVES のところまでとなる。

VSC2 MF しかし、宣言中の節は宣言完結文を必要としない。これは PERFORM 文によって開始できる。(PERFORM 文の詳細については、*手続き部 - PERFORM - ROLLBACK* の章の *PERFORM(実行)文* 節を参照。)

VSC2 MF 完結文(sentence)は、上記で説明した名前付きの段落内に明示的には含まれない。名前のない暗黙的な段落内に含まれているとみなされる。段落は、節内に明示的には含まれない。名前のない暗黙的な節内に含まれているとみなされる。

プログラムの定義、

SOOOO MF またはメソッドの定義

では、実行は、手続き部の宣言部分ではない最初の文から始まる。以降、文は記述されている順序に実行される。ただし、他の順序が示されている場合は例外である。

MF 呼出しプロトタイプでは、手続き部は実行されない。

SOOOO MF オブジェクトの定義では、宣言部分が、親であるメソッド定義のすべてで指定されている場合と同様に動作する。

SOOOO ファクトリの定義では、宣言部分が、親であるメソッド定義のすべてで指定されている場合と同様に動作する。

ここでは、算術式と条件式、共通指定と入出力指定、組み込み関数、COBOL 動詞について説明する。

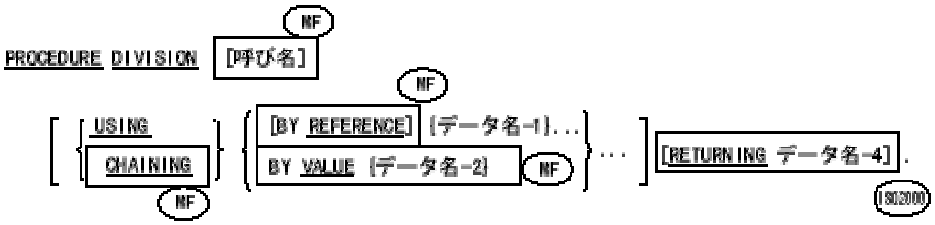
MF 手続き部の見出しは、手続き部の形式 2 では書いても書かなくてもよい。ただしこれは、、これより前に部が記述されていない場合、および手続き部が COBOL の文で始まる場合(節見出し、段落見出し、宣言節ではなく)にだけ、任意となる。

10.2 手続き部の見出し

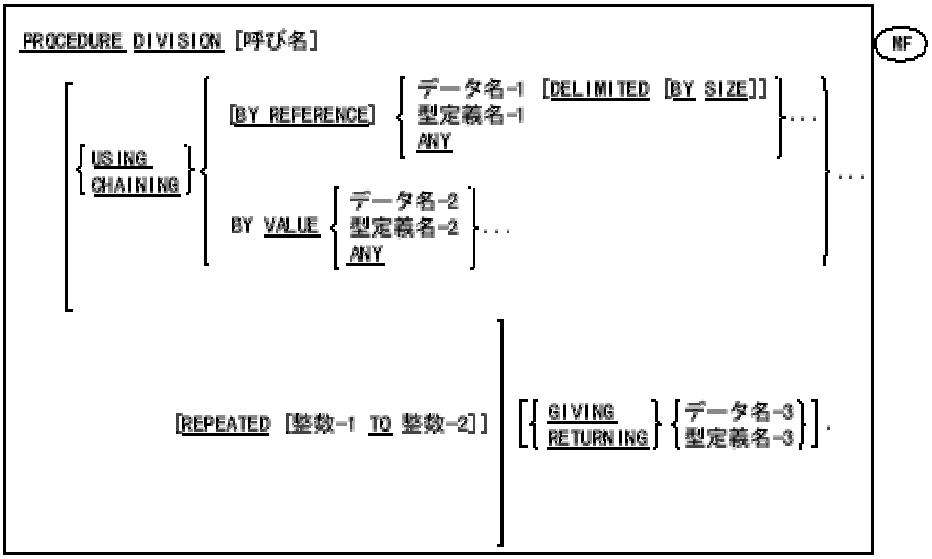
手続き部の冒頭には、下記の見出しを付けなければならない。

一般形式

形式 1



形式 2



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
   - DEFAULTCALLS - 手続き部の見出しに明示的に指定されていない場合に、省略時の呼出し方式を設定する。
   - STICKY-LINKAGE - 同じプログラム内で複数の呼出しを異なる入口点から行う間に、連絡節内のデータ項目の番地参照性を維持するか否かを制御する。

構文規則

すべての形式

1. (MF)呼び名は、目的プログラムが他のプログラムにより起動され、そのプログラムが COBOL システムの省略時設定として使用される以外の呼出し規則を使用している場合にだけ必要である。一般に、省略時設定の COBOL 呼出し規則は、実行時環境の COBOL 以外の言語の環境によって使用されるものと一致する。

   (MF)呼び名は、特殊名段落で定義しなければならない。この方法、および実行時環境でサポートされる呼出し規則とのインターフェイスを COBOL システムで記述する方法の詳細については、*環境部*の章の*特殊名段落*節を参照。

2. (MF)CHAINING および USING は同等である。
3. (ISO/IEC)(MF)データ名-1 またはデータ名-2 を USAGE OBJECT REFERENCE で指定した場合、ACTIVE-CLASS 指定は使用できない。
4. (ISO/IEC)(MF)データ名-3 は、 連絡節でレベル 01 またはレベル 77 記述項として指定しなければならない。データ名-3 のデータ記述項は、REDEFINES 句を含めてはならない。連絡節中の別の場所にあるデータ項目では、REDEFINES 句でデータ名-3 を指定してもよい。
5. (ISO/IEC)(MF)データ名-3 は、データ名-1 と データ名-2 のいずれとも同じであってはならない。

形式 1

6. データ名-1 は、呼ばれるプログラムのリンク節、

   (MF)ファイル節、または作業領域節

   で、レベル 01 またはレベル 77 項目として定義されなければならない。特定のユーザ定義の語は、データ名-1 として 2 回以上現れてはならない。

   (OSVS)(VSC2)(MF)2 回以上現れてもよい。

   データ名-1 の記述項目は、REDEFINES 句を含んではならない。

   (OSVS)(VSC2)(MF)REDEFINES 句を含んでもよい。

   しかし、データ名-1 はリンク節のどこにおいても REDEFINES 句の対象にできる。

7. (ISO/IEC)(MF)データ名-2 は、呼ばれるプログラムのリンク節、

   (MF)ファイル節、または作業領域節

   で、レベル 01 またはレベル 77 項目として定義されなければならない。データ名-2 は、字類が数字、オブジェクト、またはポインタのデータ項目として定義しなければならない。

   (MF)データ名-2 の字類はどれでもかまわない。ただし、その長さは 8 バイト以下でなければならない。

8. (ISO2002)(MF)RETURNING 指定は、関数定義および関数プロトタイプ定義の中で指定しなければならない。
9. (ISO2002)(MF)RETURNING 指定は、メソッド定義、プログラム定義、およびプログラムプロトタイプの中で指定しなければならない。

形式 2

10. (MF)形式 2 は、PROGRAM-ID 段落に記述された EXTERNAL 句のあるプログラムでだけ使用することができる。これは、呼出しプロトタイプである。
11. (MF)データ名-1 とデータ名-2 は、リンク節の 01 レベルレコードとして定義しなければならない。
12. (MF)型定義名-1、型定義名-2、型定義名-3 は、同じ原始ファイルに TYPEDEF 句でプログラマ定義の使用としてあらかじめ定義しなければならない。

一般規則

すべての形式

1. 手続き部見出しの USING 句では、5 つまでのデータ名が許される。

   (ISO2002)(MF)(COPEN)USING 句では、62 までのデータ名が許される。

2. BY REFERENCE 句および BY VALUE 句は、他の BY REFERENCE 句または BY VALUE 句があるまで、それに続くパラメータを移行する。はじめのパラメータに先行して BY REFERENCE 句または BY VALUE 句が記述されていないと、BY REFERENCE 句が想定される。

形式 1

3. (MF)呼び名は、このプログラムがそれをプログラム名あるいは入口名で起動した呼ぶプログラムにより適用されたとみなされる呼出し規則を確認する。実際に使用された呼出し規則が呼び名によって示されたものと異なる場合、COBOL システムが動作しなくなる場合がある。
4. (MF)データ名-1 がレベル 01 またはレベル 77 項目としてファイル節あるいは作業領域節に定義されている場合、目的プログラムは、データ項目がデータ名-1 と同じデータ宣言でリンク節に定義されたかのように動作し、呼ばれるプログラムの最初の文の実行に先立って、このデータ項目の内容がデータ名-1 に移動されたかのように動作する。最初のプログラムでは、これらの値はプログラムの作業領域のデータの初期化によって上書きされる。このため、呼ばれるプログラムには使用できない。
5. 呼ぶプログラムが COBOL の場合、以下の規則を適用する。呼ぶプログラムが COBOL でない場合、BY REFERENCE または BY VALUE 句を使用する必要がある時の詳細は、COBOL システムのマニュアルを参照。
6. USING 指定は、引数の渡し先であるプログラム、

   (ISO2002)関数、

(SO/IEC 1989:2002) (MF) またはメソッドが使用する仮パラメータの名前を識別する。渡された引数は、原始要素が起動する際に、以下のいずれかにより識別される。

- CALL 文の USING 指定
- (SO/IEC 1989:2002) 関数一意名中の引数
- (SO/IEC 1989:2002) (MF) INVOKE 文の USING 指定
- (SO/IEC 1989:2002) (MF) メソッドの行内呼出しで指定された引数
- (SO/IEC 1989:2002) (MF) 暗黙的 SET プロパティメソッドの起動に関するオブジェクトプロトタイプの規則で定義された引数

2 つの名前リストの間の対応は、位置を元にして決められる。

7. 内容により引数が渡された場合は、起動するランタイム要素により、実行の開始処理中に連絡節中のレコードが割り当てられた場合と同様に動作する。また、このレコードが、起動するランタイム要素の中の引数と同じ記憶域を占めていない場合と同様に動作する。

   起動する側のランタイム要素のリポジトリ段落に、起動される側のランタイム要素のプログラム指定子が含まれておらず、CALL 文に対する NESTED 指定がない場合は、この割り当てられたレコードは、引数と同じ長さとする。ここで、引数が変数反復データ項目として記述されていれば、最大長が使用される。引数は、変換されることなく、この割り当てられたレコードに転記される。このレコードはその後、起動される側のランタイム要素によって、引数として、また、参照により引き渡されるものとして扱われる。

   (SO/IEC 1989:2002) (MF) 起動される側のランタイム要素が、

   - 起動する側のランタイム要素のリポジトリ段落に、そのプログラム指定子が含まれているプログラム、
   - NESTED 指定を含む CALL 文で呼び出されたプログラム、
   - メソッド、または
   - 関数

   のいずれかである場合、この割り当てられたレコードは、以下のいずれかとなる。

   - 仮パラメータが ANY LENGTH 句で記述されている場合は、引数と同じ項類、用途、および長さのデータ項目
   - 仮パラメータが変数反復データ項目として記述されている場合は、仮パラメータと同じ記述およびバイト数 (最大長が使用される) のデータ項目

   以下のいずれかで、引数が送出し側作用対象として、また、割り当てられたレコードが受取り側作用対象として使用される。

   - 仮パラメータが数字項目の場合、ROUNDED 指定を伴わない COMPUTE 文
   - 仮パラメータの字類が指標、オブジェクト、またはポインタの場合、SET 文
   - MOVE 文

割り当てられたレコードは、その後、引数として、また、参照により引き渡されるものとして扱われる。

8. 引数が参照により引き渡される場合は、起動される側のランタイム要素は、仮パラメータが引数と同じ記憶域を占める場合と同様に動作する。
9. (ISO2002)(MF)引数が値により引き渡される場合は、起動される側のランタイム要素は、連絡節中のレコードが、起動する側のランタイム要素の起動の開始処理中に割り当てられる場合と同様に動作する。また、また、このレコードが、起動するランタイム要素の中の引数と同じ記憶域を占めていない場合と同様に動作する。この割り当てられたレコードの英数字文字位置の長さは、引数とまったく同じとする。この引数は、変換されることなく、この割り当てられたレコードに転記される。このレコードはその後、起動される側のランタイム要素によって、引数として、また、参照により引き渡されるものとして扱われる。
10. 呼ばれるプログラム中ではいつでも、データ名-1、

    (ISO2002)(MF)データ名-2、およびデータ名-3

    への参照は、呼ばれるプログラムのリンク節で与えられた項目の記述に従って決定される。この記述が呼ぶプログラムの相応するデータ項目よりも多くの文字位置を定義している場合、予期せぬ結果となる。この規則の適用に失敗したり、特定のランタイム環境用のシステム領域で許される最大サイズを超えると、システムはたいへんな損害を被ることになる。

11. (MF)プログラムが呼ばれ、USING 句の BY REFERENCE 作用対象が呼ぶプログラムのパラメータに相応する時、関連する接続が確立され、呼ぶプログラムに制御が戻るまで保たれる。プログラムが間にキャンセルせずに二度呼ばれ、同じ BY REFERENCE 作用対象が呼ぶプログラム中のパラメータと一致しない場合、この作用対象を STICKY-LINKAGE コンパイラ指令を指定せずに参照してはならない。
12. OPTIONAL 指定がデータ名-1 について使用されている場合は、対応する引数として OMITTED 指定を使用できる。そうでない場合は、OMITTED 指定を対応する引数として使用してはならない。

形式 2

13. (MF)呼出しプロトタイプ (PROGRAM-ID 段落中の EXTERNAL 句付きのプログラム) が 書き方 2 PROCEDURE DIVISION 見出しを持ち、同じ原始ファイル中の他のプログラムがこの呼出しプロトタイプのプログラム名を参照する CALL 文を持っている場合、以下の規則が提供される。
    1. 呼出し規則が CALL 文中に記述されている場合、これはプロトタイプに (明示的あるいは暗黙的に) 記述されたものと一致しなければならない。CALL 文に呼出し規則が記述されていない場合、プロトタイプに (明示的あるいは暗黙的に) 記述された呼出し規則が使用される。
    2. CALL 文に記述されたパラメータの数は、プロトタイプに記述されたパラメータの数と同じでなければならない (REPEATED 句により許されたものを除く) 。
    3. CALL 文に記述された各パラメータには、以下の規則が適用される。

a. BY REFERENCE 句または BY CONTENT 句がパラメータと関連している場合、プロトタイプ中の相応するパラメータは、BY REFERENCE で指定されなければならない。
b. BY VALUE 句がパラメータと関連している場合、プロトタイプ中の相応するパラメータは、BY VALUE で指定されなければならない。
c. CALL 文中のパラメータが BY REFERENCE 句、BY CONTENT 句、BY VALUE 句を持たない場合、プロトタイプに (明示的あるいは暗黙的に) 記述された句が使用される。
d. この規則は、相応する呼出しプロトタイプが見つからない時に適用される規則を置換える。つまり、呼出しプロトタイプが使用されない時、はじめの BY 句までのすべてのパラメータは、BY REFERENCE が指定され、BY 句の後のすべてのパラメータ (これ自身は明示的な BY 句を持たない) が最後の一つを使用するかのように扱われる。しかし、相応する呼出しプロトタイプが見つかる時は、BY 句用の定義が優先する。
e. プロトタイプ中のパラメータの字類が数字、あるいはポインタか索引データ名の場合、CALL 文中のパラメータは同じデータ定義を持たなければならない。
f. プロトタイプ中のパラメータの字類が英数字である場合、CALL 文中のパラメータも英数値クラスもので、プロトタイプ中のパラメータと少なくとも同じ長さでなければならない。
g. ANY がプロトタイプに記述されている場合、CALL 文中のパラメータはどの字類でもよい。

4. CALL 文が GIVING 句または RETURNING 句を含んでいる場合、プロトタイプも句を含んでいなければならない (逆も同様) 。CALL に記述されたデータ項目のデータ定義は、プロトタイプに記述されたデータ項目のものと同じでなければならない。
5. プロトタイプ中の DELIMITED BY SIZE 句は、英数字パラメータ用にだけ使用される。この場合、CALL 文中の対応するパラメータは暗黙に割当てられたデータ領域に移動され、バイナリ 0 (x"00") 文字がこのデータのコピーの直後に置かれる。BY SIZE 句が省略されると、CALL 文中の対応するパラメータが暗黙に割当てられたデータ領域に移動され、バイナリ 0 (x"00") 文字がコピーデータの最後のスペース以外の文字の直後に置かれる。これは、ヌルで終わるテキスト文字列を使用する C 言語のような言語とのインターフェイスを取ることを目的としている。
6. REPEATED 句は、少なくとも整数-1 と最終パラメータの整数-2 の繰返しがあることを示す。整数-1 TO 整数-2 が記述されないと、整数-1 は 0 となり、整数-2 は無限となる。

10.3 算術式

算術式 (arithmetic expression) には、数字基本項目の一意名、数字定数、これらの一意名や定数を算術演算子でつないだもの、2 つの算術式を算術演算子でつないだもの、算術式をかっこで囲んだものがある。算術式の前には単項演算子をつけることができる。

OSVS VSC2 MF 以下の記述における、

- 「数字データ項目」という語の意味には、浮動小数点数データ項目も含まれる。
- 「数字定数」という語の意味には、浮動小数点数定数も含まれる。

算術式の中に現れる一意名および定数は、算術演算を行える数字基本項目または数字定数のどちらかを表わしていなければならない。

10.3.1 算術演算子

算術式に使用できる演算子 (arithmetic operator) には、5 つの 2 項演算子 (binary arithmetic operator) と、2 つの単項演算子 (unary arithmetic operator) がある。これらの演算子は特定の記号で表わし、前後を空白で区切る。

2 項演算子	意味
+	加算
-	減算
*	乗算
/	除算
**	べき乗

単項演算子	意味
+	数字定数+1 を掛けることと同じ
-	数字定数-1 を掛けることと同じ

10.3.2 使用できる要素の組合せ

使用可能な変数、数字定数、算術演算子、およびかっこの組み合わせを表 10-1 に示す。

表 10-1: 算術式の中で使用できる要素

最初の要素	2 番目の要素				
	変数	* / ** + -	単項 + -	(	)
変数	-	P	-	-	P
* / ** + -	P	-	P	P	-
単項 + -	P	-	-	P	-
(	P	-	P	P	-
)	-	P	-	-	P

P　　許される要素の組合わせを示す

-	無効な組合わせを示す
変数	一意名または定数を示す

10.3.3 書き方と評価規則

算術式には、書き方と評価について、以下の規則が適用される。

1. 算術式の中では、要素が評価される順序を指定するために、かっこを使用できる。かっこ内の式が先に評価される。入れ子になった何組かのかっこについては、最も内側の組を最初にして、順次外側の組へと評価が進む。かっこがない場合、またはかっこ内の要素が同じ水準にある場合、評価順位は下記のようになる。

   1 番目 単項演算子の+と-

   2 番目 指数

   3 番目 乗算と除算

   4 番目 加算と減算

2. 以下に式の評価の例を示す。
3. 1 + 2 * 3
4. 上の式では、最初に乗算（下記）が実行される。
   a. 2 * 3

      次に、乗算の結果、6 を使用して加算が実行される。

   b. 1 + 6

      したがって、式の結果は 7 となる。

5. かっこを使用する目的は 2 通りある。1 つは、同じ評価順位の演算子がいくつか続く場合に、演算順序の曖昧さをなくすためである。もう 1 つは、通常の評価順位を変更することである。かっこによって評価順位が指定されない場合、評価順位が同じ一連の演算は、左から右に進められる。
6. 算術式の先頭には、記号の"("と "+"と " -" または変数だけを書くことができる。算術式の末尾には、記号の") "または変数だけを書くことができる。算術式の中では、左かっこと右かっこは対にする。左かっこが右かっこの左側に来るようにする。
7. 算術式を使用すると、作用対象の合成（composite of operands）および受取り側データ項目に関する制約を受けずに、算術演算子を組み合わせることができる。たとえば、この章で後述する *ADD 文*の構文規則 3 を参照。

10.3.4 中間結果

式が評価されると、一つ以上の中間結果が出される。概念的に、各中間結果は、一時データ項目として格納され、その値は ARITHMETIC 指令で選択したパラメーターによって決まる。

ARITHMETIC"MF"指令を選んだ場合、本 COBOL システムは、各中間結果を十進 40 桁の浮動小数点レジスタに評価する。ゼロでない各中間結果の最上位桁はこのレジスタの最上位桁に位置付けられる。小数点位置は中間結果の値に応じて浮動する。40 桁に格納できない桁は切り捨てられる。

ARITHMETIC"TRUNC20" コンパイラ指令を選択すると、Micro Focus COBOL は、ARITHMETIC"MF"を選択した場合と同じ方法で各中間結果を評価する。ただし、20 桁を超える小数点の右側の数字は、式の各段階で切り捨てられる。この指令は、COBOL WorkBench で提供されるものと同様の切り捨てが必要な場合にのみ使用すべきである。 この指令を使用する場合は、INTLEVEL コンパイラ指令の初期値、"2"の設定を変えてはならない。これは、一部の新機能 (全部で 18 桁を超える数字 PICUTURE など) を TRUNC20 とともに使用できなくなることを意味する。

OS/VS COBOL, VS COBOL II, DOS/VS COBOL, COBOL/370 および OS/390 システムでは、中間結果を、式中の要素の PICTURE 句によって決定される精度で計算するので、各バージョンによって多少異なってくる。詳細については、*IBM VS COBOL for OS/VS* マニュアル、*VS COBOL II Application Programming Guide*、および *COBOL for OS/390 & VM V2R2 Programming Guide* を参照。各中間結果は、PICTURE 9(n)V9(m) を持つ。n+m の最大値は、選択されたオプションによって 30 または 31 である。

中間結果用に OSVS または VSC2 オプションを記述した場合、このシステムは以下の制限で、選択されたメインフレームの動作をその結果を適当に切捨てて想定する。

- このシステムでは、最大の整数桁と小数桁は、それぞれ 18 とされている。中間結果の切捨ては、この制限に達した点で終了する。この場合、翻訳時に警告メッセージが出される。
- 結果は数字式の固定点だけに適用される。式が浮動小数点またはその結果が浮動小数点となるような操作を含んでいると(*手続き部 - 組み込み関数*の章を参照)、すべての中間結果の切捨てが起きる。
- 浮動小数点の結果は、その性質上不定であり、プラットフォームにより多少異なる。

10.4 条件式

条件式 (conditional expression) は、実行用プログラムの流れを制御する条件を表わす。条件を評価した結果の真理値に応じて、プログラムの処理の経路が選択される。条件式は、

ANS85 EVALUATE (評価) 、

IF (判断) 、PERFORM (実行) 、SEARCH (表引き) の各文の中に指定する。条件式は単純条件と複合条件に分類される。どちらの種類の条件も、任意の数のかっこで囲むことができる。かっこで囲んでも、条件の種類は変わらない。

OSVS OSVS システム指令を設定した場合、条件式の中で部分参照を行うことはできない。

10.4.1 単純条件

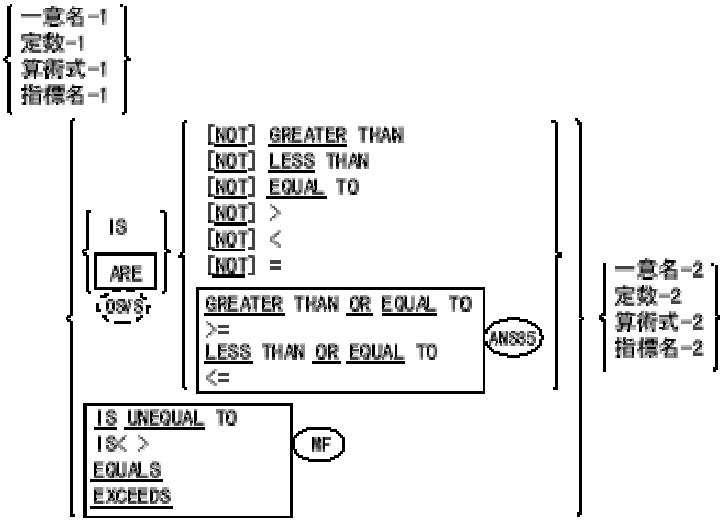
単純条件 (simple condition) には、比較条件、字類条件、条件名条件、スイッチ状態条件、符号条件がある。単純条件は真理値として、「真」または「偽」のどちらかをとる。単純条件をかっこで囲んでも、真理値は変わらない。

10.4.1.1 比較条件

比較条件 (relation condition) は、2 つの作用対象を比較する。各作用対象には、一意名によって参照されるデータ項目、定数、算術演算によって算出される値を使用できる。比較条件は、作用対象の間に関係が成立するときに、真理値「真」をとる。2 つの数字作用対象を、それぞれの USAGE 句に指定されている形式に関係なく、比較できる。その他の比較においては、作用対象の用途は同じでなければならない。作用対象のどちらかが集団項目である場合は、文字の比較規則が適用される。

OSVS OSC2 文字定数をかっこで囲むことができる。

比較条件の一般形式は下記のとおり。



比較文字の"＞" , "＜", "＝" は必要語であるが、下線が引かれていないことに注意。これは、"≧" のような他の記号との混同を避けるためである。

OSVS " = TO"と " ＞ THAN" と " ＜ THAN" も使用できる。

最初の作用対象 (一意名-1、定数-1、算術式-1) を、条件の左辺 (subject of condition) と呼ぶ。2 番目の作用対象 (一意名-2、定数-2、算術式-2) を条件の右辺 (object of condition) と呼ぶ。比較条件では少なくとも 1 つの変数を参照しなければならない。

比較演算子 (relational operator) は、比較条件の中で行う比較の種類を指定する。比較演算子としての予約語は、前後を空白で区切らなければならない。 必要語または比較文字の前に"NOT" を付けたものは、1 つの比較演算子である。たとえば、"NOT EQUAL" は「等しくな

い」ことを「真」とすることを表わし、"NOT GREATER" は「以下である」ことを「真」とすることを表わす。各比較演算子の意味を表 10-2 に示す。

下記の比較演算子は同義であり、どちらを書いてもよい。:

IS EQUAL TO	と	EQUALS;
IS NOT EQUAL TO	と	IS UNEQUAL TO;
IS GREATER THAN	と	EXCEEDS;
IS NOT GREATER THAN	と	IS LESS THAN OR EQUAL TO;
IS NOT LESS THAN	と	IS GREATER THAN OR EQUAL TO.

表 10-2: 比較演算子

意味	比較演算子
「より大きい」または「以下」	[IS / ARE] SYS [NOT] GREATER THAN [IS / ARE] SYS [NOT] >
「より小さい」または「以上」	[IS / ARE] SYS [NOT] LESS THAN [IS / ARE] SYS [NOT] <
「等しい」または「等しくない」	[IS / ARE] SYS [NOT] EQUAL TO [IS / ARE] SYS [NOT] = [IS] < > MF
「より大きい」または「等しい」	IS GREATER THAN OR EQUAL TO IS > = ANSES
「より小さい」または「等しい」	IS LESS THAN OR EQUAL TO IS < = ANSES
「等しい」	EQUALS MF
「等しくない」	IS UNEQUAL TO IS < > MF

「より大きい」	EXCEEDS (MF)
必要な比較記号、'<'、'>'、および '=' に下線が引かれない。これは、"≧" (「以上」) のような他の記号との混同を避けるためである。	

10.4.1.1.1 数字作用対象の比較

字類が数字の作用対象に関しては、その代数値が比較される。定数または算術式を表わす数字の数は意味をもたない。ゼロは符号が付いていても付いていなくても、1 つの値とみなされる。

これらの作用対象は、その用途がどのように記述されているかにかかわらず、比較することが許される。符号の付いていない数字作用対象は、比較上は、正の数として扱われる。

数字編集項目は、字類が英数字であり、これらの項目は、以下に示す文字作用対象の比較規則に従って比較される。

10.4.1.1.1 文字作用対象の比較

文字作用対象どうしの比較、または 1 つの数字作用対象と 1 つの文字作用対象の比較は、指定されている文字の照合順序に基づいて行われる。(***環境部***の章の*実行用計算機段落*節を参照。) 一方の作用対象が数字である場合、整数データ項目または整数定数にする。そして、下記のことに注意する。

1. 文字作用対象が基本データ項目または文字定数であるならば、数字作用対象は、標準データ形式で同じ大きさの英数字基本データ項目に転記されたように扱われる。そして、この英数字データ項目の内容が、文字作用対象と比較される。(***手続き部 - MERGE - OPEN***の章の *MOVE(転記)文*節を参照。)
2. 文字作用対象が集団項目であるならば、数字作用対象は、標準データ形式で同じ大きさの集団項目に転記されたように扱われる。そして、この集団項目の内容が文字作用対象と比較される。(***手続き部 - MERGE - OPEN***の章の *MOVE(転記)文*節を参照。)
3. 整数ではない数字作用対象は、文字作用対象とは比較できない。

作用対象の大きさは、作用対象中の標準データ形式文字の合計数である。

(OSVS)表意定数を持つ数字編集データ項目を比較することができる。これは英数字比較として扱われる。

(MF)用途が違っていても、数字作用対象と文字作用対象を比較できる。数字作用対象は、標準データ形式で同じ大きさの USAGE DIAPLAY 項目に転記されたように扱われる。そして、この内容が文字作用対象と比較される。

比較処理は 2 つの場合に分けられる。

1. 作用対象の大きさが等しい場合: 両方の作用対象の大きさが等しい場合は、上位の文字位置から下位へ向けて、対応する文字が順々に比較される。この比較は、対応する文字を比較して不一致が見つかるか、または作用対象の末尾にまで達した時点で終了する。両方の作用対象が等しいと判定されるのは、最初から最後まで、対応する文字がすべて等しかったときである。

   不一致であった最初の対応する文字は、文字の照合順序を比較される。そして、照合順序の大きい方の文字を含む作用対象の方が大きい作用対象であるとみなされる。

2. 作用対象の大きさが等しくない場合: 両方の作用対象の大きさが異なる場合は、短い方の作用対象の後ろに、長い方の作用対象と大きさが同じになるまで空白を付け加えたようにみなして、比較が行われる。

10.4.1.1.1 指標名および指標データ項目に関する比較

下記の項目の間でだけ、比較できる。

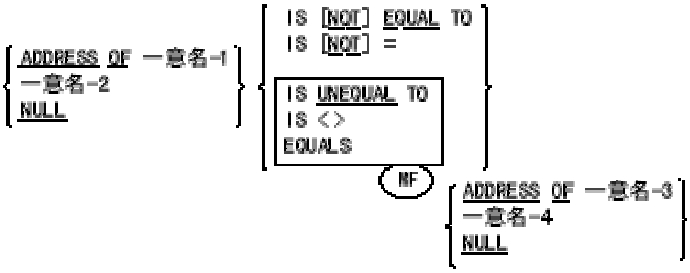
1. 2 つの指標名。結果は、指標名に対応する出現番号を比較したのと等しい。
2. 指標名と数字データ項目または数字定数。指標名の値に対応する出現番号が、数字データ項目または数字定数と比較される。
3. 指標データ名と指標名または他の指標データ項目。実際の値が、変換されることなしに比較される。
4. (OSVS)(VSC2)指標名と算術式。指標名の値に対応する出現番号が、式を計算した値と比較される。
5. 指標データ項目を上記以外のデータ項目または定数と比較すると、結果はどうなるかわからない。

10.4.1.1.1 用途がポインタであるデータ項目に関する比較

(VSC2)(MF)

用途が明示的にまたは暗黙的に、ポインタとされている 2 つのデータ項目を比較できる。ポインタの比較では、等しいか等しくないかだけを検査する比較演算子だけを使用できる。

一般形式



構文規則

1. 一意名-1 および一意名-3 は、連絡節の中の 01 レベルまたは 77 レベルの項目を参照する。
2. (MF)一意名-1 および一意名-3 は、データ部の中の任意のデータ項目を参照できる。
3. 一意名-2 および一意名-4 は、USAGE IS POINTER と指定されている項目を参照する。
4. 表意定数 NULL は、1 つの作用対象にだけ指定できる。

一般規則

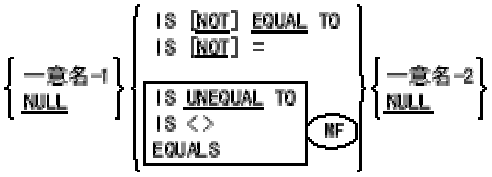
1. 2 つの番地が等しいときに、作用対象は等しいとみなされる。それ以外の場合は、作用対象は等しくないとみなされる。
2. この種の比較条件は、IF、PERFORM、EVALUATE、および SEARCH(形式 1)の各文の中で指定できる。形式 2 の SEARCH 文 (SEARCH ALL) の中では、この種の比較条件は使用できない。ポインタ･データ項目は、意味のある順序付けをすることができないからである。

10.4.1.1.2 用途が手続きポインタであるデータ項目に関する比較

(MF)(COB370)

用途が手続きポインタである、2 つのデータ項目を比較できる。

一般形式



構文規則

1. 一意名-1 および一意名-2 は、USAGE IS PROCEDURE-POINTER と指定されている項目を参照する。
2. 表意定数 NULL は、1 つの作用対象にだけ指定できる。

一般規則

1. 2 つの番地が等しいときに、作用対象は等しいとみなされる。それ以外の場合は、作用対象は等しくないとみなされる。
2. この種の比較条件は、IF, PERFORM, EVALUATE, SEARCH (形式 1) の各文の中で指定できる。 形式 2 の SEARCH 文 (SEARCH ALL) の中では、この種の比較条件は

使用できない。ポインタ･データ項目は、意味のある順序付けをすることができないからである。

10.4.1.1.3 用途がオブジェクトであるデータ項目に関する比較

2 つのオブジェクト参照を比較して、同じオブジェクトを参照しているかを確認できる。

一般形式

```
{オブジェクト一意名-1}
{NULL               }

{ IS { [NOT]EQUAL TO }
{    { [NOT]=        }
{    { <>            }
{
{ IS UNEQUAL TO      }
{ EQUALS             }

{オブジェクト一意名-2}
{NULL                }
```

構文規則

1. 表意定数 NULL は、1 つの作用対象にだけ指定できる。

一般規則

1. 同じオブジェクトを参照している場合は、作用対象は等しい。そうでない場合は等しくない。

10.4.1.2 字類条件

機能

字類条件 (class condition) は、作用対象が数字であるか英字であるか、

(ANS85)あるいは、小文字の英字であるか、大文字の英字であるか、環境部の特殊名段落中に CLASS 句によって指定されている文字集合に属する文字だけが含まれるかを、判定する。

一般形式

```
                 { NUMERIC                 }
                 { ALPHABETIC-LOWER        }
一意名-1 IS [NOT] { ALPHABETIC-LOWER        }
                 { ALPHABETIC-UPPER        } (ANS85)
                 { クラス名-1               }
```

構文規則

1. 作用対象の字類は、下記のように判定される。

a. すべての文字が、0, 1, 2, 3, ... 9 から構成される場合、作用対象は数字である。この場合、演算符号の有無は問わない。
b. すべての文字が、大文字の A, B, C, ...Z、または

ANS85小文字の a, b, c, ...z、

または大文字

ANS85と小文字

および空白の任意の組合わせから構成される場合、作用対象は英字である。

c. ANS85すべての文字が小文字の a, b, c, ...z および空白から構成される場合、作用対象は小文字の英字である。
d. ANS85すべての文字が大文字の A, B, C, ...Z および空白から構成される場合、作用対象は大文字の英字である。
e. ANS85すべての文字が特殊名段落中の字類名-1 の定義に含まれる文字から構成される場合、作用対象は字類名-1 に一致する。

2. 一意名-1 は、用途が明示的または暗黙的に DISPLAY であるデータ項目を参照しなければならない。または、NUMERIC かどうか検査する場合、用途は DISPLAY、

MFCOMPUTATIONAL、COMPUTATIONAL-X、

OSVS VSC2 MF OPENCOMPUTATIONAL-3、

MF OPENCOMPUTATIONAL-5、

ANS85または PACKED-DECIMAL

のどれかとする。

3. ANS85一意名-1 が関数一意名である場合、英数字関数

MFまたは各国語型関数

を参照しなければならない。

一般規則

1. 字類条件が語 NOT を含まず、一意名-1 が長さゼロの集団項目である場合、字類検査の結果は常に偽となる。

   NOT を指定した場合、NOT とこれに続く語は、真理値を導くための字類検査を定義する 1 つの字類条件となる。例えば NOT NUMERIC は、作用対象が文字であるかを判断する真偽検査である。字類条件が語 NOT を含んでいて、一意名-1 が長さゼロの集団項目である場合、字類検査の結果は常に真となる。

(MF)長さゼロの集団項目である場合の字類検査の真理値は、ZEROLENGTHFALSE 指令によって予約されている。

2. NUMERIC 検査は、データ記述項に英字と記述されている項目、または符号付き基本項目を含むと記述されている集団項目に対しては、使用できない。検査対象のデータ項目のデータ記述項に演算符号を付けるように記述されていない場合、その内容が数字でかつ演算符号が付いていないときにだけ、そのデータ項目は数字項目であると判定される。 検査対象のデータ項目のデータ記述項に演算符号を付けるように記述されている場合、その内容が数字でかつ有効な演算符号が付いているときにだけ、そのデータ項目は数字項目であると判定される。SIGN IS SEPARATE 句が記述されているデータ項目用の有効な演算符号は、標準データ形式文字の"+"と"-" である。SIGN IS SEPARATE 句が記述されていないデータ項目用の有効な演算符号は、*COBOL 言語の概念*の章の*文字の表現と基数の選定*節を参照。

   (VSVS)(MF)NUMERIC 検査を、データ記述項に演算子が記述されている基本項目から成る集団項目に対して使用できる。

3. ALPHABET 検査は、データ記述に数字であると記述されているデータ項目に対しては、使用できない。検査対象のデータ項目が英字であると判定されるのは、その内容が大文字の英字"A"から"Z" と空白の任意の組合わせ、

   (ANS85)および小文字の英字"a"から"z"

   と空白の任意の組合わせで構成されているときだけである。

   (VSVS)(VSC2)(MF)外部浮動小数点数項目 (USAGE DISPLAY) および内部浮動小数点数項目 (USAGE COMP-1 および USAGE COMP-2) には、字類条件を使用できない。

4. (ANS85)ALPHABETIC-LOWER 検査は、データ記述に数字であると記述されているデータ項目に対しては、使用できない。 ALPHABETIC-LOWER 検査の結果は、一意名-1 によって参照されるデータ項目の内容が、すべて小文字の英字"a" から"z" と空白で構成されているときに、真となる。
5. (ANS85)ALPHABETIC-UPPER 検査は、データ記述に数字であると記述されているデータ項目に対しては、使用できない。ALPHABETIC-UPPER 検査の結果は、一意名-1 によって参照されるデータ項目の内容が、すべて大文字の英字"A" から "Z" と空白で構成されているときに、真となる。
6. (MF)字類名-1 検査は、データ記述に数字であると記述されているデータ項目に対しては、使用してはならない。

10.4.1.3 条件名条件 (条件変数)

機能

条件名条件 (condition-name condition) は、条件変数 (condition variable) の値が条件名に対応する値のどれかと等しいか否かを判定する。

一般形式

条件名

構文規則

1. (OSVS)(VSC2)(MF)条件名に、2 バイト文字および内部浮動小数点数を使用できる。
2. (MF)条件名に、外部浮動小数点数を使用できる。

一般規則

1. 条件名にいくつかの値の範囲が関係付けられている場合、条件変数の値がその範囲内 (境界値を含む) に入るか否かが検査される。
2. 条件変数と条件名の値を比較する際の規則は、比較条件の場合と同じである。
3. 条件変数の値が条件名に対応する値の中のどれかに等しければ、検査の結果は真となる。

10.4.1.4 スイッチ状態条件

スイッチ状態条件 (switch-status condition) は、

(MF)9 つある

COBOL スイッチのそれぞれが「オン」の状態にあるか「オフ」の状態にあるかを判定する。

(MF)スイッチには、順に SWITCH-0 から SWITCH-8 という名前が付けられている。

それらの各スイッチの値 (「オン」と「オフ」) は、COBOL 実行用プログラムの実行を開始するときに、操作員によって設定される。(ランタイムスイッチの詳細については、COBOL システムのマニュアルを参照。) 条件名に対応するスイッチと「オン」または「オフ」状態は、環境部の特殊名段落の中で定義しておく。***環境部***の章の*特殊名段落*節を参照。

一般形式

条件名

条件名に定義されている状態にスイッチが設定されていると、検査の結果は真となる。

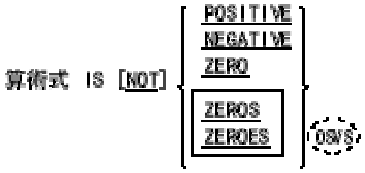
コンパイラ指令 SWITCH-TYPE を初期値の 1 に設定すると、スイッチを設定するランタイム要素の内部から COBOL スイッチが設定されているかどうかをテストする以外に、スイッチ状態条件を使用することはできない。

コンパイラ指令 DIALECT が ISO2002 に設定されると、コンパイラ指令 SWITCH-TYPE が 2 に設定される。この場合、実行単位のいずれかのランタイム要素の内部から COBOL スイッ

チが設定されているかどうかをテストする以外に、スイッチ状態条件を使用することはできない。

10.4.1.5 正負条件

正負条件 (sign condition) は、算術式の値が正か負かゼロかを判定する。 正負条件の一般形式は以下の通り。



正負条件の必要語に "NOT" を付けたものは、1 つの正負条件である。この条件に照らして代数検査が行われ、真理値が求められる。たとえば、 "NOT ZERO" は作用対象の値がゼロでない(正または負) のときに真となる。作用対象は、値がゼロよりも大きいときに正であり、値がゼロよりも小さいときに負であり、値がゼロのときゼロである。算術式は、少なくとも 1 つの変数を参照しなければならない。

OSVS 符号検査において、ZERO の代わりに ZEROS または ZEROES を使用できる。

10.4.1.6 省略引数条件

省略引数条件で、関数、メソッド、またはプログラムに引数が与えられたかどうかが決定される。

一般形式

データ名-1 IS [NOT] OMITTED

構文規則

1. データ名-1 は、省略引数条件が指定された原始要素で定義された仮パラメータでなければならない。

一般規則

1. OMITTED のテスト結果は、以下の場合に真となる。
    a. このプログラム、関数、またはメソッドを起動する文の中のデータ名-1 に対応する引数として、 一意名または定数でなく、OMITTED 指定が使用されている。
    b. 起動する側の文で、その後に続く引数が省略されており、それがデータ名-1 に対応する引数である。

c. データ名-1 に対応する引数自体が、省略引数条件が真の仮パラメータである。

2. NOT およびキーワード OMITTED を使用する場合は、これらにより、真の値に対して実行される条件が指定される。

10.4.1.6 INSTANCE-OF 条件

(MF)

INSTANCE-OF 条件は、オブジェクト参照が特定のクラスおよびインターフェイスのどちらのインスタンスであるかを決定する。

一般形式

```
オブジェクト一意名-1 IS INSTANCE OF { [ FACTORY OF ] クラス名-1  [ ONLY ] }
                                    { インターフェイス名-1                 }
```

構文規則

1. オブジェクト一意名-1 はオブジェクト参照でなければならない。
2. クラス名-1 は、この原始要素のリポジトリ段落またはクラス制御段落で指定されたクラスの名前でなければならない。
3. インターフェイス名-1 は、この原始要素のリポジトリ段落またはクラス制御段落で指定されたインターフェイスの名前でなければならない。

一般規則

1. FACTORY 指定と ONLY 指定のどちらも使用しない場合に、オブジェクト一意名-1 と名付けられたオブジェクトが、クラス名-1 と名付けられたクラスまたはそのサブクラスのインスタンス、およびインターフェイス名-1 と名付けられたインターフェイスまたはそのサブインターフェイスのいずれかであれば、INSTANCE-OF テストの結果は真となる。
2. FACTORY 指定を使用し、ONLY 指定を使用しない場合に、オブジェクト一意名-1 と名付けられたオブジェクトがクラス名-1 と名付けられたクラスのファクトリであれば、INSTANCE-OF テストの結果は真となる。
3. ONLY 指定を使用し、FACTORY 指定を使用しない場合に、オブジェクト一意名-1 と名付けられたオブジェクトがクラス名-1 と名付けられたクラスのインスタンスであり、そのサブクラスでなければ、INSTANCE-OF テストの結果は真となる。
4. FACTORY 指定と ONLY 指定の両方を使用する場合に、オブジェクト一意名-1 と名付けられたオブジェクトがクラス名-1 と名付けられたクラスのファクトリであり、そのサブクラスでなければ、INSTANCE-OF テストの結果は真となる。

10.4.2 複合条件

複合条件は、論理的結合子 (論理演算子、"AND" および "OR") による単純条件、組み合わせ条件、および複合条件の結合、または、これらの条件の論理否定 (論理演算子、"NOT") による否定で形成される。複合条件の真の値は、かっこに囲まれているかどうかにかかわらず、単純条件の個々の真の値に関する、記述されたすべての論理演算子の連携、または、論理的に結合または否定された条件の中間の真の値の結果である真の値である。

論理演算子およびその意味は以下のとおりである。

演算子	意味
AND	論理的な結合。真の値は、結合されている条件が真であれば、"真"。結合されている条件の片方または両方が偽であれば、 "偽"。
OR	論理的な包含。真の値は、包含されている条件の片方または両方が真であれば、、"真"。包含されている条件の両方が偽であれば、 "偽"。
NOT	論理的な否定、または真の値の逆。 真の値は、条件が偽であれば、"真"。条件が真であれば、 "偽"。

論理演算子の前後には、空白を入れなければならない。

10.4.2.1 否定単純条件

否定単純条件 (negated simple condition) は、単純条件に論理演算子の"NOT" を付けたものである。否定単純条件は、単純条件の真理値を逆転させる働きをする。したがって、単純条件の真理値が「偽」のときに否定単純条件の真理値は「真」となり、単純条件の真理値が「真」のときに否定単純条件の真理値は「偽」となる。否定単純条件は、単純条件の真理値を逆転させる働きをする。したがって、単純条件の真理値が「偽」のときに否定単純条件の真理値は「真」となり、単純条件の真理値が「真」のときに否定単純条件の真理値は「偽」となる。否定単純条件をかっこで囲んでも真理値は変わらない。

一般形式

NOT 単純条件

10.4.2.2 組合せ条件と否定組合せ条件

組合せ条件 (combined condition) は、論理演算子の"AND" または "OR" で条件をつないだものである。

一般形式

条件 {{AND / OR} 条件}...

構文規則

1. 使用可能な条件は以下の通り。
   - 単純条件
   - 否定単純条件
   - 組合せ条件
   - 否定組合せ条件 (組合せ条件をかっこで囲み、前に論理演算子の"NOT" を付けたもの。)
   - 上記の条件の組合せ。表 10-3 に要約した規則に従って指定する。
2. 組合せ条件の中で "AND" または "OR" のどちらか一方だけを使用するときは、かっこを使う必要はまったくない。しかし、 "AND" と "OR" と "NOT" を混ぜ合わせて使用するときは、かっこを使用することによって、最終的な真理値に影響が出る。

一般規則

1. 組合せ条件にかっこを指定しないと、論理演算子の優先順位 (結合力) に従ってどの論理演算子にどの条件が適用されるかが決まり、それに応じて暗黙的にかっこがあるものと想定される。この優先順位は"NOT","AND" , "OR" の順である。たとえば、「条件-1 OR NOT 条件-2 AND 条件-3」と指定すると、暗黙のうちに「条件-1 OR ((NOT 条件-2) AND 条件-3) 」と指定したことになる。
2. 組合せ条件にかっこを指定すると、条件と論理演算子を結合するにあたって、かっこを含めた優先順位が適用される。したがって、かっこを使用すると、論理演算子の通常の優先順位を変えることができる。たとえば、上の例において、「 (条件-1 OR (NOT 条件-2)) AND 条件-3」と指定すると、意味は変わってくる。(後述の*条件評価規則*節を参照。)

表 10-3 に、条件と条件演算子を結び付け、かっこで囲む有効な組合せを示す。左かっこと右かっこは 1 対 1 で対応し、左かっこが右かっこよりも左側にこなければならない。

表 10-3: 条件と論理演算子とかっこの組合せ

要　素	条件式中の位置	直前にきてよい要素	直後にきてよい要素
単純条件	どこでも可	OR, NOT, AND, (	OR, AND,)
OR または AND	最初と最後は不可	単純条件,)	単純条件, NOT, (
NOT	最後は不可	OR, AND, (	単純条件, (
(	最後は不可	OR, NOT, AND, (	単純条件, NOT, (
)	最初は不可	単純条件,)	OR, AND,)

表 からわかるとおり、"OR NOT" は許される組合せであるが、"NOT OR" は許されない組合せである。また、"NOT (" は許されるが、 "NOT NOT " は許されない) 。

10.4.3 略記組合せ比較条件

単純比較条件または否定単純比較条件を、論理結合語で組み合わせていくつも続けて書く場合、先行する比較条件と共通の左辺または左辺と比較演算子があり、かつかっこを使用していなければ、最初のものを除く共通の比較条件を省略できる。下記の 2 通りの方法がある。

- 比較条件の左辺を省略する
- 比較条件の左辺と比較演算子を省略する

一般形式

```
比較条件 { { AND | OR } [NOT] [比較演算子] オブジェクト }...
```

一続きの比較条件の中で、上記の両方の形の省略方法を使用できる。略記した場合、先行する明記した最後の左辺が省略した左辺を補うように挿入され、先行する明記した最後の比較演算子が省略した比較演算子を補うように挿入される。このように暗黙的に左辺および比較演算子が挿入された結果は、表 10-3 の規則に合わなければならない。省略した左辺や比較演算子が補われるこの処理は、組合せ条件の中に略記しない単純条件が出てくると終了する。

(OSVS)条件が評価される順序を、かっこを使用することによって変更できる。(下記の例を参照)

略記組合せ比較条件の中で"NOT" を使用した場合、下記のように解釈される。

1. (ANSI85)NOT の直後に GREATER, >, LESS, <, EQUAL, = のどれかが続く場合、NOT は比較演算子の一部と解釈される。ただし、GREATER THAN OR EQUAL TO, >=, LESS THAN OR EQUAL TO, <=は上記の比較演算子に含まれない。
2. 上記以外の場合、"NOT" は論理演算子として解釈される。したがって、否定比較条件の中に暗黙の左辺または比較演算子が補われる。

略記組合せ比較条件および略記否定組合せ条件と、それに相当する略さない形の例をいくつか下に示す。

略記組合せ比較条件	略さずに書いた場合
a > b AND NOT < c OR d	((a > b) AND (a NOT < c)) OR (a NOT < d)
a NOT EQUAL b OR c	(a NOT EQUAL b) OR (a NOT EQUAL c)
NOT a = b OR c	(NOT (a = b)) OR (a = c)

NOT (a GREATER b OR < c)	NOT ((a GREATER 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))))
x > a OR y AND z	x > a OR (x > y AND x > z)
OSVS x > a OR (y AND z)	x > a OR (x > y AND x > z)
OSVS x > (a OR y) AND z	(x > a OR x > y) AND x > z
OSVS x (= a OR > b)	x = a OR x > b
OSVS x = a AND (> b OR < z)	x = a AND (x > b OR x < z)
a EQUAL b OR NOT GREATER OR EQUAL c OR d	(a EQUAL b) OR (NOT (a GREATER OR EQUAL c)) OR (a GREATER OR EQUAL d)
a EQUAL b OR NOT >=c OR d	(a EQUAL b) OR (NOT (a >= c)) OR (a >= d)

10.4.3.1 条件評価規則

暗黙的な評価順序を変更する必要があるときは、かっこを使用して、組合せ条件の個々の条件が評価される順序を指定できる。

かっこ内の条件が先に評価される。かっこが入れ子になっている場合は、最も内側のかっこから順次外側へと評価が進められる。かっこが使用されていないかまたはかっこ内の条件が同じレベルにあるときは、下記の暗黙的な階層順序に従って論理評価が進められ、最終的な真理値が決定される。

1. ANS85 条件に算術式または関数が含まれている場合、その条件が評価されるときに、これらの式および関数の値が算出される。同様に、組合せ条件に否定条件が含まれる場合、この組合せ条件を評価する必要があるときに、否定条件が評価される。(前述の*書き方と評価規則*を参照。)
2. 単純条件の真理値が、下記の順で決定される。

   比較 (略記比較条件があれば、通常の形に展開してから)
   字類
   条件名
   スイッチ状態
   正負

3. 否定条件の真理値が決定される。
4. 組合せ条件の真理値が決定される。論理演算子 "AND" が先に評価され、つづいて論理演算子 "OR" が評価される。
5. 否定組合せ条件の真理値が決定される。
6. 一連の条件の評価順序がすべてかっこで指定されていない場合、同じレベルの複数の条件は左から右へ評価される。

10.5 多くの文に共通する句と一般規則

以降に説明する各文において、次のような指定が頻繁に出てくる。ROUNDED (四捨五入) 指定、ON SIZE ERROR (桁あふれ) 指定、

ANS85 NOT ON SIZE ERROR (桁あふれなし) 指定、

CORRESPONDING (対応) 指定などである。以下に、これらを個々に説明する。

以降の説明の中で、「結果の一意名」という用語を使用する。これは算術演算の結果を入れる一意名を表わす。

10.5.1 ROUNDED 指定

小数点の位置をそろえたとき、算術演算の結果の小数部の桁数が結果の一意名の小数部の桁数よりも大きいと、結果の一意名の大きさに応じて切捨てが発生する。四捨五入が要求されると、あふれる桁の最上位の値が 5 以上であると、結果の一意名の最下位の桁の絶対値に 1 が加えられる。

結果の一意名の整数部分の下位が PICTURE 文字の "P" を用いて表わされているときには、四捨五入や切捨ては、実際に記憶場所を割り当てられている部分の右端の整数部に対して行われる。

OS/VS VSC2 MF 浮動小数点数の算術演算においては、ROUNDED 指定は注記として扱われる。浮動小数点数の演算結果は、つねに四捨五入される。

10.5.2 ON SIZE ERROR 指定 ANS85 と NOT ON SIZE ERROR 指定

小数点の位置をそろえたとき、算術演算の結果の絶対値が対応する結果の一意名がとる最大値を超えると、桁あふれ条件が発生する。除数がゼロである除算が行われると、つねに桁あふれ条件が発生する。ON SIZE ERROR 指定を指定しなかったときのゼロによる除算の結果はどうなるかわからないので、注意する必要がある。一般に、桁あふれ条件は最終結果に対してだけ適用される。ただし、MULTIPLY (乗算) 文および DIVIDE (除算) 文の場合は、中間結果に対しても桁あふれ条件が適用される。

ANS85 指数の評価規則から外れると、つねに、算術演算は停止され、桁あふれ条件が発生する。

ROUNDED 指定を書くと、四捨五入の後で桁あふれの検査が行われる。その結果桁あふれ条件が発生すると、ON SIZE ERROR 指定を指定してあったか否かによって、下記のように処理が分かれる。

10.5.2.1 ON SIZE ERROR 指定がない場合

桁あふれ条件が発生すると、対応する結果の一意名の値はどうなるかわからない。この演算処理が実行される間に、桁あふれ条件が発生しなかった結果の一意名の値が、他の結果の一意名に対して発生した桁あふれ条件の影響を受けることはない。

ANS85 算術演算が終了すると、制御は算術文の末尾に移される。NOT ON SIZE ERROR 指定は指定されていても無視される。

除数がゼロの除算の結果は、ON SIZE ERROR を指定していないと、どうなるかわからない。それを避けるためには、CHECKDIV コンパイラ指令と O ランタイムスイッチを使用するとよい。

10.5.2.2 ON SIZE ERROR 指定がある場合

桁あふれ条件が発生すると、桁あふれの影響を受ける結果の一意名の値は変更されない。桁あふれ条件の発生しなかった結果の一意名の値が、この演算処理が実行される間に他の結果の一意名に対して発生した桁あふれ条件の影響を受けることはない。この演算処理が終了すると、ON SIZE ERROR 指定内の無条件命令が実行される。

CORRESPONDING 指定を伴う ADD (加算) 文および SUBTRACT (減算) 文に関しては、個々の演算処理において桁あふれ条件が発生しても、その時点では ON SIZE ERROR 指定内の無条件命令は実行されない。個々の加算または減算がすべて終了した後に、ON SIZE ERROR 指定内の無条件命令が実行される。

ANS85 桁あふれ条件が発生すると、ON SIZE ERROR 指定が指定されていても指定されていなくても、NOT ON SIZE ERROR 指定は無視される。

ANS85 ON SIZE ERROR 指定と NOT ON SIZE ERROR 指定の両方を指定したときに、実行された方の指定の中に制御を明示的に移す文が含まれていないと、必要があればその指定の実行が終わった時点で、その算術文の末尾に制御が暗黙的に移される。

10.5.2.3 NOT ON SIZE ERROR 指定

ANS85

算術演算文を実行したときに桁あふれ条件が発生しなかった場合に NOT ON SIZE ERROR 指定が指定されていれば、その中に記述されている無条件命令が実行される。この場合、ON SIZE ERROR 指定が指定されていても無視され、その中に記述されている無条件命令は実行されない。

10.5.3 CORRESPONDING 指定

以下の説明で、d1 と d2 という記号を使用する。この d1 と d2 はそれぞれ別々の集団項目を指す一意名を表わす。d1 と d2 から 1 つずつ取った一組のデータ項目は、下記の条件が満たされるとき対応するという。

1. d1 中のデータ項目および d2 中のデータ項目は、FILLER (無名項目) ではなくデータ名が同じであり、同じように修飾されている。ただし、d1 と d2 そのものは修飾語に使われていない。
2. d1 中のデータ項目および d2 中のデータ項目は、少なくとも 1 つが基本項目である。CORRESPONDING を伴う MOVE (転記) 文に関しては、結果として各データ項目に適用される転記は転記規則に合致している。CORRESPONDING を伴う ADD 文および SUBTRACT 文に関しては、両方のデータ項目とも基本数字データ項目である。
3. d1 および d2 のデータ記述にレベル番号 66、77、

   (MF)78

   または 88、USAGE IS INDEX 句、

   (MF)または USAGE IS OBJECT 句

   が含まれていない。

4. d1 または d2 に属するデータ項目で、そのデータ記述中に REDEFINES、RENAMES、OCCURS、USAGE IS INDEX、

   (MF)USAGE IS PROCEDURE-POINTER、

   (VSC2)(MF)USAGE IS POINTER、

   (MF)または USAGE IS OBJECT

   の各句が含まれるものは無視される。また、REDEFINES、RENAMES、OCCURS、USAGE IS INDEX、

   (MF)USAGE IS PROCEDURE-POINTER、

   (VSC2)(MF)USAGE IS POINTER、

   (MF)または USAGE IS OBJECT

   の各句が含まれるデータ項目の下位に属するデータ項目も同様に無視される。しかし、d1 および d2 のデータ記述中に REDEFINES 句または OCCURS 句が含まれていてもよい。あるいは、データ記述中に REDEFINES 句または OCCURS 句が含まれるデータ項目の下位に d1 および d2 が属していてもよい。

   (ANS85)d1 も d2 も、部分参照できない。

5. 上記の条件を満たす各データ項目の名前は、暗黙の修飾を使用した後では、一意でなければならない。

10.5.4 算術文

算術文には、ADD、COMPUTE、DIVIDE、MULTIPLY、SUBTRACT がある。これらの文には、下記の共通の特徴がある。

1. 作用対象のデータ記述は、同じである必要はない。必要に応じて、データ形式の変換および小数点の位置合わせが行われる。
2. 作用対象の合成とは、指定された各作用対象の小数点の位置を合わせて重ね合わせできる、仮想のデータ項目である。

   ADD、DIVIDE、MULTIPLY、および SUBTRACT 文の作用対象の合成は、18 桁を超えてはならない。

   (MF)作用対象の合成についての制限はない。

10.5.5 作用対象の重なり

ある文の送出し側項目と受取り側項目とが記憶領域の一部を共有するときで、

(ANS85)同じデータ記述項目によってまだ定義されていない場合、

その文を実行した結果はどうなるかわからない。

(MF)重なりの転記は、MOVE 文が使用され、作用対象が部分参照も添え字も使用していない場合にだけ、翻訳時に検出される。コンパイラ指令 WARNING"3"を設定してある場合、前方への重なりの転記があると警告メッセージが出される。 コンパイラ指令 FLAG"方言"を設定してある場合、その他のタイプであればフラグメッセージが出される。ただし、OSVS 以外の"方言"とする。同じ記憶域を共有する項目を送受信する他の操作は、検出されない。

(MF)COBOL 原始コードの移植性は、原始プログラム内で MOVE 文の重なりが起きない時にだけ保証されるが、この COBOL システムはこのような文を認めていない。このような文の動作は、BYTE-MODE-MOVE 指令の指定により変わる。

10.5.6 算術文における複数個の結果

ADD、COMPUTE、DIVIDE、MULTIPLY、SUBTRACT の各文は複数個の答をもつことがある。これらの文は、下記のように書かれたものとして実行される。

1. 初期評価部分のすべてのデータ項目を演算処理し、その結果を一時的に記憶し、その一時結果を使用して必要なすべての演算を行い、受取り側項目に答を収める。

2. 中間結果を記憶する領域を利用して、一連の作用対象を順々に取り上げて演算を施し、最終的に 1 つの答を得る。この形の文は、列挙されている複数の作用対象を左から右に順々に取り上げて、別々の文で記述したものとみなされる。

   例1：

   ```
   ADD a, b, c TO c, d (c), e
   ```

   は、下記のように書いたのと等しい。

   ```
   ADD a, b, c GIVING temp
   ADD temp TO c
   ADD temp TO d (c)
   ADD temp TO e
   ```

   例2：

   ```
   MULTIPLY a(i) BY i, a(i)
   ```

   は、下記のように書いたのと等しい。

   ```
   MOVE a(i) to temp
   MULTIPLY temp by i
   MULTIPLY temp BY a(i)
   ```

   ここで、temp は COBOL コンパイラによって使用される、中間結果を一時的に記憶する場所である。

10.5.7 矛盾するデータ

字類条件 (前述の*字類条件*節を参照) の中で参照された場合は除いて、手続き部で参照されたデータ項目の内容が、そのデータ項目の PICTURE 句)、

(ANS85)または関数定義

に指定されている字類と合わないと、そのデータ項目を参照した結果はどうなるかわからない。

(MF)数字項目を参照したところ、その項目に文字または無効なデータが含まれていた場合、結果はどうなるかわからない。このような条件は、実行時に検出されてエラーとされる。この動作は、F ランタイムスイッチの影響を受ける。

(MF)英字項目を参照したところ、その項目に英字でないデータが含まれていた場合、プログラムの実行は続けられるが、結果はどうなるかわからない。

10.5.8 符号付き受取り側項目

算術文または MOVE 文中の受取り側項目が、符号付きの数字または数字編集項目である場合、受取り側項目に符号が転記される。この符号の転記は、数字データの絶対値が切り捨てられたか否かに関係なく行われる。したがって、数値はゼロであるが、符号が負ということが起こり得る。

10.6 ファイル入出力の概要

10.6.1 ファイル位置指示子

ファイル位置指示子 (file position indicator) は、ファイルに一連の入出力操作を施している際に、次に呼び出されるレコードを指す働きをする。ファイル位置指示子の設定に影響する文は、CLOSE, OPEN, START, READ だけである。出力モードまたは拡張モードで開かれたファイルに対しては、ファイル位置指示子は影響を及ぼさない。

10.6.2 入出力状態

ファイル管理記述項に FILE STATUS 句を指定すると、ファイル入出力操作の結果を示す 2 文字のデータ項目がとられる。OPEN, CLOSE, READ, WRITE, DELETE, UNLOCK, START の各文を実行すると、その 結果を示す値がこのデータ項目に設定される。実行する条件に該当する USE 手続きがある場合、それが実行される前にこの値が設定される。

以下に記述するファイル状態キーは、ANSI 標準に全面的に準拠したものであるとともに、この COBOL システムで使用できる拡張機能を、何も使用していないファイルに関するものである。

(MF)たとえば、拡張機能を使用してファイルを LINE SEQUENTIAL と指定すると、返される状態キーに影響が出る。詳しいことは関連する動詞の箇所に説明してある。

10.6.2.1 状態キー 1

FILE STATUS データ項目の左端の文字を、状態キー1 という。どのような編成のものでも、ファイルの入出力操作が終了すると、このデータ項目に下記の条件のどれかを示す値が設定される。

"0" - 正常終了 (successful completion)

"1" - ファイル終了 (at end)

"2" - 無効キー (invalid key)

"3" - 永続誤り (permanent error)

"4" - (ANS85)論理誤り (logic error)

"9" - ランタイム･システム･エラーメッセージ (run-time system error message)

上記の各条件の意味は下記のとおり。

"0" 正常終了：入出力文の実行が正常に完了した。

"1" ファイル終了：順呼出しの READ 文の実行が不成功に終わった。その理由は 2 通りある。ファイル中に次の論理レコードが存在しない。または、OPTIONAL 句が記述されているファイルに対して最初の READ 文が実行されたが、 OPEN 文を実行した時点でそのファイルがそのプログラムで利用可能でなかった。

"2" 無効キー：順ファイルでないファイルに対する入出力文の実行が不成功に終わった。その理由としては、下記のものがある。

順序に誤りがある
重複キーがある
レコードが見つからない
区域外書き出しが行われた

"3" 永続誤り：入出力文の実行が不成功に終わった。その理由は 2 通りある。 順ファイルに対して区域外書き出しが行われた。または、奇偶検査誤りや伝送誤りのような入出力誤りがあった。

"4" (ANSI85)論理誤り：入出力文の実行が不成功に終わった。その理由は 2 通りある。ファイルに対する入出力操作の順序が不適切である。 または、利用者によって設定されている制限を超えた処理が行われた。

"9" ランタイム･システム･エラーメッセージ：ランタイム･システムによって定義されているエラーメッセージ条件が発生したために、入出力文の実行が不成功に終わった。状態キー1 または状態キー1 と状態キー2 の組合せによって予め定義されている条件に該当しない条件が発生した場合にだけ、この状態値が使用される。

10.6.2.2 状態キー 2

FILE STATUS データ項目の右端の文字を、状態キー2 という。状態キー2 は入出力操作の結果の、さらに詳細を示すために使用される。

状態キー1 と状態キー2 の組合せによって、入出力操作の結果が下記のように細かく定義される。

正常終了

状態キー1 の値が"0" で、入出力操作が正常に終了したことが示されている場合、状態キー2 にはその原因を示す下記のどれかの値が設定される。

"0" (すべてのファイル) それ以上詳しい情報はない。

"2" (索引ファイルだけ) 下記の 2 通りの可能性を示す。

- READ 文で読み込んだレコードの参照キーの値が、その索引における次のレコードの参照キーと等しい。
- WRITE 文または REWRITE 文によって書かれたレコードの副レコードキーの値が、最低 1 つ既にファイルに存在している。ただし、その副レコードキーには重

複が許されている。

"4" (ANS85) (すべてのファイル) 処理したレコードの長さが、そのファイルのファイル固有属性に従っていない。

"5" (ANS85) (すべてのファイル) OPEN 文が実行されたとき、参照した不定ファイルが存在しない。

"7" (ANS85) (レコード順ファイルだけ) NO REWIND, REEL/UNIT, FOR REWIND のどれかの指定を伴う CLOSE 文、または NOREWIND 指定を伴う OPEN 文が実行されたが、対象となったファイルはリール/ユニット媒体ではない。

ファイル終了条件による不成功

状態キー1 の値が"1" で、ファイル終了条件 (at end condition) が発生したことが示されている場合、状態キー2 にはその原因を示す下記のどれかの値が設定される。

"0" (すべてのファイル) 次の論理レコードが存在しないことを示す。この条件が発生する場合は下記の 2 通りある。

- ファイルの末尾に到達した。
- 不定入力ファイル (optional input file) に対して、順呼出し READ 文が初めて実行されたが、対象とするファイルが存在しない。

"4" (ANS85) (相対ファイルだけ) 使用された相対レコード番号の桁数が、ファイル用に定義されている相対キーデータ項目の桁数よりも大きい。

無効キー条件による不成功

状態キー1 の値が "2" で、無効キー条件 (invalid key condition) が発生したことが示されている場合、状態キー2 にはその原因を示す下記のどれかの値が設定される。

"1" 索引ファイルを順呼び出しした場合) 順序誤り (sequence error) があったことを示す。その原因は、次の 2 通りある。連続するキーの値が昇順になっていなかった。(*手続き部 - SEARCH - XML PARSE* の章の *WRITE(書き出し)文*節を参照。)または、READ 文が正常に実行されてから REWRITE 文が実行されるまでの間に、そのファイルの主レコードキーが変更された。

"2" (相対ファイルおよび索引ファイルだけ) キーの値が重複することを示す。その原因は、次の 2 通りある。主レコードキーの値が重複になるレコードを書き込もうとした。または、副レコードキーに DUPLICATES 指定がなされていないのに、副レコードキーの値が重複になるレコードを書き込みまたは書き換えようとした。

"3" (相対ファイルおよび索引ファイルだけ) レコードが見つからなかったことを示す。その原因は、次の 2 通りある。レコードを呼び出すときに指定したキーに対応するレコードが、ファイル中に存在しなかった。または、不定入力ファイルに対して START 文または READ 文を実行したが、対象とするファイルが存在しなかった。

"4" (ANS85)相対ファイルおよび索引ファイルだけ) 区域外書き出し (boundary violation) 条件が発生したことを示す。その原因は、次の 2 通りある。

- 外部的に定義されているファイル境界を超えて、レコードを書き込もうとした。
- 相対ファイルに対して順書き込みの WRITE 文が実行されたが、使用された相対レコード番号の桁数が、ファイに定義されている相対キーデータ項目の桁数よりも大きい。

永続誤り条件による不成功

状態キー1 の値が"3" で、永続誤り条件 (permanent error condition) が発生したことが示されている場合、状態キー2 にはその原因を示す下記のどれかの値が設定される。

"0" (すべてのファイル) 誤りの原因について、それ以上詳しい情報はないことを示す。

"4" (順ファイルだけ) 区域外書き出し条件が発生したことを示す。つまり、外部的に定義されているファイル境界を超えて、レコードを書き込もうとした。

"5" (ANS85) (すべてのファイル) INPUT、I-O、または EXTEND のどれかの指定を伴う OPEN 文が実行されたが、対象となる不定でないファイルが存在しないことを示す。

"7" (ANS85)EXTEND のどれかの指定を伴う OPEN 文が実行されたが、対象となる不定でないファイルが存在しないことを示す。 その理由には下記のものがある。

- EXTEND 指定または OUTPUT 指定がなされたが、対象となるファイルには出力操作を行うことはできない。
- I-O 指定がなされたが、対象となるファイルには入出力両用モードで開かれた相対ファイル適用できる入出力操作を行うことはできない。
- INPUT 指定がなされたが、対象となるファイルには入力操作を行うことはできない。

"8" (ANS85) (すべてのファイル) 閉じてロック (施錠) されているファイルに対して、OPEN 文を実行しようとしたことを示す。

"9" (ANS85) (レコード順、相対、索引ファイル) ファイル固有属性 (fixed file attribute) とプログラム中でファイルに対して指定された属性との間に、矛盾が検出されたことを示す。

論理誤り条件による不成功

(ANS85)

状態キー1 の値が "4" で、論理誤り条件 (logic error condition) が発生したことが示されている場合、状態キー2 にはその原因を示す下記のどれかの値が設定される。

"1" (すべてのファイル) 既に開かれているファイルに対して、OPEN 文を実行しようとしたことを示す。

"2" （すべてのファイル）開かれていないファイルに対して、CLOSE 文を実行しようとしたことを示す。

"3" （すべてのファイルに関して、順呼出しの場合だけ）DELETE 文または REWRITE 文を実行しようとしたが、その直前の入出力操作として READ 文が実行され正常終了していない。

"4" （レコード順ファイルだけ）区域外書き出しが発生したことを示す。その原因として考えられるのは、対象のファイルの RECORD IS VARYING 句によって認められている最大レコードよりも長いか最小レコードよりも短いレコードを、WRITE または REWRITE しようとしたことである。

"5" （すべてのファイル）ファイルのレコードを REWRITE しようとしたが、元のレコードの大きさと書き換えレコードの大きさが等しくない。

(MF) 行順ファイルの場合、このレコードの大きさは、空白を除去しタブを圧縮し空文字を挿入した後のレコードの物理的な大きさを指す。この場合、新しいレコードの物理的な大きさが元のレコードよりも小さくなってもよい。

"6" （すべてのファイル）入力モードまたは入出力両用モードで開かれているファイルに対して、順呼出しの READ 文を実行しようとしたが、有効な次のレコードが存在しないことを示す。その原因としては、下記のものがある。

- 先行する START 文が不成功であった。
- 先行する READ 文がファイル終了以外の条件で不成功であった。
- 先行する READ 文によってファイルの末尾に達した。

"7" （すべてのファイル）入力モードまたは入出力両用モードで開かれていないファイルに対して READ 文または START 文を実行しようとしたことを示す。

"8" （すべてのファイル）入出力両用モード、出力モード、拡張モードのどれかで開かれていないファイルに対して WRITE 文を実行しようとしたか、または順呼出し法の入出力両用モードで開かれたファイルに対して WRITE 文を実行しようとしたことを示す。

"9" （すべてのファイル）入出力両用モードで開かれていないファイルに対して、DELELTE 文または REWRITE 文を実行しようとしたことを示す。

ランタイム・システムによるエラーメッセージ

(MF)

状態キー1 の値が" 9" で、作成者が定義したエラーメッセージが発生したことが示されている場合、状態キー2 には該当するエラーメッセージ番号が 2 進数で設定される。エラーメッセージについて詳しくは、使用している COBOL システムのマニュアルを参照。

10.6.2.3 状態キー1 と 2 の有効な組合わせ

以下の表で、"S" はレコード順ファイル、

(MF)"L " は行順ファイル、

"R" は相対ファイル、"I" は索引ファイルを示す。

表 10-4 中の状態キー1 の行と状態キー2 の列が交差する部分に文字が記されている場合に、そのファイル編成に関してその状態キー1 と状態キー2 の組合せが有効である。

表 10-4:状態キー1 と 2 の有効な組合わせ

状態キー 1		状態キー 2									
		0	1	2	3	4	5	6	7	8	9
正常終了	0	SRIL		I		SRIL	SRIL		S		
ファイル終了	1	SRIL				R					
無効キー	2		I	RI	RI	RI					
永続誤り	3	SRIL				SL	SRIL		SRIL	SRIL	SRI
論理誤り	4		SRIL	SRIL	SRIL	SRIL		SRIL	SRIL	SRIL	SRIL
作成者定義	9	ランタイムシステムのエラーメッセージ (SRIL)									

10.6.3 ファイル終了条件

READ 文を実行した結果、ファイル終了条件が発生することがある。その原因の詳細については、***手続き部 - PERFORM - ROLLBACK*** の章の *READ(読み込み)文*節を参照。

10.6.4 無効キー条件

START 文、READ 文、WRITE 文、REWRITE 文、DELETE 文を実行した結果、無効キー条件が発生することがある。その原因の詳細については、***手続き部 - PERFORM - ROLLBACK*** の章の *READ(読み込み)文* および *REWRITE(書き換え)文*、***手続き部 - SEARCH - XML PARSE*** の章の *START(開始)文* および *WRITE(書き出し)文*の各節を参照。

入出力文に指定された入出力操作が実行された後で無効キー条件が発生すると、下記の動作が記してある順に行われる。

1. 該当ファイルに FILE STATUS データ項目を指定してあると、そこに無効キー条件の内容を示す値が設定される。(前述の*入出力状態*の節を参照。)
2. 無効キー条件を引き起こした文の中に INVALID KEY を指定してあると、制御は INVALID KEY 指定中の無条件文に移される。該当ファイルに USE 手続きが指定されていても、実行されない。
3. 無効キー条件を引き起こした文の中に INVALID KEY を指定しないで、該当ファイルに USE 手続きが明示的または暗黙的に指定されていると、その手続きが実行される。

無効キー条件が発生すると、そのことを認識する入出力文は不成功となる。この場合、ファイルは影響を受けない。

状態キー1 に"9" が設定されたときは、INVALID KEY 指定によって誤りが捕らえられるのではないことに注意。この場合は、明示的に状態キーを検査するか、または宣言部分を使用して、別途誤りを捕られなければならない。

10.6.5 マルチユーザーシステム上でのファイルの共有

ランタイムシステム (RTS) では、この COBOL システムのマルチユーザー機能をサポートしている。このマルチユーザー機能によって、マルチユーザー環境下で利用者がファイルを共有すること、あるプログラムがデータを更新する間は他のプログラムが該当するファイル全体またはそのレコードをアクセスできないようにすることが可能になる。

シングルユーザー環境下では、マルチユーザー用の構文は実行時に効力を発揮しない。しかし、シングルユーザーとマルチユーザーの両方の環境で使用できるように、プログラムを作成しておくことができる。

ファイルはアクティブか非アクティブのどちらかの状態をとる。アクティブ・ファイルとはいくつかの実行単位に対して開かれているものである。非アクティブ・ファイルとはどの実行単位に対しても開かれていないものである。

アクティブ・ファイルには、排他モード (exclusive mode) と共有モード (sharable mode) の 2 つのモードがある。

- MF OPEN 排他モード

  排他モードにあるファイルは、1 つの実行単位にだけ開かれている。他の実行単位はそのファイルにアクセスしようとすると、「ファイルロック」誤りを引き起こし、アクセスを拒否される。排他モードということは、1 つの実行単位だけがファイルの鍵 (ファイルロック) を保持していて、そのファイルにアクセスできるということである。実行単位においてそのファイルが閉じられると、そのファイルのロックが解除 (release) される。

- 共有モード

  共有モードにあるファイルは、任意の数の実行単位に開かれている。各実行単位は、ファイルを使用する間その中のレコードを一時に何件かロック (施錠) することによって、データを保護できる。他の実行単位は、ロック (施錠) されている個々のレコードの鍵 (レコードロック) を取得できなくなる。しかし、その点を除けば、他の実行単位がファイルにアクセスできなくなるわけではない。ファイルの編成によって、ファイルを共有できる形態に違いが生じる。

ファイルの共有方法は、以下に示すファイル構成に影響される。

- レコード順ファイル

  入力用に開かれたレコード順ファイルは、いくつかの実行単位で共有できる。しかし、そのファイル中のレコードをロックすることはできない。入出力両用または拡張用に開かれたファイルもまた、いくつかの実行単位で共有することができる。この場合は、各

実行単位はそのファイルに関してレコードロックをいくつか保持できる。出力用に開かれたファイルは、常に排他的にロックされている。

- (MF)行順ファイル

  入力用または拡張用に開かれた行順ファイルは、いくつかの実行単位で共有できる。しかし、そのファイル中のレコードをロックすることはできない。出力用に開かれたファイルは、常に排他的にロックされている。

- 相対ファイルおよび索引ファイル

  入力モードで開かれた行順ファイルは、いくつかの実行単位で共有できる。しかし、そのファイル中のレコードをロックすることはできない。入出力両用モードで開かれたファイルも、共有できる。しかし、出力用または拡張用に開かれたファイルは、排他的である。

ファイルへのアクセスを共有する各実行単位は、そのファイル中の単一のレコードまたは複数のレコードをロックすることができる。ただし、1 つの実行単位が同じファイルに対して、単一レコードのロックと複数レコードのロックの両方を選択することはできない。

10.6.5.1 単一レコードのロック

あるファイルに対して (明示的にまたは暗黙的に) 単一レコードのロックを指定した実行単位は、一時点ではそのファイル中の 1 レコードだけをロックできる。実行単位がレコードロックを獲得する方法には、手動と自動の 2 通りがある。

- 手動のレコードロック

  READ WITH LOCK 文によってレコードを呼び出した場合にだけ、実行単位はレコードロックを取得できる。

- 自動のレコードロック

  ファイル中のレコードを読むときに、実行単位はレコードロックを自動的に取得する。ただし、READ WITH NO LOCK 文を使用した場合は、例外である。

ロック解除はロックした実行単位によって、下記のどれかの方法によって行われる。

- 該当するファイル中の任意のレコードを、任意のファイル操作によって呼び出す (ただし、相対ファイルおよび索引ファイルに対する START を除く)
- 該当するファイルに対して、UNLOCK 文を実行する
- COMMIT 文を実行する
- ROLLBACK 文を実行する
- 該当するファイルを閉じる

10.6.5.2 複数レコードのロック

ファイル編成が相対か索引かレコード順の場合にだけ、複数レコードをロックできる。

複数レコードをロックすることを指定した実行単位は、1 つのファイル中で同時に何件ものレコードロックを保持できる。他の実行単位は、ロックされているレコードを更新したりロックしたりすることができなくなる。しかし、他の実行単位がロックされていないレコードへのアクセスを拒否されるわけではない。実行単位がレコードロックを獲得する方法には、手動と自動の 2 通りがある。

- 手動のレコードロック

  READ WITH LOCK 文 または READ WITH KEPT LOCK 文によってレコードを呼び出した場合にだけ、実行単位はレコードロックを取得できる。

- 自動のレコードロック

  ファイル中のレコードを読むときに、実行単位はレコードロックを自動的に取得する。ただし、ロックしないことを明示的に指定した場合は、例外である。プログラムを COBOL システムに投入したときに WRITELOCK 指令を設定しておくと、WRITE 文または REWRITE 文によって実行単位がファイルにアクセスすると、ロックされる。

ロック解除はロックした実行単位によって、下記のどれかの方法によって行われる。

- ロックされているレコードに対して、DELETE 文を実行する。この場合は、削除対象のレコードだけがロックを解除される。
- 該当するファイルに対して、UNLOCK 文を実行する
- COMMIT 文を実行する
- ROLLBACK 文を実行する
- 該当するファイルを閉じる

10.6.5.3 省略時解釈のレコードロック

表 10-5、10-6、および 10-7 に、ファイル編成別にファイルをオープンするモードごとの省略時解釈のロック型を示す。 この省略時解釈のロック型は、 プログラムを COBOL に投入するときに AUTOLOCK 指令を設定することによって、変更できる 。また、表には、個々のファイルごとに、省略時解釈のロック型を変更することができるか否かについても示す。ファイルごとにロック型を変更するには、そのファイルの SELECT 句中に適切な句を挿入する 。(構文の詳細については、*環境部*の章の *ファイル管理記述項*節を参照。)

X/Open では、自動ロックでの単一レコードロック、またはマニュアルロックでの複数レコードロックのどちらかを使用する原始プログラムに従う X/Open を制限している。

表 10-5: レコード順ファイルの省略時解釈のロック型

開くモード	指令なし	AUTOLOCK コンパイラ 指令	SELECT 文での変更

INPUT	ロックなし	ロックなし	可、ただし排他へだけ
I-O	排他	単一レコードのロック	可
OUTPUT	排他	排他	不可
EXTEND	排他	ロックなし	可、ファイルを共有可できるがレコードはロックできない

表 10-6: 行順ファイルの省略時解釈のロック型

開くモード	指令なし	AUTOLOCK コンパイラ指令	SELECT 文での変更
INPUT	ロックなし	ロックなし	不可
I-O	排他	ロックなし	不可
OUTPUT	排他	排他	不可
EXTEND	排他	ロックなし	不可

表 10-7: 相対ファイルおよび索引ファイルの省略時解釈のロック型

開くモード	指令なし	AUTOLOCK コンパイラ指令	SELECT 文での変更
INPUT	ロックなし	ロックなし	可、ただし排他へだけ
I-O	排他	単一レコードの自動ロック	可
OUTPUT	排他	排他	不可
EXTEND	排他	排他	不可

注:

1. 出力用に開いたファイルは、指定したロックのモードにかかわりなく、排他モードとされる。
2. プログラマは、省略時解釈のロック方式(表 4-15, 4-16, 4-17 を参照)を受け入れるか、またはファイル管理記述項に LOCK MODE 句 (*環境部*の *ファイル管理記述項*節を参照) を含めるかして、ロックの型を選択できる。

10.7 パラメータと戻り項目の適合

パラメータと戻り項目の適合規則は、構文規則から明示的な参照が行われる翻訳時に適用される。

注：翻訳時にパラメータと戻り項目の適合規則におり、以下がチェックされる。

- 一般的オブジェクト参照以外のオブジェクト参照に対する INVOKE 文
- 利用者定義の関数への参照
- プログラムプロトタイプ形式の CALL 文

10.7.1 パラメータ

起動する側の要素の中の引数の数は、起動される側の要素の中の仮パラメータの数と同じでなければならない。

(ISO2002)(MF)ただし、後に置かれる仮パラメータで、起動される側の要素の手続き部見出しで OPTIONAL により指定され、起動する側の要素の引数のリストでは省かれているものを除く。

(MF)作業対象の数が同じではなく、起動する側の文の方がその数が多い場合は、それらは無視される。また、手続き部見出しの方が作用対象の数が多い場合は、ランタイム要素の中でそれらが参照されなければならない。位置的な対応関係は、呼び名で暗黙的に指定される COBOL 以外の呼出し規則により異なる。作用対象の数の相違は、実行時以外にはわからないため、指令 FLAG によりこの拡張にフラグがつけられることはない。

引数と仮パラメータの適合規則は、仮パラメータまたはそれに対応する引数が集団項目であるか、またはその両方が基本項目であるかどうかにより異なる。

10.7.1.1 集団項目

仮パラメータまたは引数が集団項目である場合、対応する引数または仮パラメータは集団項目、または項類が英数字の基本項目でなければならない。さらに、仮パラメータは、対応する引数と同じか、それより少ない文字位置で記述されなければならない。

10.7.1.2 基本項目

基本項目の適合規則は、引数が参照、内容、および値のどれにより渡されるかにより異なる。

10.7.1.2.1 参照により渡される基本項目

仮パラメータまたはそれに対応する引数がオブジェクト参照の場合、対応する引数または仮パラメータは、以下の規則に従うオブジェクト参照でなければならない。

1. 引数または仮パラメータが一般的オブジェクト参照の場合、対応する仮パラメータまたは引数は一般的オブジェクト参照でなければならない。
2. 引数または仮パラメータをインターフェイス名で記述した場合、対応する仮パラメータまたは引数は、インターフェイス名で記述しなければならない。

3. 引数または仮パラメータをクラス名で記述した場合、対応する仮パラメータまたは引数は、同じクラス名で記述しなければならない。さらに、FACTORY および ONLY を同様に指定しなければならない。
4. 仮パラメータを ACTIVE-CLASS 指定で記述した場合、以下の条件のいずれかが真でなければならない。
    a. 引数が、ACTIVE-CLASS 指定で記述されたオブジェクト参照である。ここで、FACTORY 指定の有無は仮パラメータと同じである。さらに、起動されるメソッドは、定義済みのオブジェクト参照 SELF または SUPER、または ACTIVE-CLASS 指定で記述されたオブジェクト参照で呼び出される。
    b. 引数が、クラス名および ONLY 指定で記述されたオブジェクト参照である。ここで、FACTORY 指定の有無は仮パラメータと同じである。さらに、起動されるメソッドは、そのクラス名、またはそのクラス名と ONLY 指定で記述されたオブジェクト参照で呼び出される。

引数または仮パラメータがクラスポインタのものである場合、対応する仮パラメータまたは引数はクラスポインタのものでなければならない。さらに、対応する項目は同じ項類でなければならない。どちらかのポインタが制限付きポインタであれば、両方とも制限付きで同じ種類のポインタでなければならない。特殊レジスタ ADDRESS は、クラスポインタのもので、かつ、項類データポインタと見なされる。

引数または仮パラメータのどちらもクラスオブジェクトのものでもクラスポインタのものでない場合の適合規則は以下の通り。

1. 起動される側の要素であるプログラムに対し、起動する側の要素のリポジトリ段落中のプログラム指定子がなく、CALL 文に指定された NEST 指定がない場合、仮パラメータは、対応する引数と同じ長さでなければならない。
2. 起動される側の要素が以下のいずれかの場合、仮パラメータの定義および引数の定義に、PICTURE、USAGE、SIGN、JUSTIFIED、および BLANK の各句が同様に含まれなければならない。
    - 起動する側の要素中のリポジトリ段落に、そのプログラム指定子のないプログラム
    - NEST 指定を含む CALL 文で呼び出されたプログラム
    - メソッド
    - 関数

    ただし、以下の場合は例外とする。

    e. 通貨記号が一致する。ただし、対応する通貨文字列が同じ場合に限る。
    f. 終止符 PICTURE 記号が一致する。ただし、起動する側と起動される側の両方のランタイム要素について、DECIMAL-POINT IS COMMA 句が有効または無効である場合に限る。
    g. コンマ PICTURE 記号が一致する。ただし、起動する側と起動される側の両方のランタイム要素について、DECIMAL-POINT IS COMMA 句が有効または無効である場合に限る。

    さらに、以下の規則が適用される。

h. 仮パラメータを ANY LENGTH 句で記述した場合、その長さは対応する引数の長さと一致すると見なされる。
i. 引数を ANY LENGTH 句で記述した場合、対応する仮パラメータを ANY LENGTH 句で記述するものとする。

10.7.1.2.2 内容または値により渡される基本項目

仮パラメータが、ACTIVE-CLASS 指定で記述されたオブジェクト参照の場合、以下の条件のいずれかが真でなければならない。

1. 起動されるメソッドは、定義済みのオブジェクト参照 SELF または SUPER, 、または ACTIVE-CLASS 指定で記述されたオブジェクト参照により呼び出される。さらに、起動する側の要素で SET 文が有効である。この要素には、送出し側の作用対象として引数が、受取り側の作用対象として ACTIVE-CLASS 指定で記述されたオブジェクト参照が含まれる。このオブジェクト参照での FACTORY 指定の有無は、仮パラメータでの有無と同じとする。
2. 起動されるメソッドは、クラス名またはクラス名で記述されたオブジェクト参照により呼び出される。さらに、起動する側の要素で ONLY 指定および SET 文が有効である。この要素には、送出し側の作用対象として引数が、受取り側の作用対象としてクラス名と ONLY 指定で記述されたオブジェクト参照が含まれる。このオブジェクト参照での FACTORY 指定の有無は、仮パラメータでの有無と同じとする。

仮パラメータがクラスポインタのもの、または ACTIVE-CLASS 指定を用いずに記述されたオブジェクト参照である場合、その適合規則は、SET 文が、実行する側のランタイム要素で実行されたときと同じとなる。このとき、送出し側作用対象は引数とし、受取り側作用対象は対応する仮パラメータとする。

仮パラメータがクラスオブジェクトのものでもクラスポインタのものでない場合の適合規則は以下の通り。

1. 起動される側の要素であるプログラムに対し、起動する側の要素のリポジトリ段落中のプログラム指定子がなく、CALL 文に指定された NEST 指定がない場合、仮パラメータは、対応する引数と同じ数の文字位置で記述しなければならない。
2. 起動される側の要素が以下のいずれかの場合の適合規則は、仮パラメータの種類により異なる。
   - 起動する側の要素中のリポジトリ段落に、そのプログラム指定子のないプログラム
   - NEST 指定を含む CALL 文で呼び出されたプログラム
   - メソッド
   - 関数

   仮パラメータの種類と各適合規則は以下の通り。

   e. 仮パラメータが数字である場合の適合規則は、送出し側作用対象としての引数と、受取り側作用対象としての対応する仮パラメータを含む COMPUTE 文に対するものと同じとする。

f. 仮パラメータが索引データ項目またはポインタデータ項目の場合、適合規則は、送出し側作用対象としての引数と、受取り側作用対象としての対応する仮パラメータを含む SET 文に対するものと同じとする。
g. 仮パラメータが ANY LENGTH 句で記述されている場合、その長さは、対応する引数の長さに一致すると見なされる。
h. 上記のどれでもない場合、適合規則は、送出し側作用対象としての引数と、受取り側作用対象としての対応する仮パラメータを含む MOVE 文に対するものと同じとする。

10.7.2 戻り項目

戻り項目は、起動する側の文で指定しなければならない。ただし、戻り項目が、起動される側の要素の手続き部見出し、

(MF)または ENTRY 文

で指定される場合に限る。関数または行内メソッド呼出しが参照されるときは、戻り値が、起動する側の要素で暗黙的に指定される。

起動される原始要素中の戻り値は送出し側の作用対象であり、起動する側の原始要素中の対応する戻り値が受取り側作用対象である

送出し側の作用対象と受取り側作用対象の間の適合規則は、それらの作用対象の少なくとも 1 つが英数字集団項目であるか、または両方が基本項目であるかどうかにより異なる。

10.7.2.1 集団項目

送出し側と受取り側作用対象のどちらかが集団項目の場合、対応する戻り項目は集団項目、または項類が英数字の基本項目でなければならない。また、受取り側作用対象は、送出し側作用対象と同じ数の文字位置で記述しなければならない。

注：起動する側の要素中の戻り項目が 1 以外のレベル番号の集団項目で、その下位の項目の記述により、使用されると意味のないビットまたはバイトが挿入される場合は、下位の基本項目の配置が、起動する側のランタイム要素中の戻り項目と起動されるランタイム要素中の戻り項目の間で対応しない場合がある。

変数反復データ項目として記述された作用対象には、最大長が使用される。

10.7.2.2 基本項目

変数反復データ項目として記述された作用対象については、最大長が使用される。

1. 起動される原始要素中の戻り項目の記述に ACTIVE-CLASS 指定が含まれない場合、適合規則は、SET 文が、起動されるランタイム要素で実行されたときと同じとなる。このとき、送出し側作用対象は起動される原始要素中の戻り項目とし、受取り側作用対象は起動する側の原始要素中の対応する戻り項目とする。
2. 起動される原始要素中の戻り項目の記述に ACTIVE-CLASS 指定が含まれる場合、適合規則は、SET 文が、起動されるランタイム要素で実行されたときと同じとなる。このとき、受取り側作用対象は起動する側の原始要素中の戻り項目とし、送出し側作用対象は以下の規則に従って USAGE OBJECT REFERENCE で記述されるものとする。
    a. 起動されるメソッドがクラス名で呼ばれる場合、送出し側作用対象は同じクラス名と ONLY 指定で記述される。
    b. 起動されるメソッドが定義済みオブジェクト参照 SELF または SUPER、

       (MF)または SELFCLASS

       で呼ばれる場合、送出し側作用対象は ACTIVE-CLASS 指定で記述される。

    c. 起動されるメソッドが、インターフェイス名で記述されたオブジェクト参照で呼ばれる場合、は一般的オブジェクト参照とする。
    d. 起動されるメソッドがその他のオブジェクト参照で呼ばれる場合、この一意名が送出し側作用対象として使用される。この場合、用いられていれば、ONLY 指定も含まれる。
    e. 上記の規則の適用により選択された送出し側作用対象がクラス名または ACTIVE-CLASS 指定で記述されている場合、FACTORY 指定の有無は、起動される原始要素の戻り項目での有無と同じとする。

送出し側作用対象がオブジェクト参照でない場合、受取り側作用対象は同じ PICTURE、USAGE、SIGN、SYNCHRONIZED、JUSTIFIED、および BLANK 句を含むものとする。ただし、以下の例外がある。

a. 通貨記号が一致する。ただし、対応する通貨文字列が同じ場合に限る。
b. 終止符 PICTURE 記号が一致する。ただし、起動する側と起動される側の両方のランタイム要素について、DECIMAL-POINT IS COMMA 句が有効または無効である場合に限る。.
c. コンマ PICTURE 記号が一致する。ただし、起動する側と起動される側の両方のランタイム要素について、DECIMAL-POINT IS COMMA 句が有効または無効である場合に限る。

さらに、以下の規則が適用される。

a. 受取り側作用対象を ANY LENGTH 句で記述した場合、送出し側作用対象も ANY LENGTH 句で記述しなければならない。
b. 送出し側作用対象を ANY LENGTH 句で記述した場合、送出し側作用対象の長さは受取り側作用対象の長さと一致すると見なされる。

第 11 章 : 手続き部 - 組み込み関数

ANSI85

11.1 一般説明

組み込み関数機能単位において、関数とは一時的なデータ項目である。その値は、文を実行している間に関数が参照された時点で決定される。 関数名は COBOL の語であり、原始プログラム中で使用できる COBOL の語のリスト中に含まれている。(*関数の定義* 節を参照。)

11.2 関数定義と戻り値

関数の型に応じて、下記のことが示される。

- 英数字関数

  MF および各国語型

  の場合は、戻り値の大きさ。

- 数字関数および整数関数の場合は、戻り値の符号と整数であるかどうか。
- その他、関数によっては、戻り値。

11.3 関数一意名

関数一意名は、COBOL 原始プログラムの手続き部の中で使用する、関数の名前である。
(***COBOL 言語の概念***の章の*関数一意名*節を参照。)

11.4 関数からの戻り値

関数からの戻り値 (関数値) は、データ値とみなされる。関数を使用すると、実行時にこのデータ値が決定される。そのためには、プログラムから関数に、基となるデータ値をパラメータとして引き渡す必要がある。このパラメータを引数という。関数によっては、引数の取る値の範囲などに制約があるものがある。関数の実行時に引数の値がその関数の制約から外れる場合、その関数からの戻り値は保証されない。

11.5 引数

引数は、関数値を求めるために使用する値である。引数は、関数一意名中に指定する。指定する引数の型には、一意名と算術式と定数がある。各関数の定義の説明で、必要な引数の数を具体的に示してある。関数によって、引数は必要ないもの、1 つだけ必要とするもの、複数個必要とするものがある。また、中には、指定できる引数の数が変えられる関数もある。

関数一意名中に引数を指定する順序に基づいて、それぞれの引数が解釈され、関数値が決定される。指定する引数が、特定の字類またはその部分集合に属することが要求される場合がある。引数の型を下に示す。

1. 数　字：　算術式を指定する。演算符号を含む算術式の値が、関数値を求めるために使用される。
2. 英　字：　字類が英字である基本データ項目、または英字だけを含む文字定数を指定する。引数の大きさが、関数値の決定に使用されることもある。
3. 英数字：　字類が英字か英数字である基本データ項目、または文字定数を指定する。引数の大きさが、関数値の決定に使用されることもある。
4. (MF)各国語型。字類が各国語型であるデータ項目、または各国語型文字を指定する。引数の大きさが、関数値の決定に使用されることもある。
5. 整　数：　結果が必ず整数になる算術式を指定する。演算符号を含む算術式の値が、関数値を求めるために使用される。

意味のある関数値を求めるために、引数に指定できる値に制約が設けられることがある。関数の実行時に引数の値がその関数の制約から外れる場合、その関数からの戻り値は保証されない。

関数定義において引数を任意の回数繰り返し指定することが認められている場合、表を使用することができる。この場合、使用する表を識別するデータ名と、必要に応じて修飾語を指定し、その直後に添字を続ける。添字の 1 つまたはいくつかに語 ALL を指定できる。

添字に ALL を指定すると、その添字の位置に対応する表要素をすべて指定したのと同様の効果が得られる。この暗黙の指定は、表の要素を左から右に順に指定することを意味する。つまり、指定の対象となる最初（左端）の表要素は、添字に指定した語 ALL のそれぞれを、1 で置き換えた一意名によって表わされるものとなる。指定の対象となる次の表要素は、右端の ALL に対応する添字の値を 1 繰り上げた一意名によって表わされるものとなる。

それ以降、暗黙の指定の対象となる表要素は、右端の ALL に対応する添字の値を順々に 1 繰り上げていった一意名によって表わされるものとなる。そして、その ALL に対応する添字の値がその最大値に達したところで、その添字の繰り上げ処理は終わりとなる。ALL を指定した添字が複数ある場合は、次に、ALL を指定した右端の添字のすぐ左の ALL を指定した添字が 1 繰り上げられ、ALL を指定した右端の添字の値は 1 に設定し直される。右端の添字の値を順々に 1 繰り上げていくことが、再度繰り返される。そして、右端の添字の値が最大値に達するたびに、その左側の添字の値が 1 繰り上げられる。左側の添字の値が最大値に達したところで、その添字の繰り上げ処理は終わりとなる。その左側にも ALL を指定した添字がまだあれば、同様のプロセスが繰り返され、ALL を指定した左端の添字の値が最大値に達するまで、続けられる。

この処理の対象となる添字に OCCURS DEPENDING ON 句が指定してある場合は、該当する添字の値の範囲は、OCCURS DEPENDING ON 句の右辺の項目に基づいて決められる。ALL を指定した添字が評価された結果、少なくとも引数が 1 つ存在しなければならない。引数がないと、戻り値は保証されない。

例

```
01 Test-Fields.
    10 OT-Elem                  PIC 9(02).
    10 Arr.
        15 Ind occurs 5 times        PIC 9(02).
    compute OT-Elem = function sum (IND(ALL)).
```

下記のコーディングと同じ意味になる。

```
    compute OT-Elem = function sum (IND(1), IND(2), IND(3),
                                          IND(4), IND(5)).
```

ALL を指定した添字は、以下の例のように、多くの要素を持つ表でも使用できる。

```
01 Test-Group.
    03 OT-Elem              pic 9(15) binary.
    03 table-length         pic s9(9) binary.
    03 array.
        05 Ind pic 9(2) occurs 1 to 200 times depending on
                            table length
     ...
    move 100 to table-length.
    compute OT-Elem = function sum (Ind(ALL))
```

これらの文は、表の最初の 100 個の要素を合計する。table-length に指定された数が、関数の参照時に、加える要素の数として決定される。

11.6 関数の型

データ項目関数は基本データ項目であり、英数字、

(MF)各国語型文字、

数字、または整数の値を返す。 データ項目関数は基本データ項目として扱われるが、受取り側の作用対象とすることはできない。データ項目関数には、下記の型がある。

1. 英数字関数: この型の関数は、字類および項類が英数字に属する。このデータ項目中の文字数は、関数定義中に示される。英数字関数の用途は、暗黙的に DISPLAY とされる。
2. 数字関数: この型の関数は、字類および項類が数字に属する。数字関数は、つねに演算符号を持つものとみなされる。
    - 数字関数は算術式の中でのみ、

      (MF)または MOVE 文の送出し側としてのみ、

使用できる。

- 整数の作用対象が必要な箇所に、数字関数を使用することはできない。得られた値が結果的に整数となることはあるが、だからといってこの規則を無視してはならない。

3. 整数関数：この型の関数は、字類および項類が整数に属する。整数関数は、つねに演算符号を持つものとみなされる。
   - 整数関数は算術式の中でのみ、

     (MF)または MOVE 文の送出し側としてのみ、

     使用できる。

   - 整数の作用対象が必要とされる箇所で、それに符号が付いてもよい場合、整数関数を使用することができる。
4. (MF)各国語型関数：この型の関数は、字類および項類が各国語型に属する。このデータ項目中の文字数は、関数定義中に示される。各国語型関数の用途は、暗黙的に NATIONAL とされる。

11.7 日付変換関数

日付変換関数では、グレゴリオ歴を使用している。1601 年 1 月 1 日を開始日とする。この日は月曜日であるので、開始日からの通算日と曜日の関係がとらえやすくなっている。

たとえば、下記の文を実行すると、指定した日の曜日を表わす整数が返される。

```
compute DoW =
     function rem (function integer-of-date (date-field) , 7)
```

値が 0 の場合は日曜日、1 の場合は月曜日のようになる。

11.8 三角関数

三角法は三角形の辺と角の間の関係を取り扱う。角を測る単位には度、ラジアン、グラードがある。コンピュータでは三角関数の値は通常、連続的近似によって算出される。したがって、SIN, COS, TAN の各関数の引数または ASIN, ACOS, ATAN の各関数から返される値として使用するときは、角度をラジアンで表す。

1ラジアンは長さが半径に等しい円弧である。半径 (r) と円周 (c) の関係は式 C = 2 * π * *r* で表される。円の一周は 360 度である。よって、360 = 2 * π * *r* となる。このことから、下記の変換率が得られる。

1ラジアン = 180/π = 57.295780 度
1度 = π/180 = 0.01745329 ラジアン

その他に、角度を表すのに、グラード (直角の百分の1) という単位が用いられることがある。
その変換率は下記のとおりである。

1ラジアン = 200/π = 63.661977 グラード
1グラード = π/200 = 0.01570796 ラジアン

11.9 関数の定義

使用できる関数を表 11-1 に示す。

"引数" の欄には、引数の型と"type" 数を示す。具体的には、下記の型がある。

Alph	英字
Anum	英数字
Int	整数
(MF)Nat	各国語型
Num	数字

表 11-1: 関数の一覧

関数名	引数	型	戻り値
(MF)ABS	Int1 または Num1	引数による	引数の絶対値
ACOS	Num1	Num	Num1 のアークコサイン (逆余弦)
ANNUITY	Num1, Int2	Num	一括掛金 1 に対して、利率 Num1 で期間 Int2 にわたって支払われる即時年金の割合
ASIN	Num1	Num	Num1 のアークサイン (逆正弦)
ATAN	Num1	Num	Num1 のアークタンジェント (逆正接)
CHAR	Int1	Anum	プログラムの文字の照合順序中の、位置 Int1 にある文字
(ISO/IEC)(MF)CHAR-NATIONAL	Int1	Nat	各国語型プログラムの照合順序中の、位置 Int1 にある文字
COS	Num1	Num	Num1 のコサイン (余弦)
CURRENT-DATE	なし	Anum	現在の日付と時刻と、グリニッジ標準時刻との時差
DATE-OF-INTEGER	Int1	Int	整数日付に相当する標準日付 (YYYYMMDD)
DATE-TO-YYYYMMDD	Int1, Int2	Int	引数-2 の値に基づいて Int2 を YYMMDD から YYYYMMDD に変換された引数-1

DAY-OF-INTEGER	Int1	Int	整数日付に相当するジュリアン日付 (YYYYDDD) (ユリウス暦)
DAY-TO-YYYYDDD	Int1, Int2	Int	引数-2 の値に基づいて Int2 を YYDDD から YYYYDDD に変換された引数-1
(ISO2002)(MF) DISPLAY-OF	Nat1, Anum1	Anum	引数 Nat1 の USAGE DISPLAY 型の表現
(MF)E	なし	Num	*e* の値、自然基数
(MF)EXP	Num1	Num	*e* の Num1 乗
(MF)EXP10	Num1	Num	10 の Num1 乗
FACTORIAL	Int1	Int	Int1 の階乗
(MF)FRACTION-PART	Num1	Num	Num1 の端数部
INTEGER	Num1	Int	Num1 を超えない最大の整数
INTEGER-OF-DATE	Int1	Int	標準日付 (YYYYMMDD) に相当する整数日付
INTEGER-OF-DAY	Int1	Int	ジュリアン日付 (YYYYDDD) (ユリウス暦) に相当する整数日付
INTEGER-PART	Num1	Int	Num1 の整数部分
LENGTH	Alph1 または Anum1 (MF)または Nat1 または Num1	Int	文字番号の引数の長さ
(MF)LENGTH-AN	Alph1 または Anum1 または Int1 または Nat1 または Num1	Int	英数字文字番号の引数の長さ
LOG	Num1	Num	Num1 の自然対数
LOG10	Num1	Num	10 を底とする Num1 の対数
LOWER-CASE	Alph1 または Anum1 (MF)または Nat1	引数による*	引数中のすべての文字を小文字にする
MAX	Alph1 ... または Anum1 ... または Int1 ...または (MF)Nat1 ... または Num1 ...	引数による*	引数の最大値
MEAN	Num1 ...	Num	引数の算術平均
MEDIAN	Num1 ...	Num	引数のメジアン (中央値)
MIDRANGE	Num1 ...	Num	引数の最小値と最大値の平均値
MIN	Alph1 ... または Anum1 ... または Int1 ...または	引数による*	引数の最小値

	(MF)Nat1 … または Num1 …		
MOD	Int1, Int2	Int	Int1 と Int2 のモジュロ (法)
(USING)(MF)NATIONAL-OF	Anum1 (MF)または Nat1	Nat	引数 Anum1 の USAGE NATIONAL 型の表現
NUMVAL	Anum1 または (MF)または Nat1	Num	単純数字文字列の数値
NUMVAL-C	Anum1 または Anum2 (MF)または Nat1 または Nat2	Num	コンマおよび通貨記号を含むことができる、数字文字列の数値
ORD	Alph1 または Anum1 (MF)または Nat1	Int	文字の照合順序における、引数の順序番号
ORD-MAX	Alph1 … または Anum1 … または (MF)Nat1 …または Num1	Int	最大の引数の順序番号
ORD-MIN	Alph1 … または Anum1 … または (MF)Nat1 … または Num1	Int	最小の引数の順序番号
(MF)PI	None	Num	値
PRESENT-VALUE	Num1, Num2 …	Num	将来にわたる時系列データ Num2 を、利率 Num1 で割り引いた現在の値
RANDOM	Int1	Num	乱数
RANGE	Int1 …または Num1	引数による*	引数の最大値と最小値の差
REM	Num1, Num2	Num	Num1 を Num2 で割った剰余
REVERSE	Alph1 または Anum1 (MF)または Nat1	引数による*	引数中の文字の順序を逆転させる
(MF)SIGN	Num1	Int	Num1 の符号
SIN	Num1	Num	Num1 のサイン (正弦)
SQRT	Num1	Num	Num1 の平方根
STANDARD-DEVIATION	Num1 …	Num	引数の標準偏差
SUM	Int1 … または Num1 …	引数による	引数の和
TAN	Num1	Num	Num1 のタンジェント (正接)
UPPER-CASE	Alph1 または Anum1	引数に	引数中のすべての文字を大文字に

	(MF)または Nat1	よる*	する
VARIANCE	Num1 ...	Num	引数の分散
WHEN-COMPILED	なし	Anum	プログラムがコンパイルされた日付と時刻
YEAR-TO-YYYY	Int1, Int2	Int	引数-2 の値に基づいて YY から YYYY に変換された引数-1

* 引数がすべて英字である関数の型は、英数字である。

11.9.1 ABS 関数

(MF)

ABS 関数は引数の絶対値を返す。

この関数の型は引数の型に応じて下記のようになる。

引数の型	関数の型
整数	整数
数字	数字

一般形式

```
FUNCTION ABS(引数-1)
```

引数

1. 引数-1 は字類が数字でなければならない。

戻り値

1. 戻り値は下記のようになる。
    a. 引数-1 の値がゼロまたは正である場合、(引数-1)
    b. 引数-1 の値が負である場合、- (引数-1)

11.9.2 ACOS 関数

ACOS 関数は、引数-1 に指定された角または弧 (単位はラジアン) のアークコサイン (逆余弦) の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION ACOS(引数-1)
```

引数

1. 引数-1 の字類は、数字とする。
2. 引数-1 の値は、-1 以上+1 以下とする。

戻り値

1. 戻り値は、引数-1 のアークコサインの近似値である。その値は 0 以上 π 以下である。

注:度およびグラードからラジアンへの変換係数をこの章の*三角関数*節に示してある。

---

2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.3 ANNUITY 関数

ANNUITY (即時年金) 関数は、引数-2 に指定された数の期間後に、最初の一括掛金に対して、指定された期間にわたって各期末に支払われる、年金の比率の近似値を返す。利子率は引数-1 に指定する。利息は各期末に、支払の前に算出される。この関数の型は数字である。

一般形式

```
FUNCTION ANNUITY(引数-1 引数-2)
```

引数

1. 引数-1 の字類は、数字とする。
2. 引数-1 の値は、0 以上とする。
3. 引数-2 は、正の整数とする。

戻り値

1. 引数-1 の値が 0 である場合、この関数の戻り値は下記の式の近似値である。

   1 / 引数-2

2. 引数-1 の値が 0 でない場合、この関数の戻り値は下記の式の近似値である。

   引数-1 / (1 - (1 + 引数-1) ** (- 引数-2))

3. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.4 ASIN 関数

ASIN 関数は引数-1 に指定された角または弧 (単位はラジアン) のアークサイン (逆正弦) の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION ASIN（引数-1）
```

引数

1. 引数-1 の字類は、数字とする。
2. 引数-1 の値は、-1 以上+1 以下とする。

戻り値

1. 戻り値は、引数-1 のアークサインの近似値である。その値は、-π/2 以上+π/2 以下である。

   注:度およびグラードからラジアンへの変換係数をこの章の*三角関数*節に示してある。

   ---

2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.5 ATAN 関数

ATAN 関数は、引数-1 に指定された角または弧 (単位はラジアン) のアークタンジェント (逆正接) の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION ATAN（引数-1）
```

引数

1. 引数-1 の字類は、数字とする。

戻り値

1. 戻り値は、引数-1 のアークタンジェントの近似値であり、-pi/2 より大きく、+pi/2 より小さい。

注:度およびグラードからラジアンへの変換係数をこの章の*三角関数*節に示してある。

---

2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.6 CHAR 関数

CHAR 関数は、プログラムの文字の照合順序の中から、引数-1 に指定された順序位置にある文字を返す。この関数の型は英数字である。

一般形式

```
FUNCTION CHAR (引数-1)
```

引数

1. 引数-1 は、整数とする。
2. 引数-1 の値は、0 よりも大きく、文字の照合順序の文字位置の数以下とする。

戻り値

1. プログラムの文字の照合順序中の同じ文字位置に、2 つ以上の文字が割り当てられている場合、ALPHABET 句中でその文字位置に指定されている最初の文字が、この関数の戻り値とされる。
2. ALPHABET 句によって現在のプログラムの文字の照合順序が指定されていない場合、固有の照合順序が使用される。

11.9.7 CHAR-NATIONAL 関数

CHAR-NATIONAL (国別文字) 関数は国別の大小順に並べた一連の文字の中から、順位が引数の値に等しい文字を返す。

関数の型は国別である。

一般形式

```
FUNCTION CHAR-NATIONAL (引数-1)
```

引数

1. 引数-1 は整数でなければならない。
2. 引数-1 の値は、ゼロよりも大きく、国別の文字の照合順序の最大値以下でなければならない。

戻り値

1. 戻り値は、国別の大小順に並べた一連の文字の中の、順位が引数の値に等しい文字である。
2. 現行の国別の文字の照合順序に同じ順位の文字が複数ある場合には、先に出てくる方の文字が戻り値となる。

11.9.8 COS 関数

COS 関数は、引数-1 に指定された角または弧 (単位はラジアン) のコサイン (余弦) の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION COS (引数-1)
```

引数

1. 引数-1 の字類は、数字とする。

注:度およびグラードからラジアンへの変換係数をこの章の*三角関数*節に示してある。

戻り値

1. 戻り値は、引数-1 のコサインの近似値であり、-1 以上+1 以下である。
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.9 CURRENT-DATE 関数

CURRENT-DATE 関数は、現在の日時に関する 21 文字の英数字の値を返す。その細目は、暦日、時刻、関数が実行されたシステムが稼働している地域の時刻とグリニッジ標準時刻との時差である。この関数の型は英数字である。

一般形式

```
FUNCTION CURRENT-DATE
```

戻り値

1. 戻り値の文字列の文字位置（左から右に数える）ごとの内容を下に示す。

文字位置	内　容
1-4	グレゴリオ歴の暦年を表わす 4 桁の数字。
5-6	年内の月を表わす 2 桁の数字。値は 01 から 12 まで。
7-8	月内の日を表わす 2 桁の数字。値は 01 から 31 まで。
9-10	真夜中から経過した時間を表わす 2 桁の数字。値は 00 から 23 まで。
11-12	時刻の分を表わす 2 桁の数字。値は 00 から 59 まで。
13-14	時刻の秒を表わす 2 桁の数字。値は 00 から 59 まで。
15-16	1 秒の 100 分の 1 を表わす 2 桁の数字。値は 00 から 99 まで。 1 秒よりも細かく時間を測定する機能のないシステムでは、この値は 00 となる。
17	グリニッジ標準時間との時差の遅早を表わす。値は"-"、" +"、"0"のいずれかである。 この値が"-" の時は、この前の文字位置に示された時間はグリニッジ標準時間よりも遅れていることを示す。この値が "+" の時は、この前の文字位置に示された時間はグリニッジ標準時間と等しいかそれよりも進んでいることを示す。この値が"0" の時は、この関数を実行したシステムにはグリニッジ標準時間との時差を測る機能がないことを示す。 グリニッジ標準時間との時差を測る機能がないシステムでは、文字位置 17 から 21 には値 00000 が返される。
18-19	文字位置 17 の値が"-" の時は、システムで使用されている時刻がグリニッジ標準時間よりも遅れている時間数が、2 桁の数字でここに示される。その値は 00 から 12 までである。文字位置 17 の値が"+" の時は、システムで使用されている時刻がグリニッジ標準時間よりも進んでいる時間数が、2 桁の数字でここに示される。その値は 00 から 13 までであ

る。文字位置 17 の値が "0" の時は、ここには 00 が返される。

20-21　システムで使用されている時刻とグリニッジ標準時間との時差の、1 時間に満たない分数を表わす。その値は 00 から 59 までである。時差の遅早は文字位置 17 の値が"+" であるか"-" であるかによって示される。文字位置 17 の値が"0" の時は、ここには 00 が返される。

11.9.10 DATE-OF-INTEGER 関数

DATE-OF-INTEGER 関数は、グレゴリオ暦の日付を整数形式から標準日付形式（YYYYMMDD）に変換する。この関数の型は整数である。

一般形式

FUNCTION DATE-OF-INTEGER（引数-1）

引数

1. 引数-1 は、グレゴリオ暦で 1601 年 1 月 1 日以降の通算日数を表わす正の整数とする。

戻り値

1. この関数の戻り値は、引数-1 に指定した整数に相当する ISO 標準日付を表わす。
2. 戻り値は、YYYYMMDD の形式をしている。YYYY はグレゴリオ暦の暦年を表わし、MM と DD はそれぞれ月と日を表わす。
3.

11.9.11 DATE-TO-YYYYMMDD 関数

DATE-TO-YYYYMMDD 関数は引数-1 を YYnnn の形から YYYYnnn の形へ変換する。引数-2 は実行時に年に追加された時、100 年間隔の終了年を定義するか、または引数-1 が起こるところへのウィンドウの移動を定義する。この関数の型は整数である。

一般形式

FUNCTION DATE-TO-YYYYMMDD (引数-1 [引数-2])

引数

1. 引数-1 は 1,000,000 より小さい、正の整数とする。
2. 引数-2 は整数とする。
3. 引数-2 が省略された場合、この関数は 50 が指定されたものとして評価される。
4. 実行時の年と引数-2 の値の合計は、10,000 より小さく、1699 より大きいものとする。

戻り値

1. 同等の算術式は次のとおりである。

   (FUNCTION YEAR-TO-YYYY (YY, 引数-2) * 10000 + nnnn)

   ここで、

   YY = FUNCTION INTEGER (引数-1/10000)
   nnnn = FUNCTION MOD (引数-1, 10000)

   INTEGER および MOD 関数の引数-1 と YEAR-TO-YYYY 関数の引数-2 は、DAY-TO-YYYYMMDD 関数自身の引数-1 と引数-2 と同じである。

注

1. 2002 年では、FUNCTION DATE-TO-YYYYMMDD (851003, 10) の戻り値は 19851003 である。1994 年では、FUNCTION DATE-TO-YYYYMMDD (961002, (-10)) の戻り値は 18981002 である。
2. この関数はウィンドウ移動アルゴリズムを持っている。決まったウィンドウを指定する方法については、YEAR-TO-YYYY 関数の注意の項を参照のこと。

11.9.12 DAY-OF-INTEGER 関数

DAY-OF-INTEGER 関数は、グレゴリオ暦の日付を整数形式からジュリアン日付（ユリウス暦）形式（YYYYDDD）に変換する。この関数の型は整数である。

一般形式

FUNCTION DAY-OF-INTEGER（引数-1）

引数

1. 引数-1 は、グレゴリオ暦で 1601 年 1 月 1 日以降の通算日数を表わす正の整数とする。

戻り値

1. この関数の戻り値は、引数-1 に指定した整数に相当するジュリアン日付を表わす。
2. 戻り値は、YYYYDDD の形式をしている。YYYY はグレゴリオ暦の暦年を表わし、DDD は年内の通算日を表わす。

11.9.13 DAY-TO-YYYYDDD 関数

DAY-TO-YYYYDDD 関数は、引数-1 を YYnnn の形から YYYYnnn の形へ変換する。引数-2 は実行時に年に追加された時、100 年間隔の終了年を定義するか、または引数-1 が起こるところへのウィンドウの移動を定義する。この関数の型は整数である。

一般形式

FUNCTION DAY-TO-YYYYDDD (引数-1 [引数-2])

引数

1. 引数-1 は 100,000 より小さい、正の整数とする。
2. 引数-2 は整数とする。
3. 引数-2 が省略された場合、この関数は 50 が指定されたものとして評価される。
4. 実行時の年と引数-2 の値の合計は、10,000 より小さく、1699 より大きいものとする。

戻り値

1. 同等の算術式は次のとおりである。

   (FUNCTION YEAR-TO-YYYY (YY, 引数-2) * 1000 + nnn)

   ここで、

   YY = FUNCTION INTEGER (引数-1/1000)
   nnn = FUNCTION MOD (引数-1, 1000)

   INTEGER および MOD 関数の引数-1 と YEAR-TO-YYYY 関数の引数-2 は、DAY-TO-YYYYDDD 関数自身の引数-1 と引数-2 と同じである。

注

1. 2002 年では、FUNCTION DAY-TO-YYYYDDD (10004,20) の戻り値は 20101004 である。1994 年では、FUNCTION DAY-TO-YYYYDDD (95005, (-10)) の戻り値は 1995005 である。
2. この関数は、ウィンドウ移動アルゴリズムを持っている。決まったウィンドウを指定する方法については、YEAR-TO-YYYY 関数の注意の項を参照のこと。

11.9.14 DISPLAY-OF 関数

DISPLAY-OF 関数は引数中の各国語型文字を英数字型の文字表現 (外部表現) に変換する。

この関数の型は英数字である。

一般形式

```
FUNCTION DISPLAY-OF (引数-1 [引数-2])
```

引数

1. 引数-1 の字類は各国語とする。
2. 引数-2 の字類は英字または英数字とし 1 文字位置の長さを占める。引数-2 に指定された文字は、変換時に対応する英数字表現が存在しないような各国語文字に対する置き換え文字として使われる。

戻り値

1. 引数-1 中の一つ一つの各国語文字を対応する英数字表現に変換し、戻り値として返す。各国語文字と英数字との対応は稼動する環境が定義する。
2. 引数-2 が指定されると、対応する英数字表現が存在しないような各国語文字に対してこの文字が戻り値に返される。
3. 引数-2 を指定しない場合、対応する英数字表現が存在しないような各国語文字に対しては、各国語型の外部表現に変換した文字が戻り値に返される。その際に必要な最小限の制御機能が付加される。
4. 戻り値の長さは、変換された結果を保持するのに必要な、USAGE DISPLAY の文字位置の個数となる。この長さは引数の長さと各国語文字集合の性質によって異なる。

11.9.15 E 関数

(MF)

E 関数は、自然対数の底である、eの近似値を返す。

この関数の型は数字である。

一般形式

```
FUNCTION E
```

戻り値

1. 戻り値は下記のとおり。

   2.718281828459045235

11.9.16 EXP 関数

(MF)

EXP 関数は *e* を引数の値分べき乗した値の近似値を返す。

この関数の型は数値である。

一般形式

```
FUNCTION EXP（引数-1）
```

引数

1. 引数-1 は字類が数字でなければならない。

戻り値

1. 戻り値は下記のとおり。

   FUNCTION E ** 引数-1

11.9.17 EXP10 関数

MF

EXP10 関数は 10 を引数の値分べき乗した値の近似値を返す。

この関数の型は数字である。

一般形式

```
FUNCTION EXP10（引数-1）
```

引数

1. 引数-1 は字類が数字でなければならない。

戻り値

1. 戻り値は下記のとおり。

   10 ** 引数-1

11.9.18 FACTORIAL 関数

FACTORIAL 関数は、引数-1 の階乗である整数を返す。この関数の型は整数である。

一般形式

```
FUNCTION FACTORIAL (引数-1)
```

引数

1. 引数-1 は、0 以上の整数とする。

戻り値

1. 引数-1 の値が 0 の場合は、値 1 が返される。
2. 引数-1 の値が正の場合は、その階乗値が返される。
3. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.19 FRACTION-PART 関数

MF

FRACTION-PART 関数は引数の端数部分の数値を返す。

この関数の型は数字である。

一般形式

```
FUNCTION FRACTION-PART (引数-1)
```

引数

1. 引数-1 は字類が数字でなければならない。

戻り値

1. 戻り値は下記のとおり。

   (引数-1 - FUNCTION INTEGER-PART (引数-1))

   ただし、INTEGER-PART 関数の引数は FRACTION-PART 関数の引数と同じとする。

注:引数-1 の値が+1.5 であると、+0.5 が返される。引数-1 の値が-1.5 であると、-0.5 が返される。

11.9.20 INTEGER 関数

INTEGER 関数は、引数の値以下の整数の最大値を返す。この関数の型は整数である。

一般形式

```
FUNCTION INTEGER（引数-1）
```

引数

1. 引数-1 の字類は、数字とする。

戻り値

1. この関数の戻り値は、引数-1 以下の値で最大の整数である。たとえば、引数の値が-1.5 であると-2 が返され、引数-1 の値が+1.5 であると+1 が返される。

11.9.21 INTEGER-OF-DATE 関数

INTEGER-OF-DATE 関数は、グレゴリオ暦の日付を標準日付形式（YYYYMMDD）から整数形式に変換する。この関数の型は整数である。

一般形式

```
FUNCTION INTEGER-OF-DATE（引数-1）
```

引数

1. 引数-1 は、YYYYMMDD 形式の整数とする。その値は、下記の式によって算出される。

   (YYYY * 10,000) + (MM * 100) + DD

   a. YYYY はグレゴリオ暦の暦年を表わす。これは 1601 以上の整数とする。
   b. MM は月を表わす。これは 1 から 12 までの整数とする。
   c. DD は日を表わす。これは 1 から 31 までの整数とする。ただし、年月の組合わせに対応した有効な日数にすること。

戻り値

1. この関数の戻り値は、引数-1 がグレゴリオ暦で 1601 年 1 月 1 日以降、通算何日目にあたるかを示す整数である。

11.9.22 INTEGER-OF-DAY 関数

INTEGER-OF-DAY 関数は、グレゴリオ暦の日付をジュリアン日付（ユリウス暦）形式（YYYYDDD）から整数形式に変換する。この関数の型は整数である。

一般形式

```
FUNCTION INTEGER-OF-DAY（引数-1）
```

引数

1. 引数-1 は、YYYYDDD 形式の整数とする。その値は下記の式によって算出される。

   (YYYY * 1000) + DDD

   a. YYYY はグレゴリオ暦の暦年を表わす。これは 1601 以上の整数とする。
   b. DDD は日を表わす。これは 1 から 366 までの整数とする。ただし、対応する年に応じた有効な日数にすること。

戻り値

1. この関数の戻り値は、引数-1 がグレゴリオ暦で 1601 年 1 月 1 日以降、通算何日目にあたるかを示す整数である。

11.9.23 INTEGER-PART 関数

INTEGER-PART 関数は、引数-1 の整数部分の値を返す。この関数の型は整数である。

一般形式

```
FUNCTION INTEGER-PART（引数-1）
```

引数

1. 引数-1 の字類は、数字とする。

戻り値

1. 引数-1 の値が 0 の場合は、0 が返される。
2. 引数-1 の値が正の場合は、引数-1 の値以下の最大の整数が返される。たとえば、引数-1 の値が+1.5 であれば、+1 が返される。
3. 引数-1 の値が負の場合は、引数-1 の値以上の最小の整数が返される。たとえば、引数-1 の値が-1.5 であれば、-1 が返される。

11.9.24 LENGTH 関数

LENGTH 関数は、英数字文字位置にある引数の長さ、

(MF)または、引数の字類によっては各国語型文字位置にある引数の長さ

と同じ整数を返す。

この関数の型は整数である。

一般形式

```
FUNCTION LENGTH（引数-1）
```

引数

1. 引数-1 は、文字定数または任意の字類または項類のデータ項目であってよい。
2. 引数-1 またはその下位に属するデータ項目のいずれかに、OCCURS 句の DEPENDING 指定がなされている場合、DEPENDING 句に指定されたデータ項目によって参照されるデータ項目の内容は、この関数が実行される時点で使用される。

戻り値

1. 引数-1 が、英数字定数、または字類が各国語型以外の基本データ項目、または可変反復データ項目を含まない集団項目のいずれかである場合、英数字文字位置にある引数-1 の長さと同じ整数を返す。

   (MF)引数-1 が、字類が各国語型の基本データ項目または各国語型定数のいずれかである場合、各国語文字位置にある引数-1 の長さと同じ整数を返す。

2. 引数-1 が可変反復データ項目を含む集団項目である場合、その可変反復データ項目を定義している OCCURS 句の DEPENDING 指定の対象となっているデータ項目を評価した結果の文字数を表わす、整数が返される。この評価は、送出し側データ項目に関する OCCURS 句の規則に従って行われる。*データ部 - ファイルおよびデータ記述*の章の *OCCURS(反復)句*節を参照。
3. 暗黙の FILLER があれば、その文字数も戻り値に含まれる。

11.9.25 LENGTH-AN 関数

(MF)

LENGTH-AN 関数は、引数を英数字の文字位置で数えた、長さを表す整数を返す。

この関数の型は整数である。

一般形式

```
FUNCTION LENGTH-AN（引数-1）
```

引数

1. 引数-1 は任意の字類または項類の定数またはデータ項目であってよい。
2. 引数-1 またはその下位の任意のデータ項目に OCCURS 句の DEPENDING 指定が記述されている場合、LENGTH-AN 関数が評価されるときに、DEPENDING 指定中のデータ名によって参照されるデータ項目の内容が使用される

戻り値

1. 引数-1 が基本データ項目である場合、戻り値は引数-1 を英数字の文字位置で数えた長さを表す整数である。
2. 引数-1 が集団データ項目である場合、
    a. 引数-1 またはその下位の任意のデータ項目に OCCURS 句の DEPENDING 指定が記述されている場合には、戻り値は送出し側データ項目としての引数-1 を英数字の文字位置で数えた長さを表す整数である。その場合のデータ項目の送出しは、OCCURS 句の規則に従って、DEPENDING 指定中のデータ項目を評価することによって決定される。
    b. そうでない場合には、戻り値は引数-1 を英数字の文字位置で数えた長さを表す整数である。
    c. 引数-1 に暗黙的な無名項目があれば、戻り値にはその分の長さも含まれる。

11.9.26 LOG 関数

LOG 関数は、e を底とする引数-1 の対数（自然対数）の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION LOG（引数-1）
```

引数

1. 引数-1 の字類は、数字とする。
2. 引数-1 の値は、0 より大きくする。

戻り値

1. e を底とする引数-1 の対数の近似値が返される。
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.27 LOG10 関数

LOG10 関数は、10 を底とする引数-1 の対数の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION LOG10（引数-1）
```

引数

1. 引数-1 の字類は、数字とする。
2. 引数-1 の値は、0 より大きくする。

戻り値

1. 10 を底とする引数-1 の対数の近似値が返される。
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.28 LOWER-CASE 関数

LOWER-CASE 関数は、文字列中の大文字を対応する小文字に置き換えて、引数-1 と同じ長さの文字列を返す。この関数の型は引数によって決まる。具体的には下記のとおり。

引数の型	関数の型
英字	英数字
英数字	英数字
(MF)各国語型	各国語型

一般形式

```
FUNCTION LOWER-CASE（引数-1）
```

引数

1. 引数-1 は、英字、英数字、

   (MF)または各国語型文字

   とし、最低1文字の長さがなければならない。

戻り値

1. 大文字が小文字に置き換えられる点を除けば、引数-1 と同じ文字列が返される。
2. 返される文字列の長さは、引数-1 と同じである。
3. 使用している計算機の文字集合に小文字が含まれていない場合、文字の置換は行われない。

11.9.29 MAX 関数

MAX 関数は、引数-1 のうちで値が最大であるものの内容を返す。 この関数の型は引数によって決まる。具体的には下記のとおり。

引数の型	関数の型
英字	英数字
英数字	英数字
(MF)各国語型	各国語型
すべての引数整数	整数
数字 (一部に整数を含んでいてもよい)	数字

一般形式

```
FUNCTION MAX (引数-1[,...])
```

引数

1. 引数-1 を複数指定する場合、その字類はすべて同じにする。ただし、英字と英数字の引数は混在させてもよい。

戻り値

1. 引数-1 のうち、値が最大のものの内容が返される。値を比較する際は、比較条件の規則が適用される。***手続き部***の章の*比較条件*節を参照。
2. 最大値をもつ引数-1 が複数存在する場合、最も左側の引数-1 の内容が返される。
3. 英数字型の関数、

   (MF)または各国語型関数

   の場合、返される値の文字数は、選択された引数-1 の文字数と同じである。

4. 引数-1 の値が非整数の場合、結果は浮動小数点形式で戻される。

11.9.30 MEAN 関数

MEAN 関数は、引数の算術平均値を返す。この関数の型は数字である。

一般形式

```
FUNCTION MEAN ([引数-1]...)
```

引数

1. 引数-1 の字類は、数字とする。

戻り値

1. 一連の引数-1 の算術平均値が返される。
2. この算術平均値は、一連の引数-1 の和を引数-1 の数で割ったものである。
3. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.31 MEDIAN 関数

MEDIAN 関数は、引数のメジアン (値によって整列したときに中央に来る引数) の内容を返す。この関数の型は数字である。

一般形式

```
FUNCTION MEDIAN ([引数-1]...)
```

引数

1. 引数-1 の字類は、数字とする。

戻り値

1. 一連の引数-1 を値によって整列したときに、中央に来る引数の内容が返される。
2. 指定した引数-1 の数が奇数の場合、引数-1 の数の少なくとも半分が戻り値以上の値となり、少なくとも半分が戻り値以下の値となるような値が返される。指定した引数-1 の数が偶数の場合、中央に来る 2 つの引数の算術平均値が返される。
3. 引数-1 の値を整列する際の比較には、比較条件の規則が適用される。*手続き部*の章の*比較条件*節を参照。
4. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.32 MIDRANGE 関数

MIDRANGE 関数は、引数の最小値と最大値の算術平均値を返す。この関数の型は数字である。

一般形式

```
FUNCTION MIDRANGE (引数-1[,...])
```

引数

1. 引数-1 の字類は、数字とする。

戻り値

1. 引数-1 の最小値と最大値の算術平均値が返される。
2. 最小値と最大値を判定する際の比較には、比較条件の規則が適用される。***手続き部***の章の*比較条件*節を参照。
3. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.33 MIN 関数

MIN 関数は、引数-1 のうちで値が最小であるものの内容を返す。この関数の型は引数によって決まる。具体的には下記のとおり。

引数の型	関数の型
英字	英数字
英数字	英数字
(MF)各国語型	各国語型
すべての引数整数	整数
数字	数字
(一部に整数を含んでいてもよい)	

一般形式

```
FUNCTION MIN (引数-1[,...])
```

引数

1. 引数-1 を複数指定する場合、その字類はすべて同じにする。ただし、英字と英数字の引数は混在させてもよい。

戻り値

1. 引数-1 のうち、値が最小のものの内容が返される。値を比較する際は、比較条件の規則が適用される。***手続き部***の章の*比較条件*節を参照。
2. 最小値をもつ引数-1 が複数存在する場合、最も左側の引数-1 の内容が返される。
3. 英数字型の関数、

   (MF)または各国語型関数

   の場合、返される値の文字数は、選択された引数-1 の文字数と同じである。

4. 引数-1 の値が非整数の場合、結果は浮動小数点形式で戻される。

11.9.34 MOD 関数

MOD 関数は、引数-1 を引数-2 で割ったときのモジュロー (法、割り切れる値との差) を整数値で返す。この関数の型は整数である。

一般形式

```
FUNCTION MOD (引数-1 引数-2)
```

引数

1. 引数-1 と引数-2 は、整数とする。
2. 引数-2 の値は、0 であってはならない。

戻り値

1. 引数-1 を引数-2 で割ったときのモジュローが返される。その計算式は下記のとおり。

   引数-1 - (引数-2 * FUNCTION INTEGER (引数-1 / 引数-2))

2. どのような結果が得られるかを下に例示する。

引数-1	引数-2	戻り値
11	5	1
-11	5	4
11	-5	-4

-11　　-5　　-1

11.9.35 NATIONAL-OF 関数

NATIONAL-OF 関数は引数中の英数字型文字を各国語型の文字表現（内部表現）に変換する。

この関数の型は各国語である。

一般形式

```
FUNCTION NATIONAL-OF (引数-1 [引数-2])
```

引数

1. 引数-1 の字類は英字または英数字とする。英数字の場合、引数-1 には、英数字文字、各国語文字またはその両方が外部表現の形式で含まれ得る。
2. 引数-2 の字類は各国語とし 1 文字位置の長さを占める。引数-2 に指定された文字は、変換時に対応する各国語表現が存在しないような英数字文字に対する置き換え文字として使われる。

戻り値

1. 引数-1 中の一つ一つの英数字文字を対応する各国語表現に変換し、戻り値として返す。引数-1 に含まれる外部表現が持つすべての制御機能は、単に文字種別を判別するためだけに用いられ、変換対象の文字とは見なされない。
2. 引数-2 が指定されると、対応する内部形式の各国語表現が存在しないような英数字文字に対してこの文字が戻り値に返される。
3. 引数-2 を指定しない場合、対応する内部形式の各国語表現が存在しないような英数字文字に対しては、各国語型の空白文字が戻り値に返される。
4. 戻り値の長さは、変換された結果を保持するのに必要な、USAGE NATIONAL の文字位置の個数となる。この長さは引数-1 の長さによって定まる。

11.9.36 NUMVAL 関数

NUMVAL 関数は、引数-1 に指定された文字列によって表わされる数値を返す。前後の空白は無視される。この関数は数字型である。

一般形式

FUNCTION NUMVAL (引数-1)

引数

1. 引数-1 は英数字定数、

   (MF)または各国語型定数、

   または英数字データ項目、

   (MF)または各国語型データ項目

   であり、下記のどちらかの形式をとる。

   [空白] [+ / -] [空白] { 数字 [. [数字]] / . 数字 } [空白]

   または

   [空白] { 数字 [. [数字]] / . 数字 } [空白] [+ / - / CR / DB] [空白]

   ここで、空白はいくつか (0 個を含む) の空白文字の文字列を表わし、数字は 1 桁から 18 桁までの数字列を表わす。

2. 引数-1 の合計桁数は、18 を超えてはならない。
3. 特殊名段落に DECIMAL POINT IS COMMA 句を指定した場合は、引数-1 内でも小数点の代わりにコンマを使用する。

戻り値

1. 引数-1 によって表わされる数値が返される。
2. 返される数値の桁数は 18 である。
3. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.37 NUMVAL-C 関数

NUMVAL-C 関数は、引数-1 に指定された文字列によって表わされる数値を返す。引数-2 で指定された任意の通貨記号文字列、または小数点の前にある任意のコンマは無視される。この関数は数字型である。

一般形式

FUNCTION NUMVAL-C（引数-1［引数-2］）

引数

1. 引数-1 は英数字定数 、

   (MF)または各国語型定数、

   または英数字データ項目、

   (MF)または各国語型データ項目

   であり、下記のどちらかの形式とする。

   [空白] [+ / -] [空白] [通貨記号] [空白] { 数字 [, 数字]... [. [数字]] / . 数字 } [空白]

   または

   [空白] [通貨記号] [空白] { 数字 [, 数字]... [. [数字]] / . 数字 } [空白] [+ / - / CR / DB] [空白]

   ここで、空白はいくつか (0 個を含む) の空白文字の文字列を表わし、通貨記号は引数-2 に指定した文字の何文字かの文字列を表わし、数字は何桁かの数字列を表わす。

2. 特殊名段落に DECIMAL POINT IS COMMA 句を指定した場合は、引数-1 中のコンマと小数点の役割が逆転される。
3. 引数-1 の合計桁数は、18 を超えてはならない。
4. 引数-2 を指定した場合は、引数-1 と同じ字類とする。

   引数-2 に数字、コンマ、空白、正号 (+) 、負号 (-) 、小数点を指定してはならない。

5. 引数-2 を指定しないと、引数-1 の字類が英数字の時には、通貨記号文字列はプログラムに指定したものが使用される。

   (MF)または引数-1 の字類が各国語型の時には，通貨記号文字列はプログラムで指定した各国語型表現のものが使用される。通貨記号の表現は、
   FUNCTION NATIONAL-OF (cs)から戻される値である。

戻り値

1. 引数-1 によって表わされる数値が返される。
2. 返される数値の桁数は 18 である。
3. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.38 ORD 関数

ORD 関数は、プログラムの文字の照合順序中で、引数-1 が位置する順序を示す整数値を返す。順序番号の最小値は 1 である。この関数の型は整数である。

一般形式

```
FUNCTION ORD（引数-1）
```

引数

1. 引数-1 は、字類が英字、英数字、

   (MF)または各国語型文字

   の 1 文字とする。

戻り値

1. 引数-1 の字類が英字か英数字の場合、プログラムの英数字文字の照合順序中で、引数-1 が位置する順序を示す値が返される。
2. (MF)引数-1 の字類が各国語型の場合、プログラムの現行の国の照合順序中で、引数-1 が位置する順序を示す値が返される。

11.9.39 ORD-MAX 関数

引数-1 の字類が各国語型の場合、プログラムの現行の国の照合順序中で、引数-1 が位置する順序を示す値が返される。

一般形式

```
FUNCTION ORD-MAX（[引数-1]...）
```

引数

1. 引数-1 を複数指定する場合、その字類はすべて同じにする。ただし、英字と英数字の引数は混在させてもよい。

戻り値

1. 一連の引数-1 のうち、値が最大のものの順序番号が返される。
2. 最大値を決めるために値を比較する際は、比較条件の規則が適用される。***手続き部***の章の*比較条件*節を参照。
3. 最大値をもつ引数-1 が複数存在する場合、そのうちの最も左側のものの順序番号が返される。

11.9.40 ORD-MIN 関数

ORD-MIN 関数は、引数-1 の中で値が最小のものの順序番号を返す。この関数の型は整数である。

一般形式

```
FUNCTION ORD-MIN ([引数-1]...)
```

引数

1. 引数-1 を複数指定する場合、その字類はすべて同じにする。ただし、英字と英数字の引数は混在させてもよい。

戻り値

1. 一連の引数-1 のうち、値が最小のものの順序番号が返される。
2. 最小値を決めるために値を比較する際は、比較条件の規則が適用される。***手続き部***の章の*比較条件*節を参照。
3. 最小値をもつ引数-1 が複数存在する場合、そのうちの最も左側のものの順序番号が返される。

11.9.41 PI 関数

MF

PI 関数は円周率の近似値を返す。

この関数の型は数字である。

一般形式

```
FUNCTION PI
```

戻り値

1. 戻り値は下記のとおり。

   3.141592653589793238

11.9.42 PRESENT-VALUE 関数

PRESENT-VALUE 関数は、引数-2 に指定される将来の一連の期間にわたる値を、引数-1 に指定される率で割り引いた、現在の値の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION PRESENT-VALUE (引数-1 [引数-2]...)
```

引数

1. 引数-1 および引数-2 の字類は、数字とする。
2. 引数-1 の値は、-1 よりも大きくする。

戻り値

1. 各期ごとに下記の式によって算出した、一連の値の和の近似値が返される。

   引数-2 / (1+引数-1) ** n

   引数-2 の 1 つの値ごとに、1 つの期が想定される。指数 n は 1 から始まって、期が進むごとに 1 ずつ繰り上げられる。

2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.43 RANDOM 関数

RANDOM 関数は、一様分布から擬似乱数値を 1 つ返す。この関数の型は数字である。

一般形式

```
FUNCTION RANDOM [(引数-1)]
```

引数

1. 引数-1 を指定する場合、その値は 0 または正の整数とする。引数-1 は、一連の疑似乱数を発生させるためのシード値として使用される。
2. 後で別の引数-1 を指定すると、新しい一連の擬似乱数が生成される。

3. (MF)実行単位内で最初にこの関数が実行されるときに引数-1 が指定されていないと、シード値として 0 が使用される。
4. いずれの場合も、引数-1 を指定しないでこの関数を繰り返し実行すると、現在の一連の乱数中の次の値が返される。

戻り値

1. 0 以上 1 未満の値が返される。
2. 特定の処理系における特定のシード値に対して発生される一連の擬似乱数は、つねに同じである。
3. 引数-1 の値の定義域は、別に一連の擬似乱数を発生させる。この部分集合には、0 から最低 32767 までの値が含まれる。
4. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.44 RANGE 関数

RANGE 関数は、最大の引数値と最小の引数値との差を返す。この関数の型は、引数の型によって決まる。具体的には下記のとおり。

引数の型	関数の型
すべて引数整数	整数
数字 (一部に整数を含んでいてもよい)	数字

一般形式

```
FUNCTION RANGE (引数-1,...)
```

引数

1. 引数-1 の字類は、数字とする。

戻り値

1. 引数-1 の最大値から、引数-1 の最小値を差し引いた値が返される。
2. 最大値および最小値を決めるために値を比較する際は、単純条件の規則が適用される。***手続き部***の章の*比較条件*節を参照。
3. 引数-1 の値が非整数の場合、結果は浮動小数点形式で戻される。

11.9.45 REM 関数

REM 関数は、引数-1 を引数-2 で割った剰余値を返す。この関数の型は数字である。

一般形式

```
FUNCTION REM (引数-1 引数-2)
```

引数

1. 引数-1 と引数-2 の字類は、数字とする。
2. 引数-2 の値は、ゼロであってはならない。

戻り値

1. 引数-1 を引数-2 で割った剰余値が返される。その計算式は下記のとおり。

   引数-1 - (引数-2 * FUNCTION INTEGER-PART (引数-1 / 引数-2))

2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.46 REVERSE 関数

REVERSE 関数は、引数-1 の文字列の順序を逆転させた同じ長さの文字列を返す。この関数の型は、引数の型によって決まる。具体的には下記のとおり。

引数の型	関数の型
英字	英数字
英数字	英数字
(MF)各国語型	各国語型

一般形式

```
FUNCTION REVERSE (引数-1)
```

引数

1. 引数-1 は、字類が英字、英数字、

   (MF)または各国語型

   であり、長さは 1 文字以上とする。

戻り値

1. 引数-1 の文字列の長さを n、その中の任意の文字の位置を j とすると (1 ≤ j ≤ n)、返される文字列中の文字位置 j にある文字は、元の文字列の n-j+1 の位置から持って来たものである。

11.9.47 SIGN 関数

MF

SIGN 関数は引数の正負の範囲に応じて、+1, 0, -1 のどれかを返す。

この関数の型は整数である。

一般形式

```
FUNCTION SIGN (引数-1)
```

引数

1. 引数-1 の字類は数字とする。

戻り値

1. 戻り値は下記のとおり。
    a. 引数-1 が正の数である場合には、1
    b. 引数-1 の値がゼロである場合には、0
    c. 引数-1 が負の数である場合には、-1

11.9.48 SIN 関数

SIN 関数は、引数-1 に指定された角または弧 (単位はラジアン) のサイン (正弦) の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION SIN (引数-1)
```

引数

1. 引数-1 の字類は、数字とする。

注:度およびグラードからラジアンへの変換係数をこの章の*三角関数*節に示してある。

戻り値

1. 引数-1 の、サインの近似値が返される。その値は-1 以上+1 以下である。
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.49 SQRT 関数

SQRT 関数は、引数-1 の平方根の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION SQRT (引数-1)
```

引数

1. 引数-1 の字類は、数字とする。
2. 引数-1 の値は 0 または正の整数とする。

戻り値

1. 引数-1 の平方根の、近似値の絶対値が返される。
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.50 STANDARD-DEVIATION 関数

STANDARD-DEVIATION 関数は、引数の標準偏差の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION STANDARD-DEVIATION (引数-1...)
```

引数

1. 引数-1 の字類は、数字とする。

戻り値

1. 一連の引数-1 の標準偏差の近似値が返される。

2. 計算手順は下記のとおり。
    a. 一連の引数-1 の算術平均を算出する。各引数-1 の値からその平均値を引いて、差異を算出する。各差異を 2 乗する。
    b. 各 2 乗値を加算する。その和を引数-1 の数で割る。
    c. その商の平方根を求める。その平方根の絶対値を戻り値とする。
3. 引数-1 が 1 つしかない場合、または一連の引数-1 がすべて可変の反復データ項目から成りそのすべてについて反復数が 1 である場合、ゼロが返される。
4. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.51 SUM 関数

SUM 関数は、引数の値の和を返す。この関数の型は、引数の型によって決まる。具体的には下記のとおり。

引数の型	関数の型
すべて整数	整数
数字 (一部に整数を含んでいてもよい)	数字

一般形式

```
FUNCTION SUM (引数-1...)
```

引数

1. 引数-1 の字類は、数字とする。

戻り値

1. 引数の値の和が返される。
2. 一連の引数-1 がすべて整数であれば、戻り値は整数である。
3. 一連の引数-1 の中に整数でないものがあれば、戻り値は浮動小数形式の数字である。

11.9.52 TAN 関数

TAN 関数は、引数-1 に指定された角または弧 (単位はラジアン) のタンジェント (正接) の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION TAN (引数-1)
```

引数

1. 引数-1 の字類は、数字とする。

注:度およびグラードからラジアンへの変換係数をこの章の*三角関数*節に示してある。

戻り値

1. 引数-1 のタンジェントの近似値が返される。
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.53 UPPER-CASE 関数

UPPER-CASE 関数は、引数-1 の文字列中の小文字を対応する大文字に置き換えた同じ長さの文字列を返す。この関数の型は、引数の型によって決まる。具体的には下記のとおり。

引数の型	関数の型
英字	英数字
英数字	英数字
(MF)各国語型	各国語型

一般形式

```
FUNCTION UPPER-CASE (引数-1)
```

引数

1. 引数-1 は、字類、英字、英数字

   (MF)または各国語型

   であり、1 文字以上の長さとする。

戻り値

1. 小文字が対応する大文字に置き換えられる点を除けば、引数-1 と同じ文字列が返される。
2. 返される文字列の長さは、引数-1 と同じである。

11.9.54 VARIANCE 関数

VARIANCE 関数は、引数の分散の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION VARIANCE ([引数-1]...)
```

引数

1. 引数-1 の字類は、数字とする。

戻り値

1. 一連の引数-1 の分散の近似値が返される。
2. 分散は標準偏差を 2 乗したものである。(前述の *STANDARD-DEVIATION 関数*節を参照。)
3. 引数-1 が 1 つしかない場合、または一連の引数-1 がすべて可変の反復データ項目から成りそのすべてについて反復数が 1 である場合、ゼロが返される。
4. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.55 WHEN-COMPILED 関数

WHEN-COMPILED 関数は、プログラムがコンパイルされた日付と時刻を返す。この関数の型は英数字である。

一般形式

```
FUNCTION WHEN-COMPILED
```

戻り値

1. 戻り値の、文字列の文字位置 (左から右に数える) ごとの内容を下に示す。

文字位置	内　容
1-4	グレゴリオ歴の暦年を表わす 4 桁の数字。
5-6	年内の月を表わす 2 桁の数字。値は 01 から 12 まで。

7-8　　月内の日を表わす 2 桁の数字。値は 01 から 31 まで。

9-10　　真夜中から経過した時間を表わす 2 桁の数字。値は 00 から 23 まで。

11-12　　時刻の分を表わす 2 桁の数字。値は 00 から 59 まで。

13-14　　時刻の秒を表わす 2 桁の数字。値は 00 から 59 まで。

15-16　　1 秒の 100 分の 1 を表わす 2 桁の数字。値は 00 から 99 まで 。

1 秒よりも細かく時間を測定する機能のないシステムでは、この値は 00 となる。

17　　グリニッジ標準時間との時差の遅早を表わす。値は"-", " +", "0" のいずれかである。この値が"-"の時は、この前の文字位置に示された時間はグリニッジ標準時間よりも遅れていることを示す。この値が"+"の時は、この前の文字位置に示された時間はグリニッジ標準時間と等しいかそれよりも進んでいることを示す。この値が"0"の時は、この関数を実行したシステムにはグリニッジ標準時間との時差を測る機能がないことを示す。

グリニッジ標準時間との時差を測る機能がないシステムでは、文字位置 17 から 21 には値 0000 が返される。

18-19　　文字位置 17 の値が"-"の時は、システムで使用されている時刻がグリニッジ標準時間よりも遅れている時間数が、2 桁の数字でここに示される。その値は 00 から 12 までである。文字位置 17 の値が"+"の時は、システムで使用されている時刻がグリニッジ標準時間よりも進んでいる時間数が、2 桁の数字でここに示される。その値は 00 から 13 までである。文字位置 17 の値が"0"の時は、ここには 00 が返される。

20-21　　システムで使用されている時刻とグリニッジ標準時間との時差の、1 時間に満たない分数を表わす。その値は 00 から 59 までである。時差の遅早は文字位置 17 の値が"+"であるか"-"であるかによって示される。

文字位置 17 の値が"0"の時は、ここには 00 が返される。

2. この関数を含むプログラムがコンパイルされた日付と時刻が返される。プログラムが他のプログラムに含まれている場合には、含む方のプログラムがコンパイルされたときの日付と時刻が返される。
3. ここに返される値は、該当するプログラムの原始プログラム･リストおよび実行用プログラムに付けられているものと同じである。ただし、その表現形式と精度は異なる。

11.9.56 YEAR-TO-YYYY 関数

YEAR-TO-YYYY 関数は、引数-1 を、年の下位の 2 桁から 4 桁の年へ変換する。引数-2 は実行時に年に追加された時、100 年間隔の終了年を定義するか、または引数-1 が起こるところへのウィンドウの移動を定義する。この関数の型は整数である。

一般形式

FUNCTION YEAR-TO-YYYY (引数-1 [引数-2])

引数

1. 引数-1 は 100 より小さい、負でない整数とする。
2. 引数-2 は整数とする。
3. 引数-2 が省略された場合、この関数は 50 が指定されたものとして評価される。
4. 実行時の年と引数-2 の値の合計は、10,000 より小さく、1699 より大きいものとする。

戻り値

1. 最大年は次のように計算される。

   (FUNCTION NUMVAL (FUNCTION CURRENT-DATE (1:4)) + 引数-2)

   ここで、NUMVAL 関数の引数-2 は YEAR-TO-YYYY 関数自身の引数-2 と同じである。

2. 同等の算術式は次のとおりである。

   1. 以下の条件が真のとき:

      (FUNCTION MOD (最大年, 100) > = 引数-1)

      同等の算術式:

      引数-1 + 100 * (FUNCTION INTEGER (最大年/100))

   2. それ以外のときの、同等の算術式:

      引数-1 + 100 * (FUNCTION INTEGER (最大年/100) - 1)

注

1. 1995 年では、FUNCTION YEAR-TO-YYYY (4, 23) の戻り値は 2004 である。2008 年では、FUNCTION YEAR-TO-YYYY (98, (-15))の戻り値は 1898 である。
2. YEAR-TO-YYYY 関数はウィンドウ移動アルゴリズムを持っている。これをきまったウィンドウに使用するには、引数-2 を次のように指定できる。ここで、固定最大年は固定した 100 年間隔での最大年とする。

   (固定最大年 - FUNCTION NUMVAL (FUNCTION CURRENT-DATE (1:4)))

   決まったウィンドウが 1973 から 2072 の場合、2009 では引数-2 は値 63 を持ち、2019 では値 53 を持つ。

第 12 章 : 手続き部 - ACCEPT - DIVIDE

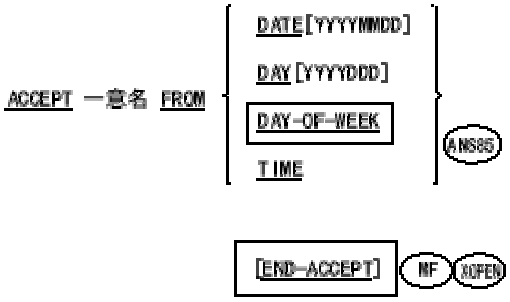
12.1 ACCEPT 文

ACCEPT (受取り) 文は、操作卓からキー入力されるデータ、またはオペレーティングシステムから提供されるデータを、指定されたデータ項目に入れてプログラムで利用できるようにする。

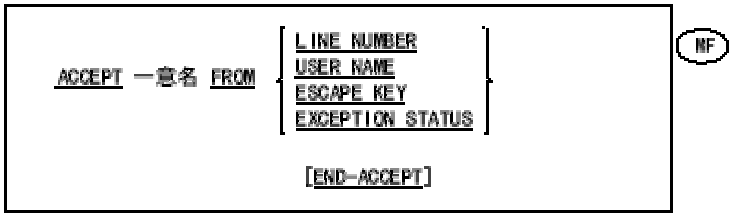
一般形式

形式 1

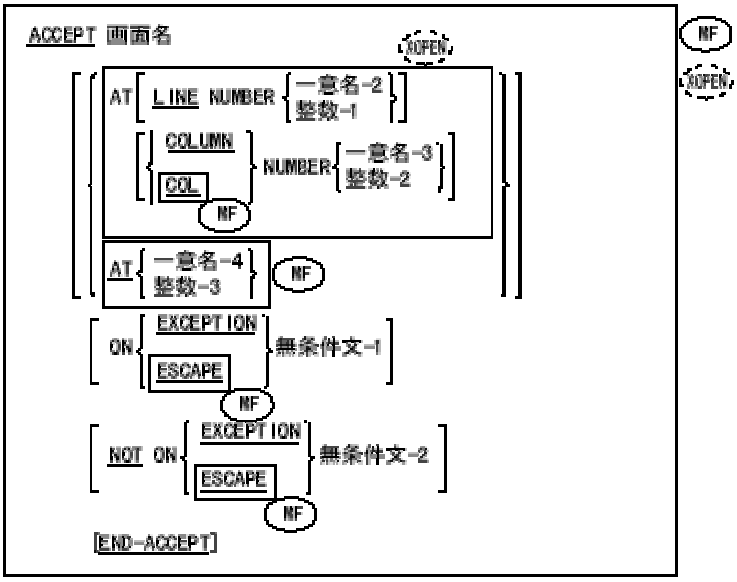
形式 2



形式 3



形式 4


ACCEPT 画面名
NOFEN
NF
NOFEN
AT
LINE NUMBER
一意名-2
整数-1
COLUMN
COL
NF
NUMBER
一意名-3
整数-2
AT
一意名-4
整数-3
NF
ON
EXCEPTION
ESCAPE
無条件文-1
NF
NOT ON
EXCEPTION
ESCAPE
無条件文-2
NF
[END-ACCEPT]

形式 5

```
ACCEPT 一意名-1                                                    (NF)

[{ AT [LINE NUMBER {一意名-2 | 整数-1}]
     [{COLUMN | COL} NUMBER {一意名-3 | 整数-2}]
  AT {一意名-4 | 整数-3}                                          }]

[FROM CRT][MODE IS BLOCK]

[ WITH { {AUTO | AUTO-SKIP}
         {BELL | BEEP}
         BLINK
         {FULL | LENGTH-CHECK}
         GRID
         [HIGHLIGHT | LOWLIGHT]
         LEFTLINE
         OVERLINE
         PROMPT [CHARACTER IS [定数-1]]
         {REQUIRED | EMPTY-CHECK}
         REVERSE-VIDEO
         {SECURE | NO-ECHO}
         SIZE IS {一意名-6 | 整数-4}
         UNDERLINE
         {FOREGROUND-COLOR | FOREGROUND-COLOUR} IS 整数-5
         {BACKGROUND-COLOR | BACKGROUND-COLOUR} IS 整数-6
         CONTROL IS {一意名-7 | 定数-2}
         {TIME-OUT | TIMEOUT} AFTER {整数-7 | 一意名-8}
         LEFT-JUSTIFY
         RIGHT-JUSTIFY
         SPACE-FILL
         TRAILING-SIGN
         UPDATE
         UPPER
         LOWER
         ZERO-FILL                                              }... ]

[ ON {EXCEPTION | ESCAPE} 無条件文-1 ]

[ NOT ON {EXCEPTION | ESCAPE} 無条件文-2 ]

[END-ACCEPT]
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
   - ACCEPTREFRESH - ACCEPT 文の前に、画面節のデータに関連するデータ領域を作業場所節中の対応する項目によって更新するか否かを指定する。
   - XOPEN - ACCEPT 文の前に、画面節のデータに関連するデータ領域を作業場所節中の対応する項目によって更新するか否かを指定する。

構文規則

形式 1

1. 形式 1 の呼び名は、環境部の特殊名段落中の機能名に関連させる。 有効な機能名に関しては、***環境部***の章の*特殊名段落*節の中の一般規則 12 を参照。
2. OSVS VSC2 MF 機能名に関連する呼び名の代わりに、機能名そのものを使用できる。
3. MF XOPEN 呼び名が特殊名段落中の ARGUMENT-NUMBER と関連するときは、一意名は符号の付かない整数とする。
4. MF XOPEN 呼び名が特殊名段落中の ARGUMENT-VALUE または ENVIRONMENT-VALUE と関連するときは、一意名は英数字データ項目とする。
5. VSC2 MF 一意名は、USAGE DISPLAY-1 (2 バイト文字) 項目でも、外部浮動小数点数データ項目でもよい。

   ISO/IEC 1989:2002 MF 一意名は、内部浮動小数点数データ項目でもよい。

6. MF XOPEN EXCEPTION 指定は、FROM が ENVIRONMENT-NAME か ARGUMENT-VALUE と共に指定された時、またはこれと関連する呼び名で指定された時にだけ、指定することができる。

形式 2

7. OSVS VSC2 MF 一意名は、内部浮動小数点数データ項目でも、外部浮動小数点数データ項目でもよい。

形式 3

8. MF 一意名の字類、項類、用途には何の制限もない。しかし、一意名に入れられる実際の値およびその値を一意名に転記することの妥当性は、FROM 指定に左右される。(詳細については、形式 3 に関する一般規則を参照)。

形式 4

9. MF 画面名は、OCCURS 句に含まれる項目であってはならない。

形式 4 and 5

10. MF LINE 指定および COLUMN 指定は、どちらから先に書いてもよい。

11. (MF) EXCEPTION と ESCAPE は同義であり、どちらを書いてもよい。
12. (MF) 一意名-4 は、PIC 9 (4) または PIC 9 (6) のデータ項目とする。

(MF) 形式 5

13. 一意名-8 は整数とする。符号付きとすることができる。
14. 整数-7 は符号付きとすることができる。
15. 形式 5 に該当する ACCEPT 文は作用対象が画面名でなく、AT 指定、CRT オプションを伴う FROM 指定、WITH 指定、MODE IS BLOCK 指定、EXCEPTION 指定のどれかが指定されているもの、または FROM 指定は指定されていないが特殊名段落に CONSOLE オプション句が指定されているものである。CONSOLE オプション句を伴う FROM 指定が指定されているか、FROM 指定が指定されておらず特殊名段落に CONSOLE IS CRT 句も指定されていない ACCEPT 文は、形式 1 の ACCEPT 文として扱われる。
16. 一意名の後ろに続ける指定は、どのような順序で書いてもよい。
17. SPACE-FILL, ZERO-FILL, LEFT-JUSTIFY, RIGHT-JUSTIFY, PROMPT, TRAILING-SIGN オプションを指定できるのは、作用対象が基本項目の場合だけである。
18. 一意名-1 に属する基本項目の用途は、USAGE DISPLAY とする。
19. 一意名-1 に属する非基本項目は 8191 バイトより大きいことがある。MODE IS BLOCK 句を使用した場合、一意名-1 の全体の長さは 8191 バイト以下にすること。

一般規則

すべての形式

1. END-ACCEPT 指定は ACCEPT 文の範囲を区切る。(*COBOL 言語の概念*の章の*明示範囲符と暗黙範囲符*節を参照。) MF(4)コンパイラ指令を設定した場合にのみ、END-ACCEPT は予約語として扱われる。

形式 1

2. ACCEPT 文は論理装置または物理装置からデータを移送する働きをする。そのデータによって、一意名が指すデータ項目の内容が置き換えられる。一意名が指すデータ項目に用途として DISPLAY が明示的または暗黙的に指定されている場合、フォーマットの変換なしに直ちにその置換が行われる。そうでない場合には、正しいフォーマットへの変換が行われる。

   データ移送のサイズは装置とランタイム環境とによって決まる (使用可能な機能名のリストについては、*環境部*の章の*特殊名段落*節を、装置およびデータ移送サイズの制限の詳細については COBOL システムのマニュアルを、それぞれ参照) 。

   装置が受取り側データ項目と同じサイズのデータを移送できる場合には、移送されたデータが受取り側データ項目に収められる。そうでない場合は、下記のように扱われる。

a. 受取り側データ項目（または現在まだ移送されたデータによって占められていない部分）のサイズが移送されるデータのサイズを超える場合は、移送されたデータが受取り側データ項目（またはまだ空いている部分）中に左詰に収められ、さらにデータが要求される。
b. 移送されたデータのサイズが受取り側データ項目（または現在まだ移送されたデータによって占められていない部分）のサイズを超える場合は、移送されたデータの左端から受取り側データ項目（または残りの部分）に収まる分だけが受け取られる。移送されたデータのうちの受取り側データ項目に収まらない残りの部分は無視される。

FROM オプションを指定しなかった場合は、FROM CONSOLE と指定したのに等しい。

3. (MF)(XOPEN)COMMAND-LINE、またはそれに関連する呼び名を指定すると、受取り側データ項目はシステムのコマンド行バッファによって上書きされる。
4. (MF)(XOPEN)機能名 ARGUMENT-NUMBER に関連する呼び名を使用すると、一意名はコマンド行中に含まれる引数を受け取る。（これには、すべての引数が含まれる。つまり、プログラム名の前にある引数、プログラム名自体、プログラム名の後ろに続く引数が含まれる。したがって、プログラム呼出しの種類によっては移植性を期待できない）。
5. (MF)(XOPEN)呼び名が ARGUMENT-NUMBER に関連する時、一意名は現在のコマンド行の引数になる。どのコマンド行の引数を現在のものとするかの決定は、下記の通りである。
   a. はじめに、最初のコマンド行の引数を現在のものとする。
   b. ARGUMENT-NUMBER に関連する呼び名をもつ DISPLAY が、現在のコマンド行の引数の数を DISPLAY 文で指定された一意名または定数の値に設定する。
   c. ARGUMENT-NUMBER に関連する呼び名をもつ DISPLAY が最後に実行されてから、ARGUMENT-VALUE に関連する呼び名をもつ ACCEPT が実行された時は、現在のコマンド行の引数の数はこの ACCEPT 文が使用される前に増やされる。

現在の引数の数が 0 に設定されていると、これは、実行単位の主プログラムのプログラム名が返されることを意味する。しかし、プログラムの呼出しにいろいろな変形があるため、主プログラム名の代わりにユーティリティプログラムまたは呼ぶプログラムの名前が返されることがある。

ACCEPT 文の実行時に現在の引数の数が

(XOPEN)99 より大きいか、0 より小さいか、

コマンド行上の実際の引数の数よりも大きい場合、無条件文-1 が指定されていると、これが実行される。

6. (MF)(XOPEN)機能名 ENVIRONMENTAL-VALUE に関連する呼び名を使用した場合、下記のようになる。
   a. ENVIRONMENT-NAME に関連する呼び名をもつ DISPLAY が前に実行されていると、指定された環境変数の値が一意名に入れられる。

b. ENVIRONMENT-NAME に関連する呼び名をもつ DISPLAY が前に実行されていないか、または指定された環境変数が存在しないと、ON EXCEPTION 指定に指定されている無条件文が実行される。この場合、一意名の値はどうなるかわからない。

7. MF OPEN 他のプログラムによって呼ばれたプログラムの中で、コマンド行の引数および引数の数を検索する効果は、それらを実行単位中の最初のプログラムによって検索する場合と同じである。

形式 2

8. ACCEPT 文は、要求された情報を一意名によって指定されたデータ項目に取り込む。その際、MOVE 文の規則が適用される。DATE、DAY、

   ANS85 DAY-OF-WEEK、

   および TIME は概念的なデータであり、COBOL プログラム中には記述されていない。

9. 指定 YYYYMMDD を含まない DATE は、(1 世紀の中の) 年、(1 年の中の) 月、(1 ヶ月の中の) 日のデータ要素から構成される。指定 YYYYMMDD を含まない DATE は、COBOL プログラムからアクセスされると、用途が DISPLAY で、6 桁の符号なしの基本整数データ項目として記述されたものとして動作する。このデータ項目の文字位置 (左から右に) は以下の通り。

文字位置	内容
1-2	グレゴリオ歴 (西暦) の年号の下 2 桁の数字
3-4	1 年の範囲内で、月を表す 01 から 12 までの 2 桁の数字。
5-6	1 ヶ月の範囲内で、日を表す 01 から 31 までの 2 桁の数字。

10. 指定 YYYYMMDD を含む DATE は、グレゴリオ歴の年号、(1 年の中の) 月、(1 月の中の) 日のデータ要素から構成される。指定 YYYYMMDD を含む DATE は、COBOL プログラムからアクセスされると、用途が DISPLAY で、8 桁の符号なしの基本整数データ項目として記述されたものとして動作する。このデータ項目の文字位置 (左から右に) は以下の通り。

文字位置	内容
1-4	グレゴリオ歴の年号の 4 桁の数字
5-6	1 年の範囲内で、月を表す 01 から 12 までの 2 桁の数字。
7-8	1 ヶ月の範囲内で、日を表す 01 から 31 までの 2 桁の数字。

11. 指定 YYYYDDD を含まない DAY は、(1 世紀の中の) 年、(1 年の中の) 日のデータ要素から構成される。指定 YYYYDDD を含まない DAY は、COBOL プログラムからアクセスされると、用途が DISPLAY で、5 桁の符号なしの基本整数データ項目として記述されたものとして動作する。このデータ項目の文字位置 (左から右に) は以下の通り。

文字位置	内容
1-2	グレゴリオ歴の年号の下 2 桁の数字
3-5	1 年の範囲内で、日を表す 01 から 366 までの 3 桁の数字。

12. 指定 YYYYDDD を含む DAY は、グレゴリオ歴の年号および (1 年の中の) 日のデータ要素から構成される。指定 YYYYDDD を含む DAY は、COBOL プログラムからアクセスされると、用途が DISPLAY で、7 桁の符号なしの基本整数データ項目として記述されたものとして動作する。このデータ項目の文字位置 (左から右に) は以下の通り。

文字位置	内容
1-4	グレゴリオ歴の年号の 4 桁の数字
5-7	1 年の範囲内で、日を表す 01 から 366 までの 3 桁の数字。

13. ANS85 DAY-OF-WEEK は、曜日を表わす単一のデータ要素から構成される。COBOL プログラムから DAY-OF-WEEK を呼び出すと、そのプログラムの中で、1 桁の符号の付かない整数の基本データ項目として記述してあるように取り扱うことができる。DAY-OF-WEEK は 1 から 7 までの値をとり、それぞれ月曜日から日曜日を表わす。
14. TIME は、時間、分、秒、百分の 1 秒の 4 つの要素から構成される。この時間は夜中の 0 時から 24 時間制で測られる。たとえば、午後 2 時 41 分は 14410000 と表わされる。COBOL プログラムから TIME を呼び出すと、そのプログラムの中で、符号の付かない 8 桁の整数の基本データ項目として記述してあるように取り扱うことができる。TIME の最小値は 00000000 であり、最大値は 23595999 である。 ハードウエアの時間測定の精度が上記よりも低い場合は、得られた値は最も近い近似値に変換される。

形式 3

15. MF ACCEPT FROM LINE NUMBER 句から返される値は、つねに数字である。この値は、使用しているシステムによって異なる。
16. MF FROM USER NAME オプションは、UNIX システム上のユーザー識別番号を返す。このような概念のないシステム上では、このオプションは空白を返す。
17. MF FROM ESCAPE KEY オプションは、終了キーによって発生させられた 2 文字のコードを返す。
18. MF EXCEPTION STATUS 項目には、3 桁のコードが入っている。このコードは、CALL 文を実行中に発生した、例外条件の種類を識別する。

    EXCEPTION STATUS を調べる場合は、CALL 文の直後で行うのがよい。 CALL と ACCEPT FROM EXCEPTION STATUS との間には、何もあってはならない。たとえば、その間にファイル入出力をはさむと、EXCEPTION STATUS が変えられてしまうので、その内容を調べる意味がなくなってしまう。.

形式 4

19. MF ACCEPT 文のこの書き方は、画面節中に定義されている画面項目を受け取って、拡張画面操作機能によって全面的に処理できるようにする。

形式 4 および 5

20. (MF) ACCEPT 文の実行順序は、つねに以下のとおりである。
    1. AT 指定
    2. BLANK 指定
    3. コンパイラ指令 ACCEPTREFRESH または XOPEN のどちらかが指定されていると、USING 指定で現在の画面項目の内容が表示される。
    4. BELL 指定
    5. ACCEPT 操作
21. (MF) AT 指定は、ACCEPT 処理を開始する画面上の絶対番地を指定する。
22. (MF) 整数-3 または整数-4 の長さが 4 桁ならば、上 2 桁は行を表わし、下 2 桁はカラムを表わす。整数-3 の長さが 6 桁ならば、上 3 桁は行を表わし、下 3 桁はカラムを表わす。
23. (MF) 行番号とカラム番号の組合わせの中には、特殊な意味をもつものがある。それらは下記のとおり。
    a. カラムの値が 1 行のカラム数よりも大きいと、その値から 1 行のカラム数が差し引かれ、行数に 1 が加えられる。カラムの値が 1 行のカラム数の範囲内におさまるまで、この処理が繰り返される。
    b. 行の値が画面の範囲から外れると、画面が 1 行上にスクロールされる。この効果は、画面の下端の行を指定したのと同じである。
    c. 与えられた行番号とカラム番号がともにゼロであると、形式 4 または 5 の前の ACCEPT 処理が終わった位置の直後から、ACCEPT が開始される。各行のカラム 1 は、前の行の最後のカラムに続くものとみなされる。
    d. 指定された行番号がゼロであるがカラム番号はゼロでないと、形式 4 または 5 の前の ACCEPT 処理が終わった位置の次の行の指定されたカラムから、ACCEPT が開始される。
    e. 指定されたカラム番号がゼロであるが行番号はゼロでないと、形式 4 または 5 の前の ACCEPT 処理が終わった位置の次のカラムの指定された行から、ACCEPT が開始される。
24. (MF) この ACCEPT 文を実行する前に、この ACCEPT 文に指定されたのと同じ画面名または一意名-1 が指定されている DISPLAY 文が実行されていなければならない。それ以降に何らかの ACCEPT 文または DISPLAY 文が実行されていてはならない。
25. (MF) ON EXCEPTION 指定を書くと、ACCEPT 処理が正常終了しなかった場合に、無条件文-1 が実行される。NOT ON EXCEPTION 指定を書くと、ACCEPT 処理が正常終了した場合に、無条件文-2 が実行される。(終了の種類については、*環境部*の章の *特殊名段落*節の CRT STATUS 句の規則を参照。)

形式 5

(MF)

27. AT 指定を指定しないと、ACCEPT 処理は行 1、カラム 1 から開始される。
28. MODE IS BLOCK 指定を指定しないと、一意名が集団項目であれば、それに属する基本項目で名前が FILLER でないものが受け取られる。受け取られる項目は、データ部内でそれらが記述されている順に画面に配置され、集団内の FILLER の長さによっ

て区切られる。この用途では、ある行の最初の位置は、その前の行の最後の位置の直後に続くものとみなされる。項目は、同じ順序で受け取られる。

CURSOR IS 句（*環境部*の章の*特殊名段落*節の CURSOR IS 句の規則を参照）内で別途指定しないかぎり、カーソルは最初の項目の始点に位置付けられる。各項目への ACCEPT 処理が終わるにつれて、カーソルは次の項目の始点に移動される。

29. 一意名-1 が集団項目であり、その下位に可変反復データ項目がある場合、ACCEPT 文は MODE IS BLOCK 句が指定されているかのように動作する。
30. 一意名-1 が集団項目であり、その下位に可変反復データ項目がある場合、ACCEPT 文は MODE IS BLOCK 句が指定されているかのように動作する。
31. MODE IS BLOCK 指定は、一意名を基本項目として扱うように指定する。こうすると、一意名が集団項目であっても、1 つの項目として表示される。
32. PROMPT オプションを指定した効果は、*データ部 - 画面節*の章の *PROMPT(プロンプト)句*節で解説している。
33. PROMPT オプションを指定しないと、空の文字位置を示すための文字は表示されない。CHARACTER オプションを抜かして PROMPT を指定すると、構成時にプロンプト用に設定されている文字が文字位置の表示に用いられる。
34. (MF) WITH 指定を用いると、ある種のオプションを ACCEPT 処理の実行時に指定できる。（これらのオプションの説明については、*データ部 - 画面節*の章を参照。）

画面記述句に指定できるオプションの他に、WITH 指定にはいくつかのオプションが加えられている。SPACE-FILL, ZERO-FILL, LEFT-JUSTIFY, RIGHT-JUSTIFY, TRAILING-SIGN, UPDATE である。ZERO-FILL はこのリストにも画面記述句にも現れている。これは、ZERO-FILL には 2 つの用法があるからである。2 番目の用法については、この章で後述する。

自由方式で、数字および数字編集の画面項目にデータを入力できるようにする構成オプションがある。COBOL では、非編集数字データ項目は、内部形式でデータを保持することを目的としたものである。しかし、この形式を用いると、そのようなデータ項目を画面上に表示できる（詳細は、ユーザーインターフェイスに関する COBOL システムのマニュアルを参照のこと）。自由方式がとられているときは、データは自動的に下記のように編集されて表示される。

- 仮想小数点を終止符で表わす。
- 符号を符号文字で表わす（負の数は"-" で、正の数は空白で）。符号は左端の数字の直前に付けられる。
- すべての整数文字位置の先行ゼロが抑制される。ただし、最末桁を除く
- 左に桁寄せされる。

SPACE-FILL、ZERO-FILL、LEFT-JUSTIFY、RIGHT-JUSTIFY、および TRAILING-SIGN オプションを使用すると、上記の形式を補正できる。

35. SPACE-FILL オプションは、自由方式の非編集数字データ項目の、すべての整数文字位置の先行ゼロを抑制して、画面に表示するようにする。このオプションは、自由方式の非編集数字データ項目に対してだけ働く。この機能は、データ項目中の初期

データが表示されるときに効力を発揮し、そのデータ項目への ACCEPT 処理が終了するときにも再び効力を発揮する。左端に付された符号があれば、右端に表示される。

36. ZERO-FILL オプションは、自由方式の非編集数字データ項目をゼロ抑制しないで、画面に表示するようにする。この機能は、データ項目中の初期データが表示されるときに効力を発揮し、そのデータ項目への ACCEPT 操作が終了するときにも再び効力を発揮する。(英字または英数字のデータ項目にこのオプションを適用したときの効果については、*データ部 - 画面節*の章の *ZERO-FILL(ゼロ補填)句*節を参照)。
37. LEFT-JUSTIFY オプションは、注記にすぎない。
38. RIGHT-JUSTIFY オプションは、入力されたデータを項目の右に桁寄せして、画面に表示するようにする。このオプションは、自由方式の非編集数字データ項目に対してだけ働く。この機能は、データ項目中の初期データ (表示された現在の内容) が表示されるときに効力を発揮し、そのデータ項目への ACCEPT 処理が終了するときにも再び効力を発揮する。
39. TRAILING-SIGN オプションは、演算符号を項目の右端の文字位置に表示するようにする。この機能は、データ項目中の初期データが表示されるときに効力を発揮し、そのデータ項目への ACCEPT 処理が終了するときにも再び効力を発揮する。このオプションは、自由方式の非編集数字データ項目に対してだけ働く。
40. UPDATE オプションは、データ項目の現在の内容 (初期データ) を表示してから、新しいデータの入力を促すようにする。新しいデータが何も入力されないと、初期データが入力されたかのように扱われる。UPDATE オプションを指定しないと、初期データが表示されるか否かは構成オプションによる (構成オプションの詳細については、ユーザーインターフェイスに関する COBOL システムのマニュアルを参照のこと)。
41. UPPER オプションは、入力された文字を強制的に大文字にする。
42. LOWER オプションは、入力された文字を強制的に小文字にする。
43. 一意名-1 に REDEFINES がかかっている場合、再定義データ領域の最初の記述が使用され、以降の記述は無視される。OCCURS または入れ子になった OCCURS が指定されている場合、反復されるデータ項目はその反復回数だけ展開される。したがって、1 つの定義が多数の項目にわたって繰り返されることになる。
44. 一意名-8 または整数-7 の値が負の場合、時間切れによる"例外条件"を発生させないという要求を表わす。この場合、キーが打鍵されるまでの時間またはキーの打鍵間隔がどんなに長くなっても、例外条件は発生しない。
45. 一意名-8 または整数-7 の値がゼロの場合、入力待ちの間は時間切れを起こさせないことを意味する。しかし、入力待ちでない (ACCEPT が処理された) ときには、直ちに時間切れとする。
46. ON EXCEPTION 指定を書くと、TIME-OUT 指定が指定してあって時間切れが発生したときに実行される。NOT ON EXCEPTION 指定を書くと、TIME-OUT 指定が指定してあるが時間切れ (または他の例外条件) が発生しなかったときに実行される。
47. 時間切れ例外条件が発生すると、ACCEPT の結果の項目は下記のようになる。
    a. 項目の一部が既に変更されていると、その時点の内容がそのまま保持される。
    b. "FULL"や"REQUIRED" をはじめとする、通常は部分的な入力を許さない属性をもつ項目でも、時間切れ例外条件が発生したときは、その要件を満たすことを求められない。
    c. 上記の規則は、キーを打つたびに時間切れ測定用の時計が再設定されるか否かにかかわらず、適用される。
48. 実行時に百分の 2,147,483,647 秒よりも大きな正の時間切れ間隔が検出されると、その値は百分の 2,147,483,647 秒 (約 8 カ月) に設定し直される。

49. TIME-OUT の値は、ACCEPT 文の実行が開始されてから時間切れ例外条件が発生するまでの時間を、秒（または十分の一秒）単位で指定する。新しい ADISCF 構成オプションは、キーボード上で何か操作が行われるたびに、時間切れ測定用の時計を"再設定"するか否かを制御する。ACCEPT 文によって時間切れ測定用の時計を再設定したりしなかったりする応用では、設定状態の変更を必要とする ACCEPT 文の前または後ろで、ADIS 実行時インターフェイスを個別に呼ぶことができる。たとえば、時間切れ測定用の時計を再設定しないように ADIS が設定されているときに、次の文を実行したとする。

```
ACCEPT INPUT-FIELD TIME-OUT AFTER +10
```

すると、ACCEPT の開始から 10 秒経過後に、時間切れ条件が発生する。この場合、ACCEPT の開始 5 秒後に何か文字を打ったとしても、時間切れ条件の発生を防ぐことにはならない。

また、時間切れ測定用の時計を再設定するように ADIS が設定されているとする。すると、上記の同じ文を実行した場合、何か文字を打つたびに TIMEOUT"時計"はゼロに設定し直される。

50. ON EXCEPTION 指定を指定しないと、時間切れ条件が発生したときに、CRT 状態キー（指定してあれば）が更新され、処理は次の論理的な指定まで続行される。このとき、ACCEPT の受取り側項目の内容は上に説明したとおりである（このことは、NOT ON EXCEPTION 指定を指定した場合にも、当てはまる）。
51. （NOT）ON EXCEPTION 指定に関する説明はすべて、（NOT）ON ESCAPE 指定にも当てはまる。

形式 1 および 5

52. (MF) 英数字データ項目に関しては、ACCEPT 文の実行中に画面から受け取られたフィールドのサイズは、宛先のフィールドのサイズとまったく同じである。したがって、フィールドの右側の部分にデータを表示したい場合には、その位置にカーソルを合わせてデータを入力しなければならない。

12.2 ADD 文

ADD（加算）文は、2 つ以上の作用対象の和をとって、その結果を格納する。

一般形式

形式 1

```
ADD {一意名-1 | 定数-1} ... TO {一意名-2 [ROUNDED]} ...

    [ON SIZE ERROR 無条件文-1]

    [NOT ON SIZE ERROR 無条件文-2]
    [END-ADD]
```

ANS85

形式 2

```
ADD {一意名-1 | 定数-1} ... TO
```

ANS85 OSVS

```
    {一意名-2 | 定数-2} GIVING {一意名-3 [ROUNDED]} ...

    [ON SIZE ERROR 無条件文-1]

    [NOT ON SIZE ERROR 無条件文-2]
    [END-ADD]
```

ANS85

形式 3

```
ADD {CORRESPONDING | CORR} 一意名-1 TO 一意名-2 [ROUNDED]

    [ON SIZE ERROR 無条件文-1]

    [NOT ON SIZE ERROR 無条件文-2]
    [END-ADD]
```

ANS85

構文規則

すべての形式

1. 形式 1 および 2 において、各一意名は必ず基本数字項目を参照すること。ただし、形式 2 においては、語 GIVING の後ろに続く各一意名は、基本数字項目の他に基本数字編集項目を参照してもよい。形式 3 においては、各一意名は、必ず集団項目を参照すること。
2. 各定数は、数字定数とする。
3. 作用対象の合成は、以下の規則に従う（*手続き部*の章の*算術文*節を参照）。
    a. 形式 1 では、作用対象の合成に、指定したすべての作用対象が用いられる。
    b. 形式 2 では、作用対象の合成には、語 GIVING の後ろに続くデータ項目を除く、指定したすべての作用対象が用いられる。

c. 形式 3 では、作用対象の合成は、対応するデータ項目の組ごとに別々に行われる。

4. 形式 3 では、CORR は CORRESPONDING の省略形である。
5. (OSVS)(VSC2)(MF)数字データ項目または定数を指定できるところではどこも、浮動小数点数定数および浮動小数点数データ項目を使用できる。

一般規則

すべての形式

1. ***手続き部***の章の *ROUNDED 指定*、*ON SIZE ERROR 指定*、*CORRESPONDING 指定*、*算術文*, *作用対象の重なり*、および*算術文における複数個の結果*の各節、***COBOL 言語の概念***の章の*明示範囲符と暗黙範囲符*節、および***翻訳集団の概念***の章の*範囲明示文*節を参照。
2. 形式 1 では、語 TO の前にある作用対象の値の和がとられ、その値が一意名-2 の現在の値に加えられて、その結果が直ちに一意名-2 に入れられる。語 TO の後ろに続く各作用対象に対して、この処理が左から右方向に繰り返される。
3. 形式 2 では、語 GIVING の前にある作用対象の値の和がとられ、その値が結果の一意名である一意名-3 によって参照される各データ項目の新しい値として格納される。
4. 形式 3 では、一意名-1 に属するデータ項目が、一意名-2 に属する対応するデータ項目に加算されて格納される。
5. (MF)計算過程で意味のある桁が失われることのないように、COBOL コンパイラによって十分な作業領域が確保される。

12.3 ALTER 文

ALTER (変更) 文は、あらかじめ定められている処理の順序を変更する。

(ANS85)ALTER 文は、ANSI'85 標準では廃要素に分類されており、ANSI 標準の次回の全面改訂の際に、削除される予定である。

(MF)この構文は、Micro Focus COBOL に組み込まれているすべての方言で全面的に使用できる。FLAGSTD 指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。

(XOPEN)準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、この動詞は明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内では、この文を使用するべきではない。

一般形式

```
ALTER 手続き名-1 TO [PROCEED TO] 手続き名-2
      [手続き名-3 TO [PROCEED TO] 手続き名-4] ...
```

構文規則

1. 各手続き名-1、手続き名-3 などは、DEPENDING 指定が含まれない単一の GO TO 文からなる段落の名前である。
2. 各手続き名-2、手続き名-4 などは、手続き部内の段落または節の名前である。

一般規則

1. ALTER 文を実行すると、手続き名-1 や手続き名-3 などに指定された段落中の GO TO 文がそれ以降に実行されたとき、それぞれ手続き名-2 や手続き名-4 などに制御が移されるようになる。独立の区分内の変更された GO TO 文は、状況によっては、初期状態に戻されることがある。(詳細については、*言語リファレンス - 追加トピック*中の*区分化の独立区分*節を参照。)
2. 区分番号が 50 以上の節内の GO TO 文を、区分番号の異なる節内の ALTER 文によって参照してはならない。

   ALTER 文のそれ以外の用法は、すべて有効である。たとえば、手続き名-1 や手続き名-3 などがオーバレイ可能な固定区分の中にあっても構わない。

12.4 CALL (呼出し) 文

CALL (呼出し) 文は、実行単位中で他の実行用プログラムに制御を移させる。

ISO/IEC MF 対象とする実行用プログラムの手続きを、番地を指定することによって、間接的に呼ぶことができる。データとデータへのポインタと手続きへのポインタを、パラメータとして呼ばれるプログラムに引き渡したり、呼ばれるプログラムから返してもらったりできる。パラメータ受渡し方式を制御することによって、COBOL 以外の言語で書かれたプログラム手続きを呼ぶことができる。

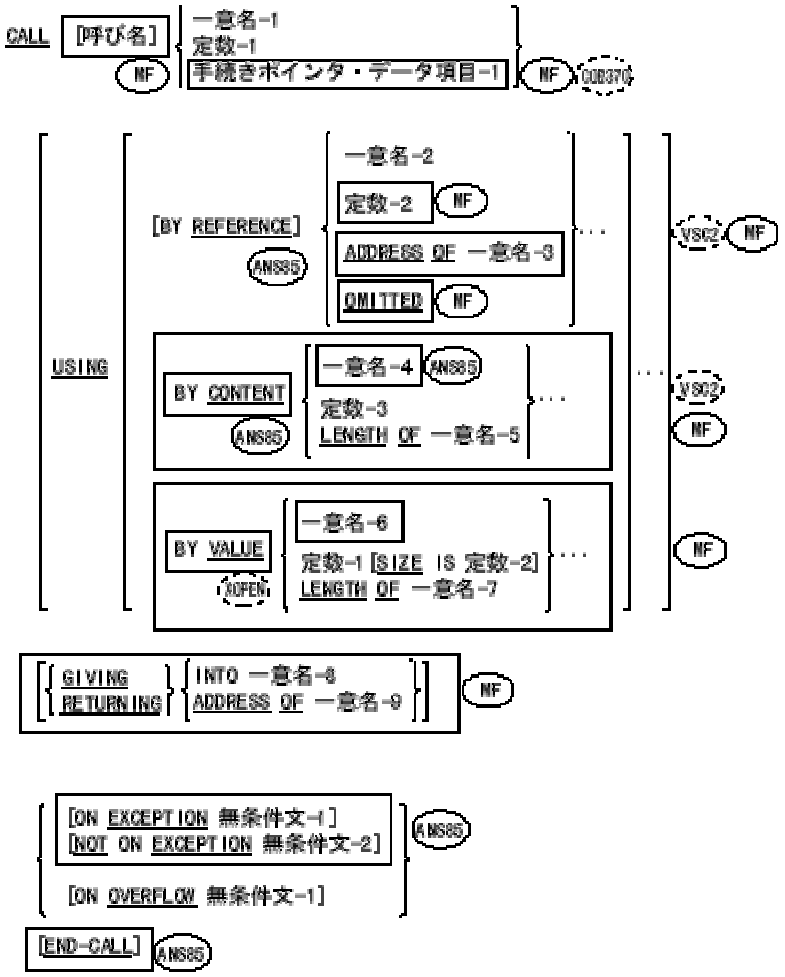
OS/390 ISO/IEC MF 呼ぶばれた COBOL プログラムは、再帰属性を持っていれば、再帰的に自分を呼ぶことができる。

例

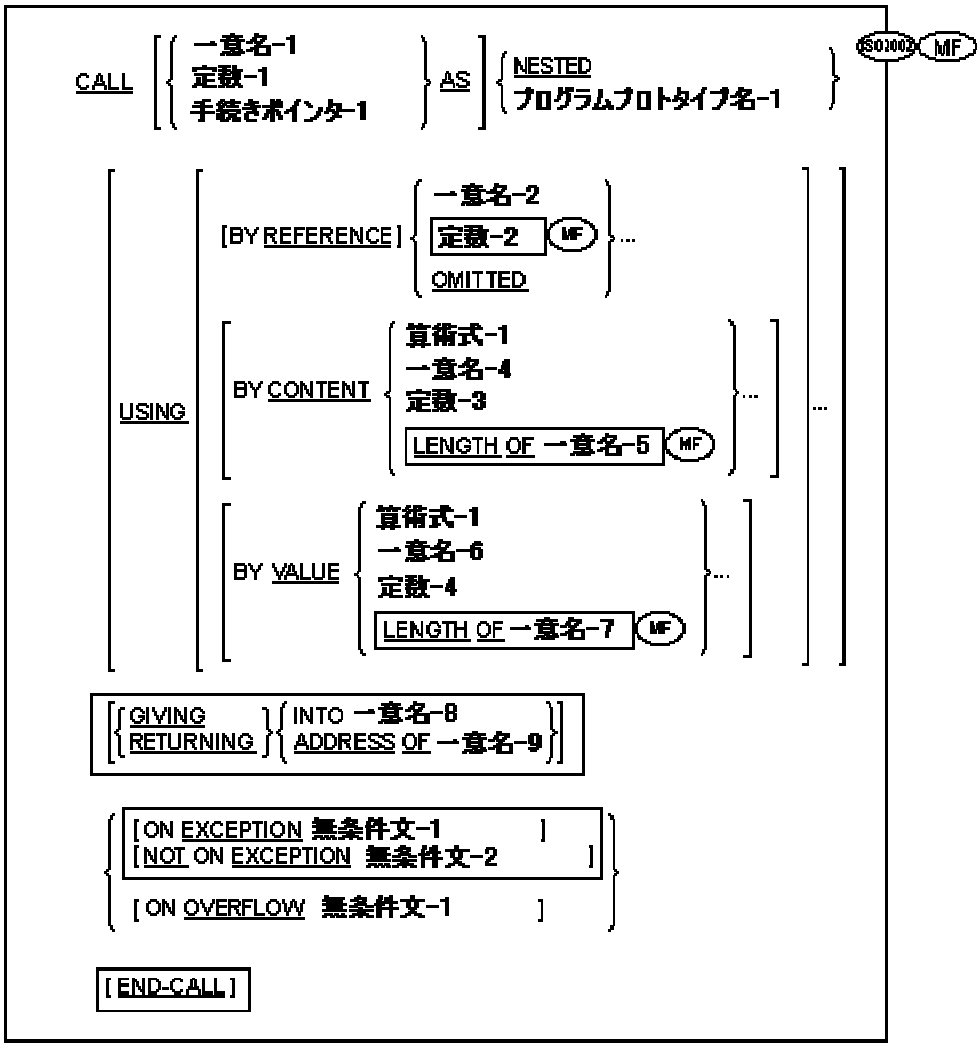
1. CALL 文の手続きポインタの使用例は、*言語リファレンス - 追加トピック*中の*例*の章の*手続きポインタの呼出しと設定*節に掲載されている。

一般形式

形式 1



形式 2 (プログラムプロトタイプ)



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
   - ALIGN - 01 レベルまたは 77 レベルのデータ項目を割り付けるメモリーの境界を指定する。
   - CASE - 大文字と小文字を等しいとみなすかどうかを指定する。
   - DEFAULTCALLS - 省略時の呼出し方式を指定する。
   - DYNAM - CALL 定数文によって呼び出されたプログラムを取り消しできるかどうかを指定する。
   - FOLD-CALL-NAME - 呼ぶプログラムと呼ばれるプログラムの間でプログラム名に関して同じ大文字と小文字の使い方が同じでないときに、呼ばれた副プログラムが見つかるように保証するか否かを指定する。
   - IBMCOMP - 語記憶モードを設定する。
   - MAPNAME - プログラム名中の英字以外の文字の取扱いに影響する。

構文規則

星印 (*) の付いている構文規則は CALL 文および CHAIN 文に共通である。それらの規則を「CALL 文」または「CHAIN 文」に適用するさいには、「CALL 文」または「CHAIN 文」を読むべきである。

すべての形式

1. * 一意名-1 は英数字データ項目、

   (ISO/MF)(MF)または各国語型データ項目

   として定義しなければならない。

2. * 定数-1、

   (MF)定数-2、

   (VSC2)(MF)および定数-3

   は英数字定数、

   (ISO/MF)(MF)または各国語型定数

   でなければならず、表意定数であってはならない。

3. * 一意名-2 および一意名-4 はファイル節、作業場所節、

   (ISO/MF)(MF)局所記憶節、

   通信節、連絡節のいずれかにデータ項目として定義されていなければならず、レベル 01 またはレベル 77 のデータ項目、

   (ANS85)または基本項目

   (ISO/MF)(MF)または任意の集団項目

   でなければならない。

   (ISO/MF)(MF)一意名-2 は、ファクトリオブジェクトまたはインスタンスオブジェクトの作業場所節またはファイル節で定義されてはならない。

4. (MF)* 一意名-5、一意名-6、一意名-7、一意名-8 はいずれも、関数一意名であってはならない。
5. (MF)* GIVING 句および RETURNING 句は、どちらを使用してもよい。
6. (MF)一意名-9 は、連絡節中の 01 レベルまたは 77 レベルのデータ項目とする。
7. (ISO/MF)(MF)*ANY LENGTH 句を使用して、一意名-2、一意名-4、一意名-6、および一意名-8 を記述してはならない。

形式 1

8. (MF)呼び名が必要なのは、呼び出す対象の実行用プログラムで用いられている呼出し方式と、COBOL システムの省略時の呼出し方式とが異なる場合だけである。通常は、COBOL の省略時の呼出し方式は、COBOL 以外の言語の主要な処理系のランタイム環境で使用されている方式と整合性がある。

   (MF)呼び名は特殊名段落中に定義しなければならない。その方法の詳細については、*環境部*の章の*特殊名段落*節を参照。また、ランタイム環境においてサポートされている呼出し方式の詳細については、COBOL のシステムのマニュアルを参照。

9. (MF)* 整数-1 は符合付きであってよい。
10. (VSC2)(MF)* 一意名-3 は連絡節、

    (MF)局所記憶節または作業場所節

    にデータ項目として定義されていなければならず、

    (VSC2)(MF)レベル 01 またはレベル 77 のデータ項目、

    (MF)またはその他のレベルのデータ項目

    でなければならない。

11. (MF)CALL 文に定数-1 を指定し（ただし、CALL 文に一意名-1 または手続きポインタ-1 を指定した場合を除く）、かつ現在の翻訳単位中に呼出しプロトタイプ（プログラム名段落中に EXTERNAL 句があるプログラム）が含まれていてその名前が定数-1 に合致する場合、構文チェックの間に下記の項目の妥当性が検査される。
    - 必要なパラメータの数
    - パラメータの型
    - 呼出し方式
12. (SO2002)*一意名-2 は、クラスオブジェクトまたはポインタのデータ項目であってはならない。

形式 2

(SO2002)(MF)

13. NESTED 指定は、プログラム定義以外で使用してはならない。
14. NESTED 指定を使用した場合は、定数-1 を指定しなければならない。定数-1 は、*COBOL 言語の概念*の章の*プログラム名に関する表記法*節で指定されたように、共通プログラムのプログラム名段落中で指定されたプログラム名と同じでなければならない。または、呼ぶプログラムの直下に含まれるプログラムのプログラム名段落中で指定されたプログラム名と同じでなければならない。
15. プログラムプロトタイプ名-1 は、リポジトリ段落中のプログラム指定子で指定しなければならない。

16. 引数について BY CONTENT 指定または BY REFERENCE 指定を使用した場合は、手続き部見出し中の対応する仮パラメータに対して、BY REFERENCE を指定しなければならない。
17. 一意名-4 が受取り側作用対象として許可された一意名であるときに、BY CONTENT を省略してはならない。ただし、一意名-4 がオブジェクトプロパティである場合は、BY CONTENT を省略してもよい。
18. 引数について BY VALUE 指定を使用した場合は、手続き部見出し中の対応する仮パラメータに対して、BY VALUE 指定を使用しなければならない。
19. 一意名-6 の字類は、数字、オブジェクト、またはポインタでなければならない。
20. 定数-4 は数字定数でなければならない。
21. OMITTED 指定を使用した場合は、手続き部見出し中の対応する仮パラメータに対して、OPTIONAL 指定を使用しなければならない。
22. 適合規則は、***手続き部***の章の*パラメータと戻り項目の適合*節で指定されたものが適用される。

一般規則

星印 (*) の付いている構文規則は CALL 文および CHAIN 文に共通である。それらの規則を「CALL 文」または「CHAIN 文」に適用するさいには、「CALL 文」または「CHAIN 文」を読むべきである。

すべての形式

1. 定数-1、または一意名-1 によって参照されるデータ項目の内容は呼ばれるプログラム

   (MF)または呼ばれる手続きの名前である。

   CALL 文を含むプログラムは呼ぶプログラムである。呼ばれるプログラムが COBOL プログラムである場合、定数-1、または一意名-1 によって参照されるデータ項目の内容はプログラム名であって、呼ばれるプログラムのプログラム名段落に記述されているものでなければならない (***見出し部***の章の*プログラム名段落*節を参照。特に大文字と小文字の区別について) 。

   (MF)あるいは、定数-1、または一意名-1 によって参照されるデータ項目の内容は入口名であって、呼ばれるプログラムの ENTRY 文中に記述されているものでなければならない。しかも、呼ばれるプログラムはすでに呼び出されており、その後で取り消されていてはならない。別の方法として、呼ばれるプログラムの名前はそのプログラムの実行プログラム用のコードが収められているファイルを指す名前であってもよい。2 つの CALL 文の間でまたは CALL 文と CANCEL 文の間で同じファイルを指す別の名前を指定した場合にどのような結果になるかは、オペレーティング･システムによって異なる。呼ばれたプログラム、プログラム･ファイル、あるいはライブラリ･ファイル中に保持されているプログラムと指定された名前の照合方法の詳細については、インターフェイスに関する COBOL システムのマニュアルを参照。

(MF)呼ばれるプログラムが COBOL プログラムではない場合のプログラム名または手続き名の構成に関する規則が、インターフェイスに関する COBOL システムのマニュアルに記載されている。

2. (MF)手続きポインタ-1 によって参照されるデータ項目の内容は呼ばれるプログラムの番地であり、そのプログラムはすでにメモリーにロードされていなければならない。呼ばれるプログラムが COBOL プログラムである場合は、呼ばれるプログラムの番地が当初に ENTRY 指定を伴う SET 文を用いて導出されており、その後でそのプログラムに対して CANCEL 文が実行されていてはならない。呼ばれるプログラムが COBOL プログラムではない場合は、呼ばれるプログラムの番地が当初に COBOL 以外のプログラムまたは手続きへの CALL を通じて導出されていなければならない。番地が NULL であるプログラムを CALL しようとすると、エラーとなる。
3. CALL 文が実行されたときに、その対象のプログラムが実行可能な状態になると、そのプログラムに制御が移される。呼ばれたプログラムから制御が戻ると、ON OVERFLOW 指定や

   (ANS85)ON EXCEPTION 指定

   は、指定されていても無視され、制御は CALL 文の末尾に移される。

   (ANS85)あるいは、NOT ON EXCEPTION 指令が指定されていると、制御は無条件文-2 に移される。制御が無条件文-2 に移されると、その無条件文-2 の中に指定されている各文に関する規則に従って、プログラムの実行が続けられる。制御の明示移行を引き起こす手続き分岐文または条件文が実行されると、その文に関する手続きに従って制御が移される。これ以外の場合は、無条件文-2 の実行が終了すると制御は CALL 文の末尾に移される。

4. CALL 文が実行されたときに、対象のプログラムがすでにメモリーにロードされていると、そのプログラムは実行可能な状態にされる。そのプログラムがまだメモリーにロードされていないと、そのプログラムを動的にロードする試みがなされる（インターフェイスに関する COBOL システム･ドキュメンテーションを参照）。対象のプログラムの所在を動的に把握でき、メモリーに連続した十分な空き領域があると、そのプログラムは動的にロードされて実行可能な状態にされる。

   CALL 文に指定されたプログラムがメモリーにロードされていないか、動的なロードに利用できないか、あるいは動的なロードに利用できるがメモリーに十分な空き領域がない場合には、その時点ではそのプログラムを実行可能にすることはできない。その場合、下記の 2 つのうちのどちらかの措置が取られる。

   a. CALL 文の中に ON OVERFLOW

      (ANS85)または ON EXCEPTION

      指定がある場合には、無条件文-1 に制御が移される。それから、無条件文-1 に指定されている各文の規則に応じて、処理が続行される。制御を明示的に移す手続き分岐または条件文が実行されると、その文の規則に応じて制

御が移される。それ以外の場合には、無条件文-1 の実行が完了すると、CALL 文の最後に制御が移され、

(ANS85)NOT ON EXCEPTION 指定はあっても無視される。

b. (ANS85)ON OVERFLOW 指定または ON EXCEPTION 指定がない場合、NOT ON EXCEPTION 指定はあっても無視される。

(MF)実行単位中の開かれているファイルに関して、明示的な CLOSE 文なしにプログラムの実行が停止され、ランタイム･エラーが表示される。

5. (ANS85)実行単位内の 2 つ以上のプログラムに、同じプログラム名をもたせることができる。呼ぶプログラム内でこのようなプログラム名を参照したとき、どちらのプログラムを指すかは、プログラム名の範囲の適用規則に従う。*COBOL 言語の概念*の章の*プログラム名の規則*節を参照。たとえば、CALL 文の中に指定されたプログラム名と同じ名前のプログラムが、実行単位中に 2 つあるとする。
   a. それらの 2 つのプログラムのうちの 1 つは、その CALL 文が含まれる独立のプログラムに直接的または間接的に含まれるか、またはその CALL 文が含まれるプログラム自体が、直接的または間接的に含まれる独立のプログラムに直接的または間接的に含まれなければならない。かつ、
   b. もう 1 つのプログラムは、別の独立したプログラムでなければならない。

   (ANS85)この例で、使用される仕組みは、下記のとおり。

   c. CALL 文中に指定された名前をもつ 2 つのプログラムの一方が、その CALL 文が含まれるプログラム中に直接的に含まれるならば、そのプログラムが呼ばれる。
   d. CALL 文中に指定された名前をもつ 2 つのプログラムの一方が共通属性をもち、その CALL 文が含まれるプログラムが直接的または間接的に含まれる独立のプログラムに直接的に含まれるならば、その共通プログラムが呼ばれる。ただし、呼ぶプログラムがその共通プログラムに含まれる場合は例外である。
   e. それ以外の場合は、別のプログラムが呼ばれる。
6. 呼ばれたプログラムが COBOL プログラムで、初期属性をもたない場合、そのプログラムおよびその中に直接的または間接的に含まれる各プログラムは、実行単位の中で最初に呼ばれたとき、および呼ばれたプログラムが取り消された後ではじめて呼ばれたときに、初期状態に置かれる。

   呼ばれたプログラムが COBOL プログラムでない場合、実行単位の中で最初に呼ばれたときにだけ初期状態に置かれる。このようなプログラムの取り消しは効果がない。

   それ以外の場合に呼ばれたときは、呼ばれるプログラムおよびその中に直接的または間接的に含まれる各プログラムの状態は、前回に呼ばれたときのままである。COBOL 以外の言語の処理系の中には、意味合いが違っているものがある。可能な場合には、インターフェイスに関する COBOL システムのマニュアルに、その詳細が記載されている。

7. (ANS85)呼ばれたプログラムが初期属性をもつ場合、そのプログラムおよびその中に直接的または間接的に含まれる各プログラムは、実行単位の中で呼ばれるたびに初期状態に置かれる。呼ばれたプログラムが初期状態にあるか使用され終わった状態にあるかは、DYNAM コンパイラ指令の設定に左右される。
8. (ANS85)呼ばれるプログラムの内部ファイル結合子に関連するファイルは、プログラムが初期状態にあるときは、まだオープンされていない。前述の *プログラムの初期状態*の節を参照。それ以外の場合に呼ばれたときは、それらのすべてのファイルの状態と位置付けは、前回に呼ばれたときのままである。
9. (ANS85)プログラムを呼ぶ過程または呼んだプログラムから戻る過程で、外部ファイル結合子に関連するファイルの状態と位置付けが変えられることはない。
10. 呼ばれるプログラムが COBOL プログラムである場合、その手続き部の見出し

    (OSVS)(VSC2)(MF)または ENTRY 文

    の中に USING 指定がある場合にのみ、CALL 文に USING 指定が含められる。その場合、各 USING 指定内の作用対象の数は同じでなければならない。

    (MF)作用対象の数が同じである必要はない。

11. (MF)呼ばれるプログラムが COBOL プログラムではない場合、そのプログラムに1つ以上のパラメータが宣言されている場合にのみ、CALL 文に USING 指定が含められる。その場合、USING 指定内の作用対象の数は呼ばれるプログラム中のパラメータの数と同じでなければならない（COBOL 以外の言語の処理系の中には、作用対象の数が同じでなくともよいものがある）。COBOL 言語では、データ項目の番地の桁寄せに制約はない。他方、COBOL 以外の言語では、通常は番地について前提があり、桁寄せされていないデータ項目が参照されると何らかのエラーを起こす。桁寄せは下記の措置のうちのいくつかを用いて実現される。
    - 集団項目に無名項目を追加する
    - ALIGN コンパイラ指令を使用するとともに、USING 指定中の作用対象を 01 レベルまたは 77 レベルのデータ項目にする
    - SYNCHRONIZED 句と IBMCOMP コンパイラ指令を併用する
12. * CALL 文

    (MF)または CHAIN 文

    の USING 指定、および呼ばれるプログラムの手続き部の見出し

    (MF)または呼ばれるプログラムが COBOL でない場合は ENTRY 文またはパラメータ・リスト

    の中の対応する USNIG 指定に中に作用対象が置かれている順序によって、

    呼ぶプログラムと呼ばれるプログラムで使用されているデータ名の対応関係が決まる。この対応は位置で決まるのであって、名前が等しいことによるのではない。CALL 文

(MF)または CHAIN 文

USING 指定の中の最初の作用対象が呼ばれるプログラムの USNIG 指定またはパラメータ･リストの最初の作用対象に対応し、それらの2番目同士が対応する、などとなる。

(MF)両方の作用対象の数が異なるために、完全には対応がとれないことがある。その場合、対応する相手がなくて残っている作用対象が CALL 文または CHAIN 文の側にある場合、それらは無視される。残っている作用対象が手続き部の見出しまたは ENTRY 文またはパラメータ･リストの側にある場合、それらは呼ばれるプログラムの中で参照されてはならない。位置による対応は、呼び名によって暗黙的に示される呼出し方式によって変わることがある。

指標名の場合は、そのような対応関係は確立されない。呼ぶプログラムと呼ばれるプログラムの指標名は常に別々の指標を参照する。

13. (ANS85)*CALL 文の USING 指定の中で参照されるパラメータの値は、その CALL 文が実行された時点で、呼ぶプログラムにとって利用可能になる。
14. * パラメータに BY REFERENCE 一意名-2 指定を

(ANS85)明示的または

暗黙的に指定すると、実行用プログラムは、呼ばれるプログラム中の対応するデータ項目が、呼ぶプログラム中のデータ項目と同じ記憶領域を占めるかのように動作する。 呼ばれるプログラム中のデータ項目のデータ記述には、呼ぶプログラム中の対応するデータ項目と同じ文字数を指定する。

(VSC2)(MF)BY REFERENCE ADDRESS OF 指定を明示的または暗黙的に指定すると、プログラムの動作は次のようになる。具体的には、USAGE POINTER を用いて追加のデータ項目が指定され、「SET データ項目 TO ADDRESS OF 一意名-3」文によって取得された値が BY REFERENCE によってそのデータ項目に渡されたかのように動作する。

(VSC2)(MF)一意名-3 が連絡節にあってそのレベル番号か 01 または 77 以外であるか、または作業場所節にある場合、そのデータ項目が BY CONTENT によって渡され、一意名-3 の番地を呼ばれるサブプログラムによっては変更できないことに等しい。

(MF)「BY REFERENCE 一意名-2」を明示的または暗黙的に指定すると、オブジェクト･プログラムは定数-2 を定数-3 のために記述されたものとして処理する。

15. (MF)(XOPEN)* BY CONTENT 指定を明示的または暗黙的に指定すると、CALL 文中の USING 句内で参照されたパラメータの値は、呼ばれたプログラム側では変更できない。しかし、呼ばれたプログラムは、その手続き部の見出しの中で、対応するデータ名によって参照しているデータ項目の値を変更できる。CALL 文の BY CONTENT 指定中の各パラメータに関するデータ記述は、手続き部の見出しの中の USING 句の中

の対応すパラメータに関するデータ記述と同じにする。つまり、変換も拡張も切捨てもあってはならない。

(MF)パラメータに関して BY CONTENT 指定を明示的または暗黙的に指定すると、オブジェクト･プログラムの動作は次のようになる。具体的には、追加のデータ項目が宣言され、そのデータ項目が BY REFERENCE 指定の中でパラメータとして使用されるかのように動作する。一意名-4 を指定すると、追加データ項目の暗黙的なデータ記述とその内容の両方が、一意名-4 のものと同じとなる。一意名-3 を指定すると、追加データ項目の暗黙的なデータ記述は、サイズと内容が一意名-3 と同じ英数字データ項目に等しくなる。「LENGTH OF 一意名-5」を指定すると、追加データ項目のデータ記述は、形式が PIC S9 (n) USAGE COMP となる。ここで、n の値は 9 以上である。このデータ記述により、使用している COBOL システムで処理可能なデータ項目の最大サイズが定義される。追加データ項目の内容は、一意名-5 に割り当てられた記憶領域のバイト数に設定される。

16. (MF)*BY VALUE 指定を明示的または暗黙的に指定すると、CALL 文の USING 指定の中で参照されているこのパラメータの値を、呼ばれるプログラムは変更できない。それにもかかわらず、呼ばれるプログラムはその手続き部の見出しの中の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できる。概念的には、システム領域（通常は「スタック」）に追加のデータ項目が宣言され、それがパラメータを渡すために COBOL 以外の言語に利用可能であり、その記憶領域が呼ばれるプログラム中のデータ項目と同じであるかのように、オブジェクト･プログラムは動作する。

(MF)一意名-6 が指定され、BINARY、COMP-4、COMP-5、および COMP-X のいずれかである場合は、追加データ項目のデータ記述とその内容の両方が、一意名-6 のものと同じとなる。整数-1 を指定すると、追加データ項目のデータ記述は、一意名-6 のデータ記述と同じとなる。ただし、その用途は COMP-5 となる。追加データ項目の内容は、COMP-5 である一意名-6 の内容と同じとなる。この変換と形式により、非 COBOL プログラムの呼出しが容易となる。

(MF)上記以外で、一意名-6 が指定された場合は、暗黙的に指定された追加データ項目のデータ記述とその内容が、一意名-6 のものと同じとなる。

(MF)整数-1 を指定すると、追加データ項目のデータ記述は符号付き数字項目 USAGE COMP-5 と同じとなる。この数字項目は、整数-2 が指定された場合は、その値により示されるバイト数で記憶される。それ以外の場合は、システム領域のサイズ（典型的な例は、システム上の POINTER のサイズ）で記憶される。

(MF)LENGTH OF 一意名-7 を指定した場合は、追加データ項目のデータ記述は PIC S9(n) USAGE COMP-5 と同じとなる。ここで、n の値は 9 以上で、COBOL システムのデータ項目の最大長を保持できるサイズのデータ項目を定義する。追加データ項目の内容は、一意名-7 に割り当てられた記憶用バイト数に設定される。

(MF)呼ばれるプログラムが COBOL プログラムの場合、CALL 文の BY VALUE 指定中の各パラメータに対応するパラメータを、手続き部見出しの USING 指定で宣言しな

ければならない。この対応するパラメータにも、BY VALUE 指定が明示的または暗黙的に指定されているものとする。

(MF)上記の、CALL 文の結果として生成された追加データ項目は、この対応するパラメータとは以下のようま関係を持つ。

(MF)追加データ項目が、BINARY、COMP-4、COMP-5、または COMP-X から COMP-5 への変換の対象であった場合、呼ぶプログラム中にある対応するパラメータのデータ記述は、追加データ項目のデータ記述と同じでなければならない。ただし、対応するパラメータの用途は、BINARY、COMP-4、COMP-5、および COMP-X のどれでもよい。呼ぶ COBOL プログラムへの記述の際に、追加データ項目の用途 COMP-5 から、呼ぶプログラム中の対応するパラメータの用途への変換が実行される。

(MF)追加データ項目が COMP-5 への変換の対象でなかった場合、呼ばれるプログラム中の各パラメータのデータ記述は、対応する追加データ項目のデータ記述と同じでなければならない。これは、追加データ項目に対し、呼ばれるプログラム中の対応するパラメータと一致させるための変換、拡張、および切り捨てのいずれも行われないことを意味する。

(MF)呼ばれるプログラムが COBOL プログラムでない場合、追加データ項目のサイズはシステム領域の最大サイズ (典型的な例は、システム上の POINTER のサイズ) を超えないものとする。そうでない場合は、システムが壊滅的に破壊される可能性がある。

17. (MF)呼ばれるプログラムから呼ぶプログラムにデータを渡す手段には、呼ぶプログラムの USING 指定中に宣言されているパラメータを使用する方法と、戻り値を使用する方法とがある。呼ばれるプログラムは EXIT GIVING 文または GOBACK GIVING 文によって値を戻す。呼ばれたプログラムから呼んだプログラムに制御が戻されると、戻り値がシステム領域に収められる。その領域は一般に、値を戻すために、COBOL 以外のプログラムによって利用可能である。

    (MF)システム領域のサイズ (通常はシステム上の POINTER のサイズ) は、COBOL システムの外で決定され、戻り値が取りうる最大値の制約要因となる。ただし、戻り値がシステム領域の最大サイズに満たないことがある。制御が戻された後で、呼んだプログラムで戻り値を利用できるようになる。それは、暗黙的に特殊なレジスタである RETURN-CODE に収められていることもあれば、GIVING 指定に指定したデータ項目に明示的に収められていることもある。

18. (MF)GIVING 指定を書かず、呼出し方式に特殊レジスタの RETURN-CODE (*環境部*の章の*特殊名段落*中節を参照) を更新するように指定されている場合、オブジェクト・プログラム動作は次のようになる。具体的には、システム領域がそのサイズのまま COBOL の数字データ項目で USAGE COPM-5 が指定されているものとして定義され、そのシステム領域を送出し側とし特殊レジスタの RETURN-CODE を受取り側として、MOVE 文を実行したかのように動作する。

19. (MF)呼ぶプログラム中に GIVING INTO 指定を書いた場合には、戻り値をシステム領域に収めるのに必要なのと同じ長さを一意名-8 に指定しなければならない。また、一意名-8 の型と用途は、呼ばれるプログラムからの値の返され方によって暗黙的に示される型と用途に等しくなければならない。呼んだプログラムに制御が戻されると、戻り値が一意名-8 に収められている。
20. (MF)呼ぶプログラムに GIVING ADDRESS OF 指定を書いた場合、呼ばれるプログラムが返す値は明示的または暗黙的に USAGE POINTER と指定されたものでなければならない。オブジェクト・プログラム動作は次のようになる。具体的には、システム領域が USAGE POINTER と指定された COBOL のデータ項目として宣言され、そのシステム領域を最初の作用対象とし「ADDRESS OF 一意名-9」を2番目の作用対象として、SET 文を実行したかのように動作する。
21. (MF)呼ばれるプログラムが値を返さない場合に GIVING 指定を書くと、エラーとなる。
22. 呼ばれたプログラムの中に、CALL 文が含まれていてもよい。しかし、

    (85/90)(SO.002)(MF)呼ぶプログラムが再帰属性を持っていない限り、

    呼ばれたプログラム呼ばれたプログラムは、呼んだプログラムを直接的または間接的に呼ぶ CALL 文を実行しない。

23. 宣言部分の範囲内で CALL 文が実行されるとき、その CALL 文は、制御が移されたがまだ実行を終了していない呼ばれたプログラムを直接的にも間接的にも参照できない。
24. (ANS85)END-CALL 指定は、CALL 文の範囲を区切る。
25. (VSC2)(MF)USAGE POINTER 、

    (MF)または USAGE PROCEDURE-POINTER

    (VSC2)(MF)が明示的または暗黙的に指定されており、呼ばれるプログラム中のデータ項目、

    (MF)または手続き

    (VSC2)(MF)の番地から当初に導出された値は、その呼ばれたプログラムに対して CANCEL 文が実行された後では無効になるので、使用してはならない。

26. (MF)使用されないパラメータをサブプログラムに渡すために、OMITTED 指定を使用することができる。その場合、ダミーのデータ項目を宣言する必要はない。
27. (MF)OMITTED パラメータは BY REFERENCE パラメータであり、呼ばれるプログラムに空の番地を渡すことに相当する。それは BY VALUE 0 SIZE 4 と指定するのと同じであるが、BY REFERENCE パラメータとして呼び出しプロトタイプの型チェックを受ける。
28. (MF)OMITTED の直前に BY REFERENCE を指定しないと、以降のパラメータはその前に有効であった呼出し方式に従って渡される。省いたパラメータは BY VALUE 0 SIZE 4 と指定したかのように扱われる。

形式 1

29. (MF)呼び名は、呼ばれるプログラムを起動するときに、このプログラム使用する呼出し規則を示す。呼ばれるプログラムで設定された呼出し規則が、呼び名が示すものと違うと、COBOL システムが動作しなくなる場合がある。
30. (MF)実行単位は、呼ばれたプログラムが発見され、実行可能な状態になる前は、それが COBOL プログラムか、それ以外のプログラムか、または手続きであるかを認識していない。通常、プログラムが COBOL プログラムかどうかは、プログラム名の形式では判定できないためである。同じプログラム名の COBOL プログラムと COBOL 以外のプログラム、または 2 個の COBOL 以外のプログラムが存在していると、エラーとなる 。
31. (ANS85)* BY CONTENT 指定、BY REFERENCE 指定、

    (MF)および BY VALUE 指定

    (ANS85)は、別の BY CONTENT 指定、BY REFERENCE 指定、

    (MF)または BY VALUE 指定

    (ANS85)が出現するまで、その後に続くパラメータすべてについて有効である。

    (ANS85)最初のパラメータの前に BY CONTENT、BY REFERENCE、

    (MF)および BY VALUE

    (ANS85)のどの指定もない場合、BY REFERENCE 指定が使用されているものとする。

形式 2

(ISO2002)(MF)

32. 一意名-1、定数-1、および手続きポインタ-1 のいずれも指定しない場合、プログラムプロトタイプ名-1 は、*環境部*の章の*リポジトリ段落*節で指定された規則に従い、呼ばれるプログラムの外部化されたプログラム名を決定する。
33. NESTED 指定を使用すると、プログラム名段落で定数-1 が指定された共通プログラムまたは含まれるプログラムが使用され、呼ばれるプログラムの特性が決定される。
34. NESTED 指定を使用しないと、プログラムプロトタイプ名-1 を使用して、呼ばれるプログラムの特性が決定される。
35. 単一の一意名または定数のみから成る引数は、算術式ではなく、一意名または定数と見なされる。
36. キーワード、BY REFERENCE、BY CONTENT、および BY VALUE のうち、どれも使用せずに引数を指定した場合、この引数が渡される方法は以下のように決定される。
    a. BY REFERENCE 指定が明示的または暗黙的に、対応する仮パラメータに使用されていると、
        1. 引数が構文規則 3 の条件を満たす場合は、BY REFERENCE が指定されたと見なされる。
        2. 引数が構文規則 3 の条件を満たさない場合は、BY CONTENT が指定されたと見なされる。

b. BY VALUE 指定が明示的または暗黙的に、対応する仮パラメータに使用されていると、BY VALUE が指定されたと見なされる。

37. 制御は、プログラムに対して指定された記述表記法に適合した方式で、呼ばれるプログラムに渡される。
38. OMITTED 指定を使用するか、その後の引数を省略した場合、そのパラメータの省略引数条件は、呼ばれるプログラム中で真と評価される。(***手続き部***の章の*省略引数条件*節を参照。)
39. 呼ばれるプログラムで省略引数条件が真であるパラメータが参照されると、実行時エラー 203 となる。ただし、引数として参照される場合、および省略引数条件の中で参照される場合を除く。

12.5 CANCEL(取消し)文

CANCEL (取消し) 文は、参照するプログラムを次回に呼んだときに必ず初期状態にあるようにする。

一般形式

```
CANCEL { 一意名-1 }...
       { 定数-1  }
```

構文規則

1. 定数-1 は、文字定数とする。
2. 一意名-1 は、英数字データ項目を参照する。

一般規則

1. 定数-1 または一意名-1 によって参照されるデータ項目の内容は、取り消す対象のプログラムを指さなければならない。その名前は取消し対象のプログラムのプログラム名段落に記されているプログラム名でなければならない (***見出し部***の章の*プログラム名段落*節、特に大文字と小文字の区別に関する記述を参照) 。別の方法として、その名前に取消し対象のプログラムの実行プログラム用のコードが収められているファイルを指定してもよい。CANCEL 文の中で同じファイルを指す別の名前を指定した場合にどのような結果になるかは、オペレーティング･システムによって異なる。前に呼ばれたプログラム、プログラム･ファイル名、あるいは登録集ファイル中のプログラムと指定された名前の照合方法の詳細については、COBOL システムのマニュアルを参照。

   その名前が ENTRY 文に含まれている入口名を指している場合、結果はどうなるかわからない。

2. 明示的または暗黙的な CANCEL 文が実行された後では、そのプログラムとそれが属する実行単位との論理関係はなくなる。明示的または暗黙的な CANCEL 文の対象で

あったプログラムが、同じ実行単位の中で後からまた呼ばれたときには、そのプログラムは初期状態にある。*翻訳集団の概念*の章の*関数、メソッド、またはプログラムの状態*節を参照。

3. 他のプログラム中の CANCEL 文の中に指定されたプログラムは、そのプログラムから呼べるものとする。*COBOL 言語の概念*の*名前の適用範囲*および *CALL (呼出し) 文*節を参照。
4. (ANS85)明示的または暗黙的な CANCEL 文が実行されると、その対象となったプログラム中に含まれるプログラムもすべて取り消される。その結果は、取り消されたプログラムの中に含まれる各プログラムに対して、それらが現れる順番と逆の順番で、有効な CANCEL 文を実行したことに等しい。
5. CANCEL 文中に指定したプログラムは、呼ばれたがまだ EXIT PROGRAM 文

   (OSVS)(VSC2)(MF)または GOBACK

   を実行していないプログラムを、直接的にも間接的にも指してはならない。

6. 取り消されたプログラムとの論理関係は、その後でそのプログラムを対象にして CALL 文を実行することによってだけ、成り立つ。
7. 呼ばれたプログラムは、CANCEL 文の作用対象として指定すること、または対象とするプログラムが属する実行単位が終了すること、

   (ANS85)または、呼ばれたプログラムが初期属性をもつ場合には、その中で EXIT PROGRAM 文を実行すること

   により、取り消される。

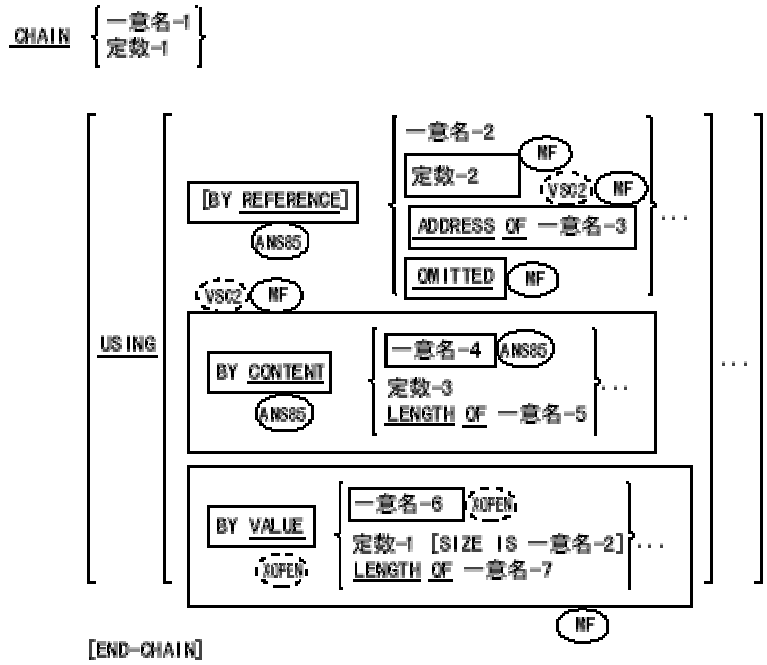
8. 実行単位の中でまだ呼ばれていないプログラム、呼ばれたが取消し済みのプログラム、または COBOL 以外のプログラムを指定して、明示的または暗黙的な CANCEL 文を実行しても、何の措置もとられない。 制御は CANCEL 文の後ろに続く、次の実行可能文に移される。
9. (ANS85)プログラム内に記述されている外部データ･レコード中のデータ項目の内容は、そのプログラムが取り消されても変わらない。
10. (ANS85)明示的または暗黙的な CANCEL 文の実行中に、暗黙の CLOSE 文が実行される。具体的には、CANCEL 文の対象プログラム中の内部ファイル結合子に関連する、開かれているファイルがすべて閉じられる。これらのファイルに関連する USE 手続きがあっても、実行されない。
11. CANCEL 文は、副プログラムを初期状態に戻すかそのままの状態で残すかを決める。CANCEL 文は、必ずしも、取り消された副プログラムが記憶されていた領域を解放するとは限らない。しかし、オペレーティングシステムの中には取り消された副プログラムが記憶されていた領域を解放するものが多い。その副プログラムの外部ファイルおよび外部データ領域は、CANCEL 文によって解放されることはない。

12.6 CHAIN(連鎖)文

(MF)

CHAIN (連鎖) 文は、実行単位内のある実行用プログラムから他の実行用プログラムへ制御を移す。ただし、後から制御が戻されることを前提としない。連鎖先のプログラムは、新しい実行単位中の主プログラムであるようにみえる。

一般形式



構文規則

1. CHAIN 文の構文は CALL 文の構文に似ている。CALL 文の項に示したように、この2つの文は、*CALL (呼出し)文*節に記述されている共通の構文規則に従う。CHAIN 文の一般形式に示されている一意名、定数、整数はすべて、CALL 文の一意名、定数、整数に対応する。

一般規則

1. 定数-1、または一意名-1 によって参照されるデータ項目の内容は連鎖先のプログラムの名前である。CHAIN 文が記述されているプログラムは連鎖するプログラムである。

   連鎖先のプログラムは COBOL プログラムでなければならず、かつ入れ子になっていてはならない。連鎖先のプログラムの名前には、そのプログラムのプログラム名段落中に記述されている名前を指定しなければならない (大文字と小文字の区別に関しては、***見出し部***の章の*プログラム名段落*節を参照) 。別の方法として、連鎖先のプログラムの名前は、連鎖先のプログラムの実行プログラム用のコードが収められているファイルを指す名前であってもよい。前に連鎖されたプログラム、プログラム･ファイル名、あるいは登録集ファイル中のプログラムと指定された名前の照合方法の詳細については、インターフェイスに関する COBOL システムのマニュアルを参照。

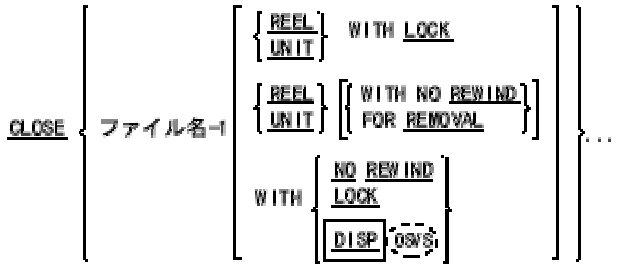
2. CHAIN 文を実行したときに、CHAIN 文に指定した連鎖先のプログラムが実行可能になれば、そのプログラムに制御が移される。連鎖するプログラムは取り消される。また、連鎖するプログラムによって呼ばれた COBOL プログラムがあれば、それらも取り消される。連鎖先のプログラムは連鎖するプログラムの制御下にはない。したがって、連鎖されたプログラム内で EXIT PROGRAM を実行しても、効果はない。
3. CHAIN 文を実行したときに、CHAIN 文に指定した連鎖先のプログラムが実行可能にならなければ、致命的なランタイム･エラーとされる。
4. 連鎖されたプログラム内に CHAIN 文があってもよい。連鎖されたプログラム内の CHAIN 文では、そのプログラム自体以外の任意のプログラムを参照できる。.
5. CHAIN 文の形式は CALL 文の形式に似ている。CALL 文の項に示したように、この2つの文は、*CALL (呼出し)文*節に記述されている共通の一般規則に従う。CHAIN 文の一般形式に示されている一意名、定数、整数はすべて、CHAIN 文の一意名、定数、整数に対応する。また、呼ぶプログラムおよび呼ばれるプログラムという表現が用いられているときは、連鎖するプログラムおよび連鎖先のプログラムと同等であるとみなす。
6. 連鎖処理によって、CHAIN 文の USING 指定内で参照されている各データ項目の値が、連鎖先のプログラムの手続き部の見出しの USING 指定中に書かれている対応するデータ名によって参照される、データ項目に転記される。
7. USING 指定中の作用対象として渡された USAGE POINTER または USAGE PROCEDURE-POINTER の指定がある任意のデータ項目の値は、COBOL データ項目から当初に導出された番地を表していてはならない。
8. END-CHAIN 指定は CHAIN 文の範囲を区切る。
9. コンパイラ指令の INTLEVEL"1"を使用した場合は、USING 指定を用いてはならない。

12.7 CLOSE(閉じる)文

CLOSE (閉じる) 文は、リール/ユニットおよびファイルの処理を終了させる (ファイルのクローズ) 。さらに、可能ならば、巻戻しやロックや取外し処理を行うこともできる。

一般形式

形式 1 (レコード順ファイル)



形式 2 (
MF 行順ファイル、
相対ファイル、および索引ファイル)

```
CLOSE ファイル名-1 [WITH LOCK] [ファイル名-2 [WITH LOCK]]...
```

構文規則

すべての形式 (すべてのファイル)

1. CLOSE 文で参照されるファイルは、すべてが同じ構成またはアクセスである必要はない。

形式 1 (レコード順ファイル)

2. (MF)REEL および UNITUNIT を指定できるのは、SELECT 句に MULTIPLE REEL または MULTIPLE UNIT MULTIPLE UNIT を指定してあるファイルに対してだけである。マルチユニット･ファイルを認識しないランタイムシステムにおいては、CLOSE REEL 文および CLOSE UNIT 文は" 空" 文である。つまり、これらの文は注記になる。CLOSE REEL 文または CLOSE UNIT 文によって閉じた装置上には、他に開いているファイルはないことが重要である。

   (OSVS)CLOSE REEL WITH LOCK 文および CLOSE UNIT WITH LOCK 文は、CLOSE REEL FOR REMOVAL と同義語として扱われる。

3. (OSVS)DISP 指定は、テープ･ファイルに対してだけ適用できる。

   (MF)DISP 指定は注記になる。

一般規則

すべての形式 (すべてのファイル)

1. CLOSE 文は、開かれているファイルに対してだけ実行できる。
2. CLOSE 文の実行が正常に終了すると、ファイル名-1 のレコード領域は使用できなくなる。CLOSE 文の実行が不成功に終わった場合は、レコード領域はどうなるかわからない。
3. (ANS85)STOP RUN (実行停止) 文が実行されたときに開かれているファイルがあると、そのファイルは閉じられる。呼ばれたプログラムの中でファイルが開かれ、後でそのプログラムの中で CANCEL (取消し) 文が実行された場合、開かれたままのファイルがあれば閉じられる。
4. (MF)CLOSE 文の実行が正常に終了すると、閉じられたファイルに関して実行単位が保持していたレコードロックまたはファイルロックはすべて解除される。
5. CLOSE 文が実行されると、ファイル名-1 に関する入出力状態の値が更新される。(*手続き部*の章の*入出力状態*節を参照。)
6. CLOSE 文の書き方とファイルの種類ごとに、実行結果をまとめて、表 12-1 に示す。

表 12-1: CLOSE 文の形式とファイルの種類の関係

CLOSE 文の形式	ファイルの種類			
	非リール/ユニット	レコード順単一リール/ユニット	レコード順複数リール/ユニット	非レコード順単一/複数リール/ユニット
CLOSE	C	C, G	A, C, G	C
CLOSE WITH LOCK	C, E	C, E, G	A, C, E, G	C, E
CLOSE WITH NO REWIND レコード順だけ	AN585 C, H	B, C	A, B, C	X
CLOSE REEL/UNIT レコード順だけ	AN585 F	AN585 F, G	F, G	X
CLOSE REEL/UNIT FOR REMOVAL レコード順だけ	AN585 F	AN585 D, F, G	D, F , G	X
CLOSE REEL/UNIT WITH NO REWIND レコード順だけ	X	X	F, B	X

7. 上の表の中に示した記号の意味を以降に説明する。入力ファイルか出力ファイルか入出力ファイルかによって意味が異なるときは、それぞれに分けて説明する。分けていなければ、説明は入力ファイルと出力ファイルと入出力ファイルのすべてに当てはまる。

   A. **前のリール/ユニットは影響を受けない**

      入力ファイルおよび入出力ファイル：

      ファイル中の現在のリール/ユニットよりも前にあるリール/ユニットがすべて処理される。ただし、既に CLOSE REEL/ UNIT 文が実行されているものを除く。現在のリール/ユニットがファイル中の最後のものでない場合、その後ろのリール/ユニットは処理されない。

      出力ファイル：

      ファイル中の現在のリール/ユニットよりも前にあるリール/ユニットがすべて処理される。ただし、既に CLOSE REEL/UNIT 文が実行されているものを除く。

   B. **現在のリールを巻き戻さない**

      現在のリール/ユニットはそのままの位置にとどめられる。

   C. **ファイルを閉じる**

      入力ファイルおよび入出力ファイル(順呼出し法)：

ファイルの末尾に達しており、かつファイル記述にラベル・レコードを指定してある場合、オペレーティングシステムのラベル方式に従って、ラベルの処理が行われる。ラベル・レコードを指定しているのにラベル・レコードが存在しない場合、またはラベル・レコードを指定していないのにラベル・レコードが存在する場合、CLOSE 文の動作はどうなるかわからない。ファイルの末尾に達しているが、ファイル記述にラベル・レコードが指定されていない場合、ラベルの処理は行われないが、ランタイムシステムに依存するその他の閉じる処理は行われる。ファイルの末尾に達していない場合、ランタイムシステムに依存する閉じる処理は行われるが、終了ラベルの処理は行われない。

入力ファイルおよび入出力ファイル(乱呼出し法、動的呼出し法); 出力ファイル(乱呼出し法、動的呼出し法、順呼出し法):

ファイル記述にラベル・レコードを指定している場合、オペレーティングシステムのラベル方式に従ってラベルの処理が行われる。ラベル・レコードを指定しているのにラベル・レコードが存在しない場合、または LABEL RECORD 句を指定していないのにラベル・レコードが存在する場合、CLOSE 文の動作はどうなるかわからない。ファイル記述にラベル・レコードを指定していない場合、ラベルの処理は行われないが、ランタイムシステムに依存するその他の閉じる処理は行われる。

D. リール/ユニットを取り外す

REEL または UNIT を指定しないで CLOSE 文をファイルに対して実行し、続いてそのファイルに対して OPEN 文を実行すると、そのファイル内のリールまたはユニットを適切な順番で再度呼び出すことができる。

E. ファイルをロックする

この実行単位の実行中は、このファイルを再び開くことはできない。

F. リール/ユニットを閉じる

入力ファイルおよび入出力ファイル(リール/ユニット媒体):

1. (ANS85)現在のリール/ユニットが該当するファイルの最後または唯一のリール/ユニットである場合、リール/ユニットの交換は行われない。現在のボリューム・ポインタは変更されず、ファイル位置指示子には次のリール/ユニットが存在しないと設定される。
2. 該当するファイルに他のリール/ユニットが存在する場合、リール/ユニットの交換が発生し、現在のボリューム・ポインタはそのファイル中の次のリール/ユニットを指すように更新される。その後標準開始リール/ユニット・ラベル手続きが実行されて、ファイル位置指示子が新しい現行ボリューム上に存在する最初のレコードの番号よりも 1 小さく設定される。現行ボリュームにデータ・レコードが存在しなければ、次のリール/ユニットの交換が発生する。

出力ファイル(リール/ユニット媒体):

下記の操作が行われる。

3. 標準終了リール/ユニット・ラベル手続きの実行
4. リール/ユニットの交換
5. 標準開始リール/ユニット・ラベル手続きの実行
6. そのファイルに対して次に WRITE 文を実行すると、次の論理レコードはそのファイルの次のリール/ユニットに書き出される。

入力ファイル、入出力ファイル、出力ファイル(非リール/ユニット媒体):

(ANS85)この CLOSE 文の実行は、正常に終了したものと扱われる。実際には、ファイルは開かれたままであり、ファイル位置指示子は変更されず、一般規則 4 に示された以外の処理は何も行われない。

G. 巻き戻す

現在のリールまたは類似の装置は、物理的始点に位置付けられる。

H. 任意指定は無視される

(ANS85)CLOSE 文は、任意指定が何もされていないかように実行される。

X. 不正

CLOSE 文の指定とファイルの種類のこの組合わせは正しくない。実行結果がどうなるかはわからない。

形式 1 (レコード順ファイル)

以下の記述において別途断らないかぎり、"リール" と"ユニット" という語は同義であり、CLOSE 文の中でまったく同じように使うことができる。順編成の大記憶ファイルの取扱いは、論理的にはテープまたは類似の順媒体と同じである。

7. 様々な記憶媒体に適用される種々の CLOSE 文の効果を示すために、すべてのファイルを下記の種類に区分する。
   a. 非リール/ユニット:巻戻しやリール/ユニットの概念が意味をもたない入出力媒体を使用したファイル。
   b. 順単一リール/ユニット:1 つのリール/ユニットに完全に格納される順ファイル。
   c. 順複数リール/ユニット:複数のリール/ユニットにまたがって格納される順ファイル。
8. 環境部のファイル管理段落中でファイルに OPTIONAL 指定をしているが、そのファイルが存在しない場合には、そのファイルに関しては標準のファイル終了処理は行われない。

9. あるファイルに対して REEL も UNIT も指定せずに CLOSE 文を実行すると、以降そのファイルを対象とする入出力文（SORT 文または MERGE 文で USING または GIVING を指定したものは除く）は明示的にも暗黙的にも実行できなくなる。ただし、そのファイルに改めて OPEN 文を実行すれば、再び入出力文を実行できるようになる。
10. ファイルが存在する媒体に対して適用しなければ、WITH NO REWIND 指定も FOR REMOVAL 指定も実行時に効力をもたない。
11. WITH LOCK を指定すると、そのファイルはその実行単位の現在の実行中には再度開くことはできなくなる。この点を除けば、通常の CLOSE と同じである。

    (MF)（この機能は、ファイルを共有するときのレコードまたはファイルのロックとは関係がない。）

すべての形式（すべてのファイル）

12. あるファイルに対して CLOSE 文を実行すると、以降そのファイルを対象とする文は、

    (MF)DELETE FILE 文を除き、

    明示的にも暗黙的にも実行できなくなる。ただし、そのファイルに改めて OPEN 文を実行すれば、再びファイルを操作する文を実行できるようになる。

13. WITH LOCK を指定すると、そのファイルはその実行単位の現在の実行中には再度開くことはできなくなる。

12.8 COMMIT(更新確定)文

(MF)

COMMIT（更新確定）文は、該当する実行単位によってロックされている、ファイル中のすべてのレコードロックを解除する。SELECT 文の WITH...ROLLBACK 句を単なる注記としてではなく実際にサポートする COBOL システムでは、COMMIT 文は現在のトランザクションの終わりを区切り、それまでのデータの更新内容を確定する。

一般形式

```
COMMIT
```

一般規則

1. COMMIT 文を実行すると、実行単位によってそれまでロックされていた、ファイル中のすべてのロックが解除される。
2. 排他ファイル上のファイルロックは、COMMIT 文の影響を受けない。
3. SELECT 文の WITH...ROLLBACK 句を単なる注記としてではなく実際にサポートする COBOL システムでは、COMMIT 文は下記の機能を果たす。

a. 現在のトランザクションが完了したことを示す。
b. トランザクション処理の間になされたデータの更新内容を確定する。

12.9 COMPUTE(計算)文

COMPUTE (計算) 文は、算術式の値をいくつかのデータ項目に代入する。

一般形式

```
COMPUTE {一意名-1 [ROUNDED]} ...
{ EQUAL } 算術式   (OSVS)(VSC2)(MF)
{ =     }
[ON SIZE ERROR 無条件文-1]
[NOT ON SIZE ERROR 無条件文-2]   (ANS85)
[END-COMPUTE]
```

構文規則

1. "=" の左側だけに現れる一意名は、基本数字項目か基本数字編集項目

   (OSVS)(VSC2)(MF)または浮動小数点数項目を指さなければならない。

2. (OSVS)(VSC2)(MF)"Equal" と "=" は同義語である。

一般規則

1. ***手続き部***の章の*算術式*、*ROUNDED 指定*、*ON SIZE ERROR 指定*、*算術文*, *作用対象の重なり*、および*算術文における複数個の答え*の各節、および ***COBOL 言語の概念***の章の*明示範囲符と暗黙範囲符*節、***翻訳集団の概念***の章の*範囲明示文*節を参照。
2. 単一の一意名または定数からなる算術式を指定すると、その単一の一意名または定数の値を一意名-1 の値に設定することになる。
3. 算術文の ADD、SUBTRACT、MULTIPLY、DIVIDE を使用した場合は、一時的なデータ項目に中間結果を入れる必要がある。COMPUTE 文を使用すると、算術処理を組み合わせながら、それらの制約を受けないで済む。式は可能なだけの桁を評価して、一意名-1 中に合うように切り捨てまたは四捨五入される。

   (OSVS)(VSC2)式の評価中に出された中間結果は、COBOL システムによって決められた PICTURE のデータ項目に転記されたように切り捨てられる。この動作は ARITHMETIC コンパイラ指令で選択する。

4. "=" の前に計算結果用の一意名を複数個指定すると、算術式を計算した結果の値が、個々の一意名-1 の新しい値として代入される。

12.10 CONTINUE 文

ANS85

CONTINUE (継続) 文は、処理操作を行う文ではない。実行可能な文が存在しないことを示す。

一般形式

```
CONTINUE
```

構文規則

1. 条件文または無条件文を使用できるところでは、どこでも CONTINUE 文を書くことができる。

一般規則

1. CONTINUE 文は、プログラムの実行には影響しない。

12.11 DELETE(削除)文

DELETE (削除) 文は大記憶ファイルからレコードを論理的に削除する。DELETE 文は相対編成または索引編成のファイルに対してだけ適用できる。

一般形式

```
DELETE ファイル名-1 RECORD
    [INVALID KEY 無条件文-1]

    [NOT INVALID KEY 無条件文-2]
    [END-DELETE]
```

ANS85

構文規則

1. 順呼出し法のファイルを対象にした DELETE 文に、INVALID KEY は指定できない。.
2. USE AFTER STANDARD EXCEPTION PROCEDURE を指定してない場合、順呼出し法ではないファイルを対象にした DELETE 文には、INVALID KEY を指定する。

   VSVS VSC2 MF この規則は強制しない。

一般規則

1. DELETE 文を実行する際には、対象とするファイルは入出力両用モードで開いておく。(*手続き部・MERGE・OPEN*の章の *OPEN(開く)文*節を参照。)
2. 順呼出しモードのファイルの場合、ファイル名-1 に関して DELETE 文の前に最後に行われた入出力文は、正常に終了した READ 文とする。オペレーティングシステムはその READ 文によって呼び出されたレコードを、ファイルから論理的に削除する。
3. 乱呼出しモードまたは動的呼出しモードのファイルの場合、オペレーティングシステムは、該当するファイルのキー･データ項目の内容によって識別されるレコードをファイルから論理的に削除する。相対ファイルの場合、このキー･データ項目は RELATIVE KEY であり、索引ファイルの場合、このキー･データ項目は RECORD KEY である。キーによって指定されたレコードがファイル中に存在しないと、無効キー条件が発生する。(*手続き部*の章の*無効キー条件*節を参照。)
4. DELETE 文の実行が正常に終了すると、該当するレコードはファイルから論理的に削除され、呼び出すことはできなくなる。
5. DELETE 文を実行することによって、ファイル名-1 のレコード領域の内容が影響を受けることはない。
6. DELETE 文を実行することによって、ファイル位置指示子が影響を受けることはない。
7. 該当ファイルに FILE データ項目を指定してあると、DELETE 文を実行したときにその内容が更新される。(*手続き部*の章の*入出力状態*節を参照。)
8. DELETE 文処理が正常または不成功に終わった後、どこに制御が移されるかは、DELETE 文中に INVALID KEY 指定、

   (ANS85)および NOT INVALID KEY 指定

   を指定したか否かに左右される。(*手続き部*の章の*無効キー条件*節を参照。)

9. (ANS85)END-DELETE 指定は DELETE 文の有効範囲を区切る。(*COBOL 言語の概念*の章の*明示範囲符と暗黙範囲符*節を参照。)
10. (MF)DELETE 文を実行する際に、対象レコードが他の実行単位によってロックされていてはならない。
11. (MF)DELETE 文の実行が正常に終了すると、その実行単位によってロックされていたレコードのロックは解除される。

12.12 DELETE FILE(ファイル削除)文

(MF)

DELETE FILE (ファイル削除) 文は、指定されたファイルをそれが格納されている物理装置から物理的に削除する。

一般形式

```
DELETE FILE {ファイル名}...
```

一般規則

1. DELETE FILE 文は、指定されたファイルをそれが格納されている物理装置から物理的に削除する。
2. DELETE FILE 文を実行するとき、指定したファイルは閉じられていなければならないが、ロックされていてはならない。

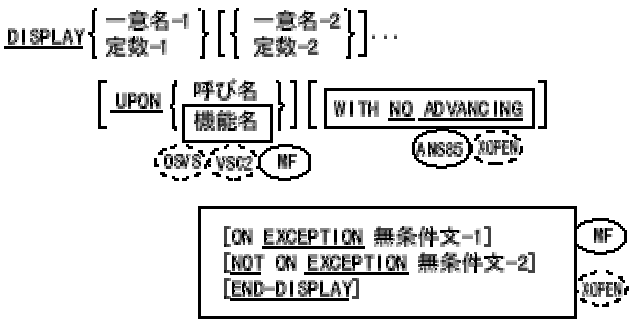
12.13 DISPLAY(表示)文

DISPLAY (表示) 文は、指定されたデータ項目のデータを CRT 画面のような適切なハードウエア装置に転送させる。

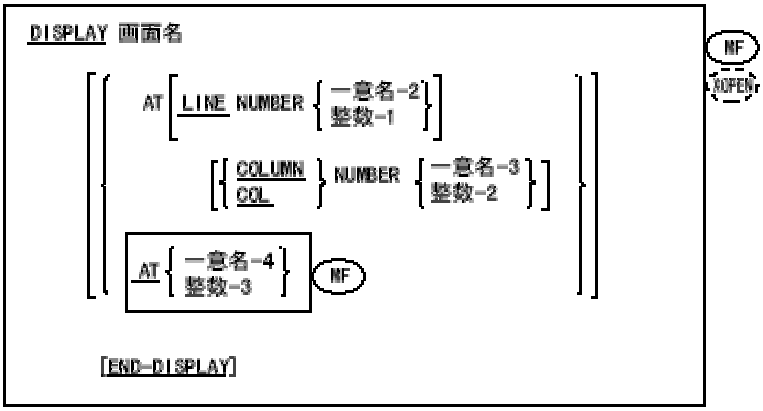
(MF)DISPLAY 文はまた、プログラムから CRT またはビデオ端末画面上の静止型定形画面にデータを転送して表示させる。表示されたその画面を、入力画面とすることができる。

一般形式

形式 1



形式 2



形式 3

```
DISPLAY {{一意名-1 | 定数-1}}                                          (NF)
   { AT [ LINE NUMBER {一意名-2 | 整数-1} ]
         [ {COLUMN | COL} NUMBER {一意名-3 | 整数-2} ] }
   { AT {一意名-4 | 整数-3} }
   [ UPON {CRT | CRT-UNDER} ] [MODE IS BLOCK]
   [ WITH { {BELL | BEEP}
            BLINK
            GRID
            ERASE {EOL | EOS}
            HIGHLIGHT
            LOWLIGHT
            LEFTLINE
            OVERLINE
            REVERSE-VIDEO
            SIZE IS {一意名-5 | 整数-4}
            UNDERLINE
            {FOREGROUND-COLOR | FOREGROUND-COLOUR} IS 整数-5
            {BACKGROUND-COLOR | BACKGROUND-COLOUR} IS 整数-6
            CONTROL IS {一意名-6 | 定数-2}
            BLANK {SCREEN | LINE} } ... ]...
[END-DISPLAY]
```

構文規則

形式 1

1. 呼び名は、環境部の特殊名段落中の機能名と関連させる。有効な機能名に関しては、***環境部***の章の*特殊名段落*節の中の「一般規則 9」を参照。
2. (OSVS)(OSC2)(NF)機能名に関連する呼び名の代わりに、機能名そのものを使用できる。
3. 各定数には、ALL 以外の表意定数を使用できる。
4. 定数が数字の場合は、符号の付かない整数とする。

VSVS VSCZ MF この制限は無視してよい。

5. MF XOPEN 呼び名が ARGUMENT-NUMBER に関連しているときは、使用される一意名-1 または定数-1 は、それぞれ符号の付かない整数または符号の付かない整数定数として、定義されているデータ項目を指さなければならない。この場合、単一の一意名-1 または定数-1 だけを使用でき、WITH NO ADVANCING 指定は使用できない。
6. MF XOPEN 呼び名が ENVIRONMENT-NAME または ENVIRONMENT-VALUE と関連するときは、使用される一意名-1 または定数-1 は、それぞれ英数字データ項目または文字定数を指さなければならない。この場合、単一の一意名-1 または定数-1 だけを使用でき、WITH NO ADVANCING 指定は使用できない。
7. ON EXCEPTION 指定および NOT ON EXCEPTION 指定は、呼び名が ENVIRONMENT-NAME または ENVIRONMENT-VALUE と関連している場合にだけ指定できる。

形式 2

8. MF 画面名は、OCCURS 句を伴う項目であってはならない。

形式 2 and 3

9. MF LINE 指定と COLUMN 指定は、どのような順番で書いてもよい。
10. MF 整数-3 および整数-4 の長さは、4 バイトまたは 6 バイトとする。

形式 3

11. MF 作用対象が画面名でない DISPLAY 文は、その中に AT 指定、CRT オプションまたは CRT-UNDER オプションを伴う UPON 指定、MODE IS BLOCK 指定が含まれるか、または UPON 指定は含まれないが特殊名段落中に CONSOLE IS CRT 句が指定されていると、形式 3 として扱われる。CONSOLE オプションを伴う UPON 指定が含まれるか、または UPON 指定が含まれず特殊名段落中に CONSOLE IS CRT 指定も指定されていないと、この形の DISPLAY 文は形式 1 として扱われる。
12. MF 一意名に続くいろいろな指定は、どんな順番で書いてもよい。
13. MF 一意名-1 内の基本データ項目の用途は、DISPLAY とする。
14. MF 一意名-1 内の非基本データ項目は、8191 バイトより大きくなることがある。MODE IS BLOCK 句を使用した場合、一意名-1 の全体の長さは 8191 バイト以下にすること。
15. MF DISPLAY の後に 2 つ以上の一意名-1 が続く場合、WITH 指定はその直前にある作用対象だけに適用される。

一般規則

形式 1

1. DISPLAY 文の作用対象の用途が DISPLAY 以外であるかまたは符号が独立していないと、その作用対象は、用途が DISPLAY で符号が独立したものに変換される。変換後の大きさが、その作用対象の大きさとされる。

2. 作用対象の 1 つとして表意定数を指定すると、その表意定数が 1 文字だけ表示される。
3. (MF)機能名 COMMAND-LINE またはそれに関連する呼び名を指定すると、データはシステムのコマンド行バッファに上書きされる。この後、ACCEPT FROMCOMMAND-LINE 文を実行することによって、このデータを読み出すことができる。この場合は、作用対象は 1 つだけ指定できる。

   (MF)使用可能なそれ以外の機能名はすべて、CONSOLE に等しいものとして扱われ、各作用対象は、指定された順番に操作卓装置に転送される。表示されるデータの全体の大きさは、各作用対象の大きさの和に等しい。表示は現在カーソルが位置している所から開始され、必要があれば、以降の行に繰り越される。

   (MF)以前のリリースでは、最後の作用対象に続く空白は表示されなかった。

4. (ANSI85)NO ADVANCING を指定すると、カーソルは最後の文字が表示された次の位置に残される。NO ADVANCING を指定しないと、カーソルは次の行の先頭に位置付けられる。カーソルが行頭にあるときは、いつでもスクロールできる。
5. (MF)(OPEN)機能名 ARGUMENT-NUMBER に関連する呼び名を指定すると、機能名 ARGUMENT-VALUE に関連する呼び名を伴う後続の ACCEPT 用の位置は、指定されたコマンド行の引数を読み出すように設定される。一意名-1 または定数-1 に関連する値が 0 より小さいか、99 より大きいか、コマンド行上の引数の合計数よりも大きいと、結果はどうなるかわからない。
6. (MF)(OPEN)機能名 ENVIRONMENT-NAME に関連する呼び名を指定すると、機能名 ENVIRONMENT-VALUE に関連する呼び名を伴う後続の ACCEPT または DISPLAY 中で読み取るかまたは設定すべき変数が、定数-1 または一意名-1 の内容として指定された変数名に設定される。ON EXCEPTION 指定は、指定されていても実行されない。したがって、ENVIRONMENT-VALUE に関連する後続の ACCEPT または DISPLAY 用の ENVIRONMENT-NAME を設定しようとしている最中に問題が発生した場合、ON EXCEPTION 条件の検出は、後続の ACCEPT または DISPLAY の中で行なわなければならない。
7. (MF)(OPEN)機能名 ENVIRONMENT-VALUE に関連する呼び名を使用すると、下記のようになる。
   a. ENVIRONMENT-NAME に関連する呼び名を伴う DISPLAY が前に実行されていると、一意名の値が指定された環境変数に入れられる。
   b. ENVIRONMENT-NAME に関連する呼び名を伴う DISPLAY が前に実行されていないと、または ACCEPT に関しては指定された環境変数が存在しないと、または DISPLAY に関して環境変数を格納するために割り当てられる領域が足りないと、ON EXCEPTION 指定に対応する無条件文が実行される。この場合、一意名中の値はどうなるかわからない。
8. (MF)(OPEN)機能名 SYSERR に関連する呼び名を使用すると、DISPLAY は CONSOLE に対するように行われるが、出力はすべてシステムの標準エラー装置に向けられる。
9. (MF)(OPEN)DISPLAY UPON ENVIRONMENT-VARIABLE を実行するときは、後に続く空白も値のうちに含められる。これを避けたければ、INSPECT REPLACING ALL SPACES BY LOW-VALUES を使用する。
10. (OSVS)(VSC2)(MF)一意名-1 においては、内部浮動小数点数は表示用の外部浮動小数点数に変換される。その結果、下記のようになる。

a. COMP-1 項目は外部浮動小数点数用の PICTURE 句-.9(8)E-99 が指定されているように表示される。
b. COMP-2 項目は外部浮動小数点数用の PICTURE 句-.9(18)E-99 が指定されているように表示される。(この PICTURE 文字列を実際に明示的に使用するとまちがいとなる。)

11. OSVS VSC2 MF 定数-1 においては、浮動小数点数定数が許される。

形式 2

12. MF この書き方の DISPLAY 文を実行すると、プログラムの画面節内に定義されている画面項目が表示され、拡張画面操作機能によって全面的に取り扱えるようになる。

形式 2 および 3

13. MF AT 指定は、DISPLAY 処理を開始する画面上の絶対番地を指定する。
14. MF DISPLAY 文が実行される順序は、つねに下記のとおりである。
    1. AT 指定
    2. BLANK 指定
    3. BELL 指定
    4. DISPLAY 処理
15. MF 整数-3 または整数-4 の長さが 4 桁ならば、上 2 桁は行を表わし下 2 桁はカラムを表わす。整数-3 または整数-4 の長さが 6 桁ならば、上 3 桁は行を表わし下 3 桁はカラムを表わす。
16. MF 行番号とカラム番号の組合せの中には、特殊な意味をもつものがある。それらは下記のとおり。
    a. カラムの値が 1 行のカラム数よりも大きいと、その値から 1 行のカラム数が差し引かれ、行数に 1 が加えられる。
    b. 行の値が画面の範囲から外れると、画面が 1 行上にスクロールされる。この効果は、画面の下端の行を指定したのと同じである。
    c. 与えられた行番号とカラム番号がともにゼロであると、形式 2 または 3 の前の DISPLAY 処理が終わった位置の直後から、DISPLAY が開始される。各行のカラム 1 は、前の行の最後のカラムに続くものとみなされる。
    d. 指定された行番号がゼロであるがカラム番号はゼロでないと、形式 2 または 3 の前の DISPLAY 処理が終わった位置の次の行の指定されたカラムから、DISPLAY が開始される。
    e. 指定されたカラム番号がゼロであるが行番号はゼロでないと、形式 2 または 3 の前の DISPLAY 処理が終わった位置の次のカラムの指定された行から、DISPLAY が開始される。

形式 3

17. MF この文の一部を繰り返し指定して、いくつものデータ項目を表示するようにできる。最初の一意名に AT 指定を指定しないと、行 1、カラム 1 から表示が開始される。それ以降に続く各データ項目は、前のデータ項目を表示し終わった後のカーソルの位置から表示が開始される。

18. (MF)MODE IS BLOCK 指定を指定しないと、一意名が集団項目であれば、それに属する基本項目で名前が FILLER でないものが表示される。これらの項目はデータ部内で記述されている順に画面上に表示され、集団内の FILLER の長さによって区切られる。この目的では、ある行の最初の位置は、その前の行の最後の位置の直後に続くものとみなされる。
19. (MF)MODE IS BLOCK 指定は、一意名を基本項目として扱うように指定する。こうすると、一意名が集団項目であっても、1 つの項目として表示される。
20. (MF)WITH 指定を用いると、各種のオプションを DISPLAY 処理の実行時に指定できる。(これらの指定の説明については、*データ部 - 画面節* の章を参照。)

    (MF)画面記述句に指定できるオプションの他に、WITH 指定にはいくつかのオプションが加えられている。SPACE-FILL, ZERO-FILL, LEFT-JUSTIFY, RIGHT-JUSTIFY, TRAILING-SIGN, UPDATE である。ZERO-FILL には 2 つの用法があるため、このリストにも画面記述句にも現れている。2 番目の用法については、この章で後述する。

    (MF)自由方式で、数字および数字編集の画面項目に、データを入力できるようにする構成オプションがある。COBOL では、非編集数字データ項目は、内部形式でデータを保持することを目的としたものである。自由方式を用いると、このようなデータ項目を画面上に表示できる。 詳細は、ユーザーインターフェイスに関する COBOL システムのマニュアルを参照。自由方式がとられているときは、データは自動的に下記のように編集されて表示される。

    - 仮想小数点を、終止符で表わす。
    - 符号を、符号文字で表わす (負の数は "-" で、正の数は空白で) 。符号は、左端の数字の直前に付けられる。
    - すべての整数文字位置の先行ゼロが抑制される。ただし、最末桁を除く。
    - 左に桁寄せされる。

    (MF)SPACE-FILL、ZERO-FILL、LEFT-JUSTIFY、RIGHT-JUSTIFY および TRAILING-SIGN の各オプションは、上記の形式を補正する。

21. (MF)一意名-1 に表意定数を指定すると、次のような特別な効果がある。SPACE を指定すると、指定したカーソル位置から画面の末尾までがクリアされる。LOW-VALUE を指定すると、指定した位置にカーソルが移動される。ALL X"01" を指定すると、指定したカーソル位置からその行の末尾までがクリアされる。ALL X"02" を指定すると、画面全体がクリアされる。ALL X"07" を指定すると、警報が鳴らされる。一意名に上記以外の表意定数を指定し、SIZE オプションを指定しないと、その表意定数の値が 1 つ表示される。
22. (MF)特別の効果をもたない表意定数に SIZE オプションを指定すると、その表意定数の値が SIZE 分の長さだけ表示される。ただし、行末から次の行頭へ表示が繰り越されるときは、表意定数の最初から表示し直される。
23. (MF)FOREGROUND-COLOR オプションを指定すると、画面全体がこの表示によってクリアされた場合にだけ、指定した色が省略時解釈の前景色となる。画面全体がクリアされるのは、BLANK SCREEN オプションを指定した時、または一意名-1 が SPACES で表示が行 1、カラム 1 から始まる時である。

24. (MF)BACKGROUND-COLOR オプションを指定すると、画面全体がこの表示によってクリアされた場合にだけ、指定した色が省略時解釈の背景色となる。
25. (MF)一意名-1 に REDEFINES がかかっている場合、再定義データ領域の最初の記述が使用され、以降の記述は無視される。OCCURS または入れ子になった OCCURS が指定されている場合、反復されるデータ項目はその反復回数分展開される。したがって、1 つの定義が多数の項目にわたって繰り返されることになる。

12.14 DIVIDE(除算)文

DIVIDE (除算) 文は、いくつかの数字データ項目を 1 つの数字データ項目で割って、商と 剰余をデータ項目の値として設定する。

一般形式

形式 1

```
DIVIDE {一意名-1} INTO {一意名-2 [ROUNDED]} ...
       {定数-1  }
  [ON SIZE ERROR 無条件文-1]
  [NOT ON SIZE ERROR 無条件文-2]
  [END-DIVIDE]
```

(ANS85)

形式 2

```
DIVIDE {一意名-1} INTO {一意名-2}
       {定数-1  }      {定数-2  }
        GIVING {一意名-3 [ROUNDED]} ...
  [ON SIZE ERROR 無条件文-1]
  [NOT ON SIZE ERROR 無条件文-2]
  [END-DIVIDE]
```

(ANS85)

形式 3

```
DIVIDE {一意名-1} BY {一意名-2}
       {定数-1  }    {定数-2  }
        GIVING {一意名-3 [ROUNDED]} ...
  [ON SIZE ERROR 無条件文-1]
  [NOT ON SIZE ERROR 無条件文-2]
  [END-DIVIDE]
```

(ANS85)

形式 4

```
DIVIDE {一意名-1} INTO {一意名-2}
       {定数-1  }      {定数-2  }
       GIVING 一意名-3 [ROUNDED]
 REMAINDER 一意名-4
 [ON SIZE ERROR 無条件文-1]
 [NOT ON SIZE ERROR 無条件文-2]      (ANS85)
 [END-DIVIDE]
```

形式 5

```
DIVIDE {一意名-1} BY {一意名-2}
       {定数-1  }    {定数-2  }
       GIVING 一意名-3 [ROUNDED]
 REMAINDER 一意名-4
 [ON SIZE ERROR 無条件文-1]          (ANS85)
 [NOT ON SIZE ERROR 無条件文-2]
 [END-DIVIDE]
```

構文規則

すべての形式

1. 各一意名は、基本数字項目とする。ただし、GIVING 指定または REMAINDER 指定に指定する一意名は、基本数字編集項目であってもよい。
2. 各定数は、数字定数とする。
3. 作用対象を合成したものは、REMAINDER データ項目を除くすべての受取り側データ項目の使用により、決定される。*手続き部*の章の*算術文*節を参照。
4. (OSVS)(VSC2)(MF)形式 1、2、または 3 のいずれかの場合、浮動小数点データ項目または定数は、数字データ項目または定数が指定できる箇所であればどこでも使用できる。形式 4 または 5 の場合、浮動小数点データ項目または定数は指定できない。

一般規則

すべての形式

1. *手続き部*の章の *ROUNDED 指定*、*ON SIZE ERROR 指定*、*算術文*, *作用対象の重なり*、および*算術文における複数個の答え*の各節、および *COBOL 言語の概念*の章の*明示範囲符と暗黙範囲符*節、*翻訳集団の概念*の章の*範囲明示文*節を参照。

形式 1

2. 形式 1 では、定数-1 または一意名-1 によって参照されるデータ項目の値が一時的データ項目に記憶される。次いで、一意名-2 の値がこの一時的なデータ項目の値で

割られる。そして、被除数 (一意名-2) の値が商で置き換えられる。同様に、一意名-2 の後ろに続く各被除数が、指定された順番に左から右に順次この一時的なデータ項目で割られる。

形式 2

3. 形式 2 では、一意名-2 または定数-2 が、定数-1 または一意名-1 のデータ項目の値で割られ、結果が一意名-3 によって参照される各データ項目に入れられる。

形式 3

4. When 形式 3 では、一意名-1 または定数-1 が、定数-2 または一意名-2 のデータ項目の値で割られ、結果が一意名-3 によって参照される各データ項目に入れられる。

形式 4 および 5

5. 形式 4 および 5 は、一意名-4 に剰余を入れたいときに使用する。COBOL では、剰余は、被除数から除数と商 (一意名-3) の積を差し引いた結果の値であると定義される。一意名-3 が数字編集項目として定義されていると、剰余を計算するために用いられる商には、編集されていない値を保持している中間項目が使用される。
ROUNDED を指定すると、剰余を計算するために用いられる商には、四捨五入ではなく切り捨てられた値を保持している中間項目が使用される。
6. 形式 4 および 5 では、REMAINDER データ項目 (一意名-4) の精度は上記の計算によって決まる。一意名-4 によって参照されるデータ項目の内容は、必要に応じて、小数点の位置合わせと切捨て (四捨五入ではない) が行われる。
7. 形式 4 および 5 において ON SIZE ERROR 指定を書くと、下記の規則が適用される。
    a. 桁あふれが商に発生した場合、剰余を計算しても意味がない。したがって、一意名-3 および一意名-4 によって参照される値は変更されない。

       剰余に桁あふれが発生した場合、一意名-4 によって参照される値は変更されない。.

8. しかし、算術演算で複数の結果を求める他の場合と同様、利用者は自分で分析を行って、実際に何が発生したのかを突き止める必要がある。

第 13 章 : 手続き部 - ENTER - INVOKE

13.1 ENTER(導入)文

ENTER (導入) 文は、同じプログラム内で 2 つ以上の言語を使用できるようにする。

(ANS85)ENTER 文は、ANSI'85 標準では廃要素に分類されており、ANSI 標準の次回の全面改訂の際に、削除される予定である。

(MF)この COBOL に組み込まれているすべての方言では、この構文は注記としての役割を果たすだけである。FLAGSTD 指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。

(XOPEN)標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、この動詞は明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内では、この文を使用するべきではない。

一般形式

```
ENTER 言語名 [手順名].
```

構文規則

1. 言語名および手順名には、任意の利用者語または英数字定数を書くことができる。

一般規則

1. この文は、注記としての役割を果たすだけである。他の言語を実際に呼び出すには、CALL 文を使用する。

13.2 ENTRY(入口)文

(OSVS)(OSC2)(MF)

ENTRY (入口) 文は、呼ばれた COBOL プログラムへの代替の入口点を設定する。

一般形式

形式 1

```
ENTRY 定数-1
[ USING { [BY REFERENCE] (データ名-1)...   (NF)
        { BY VALUE (データ名-2)...   (NF)  } ... ] .
```

形式 2

```
ENTRY 定数-1 [呼び名]                                                    (NF)
[ USING { [BY REFERENCE] { データ名-1 [DELIMITED [BY SIZE]] } ...
        {                { 型定義名-1                      }
        {                { ANY                             }
        { BY VALUE { データ名-2 } ...                                   } ...
        {          { 型定義名-2 }
        {          { ANY        }
  [REPEATED [整数-1 TO 整数-2]] [ { GIVING    } { データ名-3 } ] .
                                { RETURNING } { 型定義名-3 }
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - STICKY-LINKAGE - 同じプログラム内で異なる入口点を呼び出す間に、連絡節項目の番地を指定できる状態を維持するかどうかを制御する。
    - LITLINK - **.obj** ファイルを生成するときに、入口名の有効文字数を制御する。

構文規則

すべての形式

1. 他のプログラム内に入れ子になっているプログラム内では、ENTRY 文を使用できない。
2. 定数-1 は数字であってはならず、表意定数であってもならない。環境によっては、先頭の 8 文字だけが有効なものがある。

形式 1

3. 形式 1はプログラム定義中にのみ指定できる。
4. データ名-1 は連絡節、

(MF)ファイル節、および作業場所節

のいずれかに、レベルが 01 または 77 の項目として定義されていなければならない。

5. (MF)データ名-2 は連絡節、ファイル節、作業場所節のいずれかに、レベルが 01 または 77 の項目として定義されていなければならない。データ名-2 は字類は何であってもよいが、長さが8バイトを超えてはならない。

形式 2

6. (MF)形式 2 を使用できるのは、プログラム名段落に EXTERNAL 句が指定されているプログラム、つまり呼び出しプロトタイプ、の中だけである。
7. (MF)データ名-1 とデータ名-2 は連絡節内の 01 レベルのレコードとして定義されていなければならない。
8. (MF)型定義名-1、型定義名-2、型定義名-3 は、同じソース・ファイル中に TYPEDEF 句を用いて、プログラマによって用途が定義されたものとして、予め定義されていなければならない。

一般規則

すべての形式

1. 定数-1 の内容は入口名を表す。それはプログラムへの入口点を指しており、手続き部の開始時点での省略時の COBOL 入口点の代りの入口点となる。プログラム名段落中のプログラム名は省略時の COBOL 入口点を示すと考えられる。

   実行用の COBOL のコードが収められているプログラム・ファイルの名前は通常、基本的にプログラム名と同じであることに注意すること。呼出しが行われたときにプログラム名または入口名がまだメモリーにロードされていないと、ファイル名に基づいてプログラム・ファイルが探されるが、プログラム名に基づいて探されているように見える。そのような状況において、プログラム名を参照する呼出しは正常に行われるが、入口名を参照する呼出しはエラーとなる可能性がある。

   入口名は多くの場合にプログラム名に等しく、両者は名前の形成に関する共通の規則に従う。その規則は、*見出し部*の章の*プログラム名段落*節に記載されている。定数-1 はプログラム名段落の一般形式の定数-1 に対応する。

2. 定数-1 に入口名を指定した CALL 文を使用して呼ぶプログラムから呼ばれるプログラムを呼び出すと、ENTRY 文の後ろの次の実行可能な文に制御が移される。
3. 連絡節に宣言されているが ENTRY 文の USING 指定に宣言されていないデータ項目を参照できるのは、SET 文を実行してそれらのデータ項目をあるデータ項目にリンクしてある場合、またはコンパイラ指令の STICKY-LINKAGE が指定されている場合だけである。
4. USING 指定には最大で 62 個のデータ名を指定できる。
5. BY REFERENCE 指定と BY VALUE 指定は共に、別の BY REFERENCE 指定または BY VALUE 指定が出てくるまで、後続のパラメータに効力を及ぼす。最初のパラメー

タの前に BY REFERENCE 指定も BY VALUE 指定もない場合には、BY REFERENCE 指定があるものとみなされる。

形式 1

6. (MF)データ名-1 がファイル節または作業場所節の中のレベルが 01 または 77 の項目として定義されている場合、オブジェクト･プログラム動作は下記のようになる。具体的には、データ名-1 と同じ内容で連絡節内にデータ項目が宣言され、呼ばれるプログラム中の最初の文を実行する前にそのデータ項目の内容がデータ名-1 に転記するかのように動作する。初期プログラムでは、それらの値はそのプログラムの作業場所節のデータを初期化するさいに上書きされるので、呼ばれたプログラムからは利用できない。
7. 稼働しているランタイム要素が COBOL である場合、下記の規則が適用される。稼働しているランタイム要素が COBOL でない場合、どういうときに BY REFERENCE 指定または BY VALUE 指定を使用する必要があるかの詳細については、インターフェイスに関する COBOL システム･ドキュメンテーションを参照。
8. USING 指定はプログラムで使用される仮パラメータまたはプログラムに渡される任意の引数を指す。プログラムに渡された引数は、稼働しているソース要素によって、CALL 文の USING 指定を用いて識別される。2 つの名前リストの間の対応関係は、位置に基づいて確立される。
9. 引数が内容によって渡される場合、呼ばれるプログラムの動作は下記のようになる。具体的には、呼出しを開始するプロセスの間に呼ぶランタイム要素によって連絡節内のレコードが割り当てられ、そのレコードは呼び出すランタイム要素中の引数と同じ記憶領域を占めることはないかのように動作する。この割り当てられたレコードの長さは引数とちょうど同じ文字数である。その引数がその割り当てられたレコードに何も変換せずに転記される。その後で、そのレコードは呼ばれたプログラムによって、それは引数であり参照によって渡されたかのように処理される。
10. 引数が参照によって渡された場合、仮パラメータが引数と同じ記憶領域を占めるかのように、呼ばれるプログラムは動作する。
11. (MF)引数が値によって渡される場合には、呼ばれるパラメータの動作は下記のようになる。具体的には、呼出しを開始するプロセスの間に呼ぶランタイム要素によって連絡節内のレコードが割り当てられ、そのレコードは呼び出すランタイム要素中の引数と同じ記憶領域を占めることはないかのように動作する。この割り当てられたレコードの長さは引数とちょうど同じ文字数である。その引数がその割り当てられたレコードに何も変換せずに転記される。その後で、そのレコードは呼ばれたプログラムによって、それは引数であり参照によって渡されたかのように処理される。
12. 稼働している要素においては常に、データ名-1、

    (MF)データ名-2 およびデータ名-3

    への参照は連絡節中の記述に基づいて解明される。その連絡節中のデータ項目に定義されている文字数が、稼働している要素中の対応するデータ項目よりも多いと、結果はどのようになるか分からない。この規則に外れるか、具体的なランタイム環境において許されるシステム領域の最大許容サイズを超えるかすると、システムは壊滅的な障害を起こす可能性がある。

13. (MF) プログラムが呼ばれ、USING 指定中の BY REFERENCE の作用対象が呼ぶプログラム中のパラメータに対応する場合、参照のリンクが確立される。そのリンクは制御が呼んだプログラムに戻されるまで持続する。その後、そのプログラムを取り消すことなく、再びそのプログラムを呼んだ場合、同じ BY REFERENCE の作用対象が呼んだプログラムのパラメータに対応しなかったならば、その作用対象を参照してはならない。ただし、コンパイラ指令の STICKY-LINKAGE を指定した場合を除く。

13.3 EVALUATEl(評価)文

(ANS85)

EVALUATE (評価) 文は、多くの分岐点や合流点がある構造を記述しておいて、複数の条件を評価できるようにする。 その評価の結果によって、ランタイム要素のその後の動きが決まる。

一般形式

```
EVALUATE { 一意名-1 | 定数-1 | 式-1 | TRUE | FALSE } [ ALSO { 一意名-2 | 定数-2 | 式-2 | TRUE | FALSE } ] ...

{ { WHEN
  { ANY
    条件-1
    TRUE
    FALSE
    [NOT] { 一意名-3 | 定数-3 | 算術式-1 } [ { THROUGH | THRU } { 一意名-4 | 定数-4 | 算術式-2 } ]
    部分式-1 (MF)
  }
  [ ALSO { ANY
           条件-2
           TRUE
           FALSE
           [NOT] { 一意名-5 | 定数-5 | 算術式-3 } [ { THROUGH | THRU } { 一意名-6 | 定数-6 | 算術式-4 } ]
           部分式-2 (MF)
         } ] ... } ...

無条件文-1} ...

[WHEN OTHER 無条件文-2]

[END-EVALUATE]
```

構文規則

1. EVALUATE 文の最初の WHEN 指定の前に現れる作用対象または語 TRUE と FALSE を、個別には選択の左辺といい、総称して選択の左辺の組という。
2. EVALUATE 文の最初の WHEN 指定の中に現れる作用対象または語 TRUE と FALSE と ANY を、個別には選択の右辺といい、総称して選択の右辺の組という。
3. THROUGH と THRU は同義である。
4. THROUGH 指定によってつないだ 2 つの作用対象は、同じ字類のものにする。このように THROUGH 指定によってつないだ 2 つの作用対象は、単一の選択の右辺を構成する。
5. 選択の右辺の組に含まれる選択の右辺の数は、選択の左辺の数と等しくする。
6. 選択の右辺の組に含まれる各選択の右辺は、選択の左辺の組に含まれる同じ順番の位置にある選択の左辺と対応させる。これには、下記の規則が適用される。
    a. 選択の右辺の中に現れる一意名、定数、算術式は、選択の左辺中の対応する作用対象と比較する作用対象として、有効なものにする。(*手続き部*の章の*比較条件*節を参照。)
    b. 選択の右辺として現れる条件-1、条件-2 あるいは語 TRUE または FALSE は選択の左辺の組の中の条件式または語 TRUE または FLASE と対応させる。
    c. 語 ANY は、任意の種類の選択の左辺と対応できる。
    d. (VSC2)(MF)一意名を使えるところでは、浮動小数点数データ項目を参照できる。
    e. (VSC2)(MF)数字定数を使えるところでは、浮動小数点数定数を参照できる。
    f. (VSC2)(MF)一意名を使えるところでは、ポインタ･データ項目を参照できる。
7. (SOMM)(MF)選択の左端部分が相対作用要素、一意名を含まない字類条件、一意名を含まない記号条件、および数式を含まないである記号条件のいずれかである場合は、選択は、部分式である。
8. (SOMM)(MF)選択の右辺として部分式-1 または部分式-2 を指定する場合、これらは一意名か定数か算術式である選択の左辺と対応させる。部分式-1 および部分式-2 は一連の COBOL の語であり、対応する選択の左辺の後ろに続けたときに、有効な条件式となるものでなければならない。

一般規則

1. EVALUATE 文を実行すると、まず、選択の左辺と選択の右辺が評価されて、1 つの数値、1 つの文字値、数値の範囲、文字値の範囲、真理値のどれかが割り当てられる。これらの値は下記のように決定される。
    a. 一意名-1 および一意名-2 によって指定される選択の左辺、および一意名-3 および一意名-5 によって指定される右辺のうちで、NOT 指定または THROUGH 指定を伴わないものは、その一意名によって参照されるデータ項目の値と字類を割り当てられる。
    b. 定数-1 および定数-2 によって指定される選択の左辺、および定数-3 および定数-5 によって指定される右辺のうちで NOT 指定も THROUGH 指定も伴わないものは、その定数の値と字類を割り当てられる。定数-3 と定数-5 が表意定数の ZERO であると、それらには対応する選択の左辺の字類が割り当てられる。
    c. 算術式として式-1 および式-2 を指定された選択の左辺、および算術式-1 または算術式-3 が含まれる NOT 指定も THROUGH 指定も伴わない選択の右辺は、算術式の評価規則に従って数値を割り当てられる。(*手続き部*の章の*算術式*節を参照。)

d. 条件式として式-1 および式-2 を指定された選択の左辺、および条件-1 と条件-2 を指定された選択の右辺は、条件式の評価規則に従って真理値を割り当てられる。(*手続き部*の章の*条件式*節を参照。)
e. 語 TRUE または FALSE を指定された選択の左辺または選択の右辺は、真理値を割り当てられる。語"true" が指定された項目は、真理値の「真」を割り当てられ、語"false" が指定された項目は、真理値の「偽」を割り当てられる。
f. 語 ANY を指定された選択の右辺は、それ以上評価されない。
g. 選択の右辺に NOT 指定を付けずに THROUGH 指定を書くと、選択の左辺と比較したときの値の範囲は、比較の規則に基づいて、最初の作用対象以上で 2 番目の作用対象以下となる。(選択の右辺に NOT 指定を付けずに THROUGH 指定を書くと、選択の左辺と比較したときの値の範囲は、比較の規則に基づいて、最初の作用対象以上で 2 番目の作用対象以下となる。(*手続き部*の章の*比較条件*節を参照。) 最初の作用対象の方が 2 番目の作用対象よりも大きいと、その範囲には値は何も含まれない。の*比較条件*の節を参照。) 最初の作用対象の方が 2 番目の作用対象よりも大きいと、その範囲には値は何も含まれない。
h. 選択の対象に NOT 指定を書くと、割り当てられる値は、その値と等しくないすべての値、または NOT 指定を指定しなかった場合に割り当てられる値以外のものとなる。

2. EVALUATE 文の実行は、選択の左辺と選択の右辺の比較へと進む。そして、選択の左辺の組の中に WHEN 指定を満たすものがあるかどうかが判定される。この比較は下記のように行われる。
   a. 最初の WHEN 指定の選択の右辺の組の中の各選択の右辺が、選択の左辺の組の中の順序位置が同じ選択の左辺と比較される。
      1. 比較対象の項目に割り当てられている値が数値、文字値、数値の範囲、文字値の範囲である場合は、選択の右辺に割り当てられている値または値の範囲内のどれかの値が、比較規則に照らして選択の左辺に割り当てられている値と等しければ、その比較条件は満足される。(*手続き部*の章の*比較条件*節を参照。)
      2. 比較対象の項目に割り当てられている値が真理値である場合は、選択の左辺と選択の右辺に同じ真理値が割り当てられているときに、その比較条件は満足される。
      3. 比較対象の項目に割り当てられている値が語 ANY である場合は、その比較条件はつねに満足される。この場合は、選択の左辺の値は何であってもよい。
   b. 比較対象の選択の右辺の組に含まれる各選択の右辺に関して、上記の比較が満足されると、その選択の右辺の組が含まれる WHEN 指定が選択の左辺の組を満たすものの 1 つとして選択される。
   c. 比較対象の選択の右辺の組に含まれる選択の右辺の中に、上記の比較が満足しないものがいくつかあると、その選択の右辺の組は選択の左辺の組を満たさない。
   d. 後続の選択の右辺の組に対して、上記の比較処理が、原始要素の記述されている順序で繰り返し実行される。そして、選択の左辺の組を満たす WHEN 指定が選択されるか、またはすべての選択の右辺の組を比較し尽くした時点で、この処理は終わる。
3. 比較処理が終わると、EVALUATE 文の実行は下記のように進む。

a. WHEN 指定が選択された場合、その WHEN 指定の後ろに続く最初の無条件文-1 に処理は進む。
b. WHEN 指定が選択されなかった場合、WHEN OTHER 指定が指定されていれば、無条件文-2 に処理は進む。
c. EVALUATE 文の実行は、選択された WHEN 指定または WHEN OTHER 指定の最後まで到達した時、あるいは選択された WHEN 指定がなく WHEN OTHER 指定が指定されていない時に終了する。(*COBOL 言語の概念*の章の*明示範囲符と暗黙範囲符*節を参照。)

4. (ISO2002)(MF)選択の右辺が部分式-1 または部分式-2 によって指定されている場合、対応する選択の左辺は語 TRUE であるものとみなされる。選択の右辺はそれぞれ条件-1 または条件-2 とみなされる。ここで、条件-1 または条件-2 はそれぞれ、元の対応する選択の左辺-1 または選択の左辺-2 に続く部分式-1 または部分式-2 から導出される条件式である。

13.4 EXAMINE(検査)文

(OSVS)

EXAMINE (検査) 文は、データ項目中に現れる指定された文字を置き換えたり、数を数えたりする。

一般形式

形式 1

```
EXAMINE 一意名 TALLYING
                 { UNTIL FIRST }
                 { ALL         }
                 { LEADING     }  定数-1 [REPLACING BY 定数-2]
```

形式 2

```
EXAMINE 一意名 REPLACING { ALL         }
                         { LEADING     }
                         { FIRST       }  定数-1 BY 定数-2
                         { UNTIL FIRST }
```

構文規則

すべての形式

1. 一意名は、明示的または暗黙的に、USAGE IS DISPLAY と記述しておく。
2. 各定数は、1 文字で構成させる。一意名が数字項目である場合、定数は数字定数、値が単一の数字である文字定数、表意定数の ZERO のどれかとする。一意名の字

類がそれ以外の場合は、定数は数字でも文字でも語 ALL 以外の任意の表意定数であってもよい。

一般規則

すべての形式

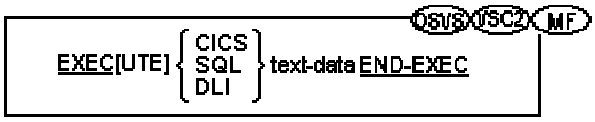
1. 検査は、下記のように行われる。
    a. 文字データ項目に関しては、検査は左端の文字から開始され、右方向に進められる。一意名に指定したデータ項目中の各文字が、順々に検査される。
    b. EXAMINE 文の対象とするデータ項目が数字の場合は、演算符号が付いてもよい。検査は符号を除く左端の文字から開始され、右方向に進められる。符号を除く各文字が順々に検査される。符号はどこに付いていても、EXAMINE 文によって完全に無視される。
2. TALLYING オプションを指定すると、数が数えられて、TALLY と呼ばれる特殊レジスタの値が置き換えられる。この数は、下記の事柄を表わす。
    a. UNTIL FIRST オプションを指定した場合は、最初に定数-1 が見つかるまでに出て来る定数-1 以外の文字の数。
    b. ALL オプションを指定した場合は、定数-1 が出て来る数。
    c. LEADING オプションを指定した場合は、最初に定数-1 以外の文字が見つかるまでに出て来る定数-1 の数。
3. REPLACING オプションを指定した場合は、下記の規則に基づいて置き換えが行われる。
    a. ALL オプションを指定すると、出て来る定数-1 はすべて定数-2 に置き換えられる。
    b. LEADING オプションを指定すると、定数-1 以外の文字が出て来るか、データ項目の右端に達したときに、定数-2 による置き換えが終了される。
    c. FIRST オプションを指定すると、最初に出て来る定数-1 が定数-2 に置き換えられる。
    d. UNTIL FIRST オプションを指定すると、定数-1 が出て来るか、データ項目の右端に達したときに、定数-2 による置き換えが終了される。

13.5 EXEC(UTE)(実行)文

EXEC (UTE) (実行) 文は、COBOL 以外のサブシステムに制御を渡すための連絡機構として設けられている。

一般形式

形式 1



形式 2

EXEC [UTE] 原文名　原文データ　END-EXEC

構文規則

1. 原文データは、文字列 END-EXEC を含まなければ、どんなテキストデータであってもよい。

一般規則

形式 1

1. 形式 EXEC(UTE) CICS、EXEC(UTE) SQL、および EXEC(UTE) DLI により、適切なシステムコンポーネント (プリプロセッサ、コプロセッサ、外部コンパイルモジュールなどと呼ばれる) が呼ばれ、原文データを処理する。ただし、対応するコンパイラ指令が指定され、そのシステムコンポーネントを起動する場合に限る。対応するコンパイラ指令が指定されていない場合は、EXEC(UTE) 文が形式 1 EXEC(UTE) 文として扱われる。これらの形式の使用について詳しくは、COBOL システムのマニュアルで CICS の使用およびデータベースへのアクセスについての情報を参照。

形式 2

2. この文は、「CALL " テキスト名" USING テキスト・データ・バッファ」文に翻訳される。(*手続き部 - ACCEPT - DIVIDE* の章の *CALL(呼出し)文*節を参照。) ここで、テキスト・データ・バッファには EXEC 文のすべてのテキスト・データが入れられ (空白は圧縮される) 、これが呼ばれるプログラムである「テキスト名」によってさらに解析される。

   このことは、呼ばれる副プログラム (テキスト名) に、テキスト文字列が渡されることを意味する。このテキスト文字列は、動詞 EXEC(UTE)で始まり END-EXEC 指定で終わる間のすべての COBOL 文が含まれる。呼ばれるプログラムは、渡されるすべての情報を解析して処理しなければならない。

   テキスト・データ内に一意名が含まれている場合、これは副プログラムに渡される一意名を表わしているのであって、データの値を表わしているのではない。

13.6 EXHIBIT(参照)文

OS/VS　MF

EXHIBIT (参照) 文は、指定された定数や一意名の内容を表示する。条件付きで表示させることもできる。一意名の場合は、内容の前に名前を表示させることができる。

一般形式

```
EXHIBIT { NAMED | CHANGED NAMED | CHANGED } { 一意名-1 | 定数-1 } ...
```

構文規則

1. EXHIBIT 文中に指定した各一意名は、どんな字類のデータであってもよい。一意名として使用できる特殊レジスタは、TALLY および RETURN-CODE のみである。
2. CHANGED と NAMED は、両方とも省略できる。

一般規則

1. EXHIBIT 文によって表示される定数および一意名は、表示行上では 1 つの空白によって区切られる。
2. 定数には、ALL 以外の表意定数を指定できる。
3. 定数に数字を指定する場合は、符号の付かない整数とする。
4. EXHIBIT NAMED 文を実行すると、指定した各一意名または定数が表示される。一意名の場合は、その名前（修飾語および添字があれば、それを含む）の後に"="（等号）を付けて、その後ろに現在の値が表示される。表示される一意名または定数は、EXHIBIT 文に指定された順番に、1 行に並べられる。
5. EXHIBIT CHANGED NAMED 文を実行すると、指定した各一意名または定数が表示される。一意名の場合は、その名前（修飾語および添字があれば、それを含む）の後に"="（等号）を付けて、その後ろに現在の値が表示される。表示される一意名または定数は、EXHIBIT 文に指定された順番に、1 行に並べられる。ただし、表示される一意名（名前と値）は、前回この EXHIBIT 文を実行した後で値に変更があったものだけである。値が変更されなかった一意名については、その名前も値も表示されない。値が変更された一意名が 1 つもなく、定数も 1 つも指定されていないと、何も表示されない（空白行の表示は抑制される）。
6. EXHIBIT CHANGED 文を実行すると、指定した各一意名または定数の現在の値が表示される。表示される一意名または定数は、EXHIBIT 文に指定された順番に、1 行に並べられる。ただし、表示される値は、前回この EXHIBIT 文を実行した後で値に変更があったものだけである。値が変更されなかった一意名については、その値は表示せず、代わりに空白が挿入される。値が変更された一意名が 1 つもなく、定数も 1 つも指定されていないと、空白行が表示される（空白行の表示は抑制されない）。
7. CHANGED オプションも NAMED オプションも指定せずに EXHIBIT 文を実行すると、指定した各一意名または定数が表示される。表示される一意名または定数は、EXHIBIT 文に指定された順番に、1 行に並べられる。

13.7 EXIT(出口)文

EXIT（出口）文は、一連の手続きの共通の終了点を設ける。

(MF) EXIT 文は、内 PERFORM、段落、または節の出口とすることもできる。

EXIT PROGRAM 文は、呼ばれたプログラムの論理的な終点を示す。

(MF)任意の言語で書かれた呼び出し元のランタイム要素に、値を返すこともできる。

(SO2002)(MF)EXIT METHOD 文は、起動されたメソッドの論理的な終点を示す。

(SO2002)EXIT FUNCTION 文は、関数の実行の論理的な終点を示す。

一般形式

形式 1

```
EXIT
```

形式 2

```
EXIT PERFORM [CYCLE]                                   (MF)
```

形式 3

```
EXIT { PARAGRAPH }                                     (MF)
     { SECTION   }
```

形式 4

```
EXIT PROGRAM [ { GIVING    } { [ADDRESS OF]一意名-1 } ]
               { RETURNING } { 整数-1               }
                                                (MF)
```

形式 5

```
EXIT METHOD    (SO2002)(MF)
```

形式 6

```
EXIT FUNCTION  (SO2002)
```

構文規則

形式 1

1. EXIT 文は、それだけで完結文を構成しなければならない。

   VSC2 MF この規則は強制しない。

2. EXIT 文は、段落中の唯一の完結文でなければならない。

   VSC2 MF この規則は強制しない。

形式 2

3. 形式 2 の EXIT 文は、内 PERFORM の中だけで使用できる。

形式 4

4. 完結文内の一連の無条件文の中に EXIT PROGRAM 文を指定する場合、これをその最後の文としなければならない。

   VSC2 MF この規則は強制しない。

5. EXIT PROGRAM 文を含む段落には、それ以外の完結文を含めてはならない。

   ANS85 この規則は強制しない。

6. MF 整数-1 は符号付きでもよい。
7. MF GIVING 指定と RETURNING 指定は同義である。
8. MF ADDRESS OF 句を指定しなかった場合、一意名-1 のサイズは 8 バイトを超えてはならない。

形式 5

9. ISO/IEC 1989:2002 MF EXIT METHOD 文を指定できるのは、メソッド手続き部の中だけである。

形式 6

10. ISO/IEC 1989:2002 EXIT FUNCTION を指定できるのは、関数手続き部の中だけである。

形式 4、5 および 6

11. 関連する USE 文中に GLOBAL 指定が書かれている宣言手続中に EXIT 文を指定してはならない。

一般規則

形式 1

1. EXIT 文は、手続き部中のある点に、利用者が手続き名を付けられるようにするだけである。 この形式の EXIT 文には、翻訳および実行に関して、それ以外の働きはない

形式 2

2. (MF)CYCLE 指定を伴わない EXIT PERFORM 文を実行すると、最も近い内 PERFORM 文に対応する END-PERFORM のすぐ後ろに続く、暗黙の CONTINUE 文に制御が移される。
3. (MF)CYCLE 指定を伴う EXIT PERFORM 文を実行すると、最も近い内 PERFORM 文に対応する END-PERFORM の直前にある、暗黙の CONTINUE 文に制御が移される。

形式 3

4. (MF)PARAGRAPH 指定を伴う形式 3 の EXIT 文を実行すると、段落の最後の文の直前にある、暗黙の CONTINUE 文に制御が移される。
5. (MF)SECTION 指定を伴う形式 3 の EXIT 文を実行すると、節の最後の段落にある最後の文の直前にある、暗黙の CONTINUE 文に制御が移される。

形式 4

6. (ANS85)呼び出し側のランタイム要素の制御下にないプログラム内で EXIT PORGRAM 文を実行すると、次の実行可能な文に制御が移されて、処理が続行される。
7. (ANS85)初期属性を持たない呼ばれたプログラム内で EXIT PROGRAM 文を実行すると、呼んだランタイム要素の中の CALL 文の次の実行可能な文に制御が移されて、処理が続行される。呼んだランタイム要素の状態は CALL 文を実行したときと同じである。ただし、呼んだランタイム要素と呼ばれたランタイム要素の間で共有されるデータ項目およびデータ･ファイルの内容は変更されている可能性がある。呼ばれたプログラムの状態も変更されない。ただし、呼ばれたプログラムによって実行されたすべての PERFORM 文は、その範囲の末尾に達したものとみなされる。
8. (ANS85)初期属性をもつ呼ばれたプログラム内で EXIT PROGRAM を実行することは、上記の 7 の処理に加えて、そのプログラムを対象とする CANCEL 文を実行することに等しい。ただし、メモリーの解放は行われない。*手続き部 - ACCEPT - DIVIDE* の章の *CANCEL(取消し)文*節を参照。)
9. (ANS85)GLOBAL 句が指定されている宣言手続きが実行されている最中に、EXIT PROGRAM 文を実行してはならない。ただし、その宣言手続きの実行中に呼ばれたプログラム内で、EXIT PROGRAM 文を実行することは構わない。
10. (MF)呼んだランタイム要素の制御下にあるプログラム内の EXIT PROGRAM 文を実行すると、システム領域に戻り値が設定される。このシステム領域は一般に COBOL 以外のプログラムが値を返すために利用できる。そのサイズは4バイトを下回ることはなく、環境によってはそれよりも大きいことがある。

    (MF)GIVING 指定を書かず、呼出し方式に特殊レジスタの RETURN-CODE (*環境部* の章の *特殊名段落*節の呼出し方式を参照) を更新するように指定した場合、オブジェクト･プログラム動作は下記のようになる。具体的には、システム領域が COBOL の

数字データ項目であり、それに USAGE COPM-5 が指定されていて、そのサイズは COBOL システムの外の操作環境によって決定されるものとして宣言されている状態で、RETURN-CODE を送出し側としシステム領域を受取り側として、MOVE 文を実行したかのように動作する。(RETURN-CODE の詳細については、*COBOL 言語の概念* の章の *特殊レジスタ* 節を参照。)

(MF) GIVING ADDRESS OF 指定を書いた場合、オブジェクト・プログラム動作は下記のようになる。具体的には、システム領域が USAGE POINTER と指定された COBOL のデータ項目として宣言され、「ADDRESS OF 一意名-9」を最初の作用対象としシステム領域を 2 番目の作用対象として、SET 文を実行したかのように動作する。システム領域よりも大きな値を間接的に渡すために、ADDRESS OF 指定を使用できる。

(MF) 「GIVING 一意名-1」指定を書いた場合、一意名-1 はシステム領域に戻り値を収めるのに必要な長さがあり、その型と用途は呼んだランタイム要素によって期待されるのものでなければならない。一意名-1 を送出し側項目とし、システム領域を受取り側項目として、MOVE 文を実行したかのように、オブジェクト・プログラムは動作する。

(MF) 「GIVING 整数-1」指定を書いた場合、整数-1 の値はシステム領域に保持可能なよりも大きくてはならない。オブジェクト・プログラム動作は下記のようになる。具体的には、システム領域が COBOL の数字データ項目であり、それに USAGE COPM-5 が指定されていて、そのサイズは COBOL システムの外の操作環境によって決定されるものとして宣言されている状態で、整数-1 を送出し側としシステム領域を受取り側として、MOVE 文を実行したかのように動作する。

形式 5

11. (2002) (MF) EXIT METHOD 文を実行すると、実行中のメソッドが停止され、呼出し元の文に制御が戻される。呼出し元のメソッド定義に RETURNING 指定があると、RETRUNING 指定によって参照されたデータ項目中の値がそのメソッド呼出しの結果となる。

形式 6

12. (2002) EXIT FUNCTION 文を実行すると、実行中の関数が停止され、呼出し元の文に制御が戻される。親の関数に含まれる RETURNING 指定により参照されるデータ項目の値が、関数呼出しの結果となる。

13.8 GOBACK(復帰)文

(OSVS) (OSC2) (MF)

GOBACK (復帰) 文は、呼ばれたプログラムの論理的な終点を示す。

(MF) 任意の言語で書かれている呼び出し元のランタイム要素またはオペレーティングシステム環境へ、値を返すこともできる。

一般形式

```
GOBACK [ { GIVING    } { [ADDRESS OF] 一意名-1 } ] (MF)
         { RETURNING } { 整数-1               }
```

構文規則

1. GOBACK 文は、一連の無条件文の最後の文とする。

   この規則は強制されないが、その場合 GOBACK の後ろに続く文は実行されない。

2. (MF)GIVING 指定と RETURNING 指定は同義である。
3. (MF)ADDRESS OF 句を指定しない場合、一意名-1 によって参照されるデータ項目のサイズは 8 バイトを超えてはならない。
4. 整数-1 は符号付きでもよい。

一般規則

1. 呼び出したランタイム要素の制御下にあるプログラム内で GOBACK 文を実行した場合、GOBACK 文と同じ句が指定されている EXIT PROGRAM 文を実行したかのように、オブジェクト・プログラムは動作する。*EXIT(出口)文*節を参照。
2. 呼び出した ランタイム要素の制御下にないプログラム内で ADDRESS OF 句を指定せずに GOBACK 文を実行した場合、GOBACK 文と同じ句が指定されている STOP RUN 文を実行したかのように、オブジェクト・プログラムは動作する。
3. 呼び出した ランタイム要素の制御下にないプログラム内で ADDRESS OF 句を指定して GOBACK 文を実行した場合、STOP RUN 文を実行したかのように、オブジェクト・プログラムは動作する。ただし、システム領域に設定される戻り値は不定であることが異なる。GOBACK 文を使用する方が、同内容の EXIT PROGRAM 文や STOP RUN 文よりも、コードが簡潔になることが多い。
4. GLOBAL 句が指定されている宣言手続きが実行されている最中に、GOBACK 文を実行してはならない。ただし、その宣言手続きを実行中に呼ばれたプログラム内で GOBACK 文を実行することはかまわない。

13.9 GO TO(飛越し)文

GO TO (飛越し) 文は、手続き部のある部分から別の部分へ制御を移転させる。

(ANS85)形式 1 の GO TO 文中の手続き名-1 を省略することは、ANSI'85 標準では廃棄要素に分類されており、ANSI 標準の次回の全面改訂の際に削除される予定である。

(MF)この構文は、Micro Focus COBOL に組み込まれているすべての方言で全面的に使用できる。FLAGSTD 指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。

(XOPEN)A 標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、形式 1 で手続き名を省略することは明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内ではこの文を使用するべきではない。

一般形式

形式 1

```
GO TO [手続き名-1]
```

形式 2

```
GO TO 手続き名-1 [手続き名-2]...DEPENDING ON 一意名
```

構文規則

すべての形式

1. 一意名は、整数をとる数字基本項目の名前である。
2. ALTER (変更) 文によって参照される段落は、段落の見出しとその後ろに書いた書き方 1 の GO TO 文だけで構成しなければならない。

形式 1

3. 手続き名-1 を伴わない形式 1 の GO TO 文は、単一の文からなる段落内にだけ書ける。

   (MF)この規則は強制しない。

4. 完結文の中の一連の無条件文の 1 つとして、形式 1 の GO TO 文を使用する場合は、その最後の文とする。

形式 2

5. 形式 2 では、少なくとも 2 つの手続き名を書く必要がある。

   (ANS85)形式 2 で手続き名が 1 つだけであってもよい。

一般規則

すべての形式

1. 形式 1 で手続き名-1 を指定しない場合は、この GO TO 文を参照する ALTER 文を、この GO TO 文の実行に先立って実行しておく。

   VSC2 MF この GO TO 文を実行する前に、この GO TO 文が含まれる段階を参照する ALTER が実行されていない場合には、この GO TO 文は CONTINUE (継続) 文と同じに扱われる。

形式 1

2. 形式 1 の GO TO 文が実行されると、制御は手続き名-1 に移される。ただし、ALTER 文によってその GO TO 文が変更されている場合には、変更された手続き名に制御が移される。

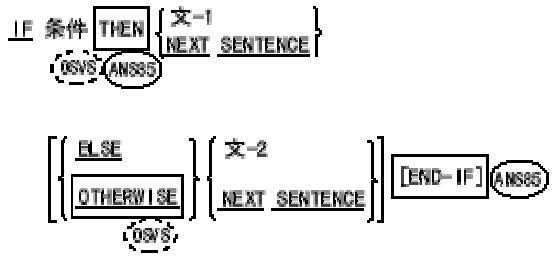
形式 2

3. 形式 2 の GO TO 文が実行されると、一意名の値が 1, 2, ... *n* のどれであるかに応じて、手続き名-1、手続き名-2 などに制御が移される。一意名の値が正または符号の付かない 1, 2, ... *n* でない場合は、制御の移転は行われず、通常の実行順序の次の文が実行される。

13.10 IF(判断)文

IF 文は、評価すべき条件を提示する (***手続き部***の章の*条件式*節を参照) 。 その結果が「真」であったか「偽」であったかによって、ランタイム要素の以降の制御の流れが変わってくる。

一般形式



構文規則

1. 文-1 と文-2 は、無条件文または条件文を表わす。条件文の前に無条件文を書いてもよい。
2. ANS85 END-IF 指定を書く場合、NEXT SENTENCE 指定を書いてはならない。

   VSC2 MF END-IF 指定と共に、NEXT SENTECE 指定を書いてもよい。

NEXT SENTENCE 指定が実行されると、制御は END-IF の次の文ではなく、後続の最も近い終止符 (.) の後ろに続く文に移される。

一般規則

1. IF 文の範囲は、下記のどれかによって区切られる。
    a. (ANS85)同じ入れ子レベルの END-IF 指定
    b. 分離符の終止符
    c. 入れ子になっている場合、上位レベルの IF 文に対応する ELSE 指定
2. IF 文が実行されると、下記の制御の移転が発生する。
    a. 条件が真のとき、文-1 を指定してある場合は、その文が実行される。文-1 に手続き分岐文または条件文が含まれると、その文の規則に従って明示的に制御が移転される。文-1 に手続き分岐文または条件文が含まれないと、ELSE 指定は指定してあっても無視されて、IF 文の末尾に制御が移される。
    b. 条件が真のとき、文-1 の代わりに NEXT SENTENCE 指定を指定してある場合は、ELSE 指定は指定してあっても無視されて、次の実行可能文に制御が移される。
    c. 条件が偽のとき、文-1 またはその代わりの NEXT SENTENCE 指定は無視され、文-2 が指定してあればそれが実行される。文-2 に手続き分岐文または条件文が含まれると、その文の規則に従って明示的に制御が移転される。文-2 に手続き分岐文または条件文が含まれないと、IF 文の末尾に制御が移される。ELSE 文-2 指定を指定してないと、文-1 は無視されて、IF 文の末尾に制御が移される。
    d. 条件が偽のとき、ELSE NEXT SENTENCE 指定を指定してある場合は、文-1 は指定してあっても無視されて、次の実行可能文に制御が移される。
3. 文-1 または文-2 のどちらか一方または両方に、IF 文を含めてもよい。この場合、IF 文は入れ子になっている、という。

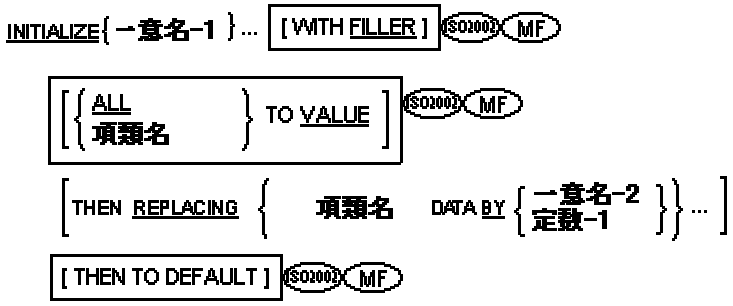
    (ANS85)IF 文の中に含まれる IF 文は、左から右方向に、IF と ELSE と END-IF の順に組になったものとみなされる。したがって、文中に出てくる ELSE は、まだ他の ELSE と組み合わされていないか、または明示的にも暗示的にも終了していない、最も近い先行する IF と組み合わされる。文中に出てくる END-IF は、明示的にも暗示的にも終了していない、最も先行する IF と組み合わされる。

13.11 INITIALIZE(初期化)文

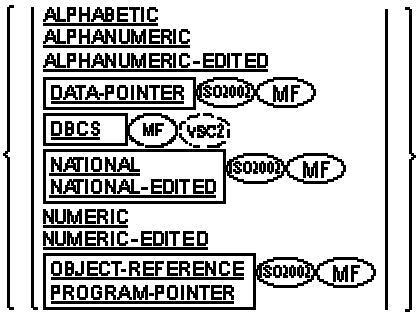
(ANS85)

INITIALIZE (初期化) 文は、選択されたデータ項目を指定された値に設定する。

一般形式



ここで 項類名 は:



構文規則

1. 一意名-1 の字類は、英字、英数字、数字、

   ISO2002 MF 各国語型、オブジェクト、またはポインタ

   でなければならない。

2. ISO2002 MF DATA-POINTER 指定、OBJECT-REFERENCE 指定、および PROGRAM-POINTER 指定のいずれかを REPLACING 指定で項類名として使用した場合は、一意名-2 を指定しなければならない。
3. ISO2002 MF REPLACING 指定で使用したデータポインタ、オブジェクト参照、およびプログラムポインタの各項類については、一意名-2 を含む SET 文が送出し側作用対象として、また、指定された項類の項目が受取り側作用対象として、有効でなければならない。（*手続き部 - SEARCH - WRITE* の章の *SET(設定)文*節を参照。）

   ANS85 REPLACING 指定で使用した、データポインタ、オブジェクト参照、およびプログラムポインタ以外の各項類については、一意名-2 または定数-1 を含む MOVE 文が送出し側作用対象として、また、指定された項類の項目が受取り側作用対象として、有効でなければならない。（*手続き部 - MERGE - OPEN* の章の *MOVE(転記)文*節を参照。）

4. REPLACING 指定を使用した場合は、定数-1 または一意名-2 が参照するデータ項目が送出し側作用対象となる。

(SOUND)(MF)REPLACING 指定を使用しない場合は、INITIALIZE 文の一般規則に従って、送出し側作用対象が決定される。

5. 一意名-1 が参照するデータ項目は受取り側領域となる。
6. REPLACING 指定内では、同じ項類は繰り返し指定できない。
7. 一意名-1 によって参照されるデータ項目またはその下位に属するデータ項目のデータ記述には、OCCURS 句の DEPENDING 指定が含まれていてはならない。

   (VSC2)一意名-1 によって参照されるデータ項目の位置が可変であってはならない。つまり、同じレコード記述内にあって OCCURS 句の DEPENDING 指定があるデータ項目の後ろに、一意名-1 を置くことはできない。ただし、OCCURS 句のある項目の下位に一意名-1 が属する場合は別である。

   (MF)一意名-1 によって参照されるデータ項目は、位置が可変であってもよい。

8. 一意名-1 によって参照されるデータ項目のデータ記述項に、RENAMES 句を含めてはならない。
9. (VSC2)(MF)数字の一意名または定数を指定できるところでは、どこでも浮動小数点数のデータ項目または定数を使用できる。

一般規則

1. 項類名中のキーワードは、本書の中で定義したデータの項類に対応する。(***COBOL 言語の概念***の章の*データの字類と項類*節を参照。)

   (SOUND)(MF)VALUE 指定で ALL が示されている場合、項類名のリスト中のすべての項類が指定されていると見なされる。

2. 一意名-1 によって参照されるデータ項目が基本項目であっても集団項目であっても、INITIALIZE 文の実行の結果は、各文の受取り側作用対象が基本項目である、一連の MOVE 文または SET 文が実行されたものと同様である。これらの文の送出し側作用対象は一般規則 4 で、また、受取り側作用対象は一般規則 3 で定義される。

   受取り側作用対象の項類がデータポインタ、オブジェクト参照、またはプログラムポインタである場合の明示文は、以下の通り。

   SET 受取り側作用対象 TO 送出し側作用対象

   そうでない場合の明示文は、以下の通り。

   MOVE 送出し側作用対象 TO 受取り側作用対象

3. 暗黙的に指定された各 MOVE 文または SET 文中の受取り側作用対象は、以下の手順を順に実行することにより決定される。
   a. まず、以下のデータ項目が、受取り側作用対象としては除外される。

        1. MOVE 文の有効な受取り側作用対象でない、あらゆる一意名。ただし、項類がデータポインタ、オブジェクト参照、またはプログラムポインタであるデータ項目を除く。
        2. FILLER 指定が使用されない場合は、明示的または暗黙的な FILLER 句を含む基本データ項目。
        3. データ記述項に REDEFINES 句または RENAMES 句を含む一意名-1、またはデータ記述項に REDEFINES 句を含むデータ項目のいずれかの下位項目である、あらゆる基本データ項目。ただし、この一意名-1 自体が REDEFINES 句を含んでいてもよい。または、REDEFINES 句を含むデータ項目の下位項目であってもよい。
    b. 次に、以下のいずれかに該当する場合は、基本データ項目が受取り側作用対象となる可能性がある。
        1. 一意名-1 が明示的に基本データ項目を参照している。
        2. 一意名-1 が参照する集団データ項目に基本データ項目が含まれる。基本データ項目が表要素の場合、出現する各基本データ項目が受取り側作用対象となる可能性がある。
    c. 最後に、可能性のある項目のうち、以下の条件の 1 つ以上が真であるものが受取り側作用対象となる。
        1. VALUE 指定が使用されており、基本データ項目の項類が VALUE 指定で明示的または暗黙的に指定された項類のどれかである。さらに、以下の条件のいずれかが真である。
            a. 基本データ項目の項類が、データポインタ、オブジェクト参照、またはプログラムポインタである。
            b. データ項目形式または表形式の VALUE 句が基本データ項目のデータ記述項で指定されている。
        2. REPLACING 指定が使用され、基本データ項目の項類が、REPLACING 指定で指定された項類のどれかである。
        3. DEFAULT が使用されている。
        4. REPLACING 指定も VALUE 指定も使用されていない。
4. 暗黙的な各 MOVE 文および SET 文の送出し側作用対象は、以下のように決定される。
    a. データ項目が、VALUE 指定を含んでいるために受取り側作用対象とされた場合、
        1. 受取り側作用対象の項類がデータポインタまたはプログラムポインタであれば、送出し側作用対象は定義済みの番地 NULL となる。
        2. 受取り側作用対象の項類がオブジェクト参照であれば、送出し側作用対象は定義済みのオブジェクト参照 NULL となる。
        3. 上記のどちらでもない場合は、データ項目もデータ記述項で指定された VALUE 句中の定数により、送出し側作用対象が決定される。データ項目が表要素であれば、初期化される対象に対応する、VALUE 句中の定数により、送出し側作用対象が決定される。実際の送出し側作用対象は定数で、MOVE 文で受取り側作用対象に転記されたときに、VALUE 句の適用により生成されるデータ項目の初期値と同じ値を結果をもたらす。
    b. データ項目が、VALUE 指定を含んでいないが、REPLACING 指定を含んでいるために受取り側作用対象とされた場合、送出し側作用対象は、

REPLACING 指定で指定された項類に関連づけられた定数-1 または一意名-2 となる。

c. データ項目が、一般規則 4a または 4b により、送出し側作用対象とされない場合、使用される送出し側作用対象は、以下の表に従い、受取り側作用対象の項類により変わる。

受取り側作用対象	表意定数
英字	表意定数英数字 SPACES
英数字	表意定数英数字 SPACES
英数字編集	表意定数英数字 SPACES
データポインタ	定義済みの番地 NULL
各国語型	表意定数各国語型 SPACES
各国語型編集	表意定数各国語型 SPACES
数字	表意定数 ZEROES
数字編集	表意定数 ZEROES
オブジェクト参照	定義済みのオブジェクト参照 NULL
プログラムポインタ	定義済みの番地 NULL

5. すべての場合において、一意名-1 によって参照されるデータ項目の内容は、INITIALIZE 文に書かれている順番 (左から右) に、初期化される。その過程で集団項目が出てきた場合には、その集団の中で記述されている順に、基本項目が初期化される。可変出現データ項目については、初期化された出現回数は、受取り側データ項目の OCCURS 句の規則により決定される。
6. 一意名-1 と一意名-2 が同じ記憶領域を指す場合には、INITIALIZE 文の実行結果はどうなるかわからない。両者が同じデータ記述項に定義されていたとしても、そのことは変わらない。(***手続き部***の章の*作用対象の重なり*節を参照。)

13.12 INSPECT(文字列検査)文

INSPECT (文字列検査) 文は、データ項目内に出てくる単一の文字または一連の文字の数を数えたり (形式 1) 、置き換えたり (形式 2) 、数を数えて置き換えたり (形式 3) 、

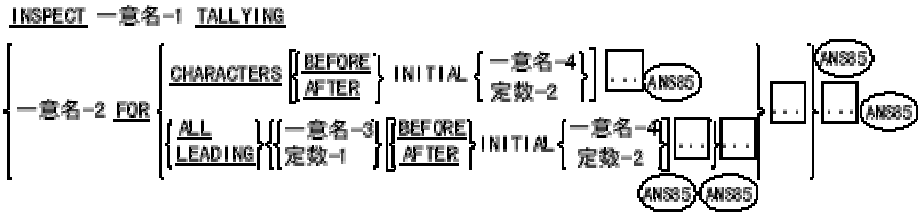
ANS85 または、変換 (形式 4) する。

例

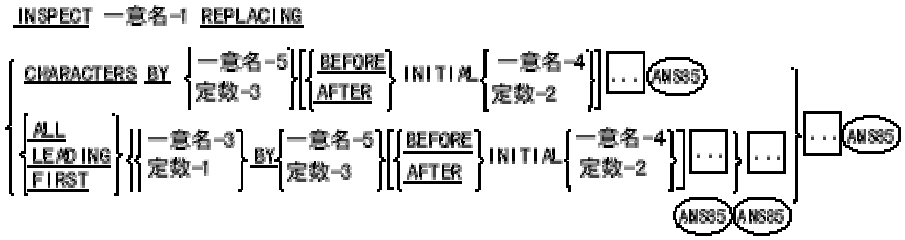
1. INSPECT TALLYING、REPLACING、CONVERTING の使用例は、***言語リファレンス - 追加トピック***の *INSPECT 文(カウント、置換、および変換)*節を参照。

一般形式

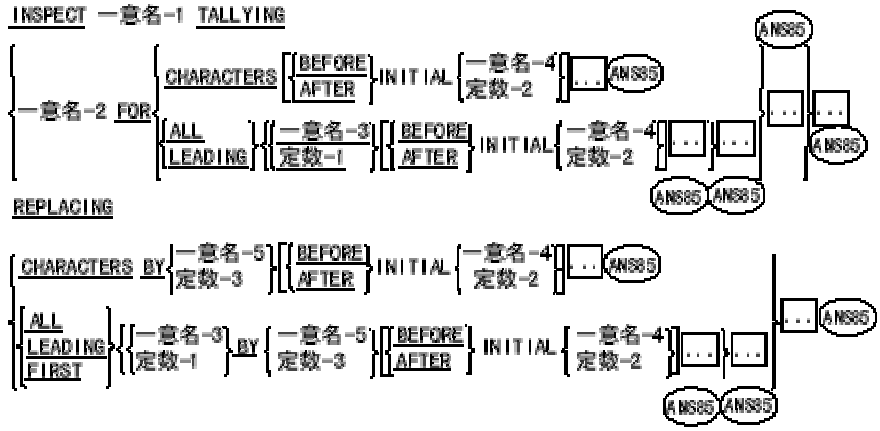
形式 1



形式 2



形式 3



ANS85 **形式 4**

```
INSPECT 一意名-1 CONVERTING { 一意名-6 } TO { 一意名-7 }   (ANS85)
                            { 定数-4   }     { 定数-5   }
    [ { BEFORE } INITIAL { 一意名-4 } ]
      { AFTER  }         { 定数-5   }
```

構文規則

すべての形式

1. 一意名-1 は、集団項目、あるいは明示的または暗黙的に USAGE IS DISPLAY と記述された基本項目とする。
2. 一意名-3 から一意名-*n* までの一意名は、

   (MF)集団項目、あるいは

   明示的または暗黙的に USAGE IS DISPLAY と記述された英字または英数字、

   (ANS85)英数字編集、数字編集、

   または数字の基本項目とする。

3. 各定数は、文字定数または表意定数とする。ただし、表意定数の ALL は使用できない。定数-1、定数-2、定数-4 が表意定数のときは、暗黙的に 1 文字のデータ項目を表わす。
4. ALL、LEADING、CHARACTERS、FIRST、

   (ANS85)または CONVERTING

   の各指定には、 BEFORE 指定、

   (ANS85)および /、

   または AFTER 指定を 2 つ以上書くことはできない。

5. 定数-1、定数-2、定数-3、定数-4、定数-5 および一意名-3、一意名-4、一意名-5、一意名-6、一意名-7 のデータ項目の長さは、定数またはデータ項目が取る限度内ならば、何文字でもよい。
6. (MF)一意名-2 (TALLYING 項目) を除くすべての一意名は、外部浮動小数点数であってもよい。英数字項目を参照する INSPECT 文に関しては、外部浮動小数点項目は英数字に再定義されているように扱われる。

形式 1 および 3

7. 一意名-2 は、基本数字データ項目とする。

形式 2 および 3

8. 定数-3 または一意名-5 のデータの大きさは、定数-1 または一意名-3 のデータの大きさと等しくする。定数-3 に表意定数を指定すると、その大きさは、定数-1 または一意名-3 のデータ項目の大きさに等しく扱われる。
9. CHARACTERS 指定を行う場合、定数-3 または一意名-5 のデータ項目の大きさは 1 文字にする。

   (ANS85)定数-2 および一意名-4 には、この制限は適用されない。

形式 4

10. (ANS85)定数-5 または一意名-7 のデータの大きさは、定数-4 または一意名-6 のデータの大きさと等しくする。定数-5 に表意定数を指定すると、その大きさは、定数-4 または一意名-6 のデータ項目の大きさに等しく扱われる。
11. (ANS85)定数-4 または一意名-6 のデータ項目中に、同じ文字が 2 回以上現れてはならない。

一般規則

すべての形式

1. (ANS85)長さを判定する目的では、一意名-1 は送出し側データ項目であるかのように扱われる。(*データ部 - ファイルおよびデータ記述の*章の *OCCURS(反復)句*節を参照。)
2. 文字列検査を行うためには、比較の周期、BEFORE 指定や AFTER 指定の範囲の設定、計数または置換の機構などが必要である。これらの設定を含めて、文字列検査は一意名-1 のデータ項目の左端の文字位置から開始され、左から右へ、右端の文字位置に達するまで進められる。この文字列検査の進め方には、字類による違いはない。具体的には、下記の一般規則 11、12、13 に説明する。
3. INSPECT 文の中で使用するデータ項目の一意名-1、一意名-3、一意名-4、一意名-5、一意名-6、一意名-7 の内容は、下記のように扱われる。
   a. 一意名-1、一意名-3、一意名-4、一意名-5、一意名-6、一意名-7 のどれかが英数字と記述されているならば、これらの各一意名の内容は文字列として扱われる。
   b. 一意名-1、一意名-3、一意名-4、一意名-5、一意名-6、一意名-7 のどれかが英数字編集か数字編集か符号なし数字項目と記述されているならば、これらの各一意名は英数字に再定義されているように扱われる。(一般規則 3a を参照。)INSPECT 文は、再定義されたデータ項目を参照するように書かれる。
   c. 一意名-1、一意名-3、一意名-4、一意名-5、一意名-6、一意名-7 のどれかが符号付き数字と記述されているならば、データ項目は符号部分を除いたその符号付き数字項目と長さの等しい符号なし数字項目に転記されたように扱われる。そして、その結果に対して上記の一般規則 3b が適用される。(*手続き部 - MERGE - OPEN* の章の *MOVE(転記)文*節を参照。)
4. 一般規則 6 から 19 において、定数-1、定数-2、定数-3、定数-4、定数-5 に関する規則は、それぞれ、一意名-3、一意名-4、一意名-5、一意名-6、一意名-7 のデータ項目の内容にも同様に当てはまる。

5. 一意名の中に添字付きのもの

   (ANS85)または関数一意名

   があると、その添字

   (ANS85)または関数一意名

   は、INSPECT 文の実行の最初に 1 回だけ評価される。

形式 1

6. (ANS85)一意名-1 は、送出し側データ項目である。
7. 必要語の ALL と LEADING は、後続の各定数-1 にかかる形容詞である。その効力の範囲は、次の形容詞が出てくるまでである。
8. 一意名-2 のデータ項目の内容は、INSPECT 文を実行することによって初期化されることはない。
9. 計数に関する規則は、下記のとおり。
   a. ALL 指定を書くと、一意名-1 の中に定数-1 が出現する回数が一意名-2 に加えられる。
   b. LEADING 指定を書くと、一意名-1 の中に定数-1 が最初に出現して以降連続して出現する回数が一意名-2 に加えられる。ただし、一意名-1 の中に定数-1 が最初に出現する場所は、定数-1 が比較の対象となる最初の比較周期が始まる点とする。
   c. CHARACTERS 指定を書くと、一意名-1 の中にその文字が出現する回数が一意名-2 に加えられる。その数え方は書き方 1 および 2 の一般規則 12e に従う。
10. 一意名-1、一意名-3、一意名-4 のどれかが一意名-2 と同じ記憶領域を占めると、INSPECT 文の実行結果はどうなるかわからない。たとえそれらのデータ項目が同じデータ記述項に記述されたものであっても、そのことは変わらない。(***手続き部***の章の*作用対象の重なり*節を参照。)

形式 1 および 2

11. 一意名-1 のデータ項目の内容を文字列検査する間、その中に定数-1 が出現するたびに、計数 (形式 1 の場合) または定数-3 による置換 (形式 2 の場合) が行われる。
12. 計数または置換するために定数-1 の出現を調べる比較は、下記のように行われる。
    a. TALLYING 指定または REPLACING 指定の作用対象は、INSPECT 文中に書かれた順に左から右に取り上げられる。一意名-1 のデータ項目の左端の文字位置から開始して、最初の定数-1 が、一意名-1 中の等しい文字数と比較される。 両者を 1 文字ずつ比較した結果が等しく、さらに下記の条件のどれかを満たす場合、定数-1 と一意名-1 中の該当部分とは等しいと判定される。
       1. LEADING 指定も FIRST 指定も書かれていない。
       2. 定数-1 に形容詞 LEADING がかかっている場合、一般規則 9 および 15 に照らして、定数-1 は先頭に現れている。

3. 定数-1 に形容詞 FIRST がかかっている場合、一般規則 9 に照らして、定数-1 は最初に現れている。

b. 最初の定数-1 が一意名-1 中の比較対象部分と一致しないと、後続の定数-1 が存在すれば、そのそれぞれについて比較が繰り返される。この比較は、両者が一致するか、後続の定数-1 がなくなるまで続けられる。後続の定数-1 がない場合には、一意名-1 のデータ項目の文字位置のうちで、前回の比較周期で取り上げられた部分のすぐ右隣の部分が、次回の比較周期での比較対象部分として取り上げられる。そして、最初の定数-1 との比較が繰り返される。

c. 比較結果が一致すると、一般規則 9 と 15 に説明してあるように、計数と置換のどちらか一方または両方が行われる。すると今度は、一意名-1 のデータ項目中の現在比較の対象となった部分の右端の文字位置のすぐ右隣の部分が、次回の比較周期での比較対象部分として取り上げられる。そして、最初の定数-1 との比較が繰り返される。

d. この比較処理は、一意名-1 のデータ項目中の右端の文字位置が比較の対象として取り上げられるか、または比較部分の左端の文字位置とみなされるまで続けられる。この状態に達すると、文字列検査は終了する。

e. CHARACTERS 指定を書くと、定数-1 に文字を 1 つ暗黙に指定したものとして、上記の 12a から 12d に記述した比較処理が行われる。ただし、一意名-1 のデータ項目の内容との比較は実際には行われない。この場合、暗黙の定数-1 は現在の比較周期において、一意名-1 のデータ項目の内容の比較対象部分の左端文字位置とつねに一致するものとみなされる。

13. 一般規則 12 に定義した比較処理は、BEFORE 指定および AFTER 指定の影響を受ける。その様子は下記のとおり。

a. BEFORE 指定も AFTER 指定も書かないと、定数-1 または CHARACTERS 指定の暗黙の作用対象が一意名-1 と比較される開始点は、一意名-1 の左端の文字位置からとなる。

b. BEFORE 指定を書くと、定数-1 または CHARACTERS 指定の暗黙の作用対象と比較される一意名-1 のデータ項目の内容の範囲は、その左端の文字位置から定数-2 が最初に出現するまで (定数-2 を含まない) となる。

定数-2 が最初に出現する位置は、一般規則 12 に記述した比較処理の最初の周期が始まる前に判定される。どれかの比較周期において、定数-1 または CHARACTERS 指定の暗黙の作用対象が比較の対象とならない場合は、一意名-1 のデータ項目の内容と一致しないものとみなされる。一意名-1 のデータ項目の中に定数-2 が出てこなかった場合は、定数-1 または CHARACTERS 指定の暗黙の作用対象は、BEFORE 指定がなかったのと同様に比較処理の対象とされる。

c. AFTER 指定を書くと、定数-1 または CHARACTERS 指定の暗黙の作用対象と比較される一意名-1 のデータ項目の内容の範囲は、最初に出現する定数-2 の右端のすぐ右の文字位置から末尾までとなる。

定数-2 が最初に出現する位置は、一般規則 12 に記述した比較処理の最初の周期が始まる前に判定される。どれかの比較周期において、定数-1 または CHARACTERS 指定の暗黙の作用対象が比較の対象とならない場合は、

一意名-1 のデータ項目の内容と一致しないものとみなされる。一意名-2 のデータ項目の中に定数-2 が出てこなかった場合は、定数-1 または CHARACTERS 指定の暗黙の作用対象は、まったく比較処理の対象とならない。

形式 2

14. 必要語の ALL と LEADING と FIRST は、後続の各 BY 指定にかかる形容詞である。その効力の範囲は、次の形容詞が出てくるまでである。
15. 置換に関する規則は、下記のとおり。
    a. CHARACTERS 指定を書くと、一意名-1 のデータ項目の中で書き方 1 と 2 の一般規則 12e に照らして一致する各文字が、定数-3 で置き換えられる。
    b. ALL 指定を書くと、一意名-1 のデータ項目の中の定数-1 に一致する各文字が、定数-3 で置き換えられる。
    c. LEADING 指定を書くと、一意名-1 のデータ項目の中の定数-1 に一致する最初以降の連続する各文字が、定数-3 で置き換えられる。ただし、一意名-1 の中に定数-1 が最初に出現する場所は、定数-1 が比較の対象となる最初の比較周期が始まる点とする。
    d. FIRST 指定を書くと、一意名-1 のデータ項目の中の定数-1 に一致する左端の文字が、定数-3 で置き換えられる。
16. 一意名-3、一意名-4、一意名-5 のどれかが一意名-1 と同じ記憶領域を占めると、INSPECT 文の実行結果はどうなるかわからない。たとえそれらのデータ項目が同じデータ記述項に記述されたものであっても、そのことは変わらない。(***手続き部***の章の *作用対象の重なり*節を参照。)

形式 3

17. 形式 3 の INSPECT 文は、同じ一意名-1 に対して、形式 1 の TALLYING 指定と形式 2 の REPLACING 指定を 2 つ続けて書いたものとして解釈され実行される。形式 1 の INSPECT 文の比較照合と計数に関する一般規則、および形式 2 の INSPECT 文の比較照合と置換に関する一般規則が、形式 3 にも当てはまる。

形式 4

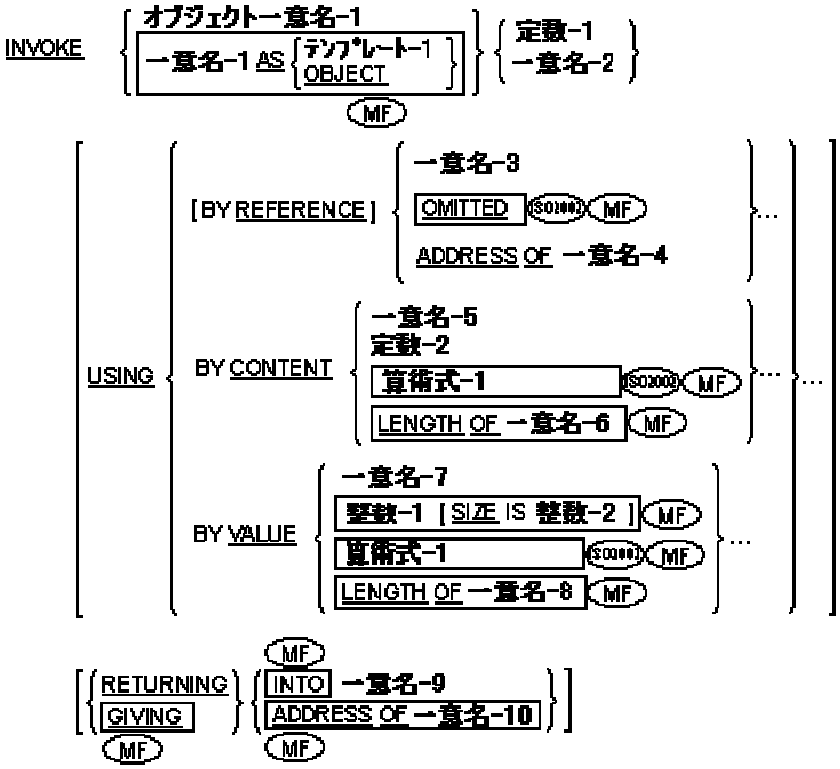
18. (ANS85)形式 4 の INSPECT 文は、定数-4 に含まれる各文字に関して 1 つずつ、同じ一意名-1 に対して ALL 指定をした、形式 2 の INSPECT 文を続けて書いたものとして解釈され実行される。これらの ALL 指定をした個々の仮想の INSPECT 文は、定数-1 の代わりに定数-4 を、定数-3 の代わりに定数-5 を用いたものと解釈される。定数-4 の文字と定数-5 の文字との対応関係は、データ項目内の順序位置によって付けられる。
19. (ANS85)一意名-4、一意名-6、一意名-7 のどれかが一意名-1 と同じ記憶領域を占めると、INSPECT 文の実行結果はどうなるかわからない。たとえそれらのデータ項目が同じデータ記述項に記述されたものであっても、そのことは変わらない。

13.13 INVOKE(メソッド呼出し)文

(ISO2002)(MF)

INVOKE 文はメソッドを呼び出す働きをする。

一般形式



構文規則

1. オブジェクト一意名-1 はオブジェクト参照またはクラス名でなければならない。
2. (MF)一意名-1 は 4 バイトのデータ項目でなければならない。
3. 定数-1 は英数字の定数でなければならない。
4. 定数-1 も定数-2 も表意定数であってはならない。
5. (ISO2002)オブジェクト一意名-1 が一般的オブジェクト参照を参照する場合、BY CONTENT 指定も BY VALUE 指定も書いてはならない。また、BY REFERENCE 指定を書かなくとも、暗黙的に指定されているものとみなされる。

   (MF)BY CONTENT 指定および BY VALUE 指定を書ける。

6. 一意名-2 は英数字のデータ項目として定義しなければならない。
7. (ISO2002)一意名-2 を指定した場合、オブジェクト一意名-1 は一般的オブジェクト参照でなければならない。

(MF)一般的オブジェクト参照であるために、オブジェクト一意名-1 は必須ではない。

8. 一意名-1 によって参照されるデータ項目が一般的オブジェクト参照ではない場合、*手続き部*の章の*パラメータおよび返却する項目*節に指定されている規則が適用される。
9. 一意名-1 によって参照されるデータ項目が一般的オブジェクト参照ではなく、引数に BY CONTENT 指定または BY REFERENCE 指定がある場合、手続き部の見出し中の対応する仮パラメータに BY REFERENCE 指定がなければならない。
10. 一意名-3 は、ファイル節、作業場所節、

    (MF)オブジェクト記憶節、

    局所記憶節、連絡節 のいずれかに定義されているデータ項目を参照しなければならない。

11. (MF)一意名-4、一意名-5、一意名-6、一意名-7、一意名-8 の いずれも、関数一意名であってはならない。
12. 引数に BY VALUE を指定した場合、手続き部の見出し中の対応する仮パラメータに BY VALUE 指定がなければならない。
13. (MF)整数-1 は符合付きでもゼロでもよい。
14. (MF)GIVING と RETURNING は同等である。
15. (MF)一意名-9 はサイズが 8 バイト以内のデータ項目であり、ファイル節、作業場所節、

    (MF)オブジェクト記憶節、

    局所記憶節、連絡節 のいずれかに定義されていなければならない。

16. (MF)一意名-10 は連絡節にレベル番号が 01 または 77 のデータ項目として定義されていなければならない。
17. 一意名-3 は送出し側と受取り側の両方の作用対象である。
18. 一意名-1、一意名-2、一意名-4、一意名-5、一意名-6、一意名-7、一意名-8 および、算術式-1 で指定されたあらゆる一意名-1 は、送出し側の作用対象である。
19. 一意名-9 と一意名-10 は受取り側の作用対象である。
20. (SOMM)(MF)OMITTED 指定が含まれる場合は、手続き部見出しの対応する仮パラメータにつして、OPTIONAL 指定が含まれなければならない。

一般規則

1. INVOKE 文を実行するプログラムまたはメソッドのインスタンスは起動力のあるランタイム要素である。
2. オブジェクト一意名-1

   (MF)または一意名-1

はオブジェクト･インスタンスを指す。オブジェクト一意名-1 をクラス名として指定した場合は、そのクラス名のファクトリ･オブジェクトを指す。定数-1、または一意名-2 によって参照されるデータ項目の内容は、そのオブジェクト･インスタンスに対して作用するそのオブジェクトのメソッドを指す。

a. 呼び出す対象のメソッドが COBOL のメソッドである場合、定数-1、または一意名-2 によって参照されるデータ項目の内容は、呼出し対象のメソッドのメソッド名段落に指定されている名前を指す。

   (SOM) (MF) オブジェクト･インスタンスが一般的オブジェクト参照により定義されている場合、呼び出されたメソッド中の仮パラメータと返却する項目のいずれも、ANY LENGTH 句を使用して記述されてはならない。

b. 呼び出す対象のメソッドが COBOL のメソッドではない場合、メソッド名を構成する規則は関係するドメインによって異なる。

3. (MF) AS 指定を書くと、COBOL のデータ項目に対してメソッドを呼び出すことができる。テンプレート-1 を指定すると、クラス･テンプレートとして使用される。テンプレートを作成することは、クラス･ライブラリ中のオブジェクトによって提供される機能である。(詳細については、使用している COBOL システムのマニュアルを参照。)
4. INVOKE 文を実行すると、指定したメソッドが実行可能になり、呼び出されたメソッドに制御が移される。呼び出されたメソッドが COBOL のメソッドである場合、***手続き部***の章の*手続き部の見出し*節に記述されている*一般規則*に従って、メソッドが実行される。そうでない場合には、メソッドが記述されている言語の作成者によって定義されている規則に従って、メソッドが実行される。メソッドから制御が戻されると、INVOKE 文の最後に制御が移される。
5. 呼び出されたメソッドを実行できるようにするさいに、その所在が把握されるまで、実行単位にはそれが COBOL のメソッドであるか否かは分からない。メソッド名の形式から COBOL のメソッドであるか否かを判定することはできない。
6. メソッドの呼出しまたはメソッドの終了のプロセスによって、何らかの外部ファイル結合子に関連するファイルの状態または位置づけが変更されることはない。
7. この処理系では、引数と連絡節内のデータ項目の間の同期を維持することはない。それは利用者の責任である。

注: COBOL 言語では、データ項目の番地の桁寄せに制約はない。他方、COBOL 以外の言語では、通常は番地について前提があり、桁寄せされていないデータ項目が参照されると何らかのエラーを起こす。桁寄せは下記の措置のうちのいくつかを用いて実現される。

- 集団項目に無名項目を追加する
- ALIGN コンパイラ指令を使用するとともに、USING 指定中の作用対象を 01 レベルまたは 77 レベルのデータ項目にする
- SYNCHRONIZED 句と IBMCOMP コンパイラ指令を併用する

INVOKE 文に RETURNING 指定を書いた場合、COBOL 以外のメソッドは適切な形式の結果を返さなければならない。

8. INVOKE 文の USNIG 指定の引数と呼び出されるメソッドの手続き部の見出しの USING 指定の中の対応する仮パラメータの順序によって、引数と仮パラメータの対応関係が決まる。この対応は位置で決まるのであって、名前が等しいことによるのではない。最初の引数が最初の仮パラメータに対応し、次にそれらの2番目同士が対応する、といったぐあいである。

   パラメータが指標名の場合は、そのような対応関係は確立されない。呼び出されたメソッドと呼び出すランタイム要素の指標名は常に別々の指標を参照する。

   起動されたランタイム要素上の USING 指定の効果については、***手続き部***の章の*手続き部の見出し*節の *一般規則*に記述されている。

9. (ISO2002)(MF)単一の一意名または定数から成る引数は、算術式ではなく、一意名または定数と見なされる。
10. INVOKE 文の USNIG 指定の中で参照されるパラメータの値は、INVOKE 文が実行されたときに、呼び出されるメソッドで利用できるようになる。
11. BY CONTENT、BY REFERENCE、BY VALUE の各指定はその後ろに続くパラメータに効力を及ぼすが、別の BY CONTENT、BY REFERENCE または BY VALUE の指定が出てきたところでその効力は停止する。

    最初のパラメータの前に BY CONTENT、BY REFERENCE、BY VALUE のどの指定もないと、BY REFERENCE 指定があるものとみなされる。

12. BY REFERENCE ADDRESS OF 指定が明示的または暗黙的に使用されている場合、USAGE POINTER を用いてデータ項目が宣言され、「SET データ項目 TO ADDRESS OF 一意名-4」文によって取得された値が BY REFERENCE によってそのデータ項目に渡されたかのように、メソッドは動作する。
13. パラメータに対して BY CONTENT 指定を明示的または暗黙的に使用すると、呼び出されたメソッドは、INVOKE 文の USING 指定に従ってこのパラメータの値を変更することができない。ただし、呼び出されたメソッドの手続き部見出し中の対応するデータ名が参照するデータ項目の値を変更することはできる。INVOKE 文の BY CONTENT 指定中の各パラメータのデータ記述は、手続き部見出しの USING 指定中の対応するパラメータのデータ記述と同じでなければならない。つまり、変換、拡張、および切り詰めはまったく行われない。

    パラメータに対して BY CONTENT 指定を明示的または暗黙的に使用している場合、実行プログラムは、追加データ項目が宣言され、BY REFERENCE 指定中で使用されたかのように動作する。一意名-5 を指定した場合は、追加項目の暗黙的なデータ記述とその内容が、一意名-5 のものと同じとなる。定数-3 を指定した場合は、追加項目の暗黙的なデータ記述が、定数-3 と同じサイズで、内容が定数-3 の値に設定された英数字データ項目と同じとなる。LENGTH OF 一意名-6 を指定した場合は、追加項目のデータ記述が、PIC S9(n) USAGE COMP-5 と同じとなる。ここで、 n の値は、9 以上で、COBOL システムでのデータ項目の最大長を保持できるサイズのデータ項目

を定義するものとする。追加データ項目の内容は、一意名-6 に割り当てられた記憶用バイト数に設定される。

14. パラメータに対して BY VALUE 指定を明示的または暗黙的に使用している場合、呼び出されたメソッドは、INVOKE 文の USING 指定に従ってこのパラメータの値を変更することができない。ただし、呼び出されたメソッドの手続き部見出し中の対応するデータ名が参照するデータ項目の値を変更することはできる。概念としては、メソッドの動作により、追加データ項目がシステム領域 (典型的な例はスタック) 内で宣言され、これを使用して COBOL 以外の言語がパラメータを渡すことができるようになる。さらに、この追加データ項目が、呼び出されたメソッド中のデータ項目と同じ記憶領域を占める。

    一意名-7 が指定され、BINARY、COMP-4、COMP-5、または COMP-X である場合、追加データ項目のデータ記述は一意名-7 のものと同じとなる。ただし、その用途は COMP-5 である。追加データ項目の内容は、COMP-5 で表示される 一意名-7 の内容と同じとする。この変換と形式により、COBOL 以外のプログラムの呼出しが容易となる。

    これ以外の場合で一意名-7 が指定された場合は、追加データ項目の暗黙的なデータ記述および内容は、一意名-7 のものと同じとなる。

    整数-1 を指定すると、追加データ項目のデータ記述は、整数-2 が指定されていれば、その値で示された記憶用バイト数を占める符号付き数字項目 USAGE COMP-5 と同じとなる。整数-2 が指定されていない場合は、システム領域のサイズ (典型的な例は、システム上の POINTER のサイズ) と同じとなる。

    「LENGTH OF 一意名-8」を指定すると、追加データ項目のデータ記述は、PIC S9(n) USAGE COMP-5 と同じとなる。ここで n の値は 9 以上で、COBOL システムでのデータ項目の最大長を保持できるサイズのデータ項目を定義するものとする。追加データ項目の内容は、一意名-8 に割り当てられた記憶用バイト数に設定される。

    呼び出されるメソッドが COBOL メソッドの場合、INVOKE 文の BY VALUE 指定中の各パラメータには、対応するパラメータが、手続き部見出しの USING 指定中で宣言されなければならない。このパラメータにも、明示的または暗黙的に BY VALUE 指定を使用しているものとする。

    上記のように INVOKE 文の結果として生成された追加データ項目と、呼び出されるメソッド中のこの対応するパラメータの関係は、以下のようになる。

    - 追加データ項目が BINARY、COMP-4、COMP-5、または COMP-X to COMP-5 からの変換の対象であった場合、呼び出されるメソッド中の対応するパラメータのデータ記述は、追加データ項目のデータ記述と同じでなければならない。ただし、対応するパラメータの用途は BINARY、COMP-4、COMP-5、および COMP-X のどれであってもよい。呼び出される COBOL メソッドの記述と同時に、追加データ項目の USAGE COMP-5 から対応するパラメータの USAGE への変換が実行される。

- 追加データ項目が COMP-5 への変換の対象でなかった場合、呼び出されるメソッド中の対応する各パラメータのデータ記述は、対応する追加データ項目と同じでなければならない。つまり、追加データ項目に対し、呼び出されるメソッド中の対応するパラメータに一致させるための変換、拡張、切り詰めはまったく行われない。

呼び出されるメソッドが COBOL メソッドでない場合、 BY VALUE 指定を使用しなければならない場合の詳細は、関連するドメインにより異なる。

15. (MF)一意名-4 が連絡節にあってそのレベル番号か 01 または 77 以外であるか、または作業場所節にある場合、そのデータ項目が BY CONTENT によって渡され、一意名-4 の番地を呼び出されたメソッドによっては変更できないことに等しい。
16. (MF)「BY REFERENCE 定数-2」を明示的または暗黙的に指定すると、メソッドは定数-2 を定数-3 のために記述されたものとして処理する。
17. (MF)「LENGTH OF 一意名-6」を指定すると、引数のデータ記述は、形式が PIC S9(9) USAGE BINARY で内容が一意名-6 に割り当てられた記憶領域のバイト数に設定されたものと等しくなる。
18. (MF)整数-1 を指定すると、引数の暗黙的なデータ記述は、符合付き数字項目で USAGE COMP-5 に等しくなる。その記憶領域内でのバイト数は、整数-2 の指定があればその値によって示され、指定がなければ 4 バイトとなる。
19. (MF)「LENGTH OF 一意名-8」を指定すると、引数のデータ記述は、形式が PIC S9(9) USAGE BINARY で内容が一意名-8 に割り当てられた記憶領域のバイト数に設定されたものと等しくなる。
20. (MF)一意名-10 を指定すると、呼び出されたメソッドは、明示的または暗黙的に USAGE POINTER と指定された値を返す。メソッドの結果は、SET 文の規則に従って、一意名-10 に代入される。
21. 呼び出されるメソッド中に INVOKE 文が含まれていてもよい。呼び出されるメソッドは、呼び出すメソッドを直接的または間接的に呼び出す INVOKE 文を実行できる。
22. (2002)(MF)OMITTED 指定が含まれるか、または後続の引数が省略されている場合、省略された引数の条件は、呼び出されたメソッド中で、そのパラメータについて真となる。
23. (2002)(MF)省略された引数の条件が真であるパラメータが、呼び出されたメソッド中で参照されている場合、RTS エラーメッセージ 203 が返される。ただし、引数として参照されている場合、および省略された引数の条件中で参照される場合を除く。

第 14 章 : 手続き部 - MERGE - OPEN

14.1 MERGE (併合) 文

MERGE (併合) 文は、指定されたキーの組に従って同じようにそろえられているいくつかのファイルを、1 つのファイルに併合する。そして、併合された順番に従って、出力手続きまたは出力ファイルへレコードを引き渡す。

一般書式

```
                                          (OSVS)
MERGE ファイル名-1 {ON {ASCENDING} KEY [IS] {データ名-1}...}...
                      {DESCENDING}

    [COLLATING SEQUENCE IS 符号系名]
     USING ファイル名-2 {ファイル名-3}...

{OUTPUT PROCEDURE IS 手続き名-1 [{THROUGH} 手続き名-2]}
                                 {THRU}
{GIVING [{ファイル名-4}...]}
         (ANS85)   (ANS85)
```

指令

1. フラグを提供し、予約語リストを変更するコンパイラ指令の他に、以下の指令が、この節で解説する構文または意味に影響を及ぼす場合がある。
   - CALLSORT"EXTSM" - USING 句または GIVING 句で 255 個までのファイルの指定を可能にする。この指令がない場合は、指定できるファイル名は 10 個までに制限される。

構文規則

1. ファイル名-1 は、データ部の整列併合ファイル記述項に記述しておく。
2. 手続き名-1 は、出力手続きの名前を表わす。
3. 手続き名-1 および手続き名-2 は、節名とする.

   (ANS85)(OSVS)この制約は解除されている。

4. (ANS85)ファイル名-1 のファイルに含まれるレコードが可変長である場合、ファイル名-2 およびファイル名-3 のファイル中に含まれるレコードの大きさは、ファイル名-1 のレコードの最小のもの以上かつ最大のもの以下にする。ファイル名-1 のファイルに含まれるレコードが固定長である場合、ファイル名-2 およびファイル名-3 のファイル中に含まれるレコードの大きさは、ファイル名-1 のレコードの最大のものを超えてはならない。

5. ファイル名-2、ファイル名-3、ファイル名-4 は、データ部の中の整列併合ファイル記述項にではなく、ファイル記述項に記述しておく。
6. THRU と THROUGH は同義語である。
7. データ名-1 はキーのデータ名であり、下記の規則に従う。
    a. キー・データ名で示すデータ項目は、ファイル名-1 のレコード中に記述しておく。
    b. キー・データ名は、修飾してもよい。
    c. キー・データ項目は、可変長であってはならない。
    d. ファイル名-1 に複数のレコード記述を含める場合、キー・データ名で示すデータ項目は 1 つのレコード記述の中だけに指定する。1 つのレコード記述の中でキー・データ名に指定した文字位置が、そのファイルの他のすべてのレコードのキーとして扱われる。
    e. キー・データ名には、OCCURS 句を含む記述項は記述できない。また、OCCURS 句を含む記述項の下位に、キー・データ名を属させてはならない。
    f. (OSVS)(VSC2)(MF)キー・データ項目は、浮動小数点項目であってもよい。
    g. (OSVS)(VSC2)(MF)キー・データ項目が外部浮動小数点数である場合、キーは英数字として扱われる。レコードが併合される順序は、使用する文字の照合順序に左右される。
    h. (OSVS)(VSC2)(MF)キー・データ項目が内部浮動小数点数である場合、キーの順序は数値順となる。
8. MERGE 文には、複数ファイル・リール中のファイル名を複数指定してはならない。
9. MERGE 文の中でファイル名を繰り返し指定してはならない。
10. MERGE 文は、手続き部の宣言部分以外のどこにでも置くことができる。
11. (ANS85)ファイル名-4 が索引ファイルである場合、最初のデータ名-1 には ASCENDING を指定する。また、データ名-1 がレコード上で占める文字位置は、そのファイルの主レコードキーと同じにする。
12. (ANS85)GIVING を指定し、その対象であるファイル名-4 のファイルに含まれるレコードが可変長である場合、ファイル名-1 のファイル中に含まれるレコードの大きさは、ファイル名-4 のレコードの最小のもの以上かつ最大のもの以下にする。ファイル名-4 のファイルに含まれるレコードが固定長である場合、ファイル名-1 のファイル中に含まれるレコードの大きさは、ファイル名-4 のレコードの最大のものを超えてはならない。

一般規則

1. MERGE 文は、ファイル名-2 およびファイル名-3 のファイルに含まれるすべてのレコードを併合する。
2. ファイル名-1 のファイルに含まれるレコードが固定長である場合、ファイル名-2 またはファイル名-3 のファイル中にファイル名-1 の固定長レコードよりも短いものがあると、ファイル名-2 またはファイル名-3 のレコードがファイル名-1 のファイルに引き渡されるときに、その後部に空白が埋められる。
3. 語 KEY の後ろに続くデータ名は、KEY 指定での区切り方とは関係なく、強さの順に左から右に並べる。つまり、一番左のデータ名が最も強いキーであり、その右のデータ名が次に強いキーである、といった具合いである。
    a. ASCENDING を指定した場合、比較の規則に従って、キー・データ項目の内容の値の小さい方から大きい方へと、順に併合される。

b. DESCENDING を指定した場合、比較の規則に従って、キー･データ項目の内容の値の大きい方から小さい方へと、順に併合される。

4. 比較の規則に従って、あるデータレコードのすべてのキー･データ項目の内容が、他のいくつかのデータレコードの対応するキー･データ項目と等しい場合、引き取りの順序は下記のようになる。
   a. MERGE 文に指定した入力ファイルの順。
   b. 1 つの入力ファイル内にキー･データ項目の値が等しいレコードが複数ある場合は、それらがすべて他の入力ファイルからよりも先に引き取られる。
5. 文字のキー･データ項目の比較に適用される文字の照合順序は、iMERGE 文の実行開始時に、下記の優先順序で決定される。
   a. 最初は、指定してあれば、MERGE 文中の COLLATING SEQUENCE に指定した文字の照合順序。
   b. 次は、プログラム用に設定してある文字の照合順序。
6. ファイル名-2 およびファイル名-3 のファイル中のレコードが、MERGE 文中に指定した ASCENDING KEY または DESCENDING KEY の順に整列されていないと、併合処理の結果はどうなるかわからない。
7. ファイル名-2 およびファイル名-3 のファイル中のすべてのレコードが、ファイル名-1 のファイルに引き渡される。MERGE 文の実行開始時にファイル名-2 およびファイル名-3 のファイルが開かれていてはならない。MERGE 文を実行すると、ファイル名-2 およびファイル名-3 の各ファイルに対して、下記の操作が行われる。
   a. ファイルの処理が開始される。これは INPUT 指定をした OPEN 文を、暗黙的に実行することである。出力手続きを指定すると、この開始処理は出力手続きに制御が移される前に行われる。
   b. 論理レコードが読み込まれて、併合操作に引き渡される。これは NEXT および AT END を指定した READ 文を、暗黙的に実行することである。

      ファイル名-1 のファイルのレコードが可変長であると、ファイル名-1 のファイルに書き出されるレコードの大きさは、ファイル名-2 またはファイル名-3 から読み込んだときのレコードの大きさとなる。このことは、ファイル名-1 のファイルの整列併合ファイル記述項に指定した、RECORD IS VARYING 句または OCCURS 句の DEPENDING ON 指定の対象となっているデータ項目の内容にかかわらず適用される。

      相対ファイルに関しては、相対キー･データ項目の内容が MERGE 文の実行後にどうなっているかはわからない。

   c. ファイルの処理が終了される。これは任意指定の何もない CLOSE 文を暗黙的に実行することである。出力手続きを指定した場合は、その中の最後の文に制御が移されるまで、この終了処理は実行されない。

      USE AFTER EXCEPTION/ERROR 手続きを指定してあれば、この暗黙の CLOSE 処理において、それも対象となる。

      ファイル名-2 またはファイル名-3 のファイル記述項中に指定した、RECORD IS VARYING 句の DEPENDING ON 指定の対象となっているデータ項目の内容は、MERGE MERGE 文の完了時点でどのようになっているかわからない。

8. 出力手続きには、RETURN (引き取り) 文を用いてファイル名-1 のファイルから順に 1 件ずつ引き取った併合済みレコードを、選択したり変更したり複写したりする任意の必要な手続きを組み込むことができる。この出力手続きの範囲には、その中から CALL 文、EXIT 文、GO TO 文、PERFORM 文によって制御を移された結果実行される、すべての文が含まれる。また、この出力手続きの範囲内の文を実行した結果として実行される宣言手続き中のすべての文も、この出力手続きの範囲に含まれる。この出力手続きの範囲内で、MERGE 文、RELEASE 文、SORT 文を起動するようなことがあってはならない。(*COBOL 言語の概念*の章の*明示指定と暗黙指定*節を参照。)
9. 出力手続きを指定すると、MERGE 文の実行中にその手続きに制御が渡される。コンパイラは出力手続きの最後の文の末尾に復帰機構を組み込む。出力手続きの最後の文に制御が移されると、復帰機構によって MERGE 文の処理が終了されて、MERGE 文の後ろの次の実行可能文に制御が移される。出力手続きに制御が移される前に、併合処理は、要求されれば併合された次の順序のレコードを引き渡せる状態に達している。出力手続き中の RETURN 文は、そのレコードを引き取るための要求である。
10. 出力手続きの実行中に、ファイル名-2、ファイル名-3、ファイル名-4 のファイルを操作したり、そのレコード領域にアクセスしたりする文を実行することはできない。
11. GIVING を指定すると、暗黙の出力手続きによって、併合されたすべてのレコードはファイル名-4 のファイルに書き出される。MERGE 文の開始時点で、ファイル名-4 のファイルが開かれていてはならない。MERGE 文を実行すると、ファイル名-4 の各ファイルに対して、下記の処理が行われる。
    a. ファイルの処理が開始される。これは OUTPUT 指定をした OPEN 文を、暗黙的に実行することである。
    b. 論理レコードが引き取られてファイルに書き出される。これは任意指定を何も含まない WRITE 文を、暗黙的に実行することである。

       ファイル名-4 のファイルのレコードが可変長であると、ファイル名-4 のファイルに書き出されるレコードの大きさは、ファイル名-1 から読み込んだときのレコードの大きさとなる。このことは、ファイル名-4 のファイルの整列併合ファイル記述項に指定した、RECORD IS VARYING 句または OCCURS 句の DEPENDING ON 指定の対象となっているデータ項目の内容にかかわらず適用される。

       相対ファイルに関しては、相対キー・データ項目の値は、最初のレコードが引き取られたときは"1" 、2 番目のレコードが引き取られたときは"2" という具合いに設定される。MERGE 文の実行が終了した後で、相対キー・データ項目の内容がどうなっているかはわからない。

    c. ファイルの処理が終了される。これは任意指定の何もない CLOSE 文を、暗黙的に実行することである。

       USE AFTER EXCEPTION/ERROR 手続きを指定してあれば、上記の暗黙の処理において、それも対象となる。しかし、その USE 手続きの中から、ファイル名-4 のファイルを操作したりそのレコード領域をアクセスしたりするような文が実行されるようなことがあってはならない。外部的に定義されているファイルの境界を超えて最初にレコードが書き出されると、そのファイルに指定し

てある USE AFTER EXCEPTION/ERROR 手続きが実行される。その USE 手続きから制御が戻されたとき、またはそのような USE 手続きが指定されていないときは、上記の 11c と同様にファイルの処理が終了される。

ファイル名-1 のファイル記述項中に指定した、RECORD IS VARYING 句の DEPENDING ON 指定の対象となっているデータ項目の内容は、GIVING を指定した MERGE 文が完了した時点でどのようになっているかはわからない。

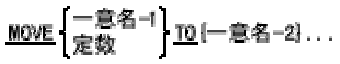
12. I ファイル名-4 のファイルに含まれるレコードが固定長である場合、ファイル名-1 のファイル中にファイル名-4 の固定長レコードよりも短いものがあると、ファイル名-4 のファイルにレコードが引き渡されるときに、その後部に空白が埋められる。
13. ***言語リファレンス - 追加トピック***の*区分化*の章に示すように、MERGE 文を含むプログラムを区分化できる。ただし、下記の制限がある。
    a. 独立区分ではない区分中の節内に MERGE 文が存在する場合、その MERGE 文から呼び出される出力手続きは、下記のどちらかの状態で組み込まれていなければならない。
        1. 非独立区分中に完全に含まれる。
        2. 単一の独立区分中に完全に含まれる。
    b. 独立区分中に MERGE 文が存在する場合、その MERGE 文から呼び出される出力手続きは、下記のどちらかの状態で組み込まれていなければならない。
        1. 非独立区分中に完全に含まれる。
        2. MERGE 文と同じ独立区分中に完全に含まれる。
14. OSVS VSC2 NF 特殊レジスタの SORT-RETURN が MERGE 文の原始要素に対して用意されている。この特殊レジスタには、併合処理が終わったときに 0 (正常終了) または 16 (不成功) が設定される。出力手続きの中でこの特殊レジスタに値 16 を設定して、途中で併合処理を終わらせることができる。この場合、併合処理は次の RETURN 文のところで終了される。

14.2 MOVE (転記) 文

MOVE (転記) 文は、編集規則に従って、データをいくつかのデータ領域に移す。

一般形式

形式 1



形式 2

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - BYTE-MODE-MOVE - 送出し側項目と受取り側項目がその記憶領域を部分的に共有する (つまり、重複部分のある) MOVE 文の動作に影響を与える。
    - DBCS (DBSPACE) - 2バイト文字の受取り側項目に埋める空白に使用する文字に影響を与える。
    - DE-EDIT - 数字編集送出し側フィールドの値をその PICTURE 句によって決定するか否かを制御する。
    - TRUNC - 受取り側のバイナリ値のサイズよりも大きい値が送出し側項目に含まれているときに、利用可能な記憶領域または PICTURE 句に基づいて、切捨てを行うかどうかを決定する。

構文規則

すべての形式

1. 一意名-1 および定数は送出し側を表わし、一意名-2 は受取り側を表わす。
2. 指標データ項目、

   (SOM) (MF) オブジェクト参照データ項目、プログラムポインタ・データ項目、

   (VSC2) (MF) ポインタ・データ項目、手続きポインタ・データ項目

   (MF) 入力ポインタ・データ項目、モニターポインタ・データ項目、ミューテックスポインタ・データ項目、セマフォポインタ・データ項目、およびスレッドポインタ・データ項目

   は、MOVE 文の作用対象にすることはできない。

3. (VSC2) (MF) 送出し側項目か受取り側項目のどちらか一方が 2 バイト文字 (USAGE DISPLAY-1) である場合には、両方とも 2 バイト文字でなければならない。表意定数の SPACE は、2 バイト文字の送出し側として使用できる。

形式 1

4. (OSVS) (VSC2) (MF) 形式 1 では、送出し側項目か受取り側項目のどちらか一方が 2 バイト文字 (USAGE DISPLAY-1) である場合には、両方とも 2 バイト文字でなければならない。表意定数の SPACE は、2 バイト文字の送出し側として使用できる。
5. 一意名-1 には数字の関数一意名を指定できる。
6. 一意名-1 が、USAGE BINARY-CHAR、USAGE BINARY-SHORT、USAGE BINARY-LONG、または USAGE BINARY-DOUBLE を用いて記述されたデータ項目である場合は、一意名-2 は数字または数字編集項目であるとする。そうでない場合にデータの移動が有効であるかどうかは、*表 14-1 MOVE 文のデータ項類*の定義に従う。

形式 2

7. CORR は、CORRESPONDING の省略形である。
8. CORRESPONDING 指定を書くときは、すべての一意名は集団項目とする。
9. MOVE CORRESPONDING 文中の集団項目内に、以下の項目が含まれていてもよい。しかし、その転記はいっさい行われない。

   VSC2 MF ポインタ・データ項目、手続きポインタ・データ項目、

   ISO/IEC MF オブジェクト参照データ項目、プログラムポインタ・データ項目、

   MF イベントポインタ・データ項目、モニターポインタ・データ項目、ミューテックスポインタ・データ項目、セマフォポインタ・データ項目、およびスレッドポインタ・データ項目

一般規則

すべての形式

1. MF 一意名-1 が数字の関数一意名であるときは、一意名-2 は数字項目か数字編集項目のどちらかでなければならない。
2. 定数または一意名-1 によって指定されるデータが、一意名-2 によって指定される各データ項目に指定された順番で転記される。一意名-2 に関する規則は、他の受取り側にも適用される。一意名-2 が添字付けまたは指標付けされていると、そのデータ項目に転記が行われる直前に添字または指標が評価される。

   ANS85 一意名-1 が部分参照されているか、添字付けされているか、あるいは関数一意名である場合、その部分参照、添字、関数一意名の評価は、受取り側作用対象の最初のデータに転記が行われる直前に、1 回だけ行われる。

   ANS85 下記の文は

   ```
   MOVE a(b) TO b, c(b)
   ```

   下記の 3 つの文に相当する。

   ```
   MOVE a(b) TO temp
   MOVE temp TO b
   MOVE temp TO c(b)
   ```

   ここで、temp は中間結果を保持する項目であり、COBOL コンパイラによって用意される。

3. 送出し側項目も受取り側項目も基本項目である MOVE を、基本項目転記という。各基本項目は次の項類のどれかに属する。具体的には、数字、英字、英数字、数字編集、英数字編集、

   ISO/IEC MF 各国語、各国語編集、

(OSVS)(VSC2)(MF)または浮動小数点数

のどれかに属する。

これらの項類は PICTURE 句に説明してある。数字定数は、数字の項類に属する。文字定数は、英数字の項類に属する。表意定数の ZERO は、数字項目または数字編集項目に転記される場合は数字の項類に属し、それ以外の場合には英数字の項類に属する。表意定数の SPACE は、英数字の項類に属する。その他の表意定数は、すべて英数字の項類に属する。

これらの項類の間の基本項目転記に関して、下記の規則が適用される。

a. 表意定数の SPACE、英数字編集、英字のデータ項目を、数字または数字編集のデータ項目に転記してはならない。.
b. 数字編集のデータ項目を、数字または数字編集のデータ項目に転記してはならない。

   (ANS85)この制限は撤廃された。

c. 数字定数、表意定数の ZERO、数字データ項目、数字編集データ項目を、英数字データ項目に転記してはならない。
d. 整数でない数字定数または整数でない数字データ項目を、英数字または英数字編集のデータ項目に転記してはならない。
e. 上記以外の基本項目転記は文法的に正しく、下記の一般規則 5 に示す規則に従って実行される。

4. 文法的に正しい基本項目転記が実行されるときには、ある形式の内部表現から別の形式の内部表現へのデータの変換が、必要に応じて行われる。また、受取り側データ項目に指定されている編集

   (ANS85)または暗黙的に示されている逆編集

   も併せて行われる。

   a. 受取り側が英数字編集または英数字のデータ項目である場合、*COBOL 言語の概念*の章の*標準桁寄せ規則*節に定義されているように、桁寄せと必要な空白の穴埋めが行われる。送出し側の項目の大きさの方が、受取り側の項目の大きさよりも大きい場には、受取り側の項目がいっぱいになった後で、右側の過剰な文字が切り捨てられる。送出し側の項目が符号付きの数字である場合、演算符号は転記されない。演算符号が独立の文字位置を占める (*データ部 - ファイルおよびデータ記述*の章の *SIGN(符号)句*節を参照) 場合、演算符号は転記されず、送出し側の項目は実際の大きさよりも 1 文字小さいものとして扱われる (標準データ形式として) 。
   b. 受取り側が数字または数字編集のデータ項目である場合、*COBOL 言語の概念*の章の*標準桁寄せ規則*節に定義されているように、桁寄せと必要なゼロの穴埋めが行われる。ただし、ゼロを置き換える編集が行われている場合は例外である。

(ANS85)送出し側の項目の項類が数字編集である場合、その項目の編集されていない値を取り出すために、暗黙の逆編集が行われる。逆編集された値には、符号が付くことがある。次いで、編集されていない値が受取り側の項目に転記される。逆編集の効果は、DE-EDIT 指令の影響を受ける。

受取り側の項目が符号付きの数字である場合、送出し側の符号は受取り側項目に転記される。(*データ部 - ファイルおよびデータ記述*の章の *SIGN(符号)句*節を参照。) 必要に応じて、符号の表現形式の変換が行われる。送出し側の項目に符号が付いていない場合には、受取り側項目用に正符号が付加される。

受取り側の項目が符号なしの数字である場合、送出し側の項目の絶対値が受取り側の項目に転記され、符号は付けられない。

送出し側の項目の項類が英数字である場合、送出し側の項目が符号なしの整数として記述されているように、データが転記される。

受取り側の項目が数字または数字編集であり送出し側の項目が英数字である場合、送出し側の項目の内容が整数でないと、結果はどうなるかわからない。送出し側の英数字項目が整数でない場合は、エラーが報告されて、ゼロがその対象に転記される。(手続き部の章の*矛盾するデータ*節を参照。)

(ANS85)送出し側の項目が数字である場合、転記し戻すと数字編集項目に同じ値が現れるように転記操作が行われる (ただし、切捨てが発生した場合は除く) 。このデータ項目に編集文字列に適合しないデータが含まれる場合、受取り側にはゼロが転記される。

c. 受取り側の項目が英字である場合、*COBOL 言語の概念*の章の*標準桁寄せ規則*節に定義されているように、桁寄せと必要な空白の穴埋めが行われる。送出し側の項目の大きさの方が受取り側の項目の大きさよりも大きい場合には、受取り側の項目がいっぱいになった後で、右側の過剰な文字が切り捨てられる。
d. (OSVS)(VSC2)(MF)受取り側の項目が浮動小数点数である場合、送出し側の項目がまず最初に内部浮動小数点数に変換され、それから転記される。外部浮動小数点数項目へ、または外部浮動小数点数項目からデータを転記する場合は、該当データはいったん内部浮動小数点数に変換される。
e. (VSC2)(MF)受取り側の項目が 2 バイト文字である場合、送出し側の項目は 2 バイト文字または各国語でなければならない。送出し側の項目の大きさと受取り側の項目の大きさが異なる場合は、送出し側の項目の右側が切り捨てられるか、受取り側の項目の右側に 2 バイト文字の空白が埋められる。

(ISO2002)(MF)受取り側の項目が各国語である場合、送出し側の項目も 2 バイト文字または各国語でなければならない。送出し側の項目の大きさと受取り側の項目の大きさが異なる場合は、送出し側の項目の右側が切り捨てられるか、受取り側の項目の右側に各国語の空白が埋められる。

5. 基本項目転記以外の転記は、英数字の基本項目同士の転記とまったく同じように扱われる。ただし、データの内部表現形式の変換は行われない。この形の転記においては、送出し側または受取り側の項目の下位に属する個々の基本項目または集団項目を考慮することなく、受取り側の項目にデータが詰め込まれる。ただし、OCCURS 句の一般規則に述べたことは例外である。
6. 表 14-1 に、各種の MOVE 文の適合性をまとめて示す。表の中に示した一般規則の参照番号は、該当事項を禁止または裏付けている規則を示す。

形式 2

7. CORRESPONDING 指定を書くと、一意名-1 に属する該当する項目が、一意名-2 に属する対応する項目に転記される。これに適用される規則については、**手続き部**の章の *CORRESPONDING 指定*節に説明してある。CORRESPONDING 指定が含まれる MOVE 文を実行することは、対応する一意名の間で個々に転記を行う MOVE 文を実行することに等しい。

   OSVS VSC2 この処理は、各宛先集団ごとに繰り返される。

表 14-1：MOVE 文のデータ項類

送出し側のデータ項目の項類	受取り側のデータ項目の項類 [1]				
	英字	英数字/英数字編集	数字 (整数/非整数)	数字編集外部/内部浮動小数点数	2 バイト文字、各国語、または各国語編集
英字	Yes/4c	Yes/4a	No/3a	No/3a	No/4e
英数字	Yes/4c	Yes/4a	Yes/4b[2]	Yes/4b	No/4e
英数字編集	Yes/4c	Yes/4a	No/3a	No/3a	No/4e
数字：整数	No/3b	Yes/4a	Yes/4b	Yes/4b	No/4e
数字：非整数	No/3b	No/3c	Yes/4b	Yes/4b	No/4e
数字編集	No/3b	Yes/4a	Yes/4b	Yes/4b	No/4e
2 バイト文字、各国語、または各国語編集	No/4e	No/4e	No/4e	No/4e	Yes/4e

[1] 各欄に該当する一般規則の番号を示す。

[2] 非整数の英数字定数が送出し側として使用された場合、エラーとなる。

14.3 MULTIPLY (乗用) 文

MULTIPLY (乗算) 文は、数字データ項目どうしを掛け合わせて、その結果をいくつかのデータ項目に入れる。

一般形式

形式 1

```
MULTIPLY { 一意名-1 } BY {一意名-2 [ROUNDED]}...
         { 定数-1   }

[ON SIZE ERROR 無条件文-1]
[NOT ON SIZE ERROR 無条件文-2]
[END-MULTIPLY]
```

ANS85

形式 2

```
MULTIPLY { 一意名-1 } BY { 一意名-2 }
         { 定数-1   }   { 定数-2   }

                GIVING {一意名-3 [ROUNDED]}...

[ON SIZE ERROR 無条件文-1]
[NOT ON SIZE ERROR 無条件文-2]
[END-MULTIPLY]
```

ANS85

構文規則

すべての形式

1. 各一意名は、数字基本項目を指さなければならない。ただし、形式 2 の語 GIVING に続く各一意名は、基本数字項目の他に基本数字編集項目であってもよい。
2. 各定数は、数字定数とする。
3. 作用対象の合成は、受取り側のデータ項目すべてを使用して決定される。*手続き部*の章の*算術文*節を参照。
4. OSVS OSC2 MF 数字データ項目または数字定数を指定できるところでは、どこでも浮動小数点数データ項目および浮動小数点数定数を使用できる。

一般規則

すべての形式

1. *手続き部*の章の *ROUNDED 指定*、*ON SIZE ERROR 指定*、*算術文*、*作用対象の重なり*、および*算術文における複数個の答え*の各節を参照。

形式 1

2. 形式 1 では、一意名-1 または定数-1 の値に一意名-2 の値が掛けられる。乗数 (一意名-2) の値は、積によって置き換えられる。同様に、一意名-2 の後続の値に対して、左から右に同じ処理が繰り返される。

形式 2

3. 形式 2 では、一意名-1 または定数-1 の値に一意名-2 の値が掛けられる。乗数 (一意名-2) の値は、積によって置き換えられる。同様に、一意名-2 の後続の値に対して、左から右に同じ処理が繰り返される。

14.4 NEXT SENTENCE (次の完結文) 文

MF

NEXT SENTENCE (次の完結文) 文は、制御を次の COBOL 完結文に移す。つまり、次の終止符 (.) に続く完結文に制御を移す。これに対し、動詞 CONTINUE は、論理的に次の COBOL の動詞に制御を移す。

一般形式

```
NEXT SENTENCE
```

構文規則

1. 条件文または無条件文を書けるところには、どこでも NEXT SENTENCE を使用できる。.

一般規則

1. NEXT SENTENCE 文は、次の終止符 (.) に続く、論理的に次となる COBOL の動詞に、実行の流れを変えさせる。

14.5 NOTE (注記) 文

OSVS

NOTE (注記) 文は、完結文または段落を注記として扱う。

一般形式

```
NOTE 文字列
```

構文規則

1. 文字列には、計算機の文字集合に属する文字を自由に組み合わせて書ける。

一般規則

1. 段落の最初の完結文が NOTE 文であると、その段落全体が注記として扱われる。
2. 段落の冒頭以外の場所に NOTE 文を置くと、次の分離符である終止符 (.) が出てくるまでのテキストが注記として扱われる。

14.6 ON (回数間隔) 文

(OSVS)

ON (回数間隔) 文は、手続き文を周期的に実行できるようにする。

一般形式

```
ON { 定数-1 }  [ AND EVERY { 定数-2 } ]
   { 一意名-1 }            { 一意名-2 }
        [ UNTIL { 定数-3 } ] { 無条件文-1 }
                { 一意名-3 } { NEXT SENTENCE }
              [ { ELSE      } { 無条件文-2 } ]
                { OTHERWISE } { NEXT SENTENCE }
```

構文規則

1. 一意名-1、一意名-2、一意名-3 は、符号なしの整数の基本項目とする。
2. 定数-1、定数-2、定数-3 は、符号なしの数字定数とする。

一般規則

1. 各 ON 文が最初に実行される前に、その ON 文用に暗黙的に定義されたカウンタ (暗黙の ON カウンタ) がゼロに初期化される。
2. 指定する一意名-1、一意名-2、一意名-3 は、ON 文が実行されるときには、正の整数値をとらなければならない。次に ON 文を実行するまでの間に、これらの値を変更すると、以降の ON 文の実行に影響が出る。
3. 暗黙の ON カウンタの内容は、実行の流れをその ON 文に移さないかぎり、変更されることはない。(呼ばれたプログラムの ON カウンタを設定し直すには、そのプログラムを取り消すしかない。EXIT PROGRAM 文を実行し、その後そのプログラムを再び呼び出した場合、CANCEL 文をはさんでいなければ ON カウンタの値は元のままである。)
4. 下記の値リストが評価される。
    a. 一意名-1 または定数-1 の現在の値

b. 一意名-1 または定数-1 の現在の値に、一意名-2 または定数-2 の現在の値を繰り返し加えた一連の値。この繰り返しは、一意名-1 または定数-1 の現在の値が、一意名-3 または定数-3 の値に達するまで行われる。

次いで、暗黙の ON カウンタが上記の一連の値と比較される。一連の値の中に ON カウンタと一致するものがあれば、無条件文-1 が実行される。そうでなければ、無条件文-2 が実行される。

14.7 OPEN (開く) 文

OPEN (開く) 文は、ファイルの処理を行うための準備 (ファイルのオープン) をする。また、見出しの確認や書込み、その他の入出力操作を行う。

(ANS85)OPEN 文の REVERSED 指定は、ANSI '85 標準では廃要素に分類されており、ANSI 標準の次回の全面改訂の際に、削除される予定である。

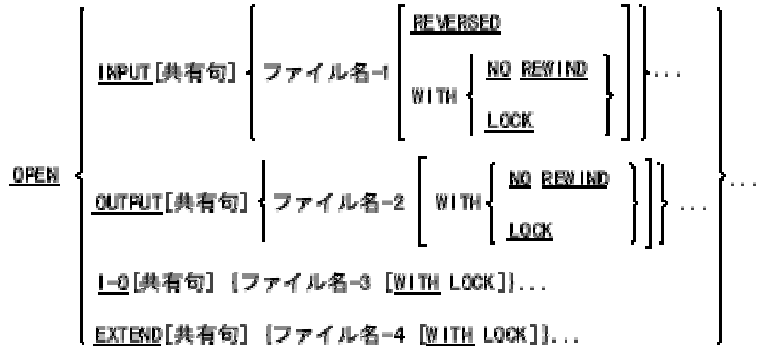
(MF)この構文は、Micro Focus COBOL に組み込まれているすべての方言で全面的に使用できる。FLAGSTD 指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。

(XOPEN)標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、OPEN 文の REVERSED 指定は明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内ではこの文を使用するべきではない。

(MF)OPEN 文は、桁詰めデータ項目用のシステム資源を割り当て、このデータ項目を初期化する。

一般形式

形式 1 (レコード順ファイル)



形式 2 (行順ファイル、相対ファイル、索引ファイル)

```
OPEN { INPUT[共有句]{ファイル名-1 [WITH LOCK]}...
       OUTPUT[共有句]{ファイル名-2 [WITH LOCK]}...
       I-O[共有句]{ファイル名-3 [WITH LOCK]}...
       EXTEND[共有句]{ファイル名-4 [WITH LOCK]}... }...
```

SHARING 句の部分は以下の通りとする。

```
[ SHARING WITH { ALL OTHER
                 NO OTHER
                 READ ONLY } ]
```

(MF)形式 3 (桁詰めデータ項目)

```
OPEN { identifier-1 }...
```

構文規則

すべての形式 (すべてのファイル)

1. OPEN 文が対象とするファイルは、編成または呼出し法が同じである必要はない。
2. (SO2002)(MF)SHARING ALL を指定して INPUT 指定を書かない場合、ファイル名-1 のファイル管理記述項に LOCK MODE 句を指定しなければならない。

形式 1 (レコード順ファイル)

3. (MF)NO REWIND は注記となる。
4. I-O 指定はディスク･ファイルに対してだけ適用できる。
5. EXTEND 指定は、LINAGE 句を指定していないファイルに対してだけ適用できる。
6. 複数ファイル･リールに対しては、EXTEND を指定できない。

形式 2 (
(MF)行順ファイル、
相対ファイル、索引ファイル)

7. (AN585)順呼出し法のファイルに対して、EXTEND を指定できる。
8. (SO2002)(MF)行順編成のファイルには SHARING 指定を書いてはならない。

形式 3 (桁詰めデータ項目)

9. (MF)一意名-1 は、イベントポインタ、モニターポインタ、ミューテックスポインタ、またはセマフォポインタのいずれかのデータ項目の USAGE を使用して、基本データ項目を参照しなければならない。

一般規則

形式 1 および 2 (すべてのファイル)

1. OPEN 文が正常に実行され終わるまで、該当するファイルを明示的または暗黙的に対象とするどのような文も実行することはできない (USING または GIVING を指定した SORT 文および MERGE 文は例外である)。
2. (SOURCE)(MF)OPEN 文が正常に実行されると、ファイルの利用可能性が判定され、ファイル名によって参照されるファイル結合子が OPEN 文のモードの状態に置かれる。そして、ファイル結合子を通じて、ファイルがファイル名と関係づけられる。

   必要語の INPUT, OUTPUT, I-O, および EXTEND を用いて、ファイル結合子を通じてファイルに適用する入出力操作を指定し、OPEN 文のモードを確立する。(表 14-4 *文と OPEN モードの許される組合せ*を参照。)

   ファイルが物理的に存在し、ファイル処理システムに認識されていれば、そのファイルは利用可能である。表 14-2 *(現在は開かれていない) 利用可能なファイルと利用不可能なファイルを開く処理*に現在は開かれていない利用可能なファイルと利用不可能なファイルを開いたときの結果を示す。表 14-3 *現在は他のファイル結合子によって開かれている利用可能な共有ファイルを開く処理*に現在は他のファイル結合子によって開かれている利用可能なファイルと利用不可能なファイルを開いたときの結果を示す。

表 14-2: (現在は開かれていない) 利用可能なファイルと利用不可能なファイルを開く処理

OPEN 文のモード	ファイルは利用可能	ファイルは利用不可能
INPUT	正常に開ける	開けない
INPUT (不定ファイル)	正常に開ける	正常に開ける。最初に読んだときに、ファイル終了条件または無効キー条件が発生する。
I-O	正常に開ける	開けない
I-O (不定ファイル)	正常に開ける	開くとファイルが作成される
OUTPUT	正常に開ける; 中身は空	開くとファイルが作成される
EXTEND	正常に開ける	開けない

EXTEND（不定ファイル）	正常に開ける	開くとファイルが作成される

表 14-3：現在は他のファイル結合子によって開かれている利用可能な共有ファイルを開く処理

開く要求		共有の度合いと OPEN 文のモード				
		他と共有せず	読出しのみ共有		他のすべてと共有	
		拡張 I-O 入力 出力	拡張 I-O 出力	入力	拡張 I-O 出力	入力
SHARING WITH NO OTHER	EXTEND I-O INPUT OUTPUT	開けない	開けない	開けない	開けない	開けない
SHARING WITH READ ONLY	EXTEND I-O	開けない	開けない	開けない	開けない	正常に開ける
	INPUT	開けない	開けない	正常に開ける	開けない	正常に開ける
	OUTPUT	開けない	開けない	開けない	開けない	開けない
SHARING WITH ALL OTHER	EXTEND I-O	開けない	開けない	開けない	正常に開ける	正常に開ける
	INPUT	開けない	正常に開ける	正常に開ける	正常に開ける	正常に開ける
	OUTPUT	開けない	開けない	開けない	開けない	開けない

3. OPEN 文の実行が正常に終了すると、対象となったファイルのレコード領域が利用可能となる。
4. 表 14-4 において、行と列の交差する格子内の"X"は、該当の行のアクセス・モードでの入出力モードの指定を、列の上部に示されている OPEN 文のモードに適用できることを示す。

表 14-4：OPEN 文モードの有効な組み合わせ

呼出し法	OPEN 文のモード				
	文	INPUT	OUTPUT	INPUT-OUTPUT	EXTEND

順呼出し	READ	X		X	
	WRITE		X		X
	REWRITE			X	
乱呼出し(非順編成ファイル)	READ	X		X	
	WRITE		X	X	
	REWRITE			X	
	START				
	DELETE			X	
動的呼出し(非順編成ファイル)	READ	X		X	
	WRITE		X	X	
	REWRITE			X	
	START	X		X	
	DELETE			X	

注: START 文と DELETE 文は、レコード順ファイル

(MF)または行順ファイル

には適用できない。

---

5. (MF)WITH LOCK を指定すると、OPEN 文は該当するファイル全体に対してロックする。(これは、そのファイルの SELECT 文の中に LOCK MODE IS EXCLUSIVE を指定することに等しい。*環境部*の章の*ファイル管理記述項*節を参照)。
6. (ANS85)OPEN 文の実行中にファイル属性の間に矛盾が検出されると、その OPEN 文の実行は不成功となる。OPEN 文の実行中にどのファイル固有属性が妥当性を検査されるかは、COBOL システムごとに定義されている。詳細については、ファイルの取扱いに関する説明書を参照。ファイルの編成や記憶媒体によって、どのファイル固有属性が妥当性を検査されるかは変わってくる。
7. (ANS85)同じ実行単位の中で 1 つのファイルを開くのに、INPUT, OUTPUT, EXTEND, I-O のどのモードでも指定できる。 ファイル結合子に関して最初に OPEN 文を実行した後では、そのファイル結合子に関して CLOSE 文を実行してからでないと、同じファイル結合子に関して再び OPEN 文を実行することはできない。対象のファイルが不定ファイルであっても、この規則は当てはまる。
8. OPEN 文を実行しても、最初のデータレコードは入手も解放もされない。
9. SELECT 句中で指定したファイルの外部名は、下記のように処理される。
    a. INPUT を指定した場合、OPEN 文を実行すると、入力ファイルを開くためのオペレーティングシステムの規則に従って、割り当てられた名前が検査される。

b. OUTPUT を指定した場合、OPEN 文を実行すると、出力ファイルを開くためのオペレーティングシステムの規則に従って、割り当てられた名前が書き出される。

10. ファイル名-1、ファイル名-2、ファイル名-3、ファイル名-4 のファイル記述項は、そのファイルを作成したときと同じにする。
11. INPUT または I/O を指定して開いたファイルに関して、OPEN 文はファイル位置指示子を設定する。その値は、索引ファイルと順ファイルに関してはファイル内に現存する最初のレコードとされ、相対ファイルに関してはレコード番号 1 とされる。ファイル中にレコードが存在しないと、索引データまたは順ファイルが次に読まれたときにファイル終了条件が発生するように、ファイル位置指示子が設定される。ファイルそのものが存在しないと、OPEN INPUT 文は誤り状態となる。
12. I-O を指定して開いたファイルの LABEL RECORDS 句に、ラベル・レコードがあると指定されていると、OPEN 文の実行に下記の処理が含められる。
    a. 入出力両用ファイルのラベルの検査に関するオペレーティングシステムの規則に従って、ラベルが検査される。
    b. 入出力両用ファイルのラベルの書出しに関するオペレーティングシステムの規則に従って、ラベルが書き出される。
13. OUTPUT を指定した OPEN 文の実行が正常に終了すると、ファイルが作成される。その時点では、そのファイルにはまだレコードは含まれていない。同じ名前のファイルが存在すると、そのファイルは削除される。そのファイルが書き込み保護されていると、誤りが発生する。
14. (ANS85)EXTEND を指定すると、OPEN 文はファイルへの次の書き出し位置を、そのファイルの最後の論理レコードの直後に位置付ける。順ファイルの場合、最後の論理レコードは最後にファイルに書き出されたレコードである。相対ファイルの場合、最後の論理レコードは現存するレコードの中で相対レコード番号が最も大きいものである。索引ファイルの場合、最後の論理レコードは現存するレコードの中で主レコードキーの値が最も大きいものである。
15. OPEN 文を実行すると、FILE STATUS データ項目の値が更新される（*手続き部の入出力状態*節を参照）。
16. (MF)該当するファイルの SELECT/ASSIGN 文の中で LOCK MODE IS EXCLUSIVE を指定してあると、OPEN 文が正常に実行されたときに、ファイルはその実行単位用に排他的にロックされる。
17. (MF)該当するファイルの SELECT/ASSIGN 文の中で LOCK MODE IS AUTOMATIC または LOCK MODE IS MANUAL を指定してあると、OPEN 文の実行が正常に終了したときに、ファイルは共有可能とされる。共有可能なファイルは、複数の実行単位から正常に開くことができる。
18. (MF)OUTPUT 用に開いたファイルおよび EXTEND 用に開いた相対および索引ファイルは、暗黙的に排他的にロックされたものと定義される。したがって、そのファイルは共有できない。
19. (MF)I-O 用に開いた共有ファイルについてだけ、レコードロックを得ることができる。
20. OPEN 文の実行が不成功に終わると、物理ファイルは影響を受けず、下記の処理が順に行われる。
    a. 該当するファイルに対応する入出力状態に、OPEN 文の実行が不成功に終わった理由を示す値が設定される。
    b. 実行すべき条件に該当する USE AFTER EXCEPTION 手続きがあれば、それが実行される（*手続き部 - SEARCH - WRITE* の章の *USE(使用)文*節を参照）。

21. I-O を指定してファイルを開くと、そのファイルには入力操作も出力操作も加えることができる。ファイルが存在しないときは、下記の規則が適用される。
    a. OPTIONAL を指定してあれば、ファイルが作成される。
    b. NOT OPTIONAL を指定してあれば、誤りが発生する。
    c. どちらの句も指定しないで、OPTIONAL-FILE 指令を原始要素のコンパイル時に指定してあれば、ファイルが作成される。
    d. どちらの句も指定しないで、NOOPTIONAL-FILE 指令を原始要素のコンパイル時に指定してあれば、誤りが発生する。
22. 入力ファイルの SELECT 句に OPTIONAL 指定をしていると、ファイルのオープン時にそのファイルの有無を尋ねてくる。そのファイルが存在しないと、最初に READ 文を実行したときに、ファイル終了条件が発生する。
23. EXTEND を指定すると、OPEN 文はファイルへの次の書き出し位置を、そのファイルの最後の論理レコードの直後に位置付ける。以降、そのファイルを対象に WRITE 文を実行すると、OUTPUT を指定したかのように、レコードがそのファイルに追加される。ファイルが存在しない場合は、作成される。
24. (ANSI85)EXTEND を指定したファイルのファイル記述項に LABEL RECORDS 句が指定してあると、OPEN 文の実行に下記の処理も含められる。
    a. 単一リール/ユニット・ファイルの場合だけ、開始ファイル・ラベルが処理される。
    b. 最後に存在するリール/ユニット上の開始リール/ユニット・ラベルは、そのファイルを INPUT を指定して開いたかのように処理される。
    c. 既存の終了ラベルは、そのファイルを INPUT を指定して開いたかのように処理される。その後で、終了ラベルは削除される。
    d. その後、そのファイルを OUTPUT を指定して開いたかのように処理が続行される。
25. I/O を指定すると、ファイルを入力と出力の両方の処理用に開ける。

    (MF)ただし、ORGANIZATION LINE SEQUENTIAL のファイルを除く。

    該当のファイルが存在しない場合には、そのファイルが作成されて、入力用には空のファイルとして使用される。

    (MF)ただし、SELECT 文に NOT OPTIONAL が指定された場合は別である。

    作成されたばかりのファイルから読込みを行うと、エラーとなる。

26. SHARING 指定は共有ファイルに対してのみ指定できる（*翻訳集団の概念*の章の*共有モード*節を参照）。
27. (SOURCE)(MF)SHARING 指定は、ファイルを共有しているファイル結合子を通じて、対象のファイルにどのような処理を行えるかを示す（*翻訳集団の概念*の章の*共有モード*節を参照）。
28. (SOURCE)(MF)T ファイル名に関連するファイル結合子のファイル管理記述項に記述されている SHARING 指定よりも、この SHARING 指定が優先する。OPEN 文に SHARING が指定されていない場合、ファイルの共有はファイル管理記述項によって全面的に指定される。OPEN 文に SHARING 指定がなく、ファイル管理記述項にも SHARING 句が指定されていない場合、該当のファイル結合子に関する共有モードは、Sharing モ

ードに応じて設定される。このとき適用される規則は、*翻訳集団の概念*の章の*共有モード*で解説する。

形式 1 (レコード順ファイル)

29. ファイルの記憶媒体が巻き戻し可能なものであるときは、下記の規則が適用される。
    a. OPEN 文を実行すると、ファイルの先頭に現在位置が位置付けられる。
    b. REVERSED 指定があるときは、OPEN 文を実行すると、ファイルの末尾に現在位置が位置付けられる。
30. REVERSED 指定があるときは、そのファイルに対して以降 READ 文を実行すると、データは逆の順に、つまり最後のレコードから読み込まれる。

形式 2 (行順ファイル)

31. (MF)行順ファイルに対しては、I-O モードを適用することはできない。ただし、REWRITE-LS 指令を設定した場合は、そのかぎりではない。

(MF)形式 3 (桁詰めデータ項目)

32. 桁詰めデータ項目に対して OPEN 文を実行する場合は、それが完了する前に、このデータ項目を参照する他の文を実行することはできない。
33. OPEN 文の実行が完了したかどうかにより、桁詰めデータ項目が使用可能かどうかが決定される。
34. OPEN の実行により、特殊レジスタ、RETURN-CODE が、桁詰めデータ項目ライブラリ手順について指定された値の一つに更新される。
35. 桁詰めデータ項目に対する OPEN 文の最初の実行の後、その後に同じデータ項目に対して実行される OPEN 文は、このデータ項目に対する CLOSE 文が実行された後に実行されるものとする。
36. 一意名-1 がイベントデータ項目を参照する場合は、OPEN 文の実行により、このデータ項目が FALSE に初期化される。
37. 一意名-1 がモニターデータ項目を参照する場合は、OPEN 文の実行により、モニターが NOT BROWSING、NOT READING、および NOT WRITING に初期化される。
38. 一意名-1 がミューテックスデータ項目を参照する場合は、OPEN 文の実行により、このデータ項目が OFF に初期化される。
39. 一意名-1 がセマフォデータ項目を参照する場合は、OPEN 文の実行により、このデータ項目がゼロに初期化される。

第 15 章 : 手続き部 - PERFORM - ROLLBACK

15.1 PERFORM(実行)文

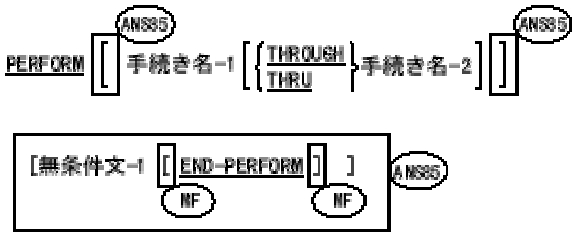
PERFORM (実行) 文は、いくつかの手続きに制御を明示的に移し、その手続きの実行が終了した時点で、暗黙的に制御を戻すために使用する。

ANS85 PERFORM 文は、その有効範囲内にあるいくつかの無条件命令の実行を制御するためにも使用する。

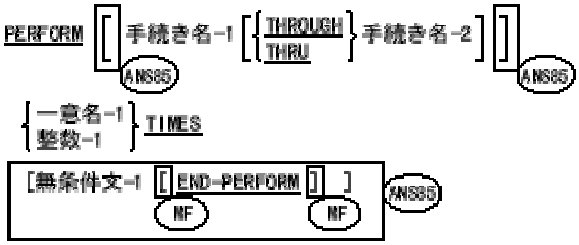
手続き部 - ENTER - INVOKE の章の *EXIT(出口)文*節を参照。

一般形式

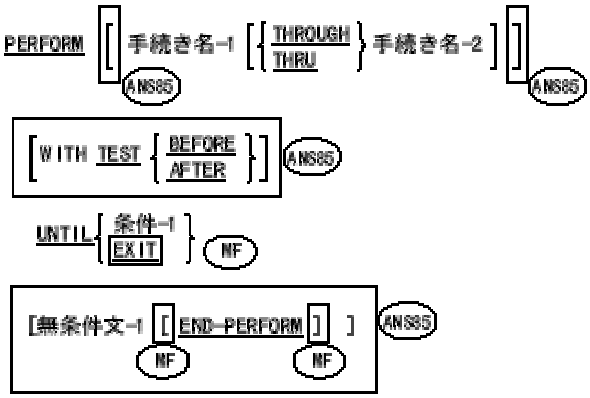
形式 1



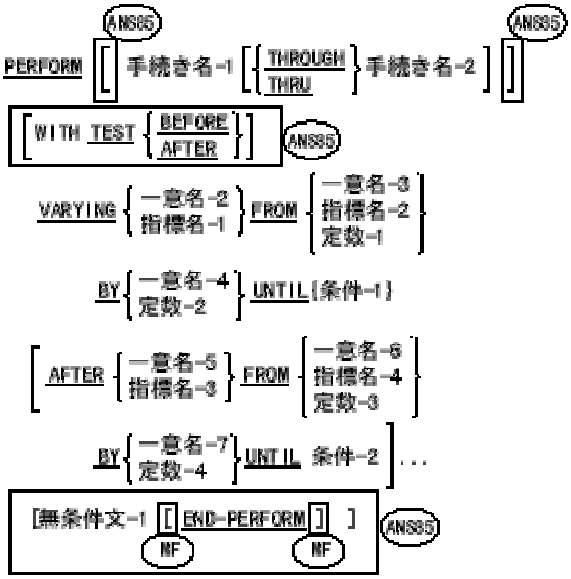
形式 2



形式 3



形式 4



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
   - PERFORM-TYPE - 重複する PERFORM の範囲の取扱いを制御する。

構文規則

すべての形式

1. (ANS85)手続き名-1 を省略するときは、無条件文-1 と END-PERFORM 指定を書く。手続き名-1 を書くときは、無条件文-1 と END-PERFORM 指定を書いてはならない。

(MF)無条件文-1 を書けば、END-PERFORM 指定を省略してもよい。

(COB370)手続き名-1 と無条件文-1 を両方とも省略してもよい。

2. 語 THROUGH と THRU は同義であり、どちらを書いてもよい。
3. 手続き名-1 と手続き名-2 を両方書いたときに、どちらか一方が手続き部の宣言部分に含まれる手続き名を指しているならば、両方の手続きとも同じ宣言節内になければならない。

形式 2 および 4

4. 各一意名は、データ部に記述された数字基本項目を表わす。形式 2 では、一意名-1 は整数とする。
5. (OSVS)(VSC2)(MF)数字データ項目または定数を指定できるところでは、どこでも浮動小数点数のデータ項目または定数を使用できる。ただし、整数が要求される箇所には使用できない。

形式 3

6. TEST 指定を書いた場合、EXIT 指定を書いてはならない。

形式 3 および 4

7. (ANS85)TEST BEFORE 指定も TEST 指定も書かないと、TEST BEFORE 指定を書いたものとみなされる。
8. 条件-1、条件-2 などは任意の条件式であってよい。(*手続き部*の章の*条件式*節を参照。)

形式 4

9. (ANS85)形式 4 では、手続き名-1 を省略するときは、AFTER 指定を書いてはならない。

   (SOURCE)(MF)この制限は適用されない。

10. 各定数は、数字定数を表わす。
11. VARYING 指定または AFTER 指定の中に指標名を指定する場合は、下記のようにする。
    a. 対応する FROM 指定および BY 指定の中の一意名は、整数データ項目とする。
    b. 対応する FROM 指定の中の定数は、正の整数とする。
    c. 対応する BY 指定の中の定数は、ゼロでない整数とする。
12. FROM 指定の中に指標名を指定する場合は、下記のようにする。
    a. 対応する VARYING 指定または AFTER 指定の中の一意名は、整数データ項目とする。
    b. 対応する BY 指定の中の一意名は、整数データ項目とする。
    c. 対応する BY 指定の中の定数は、整数とする。

13. BY 指定の中の定数は、ゼロであってはならない。
14. 形式 4 では、AFTER 指定を 2 つ書ける。

    (ANS85)形式 4 では、AFTER 指定を 6 つ書ける。

    (MF)この制限は、15 にまで拡大されている。

一般規則

すべての形式

1. 手続き名-1 を指定した PERFORM 文を「外 PERFORM 文」という。

   (ANS85)手続き名-1 を省略した PERFORM 文を「内 PERFORM 文」という。

2. 外 PERFORM 文の手続き名-1 (手続き名-2 が指定されていれば手続き名-2 まで) の範囲内に含まれている文、

   (ANS85)または、内 PERFORM 文自体の中に含まれている文

   を「指定された文の組」という。

3. (ANS85)END-PERFORM 指定は、内 PERFORM 文の範囲を区切る。(*COBOL 言語の概念*の章の*明示範囲符と暗黙範囲符*節を参照。)
4. (ANS85)内 PERFORM 文は、基本的には、外 PERFORM 文用の下記の規則に従って機能する。両者の違いは、外 PERFORM 文では、手続き名-1 (手続き名-2 が指定されていれば手続き名-2 まで) の範囲内に含まれている文が実行されるのに対して、内 PERFORM 文ではその PERFORM 文自体の中に含まれている文が実行されることである。以降の記述において、特に「内」または「外」と明示しないかぎり、外 PERFORM 文に当てはまる一般規則は、すべて内 PERFORM 文にも当てはまるものとする。
5. PERFORM 文が実行されると、制御は指定された文の組の最初の文に移される (一般規則 8b、8c、8d に示す場合を除く) 。この制御の移行は、PERFORM 文が実行されるたびに 1 回だけ発生する。>指定された文の組へ制御が移されると、PERFORM 文の末尾に戻るための制御の暗黙移行が、下記のように用意される。
   a. 手続き名-1 が段落名であり手続き名-2 は指定されていない場合、手続き名-1 の最後の文が実行されると、制御は PERFORM 文の末尾に戻る。
   b. 手続き名-1 が節名であり手続き名-2 は指定されていない場合、手続き名-1 の最後の段落の最後の文が実行されると、制御は PERFORM 文の末尾に戻る。
   c. 手続き名-2 が指定されておりそれが段落名である場合、その段落の最後の文が実行されると、制御は PERFORM 文の末尾に戻る。
   d. 手続き名-2 が指定されておりそれが節名である場合、その節内の最後の段落の最後の文が実行されると、制御は PERFORM 文の末尾に戻る。
   e. (ANS85)内 PERFORM 文の実行は、 その中に含まれる最後の文が実行された後で完了する。

f. (MF)内 PERFORM 文において、その中のすべての文の実行が終わる前に処理を終了させるために、EXIT PROGRAM 文を使用できる。

6. 手続き名-1 と手続き名-2 の間には、関連がある必要はない。ただ、手続き名-1 から始まって手続き名-2 で終わるように実行すればよい。手続き名-1 と手続き名-2 の終わりの間に、GO TO 文や PERFORM 文があってもよい。手続き名-1 から手続き名-2 の終わりに至る間にいくつかの論理経路が存在する場合は、手続き名-2 を段落名とし、その中に EXIT 文だけを置いて、すべての経路をそこに至らせるとよい。
7. 指定された文の組への制御の移行が PERFORM 文以外の手段で行われると、その文の組の末尾から次の実行文に制御が移される。つまり、PERFORM 文がその文の組を参照していないように扱われる。
8. PERFORM 文は、下記のように機能する。
   a. 形式 1 が基本的な PERFORM 文である。この形式の PERFORM によって参照される指定された文の組が 1 回実行され、その後で制御はこの PERFORM 文の末尾に戻る。
   b. 形式 2 は、 PERFORM ... TIMES 指定で、一定の回数だけ繰り返す PERFORM 文である。指定された文の組が、整数-1 または一意名-1 のデータ項目の初期値によって示される回数だけ実行される。PERFORM 文を実行するときに、一意名-1 のデータ項目の値がゼロまたは負である場合は、制御は PERFORM 文の末尾に移される。指定された文の組が指定された回数実行された後で、制御は PERFORM 文の末尾に移される。

      PERFORM 文の実行中は、一意名-1 は指定された次の実行回数を、一意名-1 のデータ項目の初期値によって示される回数から変えることはできない。

      整数-1 はゼロまたは正でなければならない。符号を付けてもよい。

   c. 形式 3 は PERFORM ... UNTIL 指定で、条件が満たされるまで繰り返す PERFORM 文である。指定された文の組が、UNTIL 指定に書かれた条件が真になるまで繰り返し実行される。条件が真になると、制御は PERFORM 文の末尾に移される。PERFORM 文に入ったときに条件が真であり、

      (ANS85)TEST BEFORE が指定、または暗黙に指定されていると、

      制御は手続き名-1 へ移行されず、PERFORM 文の末尾に移される。

      (ANS85)TEST AFTER が指定されている場合も、PERFORM 文は TEST BEFORE が指定されているのと同じように機能する。ただし、終了条件の判定が行われるのは、指定された文の組が実行されてからとなる。条件-1 中に指定された作用対象に関連する添字付けまたは部分参照は、条件が検査されるたびに評価される。

      (MF)UNTIL EXIT を指定すると、指定された文の組は、その中のどれかの文によって終了させられるまで、繰り返し実行される。外 PERFORM 文の場合は、この繰り返しを終了させられる文は、EXIT PROGRAM および STOP RUN だけである。内 PERFORM 文の場合は、EXIT PERFORM 文および GO TO 文によっても、この繰り返しを終了させることができる。

d. 形式 4 は、PERFORM ... VARYING 指定で、関連する要素の値を変化させながら条件が満たされるまで繰り返す PERFORM 文である。この形の PERFORM 文は、実行中にいくつかの一意名や指標名の値を、規則的に変化させるときに使用する。以下の記述において、VARYING、AFTER、FROM の各指定の作用対象としての一意名に関して述べることは指標名にも当てはまる。

指標名-1 を指定する場合、PERFORM 文の実行を開始する時点での一意名-3、指標名-2、定数-1 の値は、指標名-1 を用いている表の中の要素の出現番号と対応させる。指標名-3 を指定する場合、PERFORM 文の実行を開始する時点での一意名-6、指標名-4、定数-3 の値は、指標名-3 を用いている表の中の要素の出現番号と対応させる。

下に述べるように、指標名-1 または指標名-3 を増加させていく際に、指標の値は対応する表の範囲を超えてはならない。ただし、PERFORM 文の実行が終了する際には、指標の値は対応する表の範囲から 1 増分または 1 減分だけ外れることがある。

一意名-2 または一意名-5 に添字が付けられている場合、これらの一意名のデータ項目の内容が設定または増加されるたびに、その添字が評価される。一意名-3、一意名-4、一意名-5、一意名-6 に添字が付けられている場合、これらの一意名のデータ項目の内容が設定または増加の処理に使用されるたびに、その添字が評価される。条件-1 または条件-2 の中に指定されている作用対象に添字または部分参照が適用されていると、条件が検査されるたびにその添字または部分参照が評価される。

以降に、形式 4 の PERFORM 文がどのように機能するかを示す例を掲げる。

1. TEST BEFORE を明示的または暗黙的に指定した場合

   1 つの一意名のデータ項目の値を変化させる場合、PERFORM 文の実行開始時点で、一意名-2 のデータ項目の内容が、定数-1 または一意名-3 のデータ項目の現在の値に設定される。UNTIL 指定の条件-1 が偽であれば、指定された文の組が 1 回実行される。次いで、一意名-2 のデータ項目の値が指定された値 (定数-2 または一意名-4 のデータ項目の値) だけ増分または減分されて、条件-1 が再び評価される。この処理が、条件-1 が真になるまで繰り返される。PERFORM 文の実行開始時点で条件-1 が真である場合は、制御は PERFORM 文の末尾に移される。図 15-1 を参照。

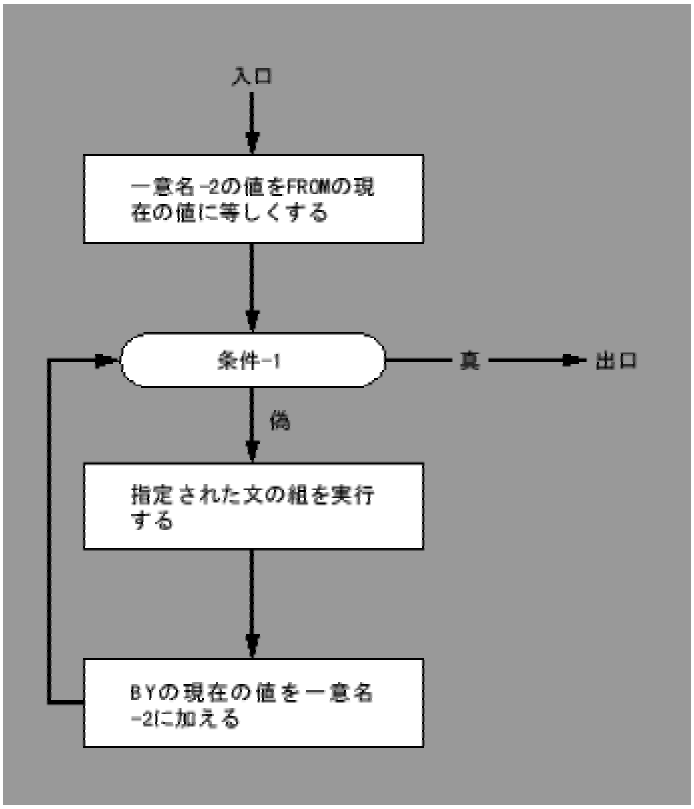


図 15-1: PERFORM 文の機能の流れ図 - VARYING, TEST BEFORE, 1 つの条件

2 つの一意名のデータ項目の値を変化させる場合、PERFORM 文の実行開始時点で、一意名-2 のデータ項目の内容が、定数-1 または一意名-3 のデータ項目の現在の値に設定される。また、一意名-5 のデータ項目の内容が、定数-3 または一意名-6 のデータ項目の現在の値に設定される。図 15-2 を参照。

これから条件-1 が評価される。この結果が真である場合は、制御は PERFORM 文の末尾に移される。この結果が偽である場合は、条件-2 が評価される。この結果が偽であれば、指定された文の組が 1 回実行される。次いで、一意名-5 のデータ項目の値が、定数-4 または一意名-7 のデータ項目の現在の値だけ増分または減分される。それから条件-2 が再び評価される。

この処理が、条件-2 が真になるまで繰り返される。条件-2 が真になると、一意名-2 のデータ項目の値が、定数-2 または一意名-4 のデータ項目の現在の値だけ増分され、一意名-5 のデータ項目の内容が、定数-3 または一意名-6 のデータ項目の現在の値に設定され直される。この後、条件-1 が再び評価される。条件-1 が真である場合は、PERFORM 文の実行は終了する。そうでなければ、条件-1 が真になるまで、この処理が繰り返される。

PERFORM 文の実行が終了した時点では、一意名-5 のデータ項目には定数-3 または一意名-6 のデータ項目の現在の値が入っている。一意名-2 のデータ項目には、増分または減分によって最後に設定された値が入っている。ただし、PERFORM 文の実行が開始された時点で条件-1 が真であった場合には、一意名-2 のデータ項目には、定数-1 または一意名-3 のデータ項目の現在の値が入っている。

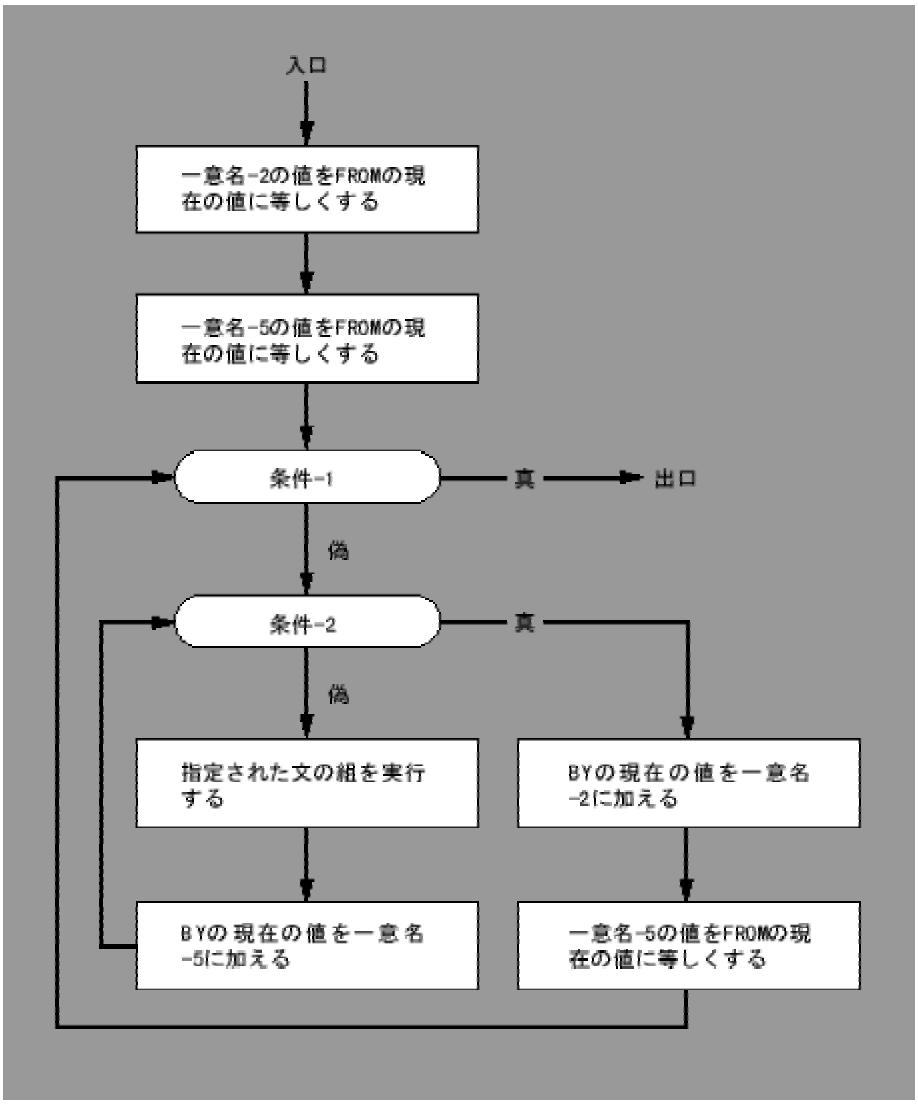


図 15-2: PERFORM 文の機能の流れ図 - VARYING, TEST BEFORE, 2 つの条件

2. (ANS85)TEST AFTER を指定した場合

1 つの一意名のデータ項目の値を変化させる場合、PERFORM 文の実行開始時点で、一意名-2 のデータ項目の内容が、定数-1 または一意名-3 のデータ項目の現在の値に設定される。そして、指定され

た文の組が 1 回実行される。それから、UNTIL 指定の条件-1 が検査される。この結果が偽であれば、一意名-2 のデータ項目の値が指定された値 (定数-2 または一意名-4 のデータ項目の値) だけ増分または減分されて、指定された文の組が再び実行される。この処理が、条件-1 の検査結果が真になるまで繰り返される。条件-1 の検査結果が真になると、制御は PERFORM 文の末尾に移される。図 15-3 を参照。

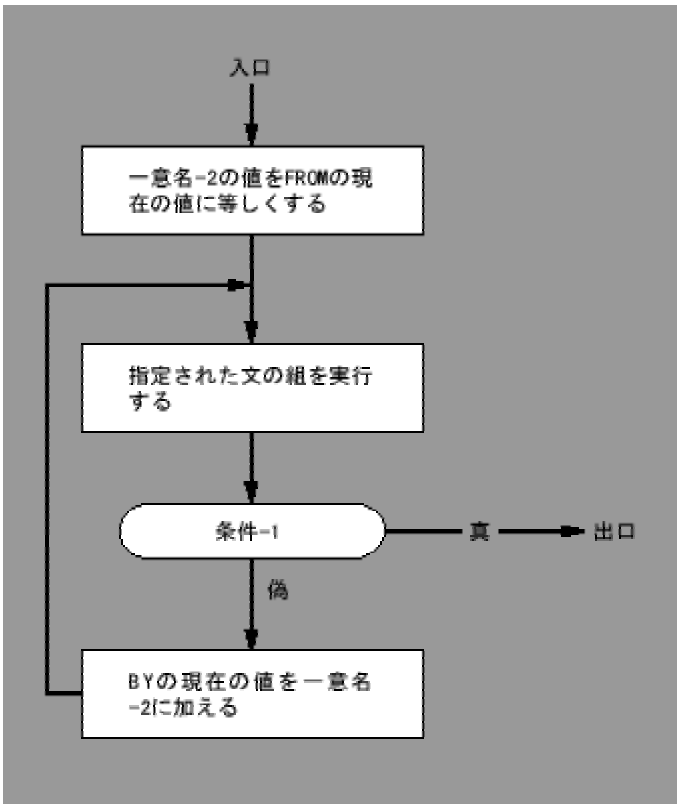


図 15-3: PERFORM 文の機能の流れ図 - VARYING, TEST AFTER, 1 つの条件

2 つの一意名のデータ項目の値を変化させる場合、PERFORM 文の実行開始時点で、一意名-2 のデータ項目の内容が、定数-1 または一意名-3 のデータ項目の現在の値に設定される。また、一意名-5 のデータ項目の内容が、定数-3 または一意名-6 のデータ項目の現在の値に設定される。そして、指定された文の組が 1 回実行される。それから条件-2 が評価される。この結果が偽であれば、一意名-5 のデータ項目の値が、定数-4 または一意名-7 のデータ項目の現在の値だけ増分または減分されて、指定された文の組が再び実行される。この処理が、条件-2 の評価結果が真になるまで繰り返される。条件-2 が真になると、今度は条件-1 が評価される。この結果が偽である場合は、一意名-2 のデータ項目の値が、定数-2 または一意名-4 のデータ項目の現在の値だけ増分または減分され、一意名-5 のデータ項目の内容が、定数-3 または一意名-6 のデータ項目の現在の値に

設定し直される。この後、指定された文の組が再び実行される。この処理が、条件-1 の評価結果が真になるまで繰り返される。条件-1 が真になると、制御は PERFORM 文の末尾に移される。図 15-4 を参照。

PERFORM 文の実行が完了すると、AFTER 指定または VARYING 指定によって変えられた各データ項目には、指定された文の組が最後に実行されたときの内容が入っている。

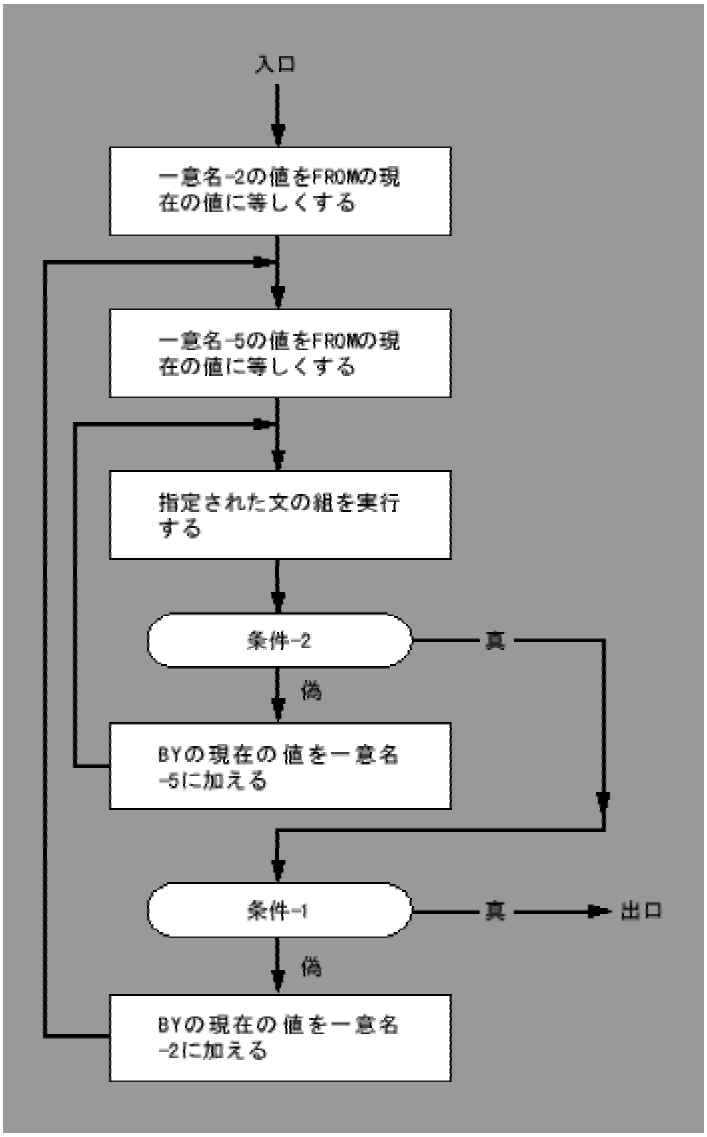


図 15-4: PERFORM 文の機能の流れ図 - VARYING, TEST AFTER, 2 つの条件

PERFORM 文に指定された文の組が実行されている間に、VARYING 変数 (一意名-2 のデータ項目および指標名-1)、BY 変数 (一意名-4 のデータ項目)、AFTER 変数 (一意名-5 のデータ項目および指標名-

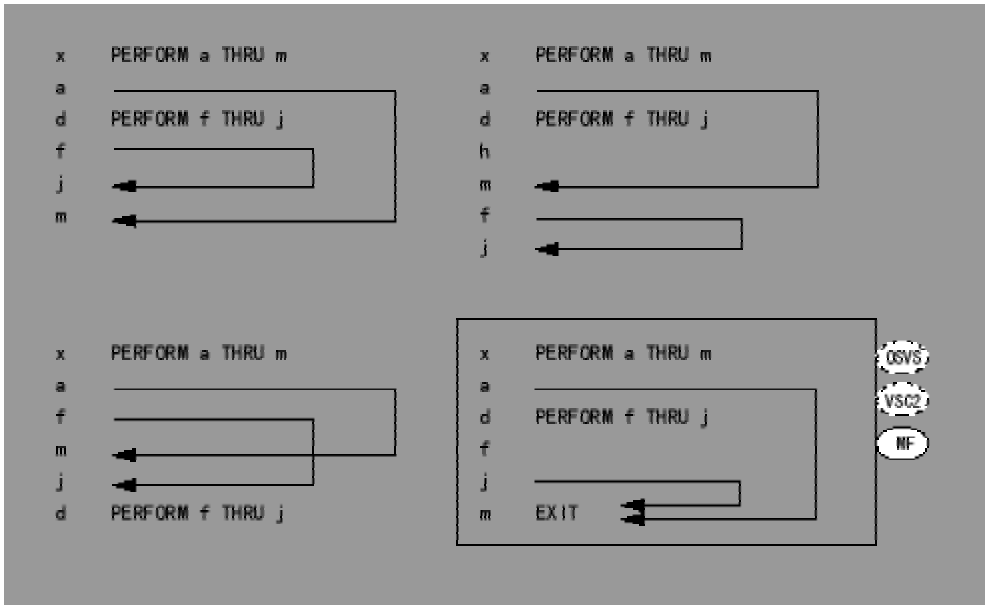
3)、FROM 変数(一意名-3 のデータ項目および指標名-2)の変化が考慮に入れられて、以降の PERFORM 文の実行に影響を及ぼす。

2 つの一意名のデータ項目を変化させる場合、一意名-2 のデータ項目が変化するたびに、一意名-5 のデータ項目は変化の全過程(FROM, BY, UNTIL)を経る。3 つ以上の一意名のデータ項目の内容を変化させる場合も、その仕組みは基本的には 2 つの場合と同じである。ただし、AFTER 指定によって変化させるデータ項目は、その前にある AFTER 指定の作用対象が増分または減分されるたびに、変化の全過程を経る。

9. PERFORM 文の範囲は、論理的には、その開始によって明示的に制御を移されて実行される文に始まり、その PERFORM 文に制御が戻されるまでに実行される、すべての文に及ぶ。したがって、PERFORM 文の範囲の中には、その中に含まれる CALL, GO TO, PERFORM などの文によって制御を移されて実行される文もすべて含まれる。また、PERFORM 文の範囲内の文を実行することによって実行される宣言手続き中の文もすべて含まれる。PERFORM 文の範囲内に含まれる文が、原始要素内で連続して現れる必要はない。
10. 次の場合、EXIT PROGRAM 文を実行したことによって制御が移されて実行される文は、PERFORM 文の範囲内にあるとはみなされない。
    a. PERFORM 文が指定されているのと同じプログラム中に指定された EXIT PROGRAM 文。
    b. PERFORM 文の範囲内にある EXIT PROGRAM 文。
11. ある PERFORM 文の範囲内に別の PERFORM 文が含まれる場合、含まれる方の PERFORM 文に連なる一連の手続きは、最初の PERFORM 文によって参照される一連の手続きに完全に含まれるか、完全にその外にあるかの、どちらかでなければならない。したがって、ある PERFORM 文の範囲内にある PERFORM 文は他の PERFORM 文の出口に制御を渡してはならない。さらに、2 つ以上の PERFORM 文が、出口を共有してはならない。

    (MF)これらの制限は強制しない。PERFORM 文を入れ子にすることも、再帰させる(ある PERFORM 文の手続きの中にその PERFORM 文を含める)ことも許される。現在実行されている PERFORM 文の最も内側の出口だけが認識される。これらの規則は PERFORM-TYPE コンパイラ指令を使用することによって変更できる。

    PERFORM 文の正しい構成の例を、下に示す。



12. 独立区分外の節内にある PERFORM 文の範囲内には、その範囲内から呼ぶ宣言節の他に、下記のどちらか 1 つを含めることができる。
    a. 1 つ以上の非独立区分に完全に含まれる節または段落
    b. 単一の独立区分に完全に含まれる節または段落

    OSVS VSC2 これらの制限は適用しない。

13. 独立区分内にある PERFORM 文の範囲内には、その範囲内から呼ぶ宣言節の他に、下記のどれか 1 つを含めることができる。
    a. 1 つ以上の非独立区分に完全に含まれる節または段落
    b. 該当する PERFORM 文と同じ独立区分に完全に含まれる節または段落

    OSVS VSC2 これらの制限は適用しない。

形式 4

14. 一意名-4 および一意名-7 のデータ項目は、値がゼロであってはならない。
15. VARYING 指定または AFTER 指定の中で指標名を使用し、対応する FROM 指定の中で一意名を使用する場合、この一意名のデータ項目の値は、正でなければならない。

15.2 READ(読み込み)文

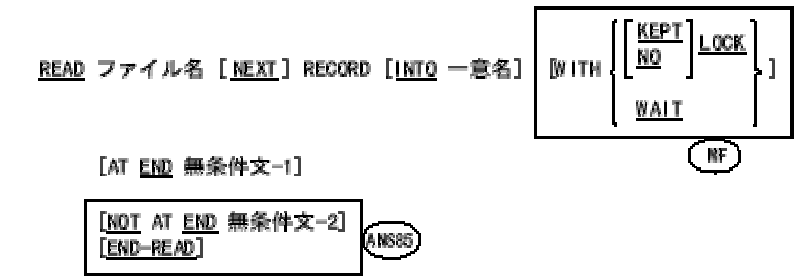
順呼出しの場合、READ (読込み) 文はファイルから次

MF または前の

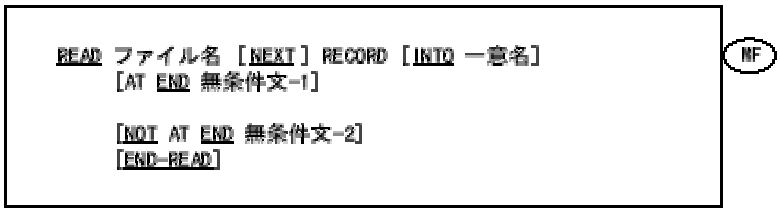
論理レコードを利用できるようにする。乱呼出しの場合、READ 文は大記憶ファイルから指定されたレコードを利用できるようにする。

一般形式

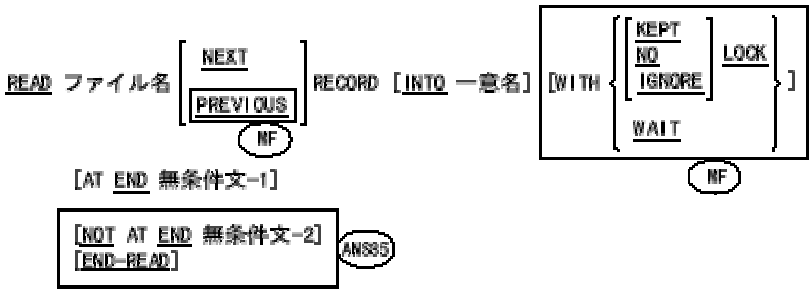
形式 1 (レコード順ファイル)



形式 2 (行順ファイル)



形式 3 (相対ファイルおよび索引ファイル)



形式 4 (相対ファイル)

```
READ ファイル名 RECORD [INTO 一意名] [WITH { [KEPT] [NO] LOCK } ]
                                          { [WAIT]      }
                                                          (NF)
    [INVALID KEY 無条件文-3]
    [NOT INVALID KEY 無条件文-4]
    [END-READ]                                            (ANS85)
```

形式 5 (索引ファイル)

```
READ ファイル名 RECORD [INTO 一意名] [WITH { [KEPT] [NO] LOCK } ]
                                           { [IGNORE]      }
                                           { [WAIT]        }
                                                            (NF)
    [ KEY IS { データ名 }  ]
             { 分割キー名 } (NF)
    [INVALID KEY 無条件文-3]
    [NOT INVALID KEY 無条件文-4]
    [END-READ]                                              (ANS85)
```

指令およびランタイム･スイッチ

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
   - RETRYLOCK - ロックされているレコードの読込みを再試行させる。
2. 下記のランタイム･スイッチによって、この項に記述した意味が影響を受ける可能性がある。
   - **B, B1** - READ NEXT 文を実行中にロックされたレコードが出てくると、それを飛ばすようにレコード･ポインタを更新させる。
   - **N** - 行順レコードを読むときに、制御文字の前の空文字の解釈を制御する。
   - **T** - 行順レコードを書くときに、タブ文字の解釈を制御する。

構文規則

すべての形式 (すべてのファイル)

1. 入力ファイルの論理レコードが可変長のとき、INTO 指定を使用してはならない。

   (ANS85) この制限は削除された。

   一意名のデータの記憶領域とファイル名のレコードの記憶領域とが、同じ記憶領域を指してはならない。

2. OSVS VSC2 MF 一意名は、浮動小数点数データ項目でもよい。
3. ファイル名に適用できる USE 手続きを設定していない場合には、INVALID KEY 指定または AT END 指定を書く。

   OSVS VSC2 MF この規則は強制ではない。

形式 1、3、4 および 5 (レコード順ファイル、相対ファイル、索引ファイル)

4. MF WITH LOCK を指定できるのは、共有ファイル中で単一のレコードを手入力でロックする場合だけである。
5. MF WITH NO LOCK を指定できるのは、共有ファイル中でレコードをマニュアルまたは自動的にロックする場合だけである。

形式 3 (相対ファイルおよび索引ファイル)

6. 動的呼出し法でレコードを順検索する場合は、NEXT

   MF または PREVIOUS

   を指定する。

形式 3、4 および 5 (相対ファイルおよび索引ファイル)

7. MF WITH KEPT LOCK を指定できるのは、共有ファイル中で複数のレコードを手入力でロックする場合だけである。

形式 4 および 5 (相対ファイルおよび索引ファイル)

8. 乱呼出し法または動的呼出し法でファイルのレコードを乱検索する場合は、形式 4 または 5 を用いる。

形式 5 (索引ファイル)

9. データ名

   ISO/JIS MF または分割キー名

   は、ファイル名のファイルのレコードキーとして指定したデータ項目の名前とする。

   OSVS MF データ名はまた、ファイル名と対応するレコードキーとして指定した、データ項目の再定義となり、同じ長さをもつデータ名を作ることもできる。

   OSVS 再定義は、レコードキーと異なる長さをもつことがある。

10. データ名

    ISO/JIS MF または分割キー名

は、修飾してもよい。

11. (MF)分割キー名は、ファイル名のファイルのレコード･キーとして指定したいくつかのデータ項目を結合したものである。

一般規則

すべての形式 (すべてのファイル)

1. READ 文を実行するときには、対象のファイルを入力モードまたは 入出力モードで開いておく。(*手続き部 - MERGE - OPEN* の章の *OPEN(開く)文* 節を参照。)
2. READ 文を実行すると、ファイル名に対応する FILE STATUS データ項目が存在すれば、その値が更新される。(*手続き部*の章の*入出力状態*節を参照。)
3. 1 つのファイルの論理レコードが複数のレコード記述によって定義されている場合、それらのレコードは自動的に同じ記憶領域を共有する。これはレコード領域を暗黙的に再定義することに相当する。現在のレコード領域の範囲を超えてデータ項目が存在する場合、READ 文の実行が終了したときにその内容はどうなっているかわからない。
4. INTO を指定すると、読み込まれたレコードは、レコード領域から一意名に指定した領域へ転記される。その際、CORRESPONDING 指定のない MOVE 文に関する規則が適用される。READ 文の実行が不成功であったときは、この暗黙的な転記は行われない。一意名に添字付けまたは指標付けがなされている場合、レコードが読み込まれてからその一意名のデータ項目に転記される直前に評価される。
5. INTO を指定すると、読み込まれたレコードは、入力レコード領域でも一意名のデータ領域でも利用可能となる。
6. ファイル位置指示子が次の論理レコードが存在しないことを示しているか、または不定入力ファイルが存在しない場合、下記の処理が順に行われる。
   a. ファイル位置指示子の内容に応じた値が、ファイル名-1 に対応する入出力状態に設定される。(*手続き部*の章の*入出力状態*節を参照。)
   b. AT END 指定をしてあると、制御は無条件文-1 に移される。この場合、ファイル名-1 のファイル結合子に対応する USE AFTER EXCEPTION 手続きが設定してあっても実行されない。その後、無条件文-1 の中の各文に関する規則に従って、処理は続行される。その中で、制御の明示をもたらす分岐文または条件文が実行された場合は、その文に関する規則に従って制御が移される。そうでなければ、無条件文-1 の実行が終了した時点で、制御は READ 文の末尾に移される。この場合、NOT AT END 指定があっても、無視される。
   c. AT END 指定をしていない場合、ファイル名-1 に対応する USE AFTER STANDARD EXCEPTION 手続きを設定する必要がある。この場合、その USE 手続きが実行される。その実行が終了すると、READ 文の末尾の次の実行文に制御が戻される。

      (ANS85)AT END 条件が発生した場合、READ 文の実行は不成功になる。

7. READ 文の実行が不成功に終わった後では、対応するレコード領域の内容はどうなっているかわからない。また、ファイル位置指示子は、次のレコードが存在しないことを示すように設定される。

形式 1 (レコード順ファイル)

8. READ 文によって利用可能となるレコードは、下記のように決定される。
    a. (ANS85)OPEN 文の実行によってファイル位置指示子の位置が設定されている場合は、ファイル位置指示子が指すレコードが利用できるようになる。
    b. (ANS85)前の READ 文の実行によってファイル位置指示子の位置が設定されている場合は、ファイル位置指示子が次のレコードを指すように >更新され、そのレコードが利用できるようになる。

    (MF)しかし、前の READ 文によってレコードがロックされている場合は、ファイル位置指示子は変更されない。そのファイル位置指示子が指すレコードが利用できるようになる。

9. READ 文の実行中にリールまたはユニットの終わりが検出されたが、まだファイルの終わりには達していない場合、下記の手続きが実行される。
    a. 標準終了リール/ユニット･ラベル手続き
    b. リール/ユニットの交換
    c. 標準開始了リール/ユニット･ラベル手続き
    d. 新しいリール/ユニットの最初のデータ･レコードの読込み
10. OPTIONAL 指定をしたファイルを開こうとしたときにそのファイルが存在しないと、そのファイルに対して最初に READ 文が実行されたときに AT END 条件が発生し、READ 文の実行は不成功に終わる。標準のファイル終了手続きは実行されない。(***環境部***の章の *ファイル管理段落*節、***手続き部 - MERGE - OPEN*** の章の *OPEN(開く)文*節、および***手続き部 - SEARCH - XML PARSE*** の章の *USE(使用)文*節を参照。) その後、 実行は、一般規則 13 に記してあるように進む。
11. (MF)INPUT 用に開かれたファイルに関しては、READ 文、READ WITH LOCK 文、および READ WITH KEPT LOCK 文のいずれもレコード･ロックを得ることはない。
12. (MF)レコードロック方式が AUTOMATIC または MANUAL である 1 つの順ファイルを複数の実行単位から EXTEND を指定して開くと、そのファイルが共有される。ただし、そのファイルの後ろに追加されるレコードの順序は保証されない。
13. (MF)I-O 用に開かれたファイルに関しては、以下のようになる。
    a. ファイルに LOCK MODE IS AUTOMATIC が指定してあると、READ 文に WITH NO LOCK を指定しないかぎり読み込まれた各レコードはロックされる。
    b. ファイルに LOCK MODE IS MANUAL が指定してあると、READ 文ではレコードロックを得ることはできない。レコードロックを得るためには、READ WITH LOCK 文を使用する。READ 文に WITH NO LOCK を指定しても、注記になる。
    c. レコードロックがいつ解除されるかは、単一のレコードをロックしたか複数のレコードをロックしたかによって異なる。(***環境部***の章のの *ファイル管理段落*節を参照。)
14. (MF)ある実行単位からは I-O または INPUT 用に開き、別の実行単位からは EXTEND 用に開いたファイルに対して READ 文を実行した際に、ファイル終了状態が発生した場合、その READ 文を実行しようとした実行単位側でそのファイルを閉じなければならない。そのファイルの状態は終了のまま残るため、その実行単位からはファイルの後ろに追加されたレコードを呼び出すことはできない。

形式 1、2 および 3 (順ファイル、相対ファイル、および索引ファイル)

15. READ 文の実行中に AT END 条件が発生しなかった場合、AT END 指定をしてあっても無視されて、以下の処理が行われる。
    a. ファイル位置指示子が設定されて、ファイル名-1 に対応する入出力状態が更新される。
    b. ファイル終了条件以外の例外条件が発生していると、ファイル名-1 に適用できる USE AFTER EXCEPTION 手続きが指定してあればそれが実行されて、その後で USE 文の規則に従って制御が移される。(***手続き部 - SEARCH - XML PARSE*** の章の *USE(使用)文*節を参照。)
    c. 例外条件が何も発生していなければ、読込み対象レコードがレコード領域中で利用できるようにされる。また、INTO を指定してあれば、暗黙の転記処理が行われる。その後で、制御は READ 文の末尾に移される。ただし、NOT AT END 指定をしてあれば、制御は無条件文-2 に移される。この場合、無条件文-2 の中の個々の文に関する規則に従って、以降の処理が続行される。制御の明示移行をもたらす分岐文または条件文が実行された場合は、その文に関する規則に従って制御が移される。制御の明示移行をもたらす分岐文または条件文が実行されなかった場合は、無条件文-2 の実行が終わると、制御は READ 文の末尾に移される。
16. NEXT 指定をした READ 文を実行したときに、対象ファイル中に次の論理レコードが存在しないと、AT END 条件が発生し READ 文の実行は不成功に終わったものとみなされる。(***手続き部***の章の*入出力状態*節を参照。)
17. (ANS85)順ファイルまたは

    順呼出し法で呼び出すファイルに関しては、NEXT 指定は書いても書かなくてもよい。READ 文の実行には影響を及ぼさない。.

形式 3 (相対ファイルおよび索引ファイル)

18. (MF)読み込んだレコードがロックされていると、ファイル位置指示子はそのレコードを指すように設定される。以降、READ NEXT 文または READ PREVIOUS 文は同じレコードを再び取り出す。

    (MF)NOT AT END 指定は、READ 文の実行が正常に終了した場合にだけ実行される。

19. (MF)PREVIOUS 指定をした READ 文を実行したときに、対象ファイル中に前の論理レコードが存在しないと、AT END 条件が発生し READ 文の実行は不成功に終わったものとみなされる。
20. AT END 条件が検出された場合、前の論理レコードが存在しないため、

    (MF)そのファイルに対して次に形式 3 の READ 文を実行するならば、NEXT を指定する。そうでなければ、

    下記の処理を続ける。

    a. 該当するファイルに対して CLOSE 文を実行し、次いで OPEN 文を実行する。いずれも正常に終了しなければならない。
    b. そのファイルに対して START 文を実行する。正常に終了しなければならない。

c. そのファイルに対して、形式 4 (相対ファイル) または形式 5 (索引ファイル) の READ 文を実行する。

21. 動的呼出し法を指定したファイルに関して、NEXT 指定をした READ 文を実行すると、

(MF)一般規則 8 に記述したように

次の論理レコードが取り出される。

22. READ 文によって利用可能となるレコードは、下記の規則に従って決定される。

a. (MF)OPEN 文の実行によってファイル位置指示子が設定されているところへ PREVIOUS を指定した READ 文を実行すると、AT END 条件が発生する。

START 文または OPEN 文の実行によってファイル位置指示子が設定されていて、ファイル位置指示子によって指されるレコードの呼出しができる場合は、そのレコードが利用可能にされる。相対ファイルのレコードを削除したり索引ファイルの副レコードキーを変更したりすることによって、レコードが呼び出しできる状態になった場合は、参照キーによって確立されている呼出し経路中の次のレコードを指すように、ファイル位置指示子が更新される。

(MF)ただし、PREVIOUS 指定をしていると、

前のレコードを指すように、ファイル位置指示子が更新される。そして、ファイル位置指示子によって指されるレコードが利用可能にされる。

b. 前の READ 文の実行によってファイル位置指示子が設定された場合、ファイル位置指示子はファイル中に存在する次のレコードを指すように更新される。

(MF)ただし、PREVIOUS 指定をしていると、前のレコードを指すように、ファイル位置指示子が更新される。

(MF)前の READ 文によってレコードがロックされた場合は、ファイル位置指示子は変更されない。そして、ファイル位置指示子によって指されるレコードが利用可能にされる。

形式 1、3、4 および 5 (順ファイル、相対ファイル、索引ファイル)

23. (MF)単一のレコードを手入力でロックするように指定したファイルを入出力両用に開くと、READ 文に WITH LOCK 指定をしたときにだけ、実行単位はレコードロックを取得する。単なる READ 文では、レコードロックを得ることはできない。ロックされているレコードを読み飛ばすためには、START 文を用いてファイル位置指示子を更新する。ただし、キーの重複を認めている副レコードキーに関しては、この方法を採ることはできない。

24. (MF)複数のレコードを手入力でロックするように指定したファイルを入出力両用に開くと、READ 文に WITH KEPT LOCK 指定をしたときにだけ、実行単位はレコードロックを取得する。単なる READ 文では、レコードロックを得ることはできない。ロックされているレコードを読み飛ばすためには、START 文を用いてファイル位置指示子を更新

する。ただし、キーの重複を認めている副レコードキーに関しては、この方法を採ることはできない。

25. (MF)WITH WAIT 指定をすると、必要があれば待ってでも、レコードロックが必ず取得される。

形式 3 (相対ファイル)

26. RELATIVE KEY を指定したファイルに対して READ 文を実行すると、利用可能にされたレコードの相対レコード番号が、RELATIVE KEY データ項目に収められる。

形式 4 (相対ファイル)

27. READ 文を実行すると、ファイルの RELATIVE KEY に指定したデータ項目中に保持されている相対レコード番号がファイル位置指示子に設定され、そのレコードが利用可能にされる。ファイル中にその相対レコード番号のレコードが存在しない場合は、無効キー条件が発生し、READ 文の実行は不成功に終わる。(***手続き部***の章の*無効キー条件*節を参照。)

形式 3 および 5 (相対ファイルおよび索引ファイル)

28. (MF)IGNORE LOCK 指定をすると、READ 文はレコードがロックされていないかのように実行される。

形式 3 (索引ファイル)

29. 索引ファイルを順呼び出しする場合、検索のキーとしている副レコードキーに値が同じものがあると、これらは WRITE 文または REWRITE 文によって書き出された順に利用可能にされる。

形式 5 (索引ファイル)

30. KEY 指定をしないと、主レコードキーが検索キーとして使用される。動的呼出しを指定すると、後に形式 3 の READ 文を実行するときにも、この検索キーが使用される。
31. KEY 指定をすると、データ名-1

    (COBOL)(MF)または分割キー名

    がその参照キーとして使用される。動的呼出しを指定すると、後に形式 3 の READ 文を実行するときにも、この検索キーが使用される。ただし、別の参照キーを指定すると、今度はそれが使用されるようになる。

32. 形式 5 の READ 文を実行すると、参照キーの値がファイル中のレコードの対応するデータ項目に格納されている値と順次比較されて、両者の値が最初に一致するものが見つけ出される。ファイル位置指示子がそのレコードを指すように設定されて、そのレコードが利用可能とされる。両者の値が一致するレコードが見つからないと、無効キー条件が発生し、READ 文の実行は不成功に終わる。(***手続き部***の章の*無効キー条件*節を参照。)

15.3 RELEASE(引き渡し)文

RELEASE (引き渡し) 文は、整列処理の最初の段階にレコードを引き渡す。

一般形式

```
RELEASE レコード名 [ FROM 一意名 ]
```

構文規則

1. RELEASE 文を使用できるのは、整列併合ファイル記述項にレコード名を記述したファイルを対象とする SORT 文に関連する、入力手続きの範囲内だけである。(*手続き部 - SEARCH - XML PARSE* の章の *SORT(整列)文*節を参照。)
2. レコード名は、関連する整列併合ファイル記述項内の論理レコードの名前とする。このレコード名は修飾してもよい。
3. (ANS85)一意名が関数一意名である場合は、英数字関数を参照しなければならない。一意名が関数一意名でない場合は、

   レコード名と一意名が同じ記憶領域を指してはならない。

4. (OSVS)(VSC2)(MF)レコード名は、浮動小数点数項目として定義してあってもよい。
5. (OSVS)(VSC2)(MF)一意名は、浮動小数点数であってもよい。

一般規則

1. RELEASE 文を実行すると、レコード名が 1 件ずつ、整列処理の最初の段階に引き渡される。
2. FROM 指定をすると、一意名のデータ領域の内容がレコード名に転記されて、それからレコード名の内容が整列ファイルに引き渡される。この転記は、CORRESPONDING 指定のない MOVE 文に関する規則に従って行われる。レコード領域内のデータは利用できなくなるが、一意名のデータ領域のデータは引続き利用できる。
3. RELEASE 文の実行が終了した後は、レコード領域中の論理レコードは利用できなくなる。ただし、対応する整列併合ファイルを SAME RECORD AREA 句に指定しておけば、利用できる。この場合、その論理レコードは、 その SAME RECORD AREA 句に指定した 他のファイルのレコードとしても利用できる 。入力手続きから制御が戻されると、整列用ファイルには RELEASE 文の実行によって引き渡されたすべてのレコードが格納されている。

15.4 RETURN(引き取り)文

RETURN (引き取り) 文は、整列処理の最後の段階から整列済みのレコードを引き取るか、または併合処理中に併合済みのレコードを引き取る。

一般形式

```
RETURN ファイル名 RECORD [INTO 一意名-1]
    AT END 無条件文-1

    [NOT AT END  無条件文-2]  (ANS85)
    [END-RETURN]
```

構文規則

1. ファイル名は、データ部の中の整列併合ファイル記述項に記述しておく。
2. RETURN 文を使用できるのは、ファイル名を対象とする SORT 文または MERGE 文に関連する、出力手続きの範囲内だけである。
3. 入力ファイルのレコード記述に大きさの異なる論理レコードが含まれているときは、INTO 指定をしてはならない。一意名のデータ領域とファイル名のレコード領域が、同じ記憶領域を指していてはならない。
4. (OSVS)(VSC2)(MF)一意名-1 は、浮動小数点数項目であってもよい。

一般規則

1. 1 つのファイルに論理レコードのレコード記述がいくつもある場合、これらのレコードは自動的に同じ記憶領域を共有する。これは、暗黙的にレコードを再定義することと同じである。現在のデータ･レコードの範囲を超えるデータ項目は、RETURN 文の実行が終了した時点でその内容は、どうなっているかわからない。
2. RETURN 文の実行が終了すると、ファイル名のファイル中に存在する次のレコードが、そのレコード領域において利用可能とされる。このレコードの順序は、SORT 文または MERGE 文中に指定されているキーによって決定される。ファイル名のファイル中に次の論理レコードが存在しないとファイル終了条件が発生し、AT END 指定の無条件文-1 に制御が移される。この場合、無条件文-1 の中の各文に関する規則に従って、処理は続行される。その中で、制御を明示的に移行する手続きの分岐または条件文が実行された場合は、その文に関する規則に従って制御が移される。 そうでなければ、無条件文-1 の実行が終了した時点で、制御は RETURN 文の末尾に移される。

   (ANS85)この場合、NOT AT END 指定があっても、無視される。

   ファイル終了条件が発生すると、RETURN 文の実行は不成功に終わり、ファイル名のレコード領域の内容はどうなるかわからない。AT END 指定中の無条件文-1 の実行が終了した後では、現在の出力手続きの一環としてそれ以上 RETURN 文を実行することはできない。

3. (ANS85)RETURN 文の実行中にファイル終了条件が発生しなかった場合、レコードが利用可能にされる。次いで INTO 指定に基づく暗黙の転記が行われた後で、無条件文-2 を指定してあれば、制御はそちらに移される。無条件文-2 を指定してなければ、制御は RETURN 文の末尾に移される。

4. (ANS85) END-RETURN 指定は、RETURN 文の範囲を区切る。
5. 下記のどちらかの場合は、RETURN 文中に INTO 指定を含めることができる。
   - 整列併合ファイル記述のもとに、レコードが 1 件だけ記述されている。
   - ファイル名のファイルのすべてのレコードおよび一意名のデータ項目が、集団項目または基本英数字項目である。
6. INTO 指定を伴う RETURN 文を実行した結果は、下記の規則を順に適用することと同じである。
   a. INTO 指定を伴わない同じ内容の RETURN 文を実行する。
   b. 現在のレコード領域の内容を、一意名のデータ領域に転記する。転記の仕方は CORRESPONDING 指定のない MOVE 文の規則に従う。現在のレコードの大きさは、RECORD 句の指定によって決まる。ファイル記述項に RECORD IS VARYING 句が指定してある場合は、暗黙的に集団項目転記が行われるが、この転記は不成功に終わる。一意名に添字付けがされていれば、レコードが読み出された後、一意名のデータ項目に転記される直前に評価される。レコード領域および一意名-1 のデータ項目の、両方のデータが利用できる。

15.5 REWRITE(書き換え)文

REWRITE (書き換え) 文は、ディスク･ファイル中に存在するレコードを論理的に置き換える。

一般形式

形式 1 (行順ファイル)

```
REWRITE レコード名 [FROM 一意名]                    (NF)

    [END-REWRITE]
```

形式 2 (レコード順ファイル)

```
REWRITE レコード名 [FROM 一意名]

    [END-REWRITE]
```

形式 3 (相対ファイルおよび索引ファイル)

```
REWRITE レコード名 [FROM 一意名]
    [INVALID KEY 無条件文-1]

    [NOT INVALID KEY 無条件文-2]
    [END-REWRITE]                                   (ANS85)
```

指令およびランタイム･スイッチ

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
   - WRITE-LOCK - 複数のレコード･ロックを使用するときに、REWRITE 文用のレコード･ロックを取得する。
2. 下記のランタイム･スイッチによって、この項に記述した意味が影響を受ける可能性がある。
   - N - 行順レコードを書くときに、制御文字の前の空文字の挿入を制御する。
   - T - 行順レコードを書くときに、タブ文字の挿入を制御する。

構文規則

すべての形式 (すべてのファイル)

1. ANS85 一意名が関数一意名である場合、英数字関数を参照しなければならない。一意名が関数一意名でない場合、

   レコード名と一意名が、同じ記憶領域を指してはならない。

2. レコード名は、データ部のファイル節の中の論理レコードの名前である。レコード名は修飾してもよい。
3. OSVS VSC2 MF レコード名は、浮動小数点数項目または 2 バイト文字項目であってもよい。
4. OSVS VSC2 MF 一意名は、USAGE DISPLAY-1 (DBCS) 項目または浮動小数点数項目であってもよい。

形式 3 (相対ファイルおよび索引ファイル)

5. 順呼出し法でファイルを処理している場合には、REWRITE 文には INVALID KEY を指定できない。

   OSVS VSC2 ただし、順呼出し法で索引ファイルを処理している場合には、REWRITE 文に INVALID KEY を指定してもよい。

6. 乱呼出し法または動的呼出し法でファイルを処理しているが、そのファイルに適用できる USE 手続きを指定していない場合には、REWRITE 文に INVALID KEY を指定する。

一般規則

すべての形式 (すべてのファイル)

1. 該当するレコード名のレコードが含まれるファイルは、ディスク･ファイルとする。この文を実行する時点では、入出力モードで開いておく。(*手続き部 - MERGE - OPEN* の章の *OPEN(開く)文*節を参照。)

2. ファイルを順呼出し法で処理している場合、REWRITE 文を実行する前に実行した直前の入出力文は、正常に終了した READ 文でなければならない。オペレーティングシステムは、READ 文によって呼び出されたレコードを論理的に置き換える。
3. FROM 指定をした REWRITE 文を実行することは、

   MOVE 一意名 TO レコード名

   を実行し、それから FROM 指定をしない REWRITE 文を実行することと同じである。暗黙の MOVE 文が実行される前のレコード領域の内容は、REWRITE 文の実行に影響を及ぼさない。

4. ファイル位置指示子は、REWRITE 文の実行の影響を受けない。
5. REWRITE 文を実行すると、更新対象のファイルに対応する FILE STATUS データ項目を指定してあれば、その値が更新される。(***手続き部***の章の*入出力状態*節を参照。)
6. (ANS85)END-REWRITE 指定は、REWRITE 文の範囲を区切る。
7. (MF)書き換えようとするレコードが他の実行単位によってロックされていると、REWRITE 文の実行は不成功に終わる。

形式 1 および 2 (順ファイル)

8. レコード名のレコードの文字数は、新しい書き換え用のレコードの文字数と等しくする。

注: 圧縮された順ファイルには REWRITE 文を使用しないように勧める。その理由は、圧縮された新しいレコードの長さが圧縮された古いレコードの長さと同じでないと、REWRITE 処理は正常に終了しないからである。

---

9. REWRITE 文の実行が正常に終了すると、レコード領域中の論理レコードは解放され、利用できなくなる。ただし、対応するファイルを SAME RECORD AREA 句に指定しておけば、利用できる。この場合、その論理レコードは 、そのファイルの論理レコードとしてだけではなく、その SAME RECORD AREA 句に指定した 他のファイルのレコードとしても、利用できる。

形式 1 (行順ファイル)

10. (MF)書き込もうとするレコードの圧縮された長さが、元の圧縮されたレコードの長さ以下のとき、REWRITE 文を使用して正常に終了させることができる。ここで、圧縮された長さとは、後行の空白を削除しタブを圧縮し空文字を挿入したものを指す。レコードを読み込むと、空文字とタブ文字が拡張されて、レコードの長さが長くなる。

形式 3 (相対ファイルおよび索引ファイル)

11. REWRITE 文の実行が正常に終了すると、レコード領域中の論理レコードは解放され、利用できなくなる。ただし、対応するファイルを SAME RECORD AREA 句に指定しておけば、利用できる。この場合、その論理レコードは、そのレコード名を含むファイルの論理レコードとしてだけではなく、その SAME RECORD AREA 句に指定した 他のファイルのレコードとしても利用できる。

形式 3 (相対ファイル)

12. 乱呼出し法また動的呼出し法で処理しているファイルに関しては、オペレーティングシステムは、そのファイルの RELATIVE KEY データ項目の内容によって指定されるレコードを論理的に置き換える。ファイル中に該当するレコードが存在しないと、無効キー条件が発生する(*手続き部*の章の*無効キー条件*節を参照。) この場合、更新処理は行われず、レコード領域中のデータは元のまま残る。

形式 3 (索引ファイル)

13. 順呼出し法で処理しているファイルに関しては、書き換える対象のレコードは主レコードキーの値によって指定する。REWRITE 文を実行するとき、置き換えようとするレコードの主レコードキー・データ項目中の値は、ファイルから最後に読み込んだレコードの主レコードキーの値と等しくなければならない。
14. 乱呼出し法または動的呼出し法で処理しているファイルに関しては、書き換える対象のレコードは主レコードキー・データ項目の値によって指定する。
15. これから書き換えられるレコードの副レコードキー・データ項目の内容は、置き換える側のものと違っていても構わない。オペレーティングシステムは、更新後にどのレコードキーを使用してもレコードを呼び出すことができるように、REWRITE 文の実行中にレコードキー・データ項目の内容を使用する。
16. 下記のどれかの場合に、無効キー条件が発生する。
    a. 順呼出し方式でファイルを処理しているときに、新しく置き換える側のレコードの主レコードキー・データ項目の値が、ファイルから最後に読み込んだレコードの主レコードキーの値と等しくない。
    b. 主レコードキー・データ項目の値と主レコードキーの値が一致するレコードが、ファイル中に存在しない。
    c. 副レコードキー・データ項目の値と副レコードキーの値が一致するレコードが、DUPLICATES 句を指定していないファイルに関して、既にファイル中に存在する。

    上記の場合、更新処理は行われず、レコード領域中のデータは元のまま残る。(*手続き部*の章の*無効キー条件*節を参照。)

15.6 ROLLBACK(更新取消し)文

MF

ROLLBACK (更新取消し) 文は、実行単位によって押さえられているすべてのファイル中のすべてのレコードロックを解除する。

一般形式

ROLLBLACK

一般規則

1. ROLLBACK 文を実行すると、実行単位によって押さえられているすべてのファイル中のすべてのレコードロックが解除される。
2. 排他ファイル中のファイルロックは、ROLLBACK 文の影響を受けない。
3. 使用している COBOL システムにおいて、SELECT 文の WITH...ROLLBACK 句が単なる注記ではなく実際に機能する場合、ROLLBACK 文は下記の処理を行う。
    a. 現在のトランザクションを終了させる。
    b. 現在のトランザクションの処理中に行われたすべての変更を取り消す。

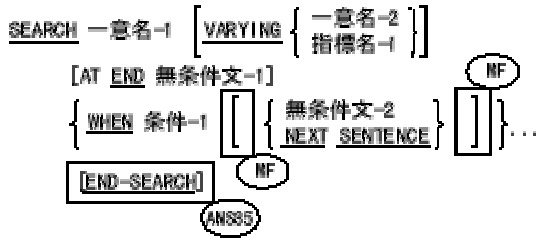
第 16 章 : 手続き部 - SEARCH - XML PARSE

16.1 SEARCH(表引き)文

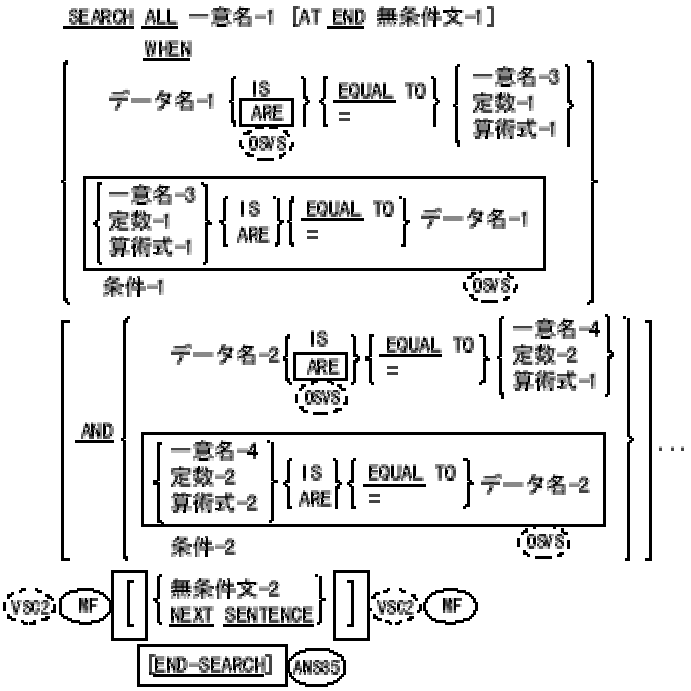
SEARCH (表引き) 文は、表を検索して、指定された条件を満足する表要素を探し出し、対応する指標名がその表要素を指すようにするために使用する。

一般形式

形式 1



形式 2



比較文字の"=" は必須であるが、他の記号と混同することを避けるために、下線を引いていないことに注意。

構文規則

すべての形式 (すべてのファイル)

1. 一意名-1 には添字または指標を付けてはならない。このデータ記述には、 OCCURS 句および INDEXED BY BY 句が含まれていなければならない。形式 2 では、一意名-1 のデータ記述の OCCURS 句内に、KEY IS 指定も含まれていなければならない。
2. 一意名-2 を指定する場合、指標データ項目 (USAGE IS INDEX) または想定される小数点の右に何もない整数データ項目とする。
3. ANS85 END-SEARCH 指定を書いた場合、NEXT SENTENCE 指定を書いてはならない。

   VSC2 MF END-SEARCH 指定と NEXT SENTENCE 指定を同時に書いてもよい。NEXT SENTENCE 指定が実行されると、制御は END-SEARCH 指定の後ろに続く次の文に移されるのではなく、最も近い終止符 (.) の後ろの文に移される。

4. MF 無条件文-1 および無条件文-2 の代わりに、条件文を使用してもよい。

形式 1

5. 条件文-1、条件文-2 などには、***手続き部***の章の*条件式*節で説明した任意の条件を書くことができる。
6. OSVS VSC2 MF 一意名-1 (つまり表引きの対象とする表要素) は、内部浮動小数点数項目または外部浮動小数点数項目であってもよい。

形式 2

7. 使用する条件名は、すべて単一の値をもつものとして定義しておく。条件名に関連するデータ名は、一意名-1 の KEY 句の中に記述しておく。データ名-1 およびデータ名-2 は、それぞれ修飾できる。データ名-1 およびデータ名-2 は、それぞれ他の必要な指標や定数とともに、一意名-1 用の最初の指標名によって指標付けしておき、一意名-1 の KEY 句の中に記述しておく。一意名-3、一意名-4、算術式-1 または算術式-2 の中で使用する一意名は、一意名-1 の KEY 句の中に記述してあってはならず、一意名-1 用の最初の指標名によって指標付けしてあってもならない。

   一意名-1 の KEY 句の中のデータ名を使用するとき、または一意名-1 の KEY 句の中のデータ名に関連する条件名を使用するとき、一意名-1 の KEY 句の中のそのデータ名の前にあるすべてのデータ名またはそれらに関連する条件名を使用する。

8. OSVS VSC2 MF 一意名-1 (つまり表引きの対象とする表要素) は、浮動小数点数であってはならない。
9. OSVS VSC2 MF キー・データ項目であるデータ名-1 およびデータ名-2 は浮動小数点数であってならない。しかし、キー・データ項目と比較する対象の一意名-3、一意名-4、定数-1、定数-2 は、浮動小数点数であってもよい。

一般規則

すべての形式

1. SEARCH 文の範囲は、下記のどれかによって区切られる。
   - (ANS85)同じ入れ子レベルにある END-SEARCH 指定
   - 区切り文字の終止符
   - 前の IF 文に対応する ELSE 指定

     (ANS85)または END-IF 指定

2. 実行された無条件文-1 または無条件文-2 に GO TO 文が含まれていなければ、制御は次の実行完結文に移される。
3. 2 次元または 3 次元の表を作成するための OCCURS 句が含まれるデータ項目の下位に一意名-1 が属する場合、OCCURS 句の INDEXED BY 句を通じて、表の各次元ごとに指標名を付ける。SEARCH 文が実行されると、一意名-1 用の指標名 (および指定されていれば一意名-2 または指標名-1) だけが、値を変えられる。2 次元または 3 次元の表の全体を表引きするためには、SEARCH 文を数回実行する必要がある。指標の値を調整する必要がある場合には、SEARCH 文を実行する前に、SET 文を実行しておく。

形式 1

4. 形式 1 の SEARCH 文を用いると、逐次表引き処理が行われる。このときの指標の値が、その開始点となる。
   a. SEARCH 文の実行が開始される時点で、一意名-1 用の指標データ項目の値が出現番号の最大値よりも大きいと、その SEARCH 文の実行は直ちに停止される。一意名-1 の出現番号の最大値の詳細については、OCCURS 句に説明してある。(*データ部 - ファイルおよびデータ記述*の章の *OCCURS(反復)句*節を参照。) このとき、AT END 指定が書いてあれば、無条件文-1 が実行される。AT END 指定が書いてなければ、制御は次の実行完結文に移される。
   b. SEARCH 文の実行が開始される時点で、一意名-1 用の指標データ項目の値が出現番号の最大値を超えないと、その SEARCH 文の処理が進められる。(一意名-1 の出現番号の最大値の詳細については、*データ部 - ファイルおよびデータ記述*の章の *OCCURS(反復)句*節で説明。) まず、条件が書かれている順に評価される。この際、指定されていれば、検査の対象となる要素の出現番号を判定するために、指標データ項目の値が使用される。条件が満足されていない場合は、指標データ項目の値が増やされて、次の要素が参照できるようにされる。この処理が順次繰り返される。一意名-1 用の指標データ項目の値が出現番号の最大値を超えると、上記の一般規則 1a に示すとおり、その表引き処理は終了する。この間にどこかの時点で条件が満足されると、表引き処理は直ちに停止され、その条件に付随する無条件文が実行される。このとき、その指標データ項目には、条件を満足させた要素の出現番号が記録されている。

5. VARYING 指定を書かなかった場合、表引き処理に使用される指標名は、一意名-1 の INDEXED BY 句の中の最初の (または唯一の) 指標名だけである。一意名-1 用のその他の指標名の指標データ項目の内容は変わらない。
6. VARYING 指標名-1 指定を書いた場合、一意名-1 の INDEXED BY 句の中に指標名-1 が書いてあれば、その指標名がこの表引き処理に使用される。そうでない場合、または VARYING 一意名-2 指定を書いた場合、一意名-1 の INDEXED BY 句の中の最初の (または唯一の) 指標名が表引きに使用される。この場合、さらに下記の処理も行われる。
    a. VARYING 指定で指標名-1 を書いた場合、この指標名-1 が別の表の INDEXED BY 指定に書いてあると、指標名-1 の指標データ項目が表わす出現番号は、一意名-1 用の指標名の指標データ項目が表わす出現番号と、同時に同じだけ増加される。
    b. VARYING 指定で一意名-2 を書いた場合、この一意名-2 が指標データ項目であると、この指標データ項目が表わす出現番号は、一意名-1 用の指標名の指標データ項目が表わす出現番号と、同時に同じだけ増加される。一意名-2 が指標データ項目でないと、この値は一意名-1 用の指標が増加されるときに、1 だけ増加される。

形式 2

7. SEARCH 文で、SEARCH ALL の処理結果が正しく得られるためには、下記の条件が共に満たされる必要がある。
    a. 表の中のデータは、一意名-1 のデータ記述の ASCENDING KEY 句または DESCENDING KEY 句に指定された順番に並んでいること。
    b. WHEN 句中に書いたキーの内容が、表の要素を一意に識別するのに十分であること。
8. 形式 2 の SEARCH 文を用いると、非逐次表引き処理が行われる。このとき、データ名-1 用の指標データ項目の初期値は無視され、表引き処理につれて指標データ項目の値は変化する。ただし、その値が許される最大値よりも大きくなることも、許される最小値よりも小さくなることもない。(表の大きさについては、*データ部 - ファイルおよびデータ記述*の章の *OCCURS(反復)句*節で説明。) 許容範囲内のどの指標値に関しても WHEN 句に指定された条件のうちで満たされないものがある場合、AT END 指定が書いてあれば、その無条件文-1 に制御が移される。AT END 指定が書いてなければ、次の実行完結文に制御が移される。どちらの場合も、指標データ項目の最後の値はどうなるかわからない。すべての条件が満たされたときは、そのときの要素の出現番号が指標データ項目に記録され、制御は無条件文-2 に移される。
9. 表引き処理に使用される指標名は、一意名-1 の INDEXED BY 句の中の最初の (または唯一の) 指標名だけである。一意名-1 用のその他の指標名の指標データ項目の内容は変わらない。
10. OSVS VSC2 MF 無条件文-2 も NEXT SENTENCE 指定も必要ない。これらが指定されていないと、SEARCH 文は表の中の条件に一致した値に指標を設定する。

図 16-1 に、WHEN 指定が 2 つ含まれる形式 1 の表引き処理の流れ図を示す。

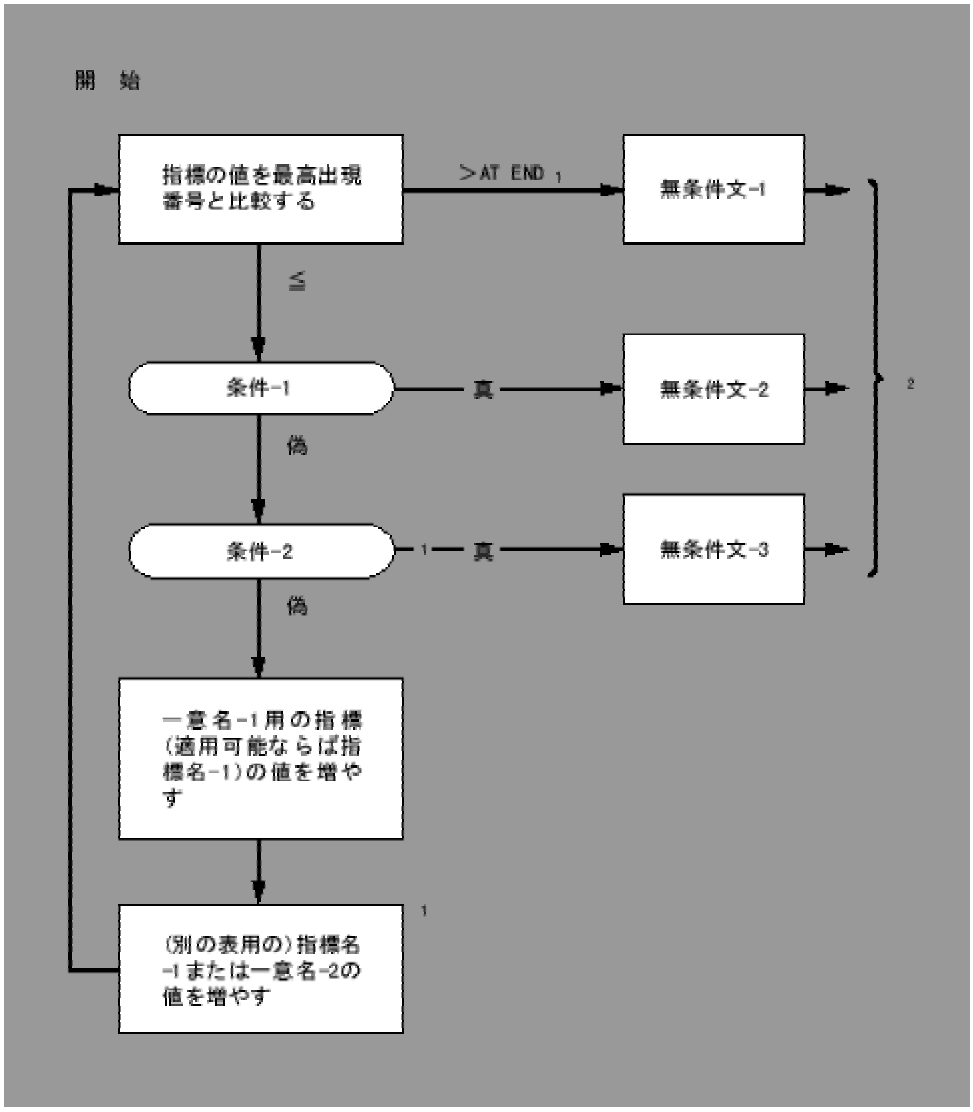


図 16-1：WHEN 指定が 2 つ含まれる表引き処理の流れ図

[1] これらの処理は、SEARCH 文中に指定されたときにだけ行われる。

[2] 各無条件文に GO TO 文が含まれていなければ、制御は次の実行完結文に移される。

16.2 SERVICE(サービス)文

OSVS OSC2

SERVICE (サービス) 文は通常、CICS プログラム内の連絡節中の項目の番地参照性を確立するために使用する。

一般形式

```
SERVICE { LABEL
        { RELOAD 一意名 }
```

構文規則

1. SERVICE LABEL は、メインフレームの CICS プリプロセッサによって、制御の流れを示すために生成される。これは、一般的な用途に向けたものではない。
2. SERVICE LABEL 文は手続き部にだけ書ける。宣言節には書けない。
3. 一意名は、連絡節内の BLL セル･ブロックで示される最初の 01 レベルの項目か、EXEC CICS 文に続いて番地参照性を再確立する必要があるその他の連絡節内の 01 レベルの項目とする。

一般規則

1. SERVICE LABEL 文の次の文のところで、有効性を失っているレジスタがすべて再ロードされる。
2. CICS プログラム中で OS/VS COBOL BLL 機構を使用する場合と、手続き部の冒頭でパラメータ･リストへの番地参照性が確立されなければならない。このために、手続き部の冒頭に「SERVICE RELOAD 一意名」文を追加する。ここで、一意名は連絡節内の最初の項目であり、連絡節内の他のすべての記述項に対するポインタを保持している。
3. OSVS 指令を適用して翻訳したプログラム中に、局所モードの EXEC CICS 文が含まれている場合、各 CICS コマンドの後ろに SERVICE RELOAD 文を続けて、一意名は CICS コマンド中に使用されたのと同じポインタ（BLL セル･リファレンス）とする。
4. OSVS 指令を適用して翻訳した CICS プログラムにおいては、大きさが 4096 バイト以上ある連絡節を参照するたびに、SERVICE RELOAD 文を使用して、4096 バイトを超える部分への番地参照性を再確立させる。
5. VS2 指令を適用して翻訳した CICS プログラムにおいては、SERVICE RELOAD 文は注記の役割を果たすだけである。

16.3 SET(設定)文

(ANS85)SET を使用して、外部スイッチの状態を変更することができる。

(ANS85)SET を使用して、条件変数の値を変更することができる。

(VSC2)(MF)SET 文を使用して、ポインタ変数にデータ項目の番地を割り当てることができる。また、ポインタ変数内容を調整することができる。

SET 文は、表操作処理のための参照点を決めるために、表要素の指標を設定する。

(MF)SET 文を使用して、プログラムの番地または手続きポインタ･データ項目へのプログラムの入口点を割り当てることができる。

(ISO2002)(MF)SET 文を使用して、プログラムポインタ･データ項目へのポインタの番地を割り当てることができる。

(ISO2002)(MF)SET 文を使用して、オブジェクト参照を割り当てることができる。

(MF)SET 文を使用して、桁詰めデータ項目の値を変更することができる。

(MF)SET 文を使用して、桁詰めデータ項目の番地を他の桁詰めデータ項目に割り当てることができる。

一般形式

形式 1

```
SET { [呼び名-1]... TO { ON  } } ...                    (ANS85)
    {                  { OFF } }
```

形式 2

```
SET { [条件名1]... TO { TRUE  } } [...] (MF)            (ANS85)
    {                 { FALSE } }
```

形式 3

```
SET { ADDRESS OF 一意名-1 } ... TO { ADDRESS OF 一意名-2 }   (MF)
    { ポインタ名-1          }      { ポインタ名-2        }   (VSC2)
                                   { NULL                }
                                   { NULLS               }
```

形式 4

```
SET [ポインタ名-3...] { UP   } BY { 一意名-3            }   (MF)
                      { DOWN } { 整数-1                }
                               { LENGTH OF 一意名-4    }
```

形式 5

```
SET { 指標名-1 }... TO { 指標名-2 }
    { 一意名-5 }       { 一意名-6 }
                       { 整数-2   }
```

形式 6

```
SET { 指標名-3 }... { UP   } BY { 一意名-7 }
    {             } { DOWN } {  {  整数-3   }
```

形式 7

```
SET 〔手続きポインタ名-1〕 TO { 手続きポインタ名-2                 } (NF)
                              { ENTRY { 一意名-8 }                }
                              {       { 整数-1   }                }
                              { NULL                              }
                              { NULLS                             }
```

形式 8

```
                                   (SO1000) (MF)
SET { 一意名-9 }... TO 一意名-10
```

形式 9 (イベントポインタの値)

```
                                   (MF)
SET { イベントポインター1 }... TO { TRUE  }
                                  { FALSE }
```

形式 10 (イベントポインタの番地)

```
                                   (MF)
SET { イベントポインター1 }... TO イベントポインター2
```

形式 11 (モニターポインタの値)

```
SET {モニターポインター1 }... TO [ NOT ] { BROWSING | READING | WRITING } [ CONVERTING FROM { BROWSING | WRITING } ]   (MF)
```

形式 12 (モニターポインタの番地)

```
SET {モニターポインター1} ... TO モニターポインター2   (MF)
```

形式 13 (ミューテックスポインタの値)

```
SET {ミューテックスポインター1 }... TO { ON | OFF }   (MF)
```

形式 14 (ミューテックスポインタの番地)

```
SET {ミューテックスポインター1} ... TO ミューテックスポインター2   (MF)
```

形式 15 (セマフォポインタの値)

```
SET { セマフォポインター1 }... { UP | DOWN } BY { 整数-4 | 一意名-11 }   (MF)
```

形式 16 (セマフォポインタの番地)

```
SET { セマフォポインター1 }... TO セマフォポインター2   (MF)
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある
    - STICKY-LINKAGE - SET 文によってポインタ･データ項目に入れられたデータ項目の番地を、副プログラムの呼出しの間に保持しておくか否かを決定する。

構文規則

すべての形式

1. 表意定数の NULL や NULLS あるいはポインタ名-2 や手続きポインタ名-2 によって参照されるデータ項目は送出し側を表す。「ADDRESS OF 一意名-2」、「ENTRY 一意名-8」、「ENTRY 定数-1」、または送出し側に保持されている値は送出し値を表す。

   ポインタ名-1 やポインタ名-3 や手続きポインタ名-1 によって参照されるデータ項目あるいは「ADDRESS OF 一意名-1」によって暗黙的に示される COBOL システム領域は受取り側を表す。

   一意名-3、整数-1、「LENGTH OF 一意名-4」は増分値を表す。

形式 1

2. (ANS85)呼び名-1 は、状態を変更できる外部スイッチに関するものとする。

   (MF)SET 文で参照できる外部スイッチについては、***環境部***の章の*特殊名段落*節を参照。

形式 2

3. (ANS85)条件名-1 は、条件変数に関するものとする。
4. (MF)FALSE 指定を書いた場合、条件名-1 のデータ記述項の VALUE 句に FALSE を指定しておく。
5. (MF)各 FALSE 指定または TRUE 指定は、それらの前に来る条件名-1 の反復要素および前の FALSE 指定または TRUE 指定に続く条件名-1 の反復要素 (もしあれば) に適用される。

形式 3

6. (VSC2)(MF)一意名-1 は、連絡節内の 01 レベルまたは 77 レベルの項目とする。
7. (SOURCE)(VSC2)(OS/390)(MF)一意名-2 は、77 レベル、または 01 から 49 までのレベルのデータ項目を参照しなければならない。

   (SOURCE)(VSC2)一意名-2 は、連絡節内で宣言されていなければならない。

   (OS/390)一意名-1 は、連絡節内、局所記憶節、または作業記憶節で宣言されていなければならない。

8. (VSC2)(MF)ポインタ名-1 とポインタ名-2 はそれぞれ、USAGE IS POINTER を伴う基本データ項目を参照する一意名でなければならない。

形式 4

9. (MF)ポインタ名-3 は USAGE IS POINTER を伴う基本データ項目を参照する一意名でなければならない。
10. (MF)一意名-3 は整数の基本項目でなければならない。
11. (MF)整数-1 は符合付きであってもよい。

形式 5

12. 一意名-5 と一意名-6 はそれぞれ、指標データ項目または整数として記述されている基本項目を参照しなければならない。両方を指定した場合は、同一の基本項目を参照してはならない。

形式 5 および 6

13. 整数-2 と整数-3 は、符号付きであってもよい。整数-2 は、正の数とする。

形式 6

14. 一意名-7 は整数の基本項目を参照しなければならない。

形式 7

15. (MF)手続きポインタ名-1 と手続きポインタ名-2 はそれぞれ、USAGE IS PROCEDURE-POINTER を伴う基本データ項目を参照する一意名でなければならない。
16. (MF)一意名-8 は英数字のデータ項目として定義されていなければならない。その値が COBOL のプログラム名であっても COBOL 以外のプログラム名であってもよいようにするためである。
17. (MF)定数-1 は文字定数でなければならない。

形式 8

18. 一意名-9 は字類がオブジェクトであり、受取り側項目として許されている項目でなければならない。
19. 一意名-10 はクラス名またはオブジェクト参照でなければならない。定義済みオブジェクト参照である SUPER を指定してはならない。
20. 一意名-9 によって参照されるデータ項目が一般的オブジェクト参照である場合、一意名-10 に指定できる定義済みオブジェクト参照は SELF と NULL だけである。
21. 一意名-9 によって参照されるデータ項目の記述にインターフェイス-1 を指すインターフェイス名がある場合、一意名-10 によって参照されるデータ項目は下記のどれかでなければならない。
    a. int-1 を識別するインターフェイス、または int-1 から継承するインターフェイス名を用いて記述されたオブジェクト参照
    b. 下記の規則に従うクラス名を用いて記述されたオブジェクト参照
        1. FACTORY 指定を書いた場合は、指定したクラスのファクトリ・オブジェクトのインターフェイスは int-1 を使用しなければならない。
        2. FACTORY 指定を書かなかった場合は、指定したクラスのインスタンス・オブジェクトのインターフェイスは int-1 を使用しなければならない。

c. ACTIVE-CLASS 指定を用いて記述されたオブジェクト参照。ただし、下記の規則に従う。
    1. FACTORY 指定を書いた場合は、一意名-10 によって参照されるデータ項目を含むクラスのファクトリ・オブジェクトのインターフェイスは int-1 を使用しなければならない。
    2. FACTORY 指定を書かなかった場合は、一意名-10 によって参照されるデータ項目を含むクラスのインスタンス・オブジェクトのインターフェイスは int-1 を使用しなければならない。
d. ファクトリ・オブジェクトが int-1 を使用するクラス名
e. 以下の規則に従う、定義済みオブジェクト参照の SELF
    1. SET 文がクラスのファクトリ定義中のメソッドに含まれている場合、そのファクトリ定義は、int-1 を参照する IMPLEMENTS 句中で記述されていなければならない。
    2. SET 文がクラスのインスタンス定義中のメソッドに含まれている場合、そのインスタンス定義は、int-1 を参照する IMPLEMENTS 句で記述されていなければならない。
f. 定義済みオブジェクト参照の NULL

22. 一意名-9 によって参照されるデータ項目の記述にクラス名がある場合、一意名-10 によって参照されるデータ項目は下記のどれかでなければならない。
    a. クラス名を用いて記述されたオブジェクト参照。ただし、下記の規則に従う。
        1. 一意名-9 によって参照されるデータ項目に ONLY 指定が書いた場合、一意名-10 によって参照されるデータ項目は ONLY 指定を用いて記述されていなければならず、一意名-10 によって参照されるデータ項目の記述中に指定されているクラス名は一意名-9 によって参照されるデータ項目の記述中に指定されているクラス名と同じである。
        2. 一意名-9 によって参照されるデータ項目に ONLY 指定を書かなかった場合、一意名-10 によって参照されるデータ項目は一意名-9 によって参照されるデータ項目の記述中に指定されているのと同じクラスまたはそのサブクラスを参照しなければならない。
        3. FACTORY 指定の有無は一意名-9 によって参照されるデータ項目の記述と同じである。
    b. ACTIVE-CLASS 指定を用いて記述されたオブジェクト参照。ただし、下記の規則に従う。
        1. 一意名-9 によって参照されるデータ項目の記述に ONLY 指定があってはならない。
        2. 一意名-10 によって参照されるデータ項目を含むクラスは一意名-9 によって参照されるデータ項目の記述中に指定されているのと同じクラスまたはそのサブクラスでなければならない。
        3. FACTORY 指定の有無は一意名-9 によって参照されるデータ項目の記述と同じでなければならない。
    c. 定義済みオブジェクト参照の SELF。ただし、下記の規則に従う。
        1. 一意名-9 によって参照されるデータ項目の記述 ONLY 指定があってはならない。
        2. SET 文を含むオブジェクトのクラスは一意名-9 によって参照されるデータ項目の記述中に指定されているのと同じクラスまたはそのサブクラスでなければならない。

3. 一意名-9 によって参照されるデータ項目の記述に FACTORY 指定がない場合、SET 文を含むオブジェクトはインスタンス･オブジェクトとする。
4. 一意名-9 によって参照されるデータ項目の記述に FACTORY 指定がある場合、SET 文を含むオブジェクトはファクトリ･オブジェクトとする。

d. クラス名。ただし、一意名-9 によって参照されるデータ項目の記述に FACTORY 指定があるものとして、そのクラス名は一意名-9 によって参照されるデータ項目の記述中に指定されているのと同じクラスまたはそのサブクラスを参照しなければならない。
e. 定義済みオブジェクト参照の NULL

23. 一意名-9 によって参照されるデータ項目 ACTIVE-CLASS 指定を書いた場合、一意名-10 によって参照されるデータ項目は下記のどちらかでなければならない。
    a. ACTIVE-CLASS 指定を用いて記述されたオブジェクト参照。ただし、FACTORY 指定の有無は一意名-9 によって参照されるデータ項目の場合と同じでなければならない。
    b. 定義済みオブジェクト参照の SELF。ただし、下記の規則に従う。
        1. 一意名-9 によって参照されるデータ項目の記述に FACTORY 指定がない場合、SET 文を含むオブジェクトはインスタンス･オブジェクトでなければならない。
        2. 一意名-9 によって参照されるデータ項目の記述に FACTORY 指定がある場合、SET 文を含むオブジェクトはファクトリ･オブジェクトでなければならない。
    c. 定義済みオブジェクト参照の NULL

形式 9

24. (MF)イベントポインタ-1 は、USAGE EVENT-POINTER を含むデータ項目として定義しなければならない。

形式 10

25. (MF)イベントポインタ-1 およびイベントポインタ-2 は、USAGE EVENT-POINTER を含むデータ項目として定義しなければならない。

形式 11

26. (MF)モニターポインタ-1 は、USAGE MONITOR-POINTER を含むデータ項目として定義しなければならない。
27. (MF)NOT を使用した場合は、CONVERTING 指定を使用してはならない。

形式 12

28. (MF)モニターポインタ-1 およびモニターポインタ-2 は、USAGE MONITOR-POINTER を含むデータ項目として定義しなければならない。

形式 13

29. (MF)ミューテックスポインタ-1 は、USAGE MUTEX-POINTER を含むデータ項目として定義しなければならない。

形式 14

30. (MF)ミューテックスポインタ-1 およびミューテックスポインタ-2 は、USAGE MUTEX-POINTER を含むデータ項目として定義しなければならない。

形式 15

31. (MF)セマフォポインタ-1 は、USAGE SEMAPHORE-POINTER を含むデータ項目として定義しなければならない。
32. (MF)一意名-11 は整数を参照しなければならない。

形式 16

33. (MF)セマフォポインタ-1 およびセマフォポインタ-2 は、USAGE SEMAPHORE-POINTER を含むデータ項目として定義しなければならない。

一般規則

形式 1

1. (ANS85)指定した呼び名-1 に関する各外部スイッチの状態は、ON を指定した場合はオンに、OFF を指定した場合はオフに、設定される。(***手続き部***の*スイッチ状態条件*節を参照。)

形式 2

2. (ANS85)条件名-1 に関する VALUE 句中の定数が、VALUE 句の規則に従って、条件変数に入れられる。(***データ部 - ファイルおよびデータ記述***の章の *VALUE(値)句*節を参照。) VALUE 句の中に複数の定数を指定してある場合には、条件変数の値は VALUE 句内の最初の定数に等しく設定される。
3. (ANS85)複数の条件名を指定した場合、その SET 文を実行した結果は、その中の各条件名-1 用にその順番どおりに別々の SET 文を書いて実行したのと等しい。
4. (MF)FALSE 指定を書くと、条件名-1 に関する VALUE 句の FALSE 指定の中の定数が、VALUE 句の規則に従って条件変数に入れられる。(***データ部 - ファイルおよびデータ記述***の章の *VALUE(値)句*節を参照。)

形式 3

5. (VSC2)(MF)送出し値はデータ項目の番地を表す。ポインタ名-2 を指定した場合、送出し値はポインタ名-2 によって参照されるデータ項目に関して保持されている値である。「ADDRESS OF 一意名-2」を指定した場合、送出し値は一意名-2 の番地を表す。
6. (VSC2)(MF)ポインタ名-1 を指定した場合、送出し値はポインタ名-1 によって参照されるデータ名に転記される。

7. (VSC2)(MF)「ADDRESS OF 一意名-1」を指定した場合、送出し値は COBOL のシステム領域に転記される。それ以降、ランタイム要素は一意名-1 によって参照される記憶領域が送出し値によって表される番地にあるものとして動作する。

   (MF)副プログラムを呼び出すときにリンクを維持するか否かは、STICKY-LINKAGE 指令の影響を受ける。

形式 4

8. (MF)SET 文が実行される前に、ポインタ名-3 によって参照されるデータ項目の値は論理レコード内のデータ項目の番地である当初の番地を表していなければならない。SET 文が実行された後では、ポインタ名-3 によって参照されるデータ項目の値は新しい番地を表す。当初の番地と新しい番地が共に同じ論理レコード（あるいは、番地空間がセグメント化れている環境では、同じセグメント）内に収まらない場合、ポインタ名-3 によって参照されるデータ項目の値を使用すると、結果はどうなるか分からない。
9. (MF)UP 句を指定した場合、増分値に指定された値を当初の番地に加えて、新しい番地が算出される。
10. (MF)DOWN 句を指定した場合、増分値に指定された値を当初の番地から引いて、新しい番地が算出される。

形式 5

11. 指標名-1 または一意名-5 が現れるたびに、以下の手順が繰り返される。このときつねに、指標名-2 または一意名-6 の値が使用される。これは、これらが実行される文の先頭にあるためである。一意名-5 に関連づけられた添え字や指標はどれも、そのデータ項目の値が変更される直前に評価される。
    a. 指標名-1 を指定すると、指標名-1 に値が設定され、指標名-2、一意名-6、または整数-2 が参照する表要素の出現回数に対応する表要素を参照する。一意名-6 が指標データ項目を参照する場合、または指標名-2 が一意名-1 と同じ表に関連している場合、変換は行われない。
    b. 一意名-5 が指標データ項目を参照する場合、それを指標名-2 または、同様に指標項目を参照する一意名-6 のどちらかの内容と等しく設定できる。どちらの場合も、変換は行われない。
    c. 一意名-5 が指標データ項目を参照しない場合、指標名-2 の値に対応する出現番号にのみ設定できる。この場合、一意名-6 も整数-2 も使用できない。
12. SET 文における各種の作用対象の有効な組合わせを、表 16-1 に示す。この中に、適用される一般規則の番号も示す。

表 16-1：指標を作用対象とする SET 文の有効な作用対象の組合わせ

送出し側項目	受取り側項目		
	整数データ項目	指標名	指標データ項目
整数定数	無効/11c	有効/11a	無効/11b
整数データ項目	無効/11c	有効/11a	無効/11b
指標名	Valid/11c	有効/11a	有効/11b[1]

指標データ項目	無効/11c	有効/11a[1]	有効/11b[1]

13. [1] =変換は行われない

形式 5 および 6

13. 指標名は対応する表を定義する際に、OCCURS 句の INDEXED BY 指定を用いて定義しておく。
14. 指標名-1 を指定する場合、SET 文を実行した後の指標の値は、対応する表の要素の数に見合っていなければならない。 SEARCH 文または PERFORM 文を実行した後の指標名の指標の値は、どうなっているかわからない。(*手続き部 - PERFORM - ROLLBACK*の章の *PERFORM(実行)文*および *SEARCH(表引き)文*節を参照。)

    指標名-2 を指定する場合、SET 文を実行する前の指標の値は、表中の、指標名-1 に関連する要素の数に対応していなければならない。

    指標名-3 を指定する場合、SET 文を実行する前および後の指標の値は、表中の、指標名-3 に関連する要素の数に対応していなければならない。

形式 6

15. 指標名-3 の内容が、整数-3 または一意名-7 によって表わされる出現番号の値だけ、増分 (UP BY) または減分 (DOWN BY) される。指標名-3 に複数の項目を指定すると、そのすべてについて上記の処理が繰り返される。その際、初回と同じように、一意名-7 の値が使用される。

形式 7

16. (MF)送出し値は COBOL または COBOL 以外のプログラム内の手続きの開始番地を表す。
17. (MF)送出し値は手続きポインタ-1 によって参照されるデータ項目に転記される。
18. (MF)手続きポインタ名-2 を指定した場合、送出し値は手続きポインタ-2 によって参照されるデータ項目内に収められている値となる。
19. (MF)定数-1、または一意名-8 によって参照されるデータ項目の内容は、参照される手続きの名前である。参照される手続きが COBOL の手続きである場合、参照される手続きの名前にはプログラム名または入口名が収められていなければならない。前者は参照されるプログラムのプログラム名段落に記述されているものである。後者は参照される手続きの ENTRY 文に記述されているものである。

    (MF)呼ばれるプログラムが COBOL のプログラムではない場合、プログラムまたは手続きの名前を形成するための規則がインターフェイスに関する COBOL システムのマニュアルに記載されている。

    (MF)参照される手続きが前に利用可能にされており、SET 文を実行する時点でもまだ利用可能な場合には、送出し値は参照される手続きの番地を表す。

(MF)参照される手続きが SET 文を実行する時点でもまだ利用可能ではない場合には、送出し値は COBOL システムのエラー手続きの番地を表す。

形式 8

20. 一意名-10 がオブジェクト参照である場合、一意名-10 によって識別されるオブジェクトへの参照が、一意名-9 によって参照される各データ項目に指定された順に収められる。
21. 一意名-10 がクラス名である場合、一意名-10 によって識別されるクラスのファクトリ・オブジェクトへの参照が、一意名-9 によって参照される各データ項目に指定された順に収められる。

形式 9 および 10

22. (MF)複数のイベントポインタ-1 を指定すると、各イベントポインタ-1 に対して、SET 文での指定と同じ順序で別の SET 文が書かれたときと同じ結果となる。

形式 9

23. (MF)SET 文の実行により、イベントポインタ-1 が参照するイベントの値が、TRUE または FALSE に設定される。

注:イベントが FALSE に設定されると、そのイベントを参照する WAIT 文を実行するどのスレッドも、そのイベントが別のスレッドで TRUE に設定されるまで実行されない。

形式 10

24. (MF)イベントポインタ-1 は、イベントポインタ-2 が参照するものと同じイベントデータ項目を参照するよう設定される。

形式 11 および 12

25. (MF)複数のモニターポインタ-1 を指定すると、各モニターポインタ-1 に対して、SET 文での指定と同じ順序で別の SET 文が書かれたときと同じ結果となる。

形式 11

26. (MF)NOT 指定を使用しない場合、モニターポインタ-1 が参照するモニターの値は、BROWSING、READING、または WRITING のいずれかに設定され、これにより、対応する形式のロックが設定する。この特定のモニターに対するこの特定の種類のロックは結果として、一致する NOT 指定または一致する CONVERTING FROM 指定を含む SET 文により解除されなければならない。

たとえば、ロックが以下の文で設定されたとする。

```
SET モニター-1 TO READING
```

このロックは以下の文で解除できる。

```
SET モニター-1 TO NOT READING
```

27. (MF)モニターに対して設定された現在の種類のロックは、CONVERTING 指定を使用して変更される。FROM 指定で指定された種類のロックは、そのスレッドにより現在設定されていなければならない。さらに、文が正常に実行されると、微細な操作により、TO 指定で指定された種類のロックに変更される。
28. (MF)間にロックを解除する SET 文を挟まない、連続した SET モニター文を実行すると、入れ子のロックを設定できる。一度、READING ロックが設定されると、そのスレッド中に BROWSING ロックも WRITING ロックも入れ子にすることはできない。 一度、BROWSING ロックまたは WRITING ロックが設定されると、そのスレッド中に他のどのレベルのロックも入れ子にすることはできない。

形式 12

29. (MF)モニターポインタ-1 は、モニターポインタ-2 が参照するものと同じモニターデータ項目を参照するよう設定される。

形式 13 および 14

30. (MF)複数のミューテックスポインタ-1 を指定すると、各ミューテックスポインタ-1 に対して、SET 文での指定と同じ順序で別の SET 文が書かれたときと同じ結果となる。

形式 13

31. (MF)SET 文の実行により、ミューテックスポインタ-1 が参照するミューテックスの値は ON または OFF に設定される。
32. (MF)ミューテックスが ON に設定された場合、そのスレッド中でそのミューテックスが OFF に設定されるまで、ON に設定しようとするスレッドはどれも実行されない。

形式 14

33. (MF)ミューテックスポインタ-1 は、ミューテックスポインタ-2 が参照するものと同じミューテックスデータ項目を参照するよう設定される。

形式 15 および 16

34. (MF)複数のセマフォポインタ-1 を指定すると、各セマフォポインタ-1 に対して、SET 文での指定と同じ順序で別の SET 文が書かれたときと同じ結果となる。

形式 15

35. (MF)セマフォポインタ-1 が参照するセマフォデータ項目の内容は、整数-4 の値に、または一意名-11 が参照するデータ項目に対応する値ずつ、大きく (UP BY の場合) または小さく (DOWN BY の場合) なる。その後、セマフォポインタ-1 が出現するたびにこの処理が繰り返され、そのたびに、一意名-11 が参照するデータ項目の値が使用される。これは、この値が実行される文の先頭にあるためである。
36. (MF)セマフォが DOWN BY n に設定され、n より小さい場合は、そのスレッドは、別のスレッドがセマフォを n 以上に上げるまで実行されない。

形式 16

37. (MF)セマフォポインタ-1 は、セマフォポインタ-2 が参照するものと同じセマフォデータ項目を参照するよう設定される。

16.4 SORT(整列)文

SORT (整列) 文は、ファイル中のレコードを整列させる。この処理は 3 つの段階に分かれる。最初の段階では、入力手続きを実行するかまたは他のファイルからレコードを転送するかして、整列用ファイルを作成する。次の段階では、指定されたキーの組に基づいて、整列用ファイル中のレコードの順序をそろえる。最後の段階では、整列させたレコードをその順番に出力手続きまたは出力ファイルに引き渡す。

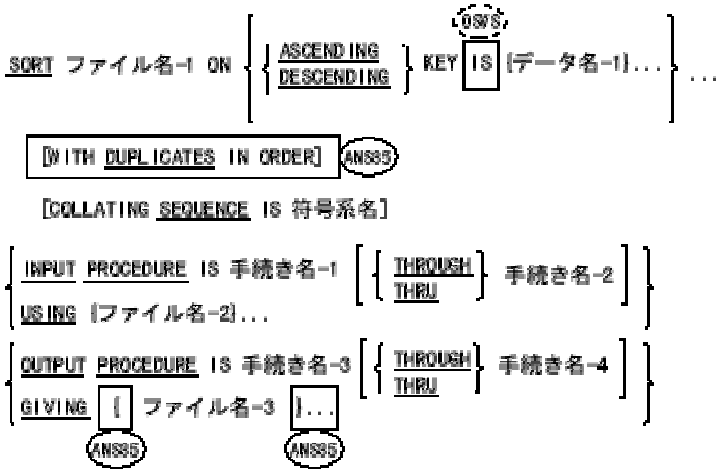
(MF)表の中の要素を整列させるためにも、SORT 文を使用できる。

例

SORT 動詞の使用例は、*言語リファレンス - 追加トピック* の *例*の章の*入出力手続きを使用したファイルの SORT* および *SORT の表の項目*節に掲載されている。

一般形式

形式 1



形式 2

```
SORT データ名-2 [ ON { ASCENDING } KEY [データ名-1]... ]...   (MF)
                     { DESCENDING }
      [WITH DUPLICATES IN ORDER]
      [COLLATING SEQUENCE IS 符号系名]
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
   - CALLSORT - SORT および MERGE の処理を行うために使用するプログラムを指定する。

構文規則

すべての形式

1. 形式 1 の SORT 文は、手続き部の宣言部分、SORT 文または MERGE 文に関連する入力手続き、出力手続きの場所には置くことができない。

   (MF)形式 2 の SORT 文は宣言節に置いてもよい。

2. データ名-1 はキーのデータ名であり、下記の規則に従う。
   a. キー・データ名は修飾してもよい。
   b. キー・データ名のデータ項目は、可変長項目であってはならない。
   c. キー・データ項目は、浮動小数点数項目でもよい。
   d. (OSVS)(OSC2)(MF)キーが外部浮動小数点数であると、コンパイラはそのデータ項目を数字データとしてではなく文字データとして扱う。レコードがそろえられる順序は、使用する文字の照合順序に左右される。
   e. (OSVS)(OSC2)(MF)キー・データ項目が内部浮動小数点数である場合、キーの順序は数値順となる。

形式 1

3. ファイル名-1 は、データ部の整列併合ファイル記述項に記述しておく。
4. データ名-1 はキーのデータ名であり、下記の規則に従う。
   a. キーとするデータ項目は、ファイル名-1 のレコード中に記述しておく。
   b. ファイル名-1 に複数のレコード記述を含める場合、キー・データ項目は 1 つのレコード記述の中だけに指定する。
   c. >キー・データ項目には、OCCURS 句を含む記述項は記述できない。また、OCCURS 句を含む記述項の下位にキー・データ項目を属させてはならない。
   d. ファイル名-1 のファイルのレコードが可変長である場合、データ名をキーに指定するデータ項目はすべて、そのファイルの最初の x 文字位置のレコード

の範囲内に入っていること。ここで x は、ファイル名-1 によって参照されるファイルに指定された最小レコードサイズを示す。

5. 語 THRU と THROUGH は同義語である。
6. ファイル名-2 とファイル名-3 はデータ部の中の整列併合ファイル記述項にではなく、ファイル記述項に記述しておく。
7. (ANS85)ファイル名-2 のファイルとファイル名-3 のファイルは、同じマルチ・ファイル・リールに収録されていてもよい。
8. (ANS85)ファイル名-3 が索引ファイルを指す場合、最初のデータ名-1 には ASCENDING を指定する。また、データ名-1 がレコード上で占める文字位置は、そのファイルの主レコードキーと同じにする。
9. 1 つの SORT 文の中で使用しているファイル名の組は、SAME SORT AREA 句または SAME SORT-MERGE AREA 句には指定できない。GIVING 指定の対象となっている複数のファイルは、同じ SAME 句には指定できない。
10. (ANS85)GIVING を指定したとき、ファイル名-3 のファイルに含まれるレコードが可変長である場合、ファイル名-1 のファイル中に含まれるレコードの大きさは、ファイル名-3 のレコードの最小のもの以上かつ最大のもの以下にする。ファイル名-3 のファイルに含まれるレコードが固定長である場合、ファイル名-1 のファイル中に含まれるレコードの大きさは、ファイル名-3 のレコード最大のものを超えてはならない。

    (ANS85)これらのレコードを構成する基本データ項目のデータ記述が異なる場合、対応するレコード同士の文字数が等しくなるように記述することは、プログラマの責任である。

11. (ANS85)USING を指定し、ファイル名-1 のファイルに含まれるレコードが可変長である場合、ファイル名-2 のファイル中に含まれるレコードの大きさは、ファイル名-1 のレコードの最小のもの以上かつ最大のもの以下にする。ファイル名-1 のファイルに含まれるレコードが固定長である場合、ファイル名-2 のファイル中に含まれるレコードの大きさは、ファイル名-1 のレコードの最大のものを超えてはならない。
12. 手続き名-1 は、入力手続きの名前を表わす。手続き名-3 は、出力手続きの名前を表わす。
13. 手続き名-1、手続き名-2、手続き名-3、手続き名-4 は、節名とする。

    (ANS85)(SVS)この制限は削除された。

14. USING 句または GIVING 句に 10 個以上のファイル名を指定したい場合、コンパイラ指令の CALLSORT"EXTSM"を使用しなければならない。この指令を使用すると、255 個までのファイルを指定できる。

形式 2

15. (MF)データ名-2 は修飾してもよい。データ名-2 のデータ記述項には、OCCURS 句を記述しておく。
16. (MF)データ名-1 のデータ項目は、データ名-2 のデータ項目と同じであるか、またはデータ名-2 のデータ項目の下位に属する項目とする。
17. (MF)データ名-1 とデータ名-2 が同じでない場合、データ名-1 のデータ項目のデータ記述項に OCCURS 句が記述してあってはならない。また、データ名-1 のデータ項目

は、データ名-2 の下位に属し OCCURS 句を伴うデータ項目の下位にも属してはならない。

18. (MF)データ名-2 の表の記述中に KEY 指定がある場合にだけ、KEY 指定を省略できる。
19. (MF)OCCURS 句を伴うデータ項目の下位にデータ名-2 のデータ項目が属す場合、データ名-2 のデータ項目の親である OCCURS 句を伴うデータ項目には、指標名を付ける。

一般規則

すべての形式

1. 語 ASCENDING および DESCENDING は、別の ASCENDING または DESCENDING が出てくるまで、後ろに繰り返し続くすべてのデータ名-1 に効力を及ぼす。
2. データ名-1 のデータ項目は、キー・データ項目である。このキー・データ項目によって、ファイル名-1 のファイルからレコードが引き取られる順序または整列が終了した後で表の要素が格納される順序が決まる。キーの強さの順序は、SORT 文の中に記述した順である。ASCENDING または DESCENDING の指定は、キーの強さには関係ない。
3. DUPLICATES 指定をした場合、あるレコードまたは表要素のすべてのキー・データ項目の内容が他のいくつかのレコードまたは表要素の対応するキー・データ項目の内容と等しいとき、引き取られるレコードの順序は下記のようになる。
    a. SORT 文の中または実行時スイッチに指定された入力ファイルの順序。 入力ファイルからレコードを読み込んだ場合は、その順序が保たれる。
    b. 入力手続きが用いられているときは、入力手続きからレコードが引き渡された順序。
    c. 整列処理が実行される前の、表要素の内容の相対順序。
4. DUPLICATES 指定をしなかった場合、あるレコードまたは表要素のすべてのキー・データ項目の内容が他のいくつかのレコードまたは表要素の対応するキー・データ項目の内容と等しいとき、引き取られるレコード順序または表要素の内容の相対順序はどうなるかわからない。
5. 文字のキー・データ項目の比較に適用される文字の照合順序は、SORT 文の実行開始時に、下記の優先順序で決定される。
    a. 最初は、指定してあれば、SORT 文中の COLLATING SEQUENCE に指定した文字の照合順序。
    b. 次は、プログラム用に設定してある文字の照合順序。

形式 1

6. SORT 文は、ファイル名-2 のファイル中のすべてのレコードまたは入力手続きによって、ファイル名-1 のファイルに引き渡されるレコードを受け取る。その後、データ名-1 のデータ項目の値と ASCENDING または DESCENDING 指定によって定まる順に、それらのレコードを並べ変え、出力手続きまたはファイル名-3 のファイルに引き渡す。
7. ファイル名-1 のファイルから返された 2 つのレコードの相対順序を決めるために、比較条件の作用対象比較規則に従い、対応するキー・データ項目の内容が、最も強いキー・データ項目から順に比較される。

    a. 対応するキー･データ項目同士の内容が異なり、キーに ASCENDING 指定がなされている場合、キーの値が小さい方のレコードが先に返される。
    b. 対応するキー･データ項目同士の内容が異なり、キーに DESCENDING 指定がなされている場合、キーの値が大きい方のレコードが先に返される。
    c. 対応するキー･データ項目同士の内容が等しい場合、次に強いキー･データ項目の内容に基づいて、順序が決定される。
8. SORT 文の実行は、下記の 3 つの段階に区分される。
    a. ファイル名-1 のファイル中で、レコードを利用できるようにする。このためには、入力手続き中の RELEASE 文を実行するか、またはファイル名-2 に対して暗黙的に READ 文を実行する。この段階の処理が始まるときに、ファイル名-2 のファイルは開かれていてはならない。この段階の処理が終わるとき、ファイル名-2 のファイルは閉じられている。
    b. ファイル名-1 のファイル中のレコードの順序がそろえられる。この段階では、ファイル名-2 およびファイル名-3 のファイルの処理は、何も行われない。
    c. ファイル名-1 のファイルのレコードがそろえられた順序に従って、利用できるようにされる。このために、順序をそろえられたレコードがファイル名-3 のファイルに書き出されるか、RETURN 文を実行することによって出力手続きで利用できるようにされる。この段階の処理が始まるときに、ファイル名-3 のファイルは開かれていてはならない。この段階の処理が終わるとき、ファイル名-3 のファイルは閉じられている。
9. 入力手続きでは、RELEASE 文によって 1 件ずつファイル名-1 のファイルに引き渡されるレコードを、必要に応じて選択したり修正したり複写したりする。この入力手続きの範囲には、その中から CALL 文、PROGRAM 指定のない EXIT 文、GO TO 文、PERFORM 文によって制御を移された結果実行されるすべての文が含まれる。この入力手続きの範囲内で、MERGE 文、RETURN 文、SORT 文を起動するようなことはできない。
10. 入力手続きを指定すると、SORT 文によってファイル名-1 のファイル中のレコードの並べ替えが開始される前に、入力手続きに制御が渡される。コンパイラは、入力手続きの最後の文の末尾に復帰機構を組み込む。入力手続きの最後の文に制御が渡ると、ファイル名-1 のファイルに引き渡されているレコードの整列処理が開始される。
11. USING を指定すると、ファイル名-2 のファイル中のすべてのレコードが、ファイル名-1 のファイルに書き出される。SORT 文を実行すると、ファイル名-2 の各ファイルに対して、下記の処理が行われる。
    a. ファイルの処理が開始される。これは INTPUT 指定および WITH LOCK 指定をした OPEN 文を実行するように行われる 。
    b. 論理レコードが、入力ファイルから整列用ファイルに引き渡される。これは NEXT および AT END 指定をした READ 文が実行されるように行われる。ファイル名-1 のファイルに含まれるレコードが固定長である場合、ファイル名-2 のファイル中にファイル名-1 の固定長レコードよりも短いものがあると、ファイル名-2 のレコードがファイル名-1 のファイルに引き渡されるときに、ファイル名-2 のレコードの後部に空白が埋められる。ファイル名-1 のファイルのレコードが可変長であると、ファイル名-1 のファイルに書き出されるレコードの大きさは、ファイル名-2 またはファイル名-3 から読み込んだときのレコードの大きさとなる。このことは、ファイル名-1 のファイルのファイル記述項に指定した RECORD VARYING DEPENDING ON 句または OCCURS DEPENDING ON 句

の DEPENDING ON 指定の対象となっているデータ項目の内容にかかわらず、適用される。

c. ファイルの処理が終了される。これは任意指定の何もない CLOSE 文が実行されたように行われる。この終了処理が済んだ後で、SORT 文によるファイル名-1 のファイルの整列処理が開始される。

相対ファイルに関しては、GIVING 指定の対象にファイル名-2 を指定しないと、SORT 文の実行が終了したときに、RELATIVE KEY データ項目の内容はどうなっているかわからない。

USE AFTER EXCEPTION 手続きを指定してあれば、上記の暗黙の処理においてそれも対象となる。しかし、その USE 手続きの中から、ファイル名-2 のファイルを操作したり、そのレコード領域をアクセスしたりするような文が実行されるようなことがあってはならない。

ファイル名-2 のファイル記述項中に指定した RECORD VARYING DEPENDING ON 句の対象となっているデータ項目の値は、SORT 文が完了した時点でどのようになっているかはわからない。

12. 出力手続きでは、RETURN 文によって 1 件ずつファイル名-1 のファイルから引き取られる整列済みのレコードを、必要に応じて選択したり修正したり複写したりする。この出力手続きの範囲には、その中から CALL 文、PROGRAM 指定または METHOD 指定のない EXIT 文、GO TO 文、PERFORM 文によって制御を移された結果実行されるすべての文が含まれる。また、この出力手続きの範囲内の文を実行した結果として実行される宣言手続き中のすべての文も、この出力手続きの範囲に含まれる。この出力手続きの範囲内で、MERGE 文、RELEASE 文、SORT 文を起動するようなことはできない。
13. 出力手続きを指定すると、SORT 文によってファイル名-1 のファイル中のレコードの並べ替えが終了した後で、出力手続きに制御が渡される。コンパイラは、出力手続きの最後の文の末尾に復帰機構を組み込む。出力手続きの最後の文に制御が渡ると、復帰機構によって SORT 文の処理が終了され、SORT 文の次の実行文に制御が移される。出力手続きに入る前に、レコードの並べ替えは済んでいる。要求されれば次のレコードを引き渡せる状態に達している。出力手続き中の RETURN 文は、次のレコードを求める要求である。
14. GIVING を指定すると、順序をそろえられたすべてのレコードがファイル名-3 のファイルに書き出される。これは SORT 文に暗黙的に出力手続きを使用することである。SORT 文を実行すると、ファイル名-3 の各ファイルに対して下記の処理が行われる。
    a. ファイルの処理が開始される。これは OUTPUT 指定をした OPEN 文を、暗黙的に実行することである。この開始処理は、入力手続きがあれば、その実行が済んだ後で行われる。
    b. 論理レコードが、整列用ファイルから出力ファイルに書き出される。これは何も任意指定をしない WRITE 文を、暗黙的に実行することである。ファイル名-3 のファイルに含まれるレコードが固定長である場合、ファイル名-1 のファイル中にファイル名-3 の固定長レコードよりも短いものがあると、ファイル名-1 のレコードがファイル-3 のファイルに書き出されるときに、ファイル名-1 のレコードの後部に空白が埋められる。ファイル名-3 のファイルのレコードが可変

長であると、ファイル名-3 のファイルに書き出されるレコードの大きさは、ファイル名-1 から読み込んだときのレコードの大きさとなる。このことは、ファイル名-3 のファイルのファイル記述項に指定した RECORD VARYING DEPENDING ON 句または OCCURS DEPENDING ON 句の対象となっているデータ項目の内容にかかわらず、適用される。

出力ファイルが相対ファイルである場合、最初に引き取られるレコードの RELATIVE KEY データ項目の値は "1" 、2 番目に引き取られるレコードの RELATIVE KEY データ項目の値は "2" 、という具合いになる。SORT 文の実行が終了した後の RELATIVE KEY データ項目の内容は、どうなっているかわからない。

c. ファイルの処理が終了される。これは任意指定の何もない CLOSE 文を、暗黙的に実行することである。

USE AFTER EXCEPTION 手続きを指定してあれば、上記の暗黙の処理において、それも対象となる。しかし、その USE 手続きの中から、ファイル名-3 のファイルを操作したりそのレコード領域を呼び出したりするような文が実行されるようなことがあってはならない。外部的に定義されているファイル境界を超えて最初に書き出しが試みられたときに、そのファイルに関して USE AFTER EXCEPTION 手続きを指定してあれば、その手続きが実行される。USE 文の規則に従って制御が戻されると、さらに暗黙の書き出し処理が行われることはない。上記のように、そのファイルの処理は終了される。

15. ファイル名-1 のファイル記述項中に指定した RECORD VARYING DEPENDING ON 句の対象となっているデータ項目の値は、GIVING 指定をした SORT 文が完了した時点でどのようになっているかはわからない。
16. SORT 文を含むプログラムを区分化できる。(*言語リファレンス - 追加トピック*の*区分化*節を参照。) ただし、下記の制限がある。
    a. 独立区分ではない区分中の節内に SORT 文が存在する場合、その SORT 文から呼び出される入力手続きまたは出力手続きは、下記のどちらかの状態で組み込まれていなければならない。
        - 非独立区分中に完全に含まれる。
        - 単一の独立区分中に完全に含まれる。
    b. 独立区分中に SORT 文が存在する場合、その SORT 文から呼び出される入力手続きまたは出力手続きは、下記のどちらかの状態で組み込まれていなければならない。
        - 非独立区分中に完全に含まれる。
        - SORT 文と同じ独立区分中に完全に含まれる。
17. OSVS VSC2 MF SORT ランタイム要素用に、特殊レジスタの SORT-RETURN が用意されている。この特殊レジスタには、整列処理が終わったときに 0 (正常終了) または 16 (不成功) が設定される。入力/出力手続きの中で、この特殊レジスタに値 16 を設定して、途中で整列処理を終わらせることができる。この場合、整列処理は次の RETURN 文または RELEASE 文のところで終了される。

形式 2

17. (MF)SORT 文は、データ名-2 が参照する表の要素を並べ替え、並べ替えた表を、ASCENDING または DESCENDING 指定により決定された順序、またはデータ名-2 に関連づけられた KEY 指定により決定された順序で、データ名-2 に表示する。
18. (MF)並べ替えた後の表要素の相対順序を決めるために、比較条件の規則に従い、対応するキー･データ項目の内容が、最も強いキー･データ項目から順に比較される。
    a. 対応するキー･データ項目同士の内容が異なり、キーに ASCENDING 指定が含まれている場合、キーの値が小さい方の表要素に小さい出現番号が割り当てられる。
    b. 対応するキー･データ項目同士の内容が異なり、キーに DESCENDING 指定が含まれている場合、キーの値が大きい方の表要素に小さい出現番号が割り当てられる。
    c. 対応するキー･データ項目同士の内容が等しい場合、次に強いキー･データ項目の内容に基づいて、順序が決定される。
19. (MF)データ名-2 の表の要素の出現番号は、OCCURS 句の規則に従って決定される。
20. (MF)OCCURS 句を伴うデータ項目の下位に属するデータ項目をデータ名-2 が指す場合、親の表の最初または唯一の指標名を、希望する値に設定してから SORT 文を実行する。
21. (MF)KEY 指定をしないと、データ名-2 が参照する表のデータ記述項中の KEY 指定によって、順序が決定される。
22. (MF)KEY 指定をすると、この指定の方が、データ名-2 の表のデータ記述項中の KEY 指定よりも優先する。
23. (MF)KEY 指定をしてもデータ名-1 を省略すると、データ名-2 が参照するデータ項目がキー･データ項目とされる。
24. (MF)データ名-2 が参照する表の要素を整列した結果は、同じ表に書き戻される。
25. (MF)COLLATING SEQUENCE 指定は注記としてのみ処理される。

16.5 START(位置決め)文

START (位置決め) 文は、相対ファイルまたは索引ファイルの中で、以降に検索するレコードに論理的に位置付ける。この文は順編成のファイルには適用できない。

(MF)START 文は、同期または非同期でスレッドの実行を開始する。

一般形式

形式 1 (相対ファイル)

START ファイル名-1 [KEY { IS EQUAL TO
IS =
IS GREATER THAN
IS >
IS NOT LESS THAN
IS NOT <
ANS85
IS GREATER THAN OR EQUAL TO
IS >=
IS LESS THAN
IS <
IS NOT GREATER THAN
IS NOT >
IS LESS THAN OR EQUAL TO
IS <=
IS NOT GREATER THAN OR EQUAL TO
IS NOT >=
IS NOT LESS THAN OR EQUAL TO
IS NOT <=
NF } データ名-1]

[INVALID KEY 無条件文-1]
[NOT INVALID KEY 無条件文-2]
[END-START]
ANS85

形式 2 (索引ファイル)

```
START ファイル名-1 [ KEY { IS EQUAL TO
                           IS =
                           IS GREATER THAN
                           IS >
                           IS NOT LESS THAN
                           IS NOT <                    (ANS85)
                           IS GREATER THAN OR EQUAL TO
                           IS >=
                           IS LESS THAN                (NF)
                           IS <
                           IS NOT GREATER THAN
                           IS NOT >
                           IS LESS THAN OR EQUAL TO
                           IS <=
                           IS NOT GREATER THAN OR EQUAL TO
                           IS NOT >=
                           IS NOT LESS THAN OR EQUAL TO
                           IS NOT <=
                         } ]

{ データ名-1 }                    [ WITH SIZE { 一意名-1 } ] ]
{ 分割キー名-1 }  (NF)                        { 定数-1 }
                                              (NF)
[INVALID KEY 無条件文-1]

[NOT INVALID KEY 無条件文-2]
[END-START]                                   (ANS85)
```

形式 3 (スレッド)

```
START { 定数-1
        一意名-1
        手続きポインター1 } ...                    (NF)
[ USING [ BY CONTENT ] 一意名-2]
[ RETURNING INTO 一意名-3
  IDENTIFIED BY スレッドポインター1 ]
[ STATUS IS 一意名-4 ]
[ ON EXCEPTION 無条件文-1          ]
[ NOT ON EXCEPTION 無条件文-2      ]
[ END-START ]
```

必須の比較文字 ">"、"<"、">="、"<="、"=" には下線を引いていないことに注意。これは、"≥" (以上)のような他の記号と混同することを避けるためである。

構文規則

すべての形式 (相対ファイルおよび索引ファイル)

1. ファイル名-1 は、相対ファイルまたは索引ファイルの名前にする。
2. ファイル名-1 は、順呼出し法または動的呼出し法のファイルの名前にする。
3. データ名-1 は修飾してもよい。
4. ファイル名-1 に適用できる USE 手続きを指定していないときは、INVALID KEY 句を指定する。

   OSVS VSC2 MF この規則は強制しない。

形式 1 (相対ファイル)

5. KEY 指定をする場合、データ名-1 は対応するファイル管理記述項の RELATIVE KEY に指定したデータ項目とする。

形式 2 (索引ファイル)

6. KEY 指定をする場合、ファイル名-1 のレコードキーとして指定したデータ項目を、データ名-1 のデータ項目として使用できる。さらに、ファイル名-1 のレコードキーとして指定したデータ項目の下位に属し、そのレコードキー・データ項目と左端の文字位置が同じで項類が英数字のデータ項目を、データ名-1 のデータ項目として使用できる。
7. MF 分割キー名-1 のデータ項目には、ファイル名-1 のレコードキーとして指定したデータ項目のいくつかを使用できる。
8. MF WITH SIZE 指定は、位置決め処理で使用するキー中の文字数を示す。
9. MF WITH SIZE 指定の対象とした場合、一意名-1 は基本整数データ項目の名前とする。

形式 3 (スレッド)

MF

10. 定数-1 は、数字定数以外の定数でなければならず、表意定数であってはならない。
11. 一意名-1 は、英数字データ項目として定義されなければならず、その値は COBOL プログラム名と COBOL 以外のプログラム名のどちらでもよい。
12. 一意名-3 は、USAGE POINTER またはサイズが 4 バイトのデータ項目のいずれかとして定義しなければならない。この定義は、新しいスレッドに対する EXIT PROGRAM / STOP RUN 文中の GIVING/RETURNING 項目の定義により異なる。一意名-3 は、USAGE POINTER 項目として定義した場合は、関数一意名であってはならない。
13. スレッドポインタ-1 は、USAGE THREAD-POINTER として定義しなければならない。
14. 一意名-4 は、4 桁以上の長さの整数データ項目として定義しなければならない。

一般規則

すべての形式 (すべてのファイル)

1. START 文を実行するときには、ファイル名-1 のファイルは INPUT モードまたは I-O モードで開いておく。(*手続き部 - MERGE - OPEN* の章の *OPEN(開く)文*節を参照。)

2. KEY 指定をしないと、比較演算 IS EQUAL TO を暗黙的に指定したものとみなされる。
3. START 文を実行すると、ファイル名-1 に対応する FILE STATUS データ項目を指定してあれば、その値が更新される。(*手続き部*の章の*入出力状態*節を参照。)
4. (MF)START 文が、レコードロックを取得したり検出したり解除したりすることはない。

形式 1 (相対ファイル)

5. KEY 指定中の比較演算子によって表わされる比較は、ファイル名-1 のファイル中のレコードのキー (下記の 6 項を参照) とデータ名-1 との間で行われる。
   a. (ANS85)比較演算子で、キーをデータ名-1 と等しいか、より大きいか、以上と指定すると、

      ファイル中に現存する論理レコードのうちで、その比較条件を満足する最初のものにファイル位置指示子が位置付けられる。

   b. (MF)比較演算子で、キーをデータ名-1 より小さいか、以下と指定すると、ファイル中に現存する論理レコードのうちで、その比較条件を満足する最後のものにファイル位置指示子が位置付けられる。
   c. 比較条件を満たすレコードがファイル中に存在しないと、無効キー条件が発生する。この場合、START 文の実行は不成功に終わり、ファイル位置指示子の内容はどうなるかわからない。(*手続き部*の章の*無効キー条件*節を参照。)
6. 上記の 5 項の比較では、ファイル名-1 の RELATIVE KEY 句に指定したデータ項目が使用される。したがって、ファイル名-1 には RELATIVE KEY 句を指定しておく。

形式 2 (索引ファイル)

7. KEY 指定中の比較演算子によって表わされる比較は、ファイル名-1 のファイル中のレコードのキー (下記の 8 項を参照) とデータ名-1 との間で行われる。ファイル名-1 が索引ファイルであり、作用対象の大きさが等しくないと、短い方を超える長い方の右側の部分が切捨てられたように比較が進められる。文字の比較に関するその他の規則が、すべてこの比較に適用される。ただし、PROGRAM COLLATING SEQUENCE 句は、この比較には効力をもたない。(*手続き部*の章の*比較条件*節の*文字作用対象の比較*を参照。)
   a. (ANS85)比較演算子で、キーをデータ名-1 と等しいか、より大きいか、以上と指定すると、

      ファイル中に現存する論理レコードのうちで、その比較条件を満足する最初のものにファイル位置指示子が位置付けられる。

   b. (MF)比較演算子で、キーをデータ名-1 より小さいか、以下と指定すると、ファイル中に現存する論理レコードのうちで、その比較条件を満足する最後のものにファイル位置指示子が位置付けられる。キーの値が重複するものが存在すると、そのうちの最後のものにファイル位置指示子が位置付けられる。
   c. 比較条件を満たすレコードがファイル中に存在しないと、無効キー条件が発生する。この場合、START 文の実行は不成功に終わり、ファイル位置指示子の内容はどうなるかわからない。(*手続き部*の章の*無効キー条件*節を参照。)

8. 上記の 7 項の比較において KEY 指定をした場合は、データ名のデータ項目 (キー) が使用される。
9. 上記の 7 項の比較において KEY 指定をしなかった場合は、ファイル名-1 の RECORD KEY 句に指定したデータ項目 (キー) が使用される。
10. START 文の実行が正常に終了すると、下記のように参照キーが設定され、後続の形式 3 の READ 文で使用できるようになる。(*手続き部 - PERFORM - ROLLBACK*の章の *READ(読み込み)文*節を参照。)
    a. KEY 指定をしなかった場合は、ファイル名-1 の主レコードキーが参照キーとなる。
    b. KEY 指定をし、ファイル名-1 のレコードキーとしてデータ名-1

       (MF)または分割キー名-1

       を指定した場合は、そのレコードキーが参照キーとなる。

    c. KEY 指定をし、ファイル名-1 のレコードキーとしてデータ名-1

       (MF)または分割キー名-1

       を指定しなかった場合は、データ名-1

       (MF)または分割キー名-1

       と左端の文字位置が同じであるレコードキーが参照キーとなる。

11. START 文の実行が不成功に終わったときは、参照キーはどうなっているかわからない。

形式 3 (スレッド)

(MF)

12. 定数-1、一意名-1、または手続きポインタ-1 は、翻訳単位でもっとも外部にあるプログラム名、プログラム中の入口点、または他の言語のラベルであるプログラム名を参照しなければならない。これが、新規作成されたスレッドの実行開始点となる。
13. 一意名-2 を指定した場合は、参照により渡される単一のパラメータが受け入れられるよう、新規作成されたスレッドの開始点を定義しなければならない。
14. BY CONTENT 指定を使用した場合は、一意名-2 の内容が、システムにより割り当てられたスレッドセーフの作業領域にコピーされ、この作業領域への参照が新規作成されたスレッドの開始点に渡される。この作業領域は、作成したプログラムのデータ領域の有効期限にかかわらず、このスレッドの実行中は有効のままである。
15. BY CONTENT 指定を使用しない場合は、 一意名-2 への直接参照が新規作成されたスレッドの開始点に渡される。プログラマは、新規作成されたスレッドが参照する間、このデータ領域が確実に有効であるようにしておく必要がある。
16. RETURNING 指定を使用した場合は、新規作成されたスレッドが動作を完了し、一意名-3 の値を返し、制御を START 文に戻す。

17. IDENTIFIED BY 指定を使用した場合は、新規作成されたスレッドの実行が開始され、新規作成されたスレッドを参照するハンドルがスレッドポインタ-1 に配置され、 制御が START 文に返される。ハンドルを使用して、WAIT 文は戻り値を取得し、実行の同期を取り、スレッドの資源を解放する。このハンドルは、CBL_THREAD_登録集ルーチンで使用されるような有効なスレッドハンドルである。
18. RETURNING 指定も IDENTIFIED 指定も使用しない場合は、新規作成されたスレッドの実行が開始され、制御が START 文に返される。スレッドの資源はすべて、スレッドが終了したときに自動的に解放される。
19. STATUS 指定を使用した場合は、START 文の実行により、CBL_THREAD_登録集ルーチンに対して指定された戻り符号のいずれかが、一意名-4 中に配置される。
20. START 文が不成功に終わった場合、一意名-4 が指定されていれば、CBL_THREAD_登録集ルーチンに対して指定された戻り符号のいずれかがその中に配置される。さらに、以下のいずれかが行われる。
    a. ON EXCEPTION 指定を使用した場合は、制御が無条件文-1 に渡される。その後、無条件文-1 で指定された各文の規則に従って、実行が続行される。手続きの分岐、または制御の移動を明示的に指示する条件文が実行された場合は、制御が文の規則に従って渡される。そうでない場合は、無条件文-1 の実行が完了すると同時に、START 文の最後に制御が渡され、NOT ON EXCEPTION 指定が使用されていても無視される。
    b. 上記以外の場合で、NOT ON EXCEPTION 指定または STATUS 指定を使用した場合は、START 文の最後に制御が渡され、NOT ON EXCEPTION 指定が使用されていても無視される。
    c. 上記以外の場合、実行単位は実行時エラーとともに終了する。
21. START 文が正常に実行された場合は、以下が記載された順序通りに行われる。
    a. STATUS 指定を使用した場合は、ZERO が一意名-4 に転記される。
    b. NOT ON EXCEPTION 指定を使用した場合は、制御が無条件文-2 に渡される。 その後、無条件文-2 で指定された各文の規則に従って、実行が続行される。手続きの分岐、または制御の移動を明示的に指示する条件文が実行された場合は、制御が文の規則に従って渡される。そうでない場合は、無条件文-2 の実行が完了すると同時に、START 文の最後に制御が渡され、ON EXCEPTION 指定が使用されていても無視される。

16.6 STOP(停止)文

STOP (停止) 文は、実行単位の実行を、永久的にまたは一時的に停止させる。

(ANS85) 「STOP 定数」文は、ANSI'85 標準では廃要素に分類されており、ANSI 標準の次回の全面改訂の際に、削除される予定である。

(MF) この構文は、Micro Focus COBOL に組み込まれているすべての方言で全面的に使用できる。FLAGSTD 指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。

(XOPEN)標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、「STOP 定数」の書き方は明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内ではこの文を使用するべきではない。

一般形式

形式 1

```
STOP { RUN    }
     { 定数-1 }
```

形式 2

```
STOP RUN [ { GIVING    } { [ADDRESS OF] 一意名-1          } ]   (MF)
           { RETURNING } { 整数-1 [SIZE IS 整数-2]        }
```

構文規則

1. 定数-1 は語 ALL で始まる表意定数であってはならない。
2. 定数-1 が数字定数の場合、符号なしの整数とする。

   (OSVS)(VSC2)符号付きの整数を使用できる。

3. (MF)整数-1 は 符号を付けてもよい。
4. 完結文の中の一連の無条件文の中に STOP RUN 文を書く場合、その中の最後の文にする。

   (OSVS)(VSC2)(MF)必ずしも、一連の無条件文の最後に STOP RUN 文を書かなくてもよい。しかし、一連の文の中で STOP RUN 文よりも後ろにあるものは実行されない。

5. (MF)GIVING と RETURNING は同義語であり、どちらを書いてもよい。
6. (MF)一意名-1 は 8 バイト以下のサイズとする。.
7. (MF)STOP RUN GIVING 文が、START 文により作成され (形式 3 スレッド) たスレッド内で実行され、ADDRESS 指定が使用されない場合、一意名-1 は、USAGE POINTER またはサイズが 4 バイトのデータ項目として定義しなければならない。一意名-1 の定義は、START 文中の RETURNING 項目の定義により異なる。 一意名-1 は、USAGE POINTER 項目として定義された場合に関数一意名であってはならない。

一般規則

1. RUN 指定を書くと、実行単位の実行が停止され、制御はオペレーティングシステムに移される。

2. STOP RUN 文を実行中に、実行単位内の開かれている各ファイルに関して、選択指定の何もない CLOSE 文が暗黙的に実行される。それらのファイルに関連する USE 手続きがあっても、実行されない。
3. (MF)STOP RUN 文の実行により、戻り値がシステム領域で設定される。COBOL 以外のランタイム要素は通常、このシステム領域を使用して戻り値を返す。オペレーティングシステムが、動作するプログラムからオペレーティングシステムへ値を返す機能をサポートしている場合、値はシステム領域から返される。

   (MF)GIVING 指定を書かないと、実行単位の動作は下記のようになる。具体的には、システム領域が COBOL の数字データ項目であり、それに USAGE COPM-5 が指定されていて、そのサイズは COBOL システムの外の操作環境によって決められているものとして宣言されている状態で、RETURN-CODE を送出し側としシステム領域を受取り側として、MOVE 文を実行したかのように動作する。(RETURN-CODE 詳細については、*COBOL 言語の概念*の章の*特殊レジスタ*節を参照。)

   (MF)「GIVING 一意名-1」指定を書いた場合、一意名-1 はシステム領域に戻り値を収めるのにちょうど必要な長さであり、オペレーティング・システムによって期待される型と用途のものでなければならない。通常、一意名-1 は PIC S9 (9) USAGE COMP-5 を明示的または暗黙的に指定して宣言する必要がある。一意名-1 を送出し側項目とし、システム領域を受取り側項目として、MOVE 文を実行したかのように、実行単位は動作する。

   (MF)「GIVING 整数-1」指定を指定した場合、整数-1 の値はシステム領域に保持できるよりも大きくてはならない。実行単位は、整数-1 を送出し側としそのシステム領域を受取り側として、MOVE 文を実行したかのように動作する。

4. 「STOP 定数-1」を指定すると、操作員に定数-1 が通知される。実行用プログラムの再開が行われると、次の実行文から処理が継続される。実行キーまたはそれと機能的に等しいキーを押すと、実行が再開される。
5. (MF)START 文または CBL_CREATE_THREAD 登録集ルーチンで作成されたスレッド中で実行されると、STOP RUN GIVING 文は実行単位を終了しない。単に、戻り値を返し、スレッドを終了させる。 以下の文と同じである。

   ```
   CALL 'CBL_THREAD_EXIT' USING BY VALUE ADDRESS OF スレッドパーム
   ```

6. (MF)START 文または CBL_THREAD_CREATE 登録集ルーチンにより作成されていないスレッド (つまり、主なスレッドまたは別のプログラム言語で書かれたプログラムによる作成されたスレッド) 中の STOP RUN は、実行中の CBL_THREAD_CREATE すべてが終了するのを待ち、その後、実行単位を終了する。

16.7 STRING(連結)文

STRING (連結) 文は、いくつかのデータ項目の内容の一部または全部をつなぎ合わせて、1 つのデータ項目を作る。

一般形式

```
STRING {{一意名-1 | 定数-1} ... DELIMITED BY {一意名-2 | 定数-2 | SIZE}} ...
    INTO 一意名-3
    [WITH POINTER 一意名-4]
    [ON OVERFLOW 無条件文-1]
    [NOT ON OVERFLOW 無条件文-2]   (ANS85)
    [END-STRING]
```

構文規則

1. 各定数には、任意の表意定数を使用できる。ただし、ALL を除く。
2. 定数は、すべて文字定数とする。一意名-4 を除く一意名はすべて、明示的または暗黙的に USAGE IS DISPLAY と記述されていなければならない。
3. 一意名-3 は、基本英数字データ項目とする。編集記号または JUSTIFIED 句を伴っていてはならない。
4. 一意名-4 は、整数の基本データ項目とする。その大きさは、一意名-3 によって参照される領域の大きさに、1 を加えた値を収められるほどでなければならない。一意名-4 の PICTURE 文字列には、"P" を含めることはできない。
5. 一意名-1 または一意名-2 が基本数字データ項目である場合、PICTURE 文字列には、"P" が含まれない整数として記述されていなければならない。
6. (ANS85)一意名-3 を、部分参照してはならない。
7. (OS/390)(MF)一意名-3 を、部分参照できる。

一般規則

1. 一意名-1 と定数-1 は、送出し側項目を表わす。一意名-3 は、受取り側項目を表わす。
2. 一意名-2 と定数-2 は、転記の区切り文字を示す。SIZE 指定を書くと、一意名-1 または定数-1 に指定したデータ項目の内容のすべてが転記される。区切り文字として表意定数を指定すると、それは 1 文字として扱われる。
3. 定数-1 または定数-2 として表意定数を指定すると、暗黙的に用途が DISPLAY である 1 文字として扱われる。
4. STRING 文が実行されると、下記の規則に従ってデータが転記される。
    a. 英数字間の転記規則に従って、定数-1 または一意名-1 のデータ項目の内容が一意名-3 に転記される。ただし、空白文字の穴埋めは行われない。(*手続き部 - MERGE - OPEN の章の MOVE(転記)文節を参照。*)
    b. SIZE 指定を書かずに DELIMITED 指定を書くと、一意名-1 によって参照されるデータ項目の内容または定数-1 の値が受取り側データ項目に移送される。その順序は STRING 文に指定されたとおりであり、左端の文字から始まって左から右へ順に進み、データ項目の末尾に達するか、定数-2 に指定された文字が出てくるか、一意名-2 の内容によって指定された文字が出て来た時

点で終了する。定数-2 に指定された文字または一意名-2 によって参照されたデータ項目に指定された文字は移送されない。

c. DELIMITED BY 指定に SIZE と書くと、定数-1 の値または一意名-1 のデータ項目の内容の全体が、一意名-3 のデータ項目に転記される。送出し側のすべてのデータが転記されるか、または一意名-3 のデータ項目の末尾に達した時点で、この転記処理は終了する。

5. POINTER 指定を書くと、プログラマが一意名-4 を明示的に利用できるようになる。この初期値を設定するのは、プログラマの役割である。一意名-4 の初期値は、1 以上とする。
6. POINTER 指定を書かないと、暗黙的に一意名-4 を指定し、その初期値を 1 としたものとして、以下の一般規則が適用される。
7. 送出し側項目から受取り側項目に文字列が転記される処理は、文字が 1 つずつ取り出されて、一意名-3 のデータ項目の内の一意名-4 の値で示される文字位置に送られるように処理される。次の文字が送られる前に、一意名-4 の値が 1 繰り上げられる。一意名-4 の値は、STRING 文の上記の処理以外では変えられることはない。
8. STRING 文の実行が終了した時点で、一意名-3 の内容が変わっているのは、直接的に転記の対象となった部分だけである。一意名-3 のそれ以外の部分には、STRING 文を実行する前のデータ内容が残っている。.
9. 各文字を転記する前の時点で、一意名-4 のデータ項目の値が 1 より小さいか一意名-3 のデータ項目の文字数よりも大きいと、それ以上転記は行われない。この場合、

   (ANS85)NOT ON OVERFLOW 指定はあっても無視されて、

   STRING 文の末尾に制御が移される。ただし、ON OVERFLOW 指定があれば、無条件文-1 に制御が移される。無条件文-1 に制御が移された場合は、その中に記述されている文に関する規則に従って、プログラムの実行が進められる。手続き分岐文あるいは明示的に制御を移行させる条件文が実行されると、その文に関する規則に従って、制御が移される。そうでなければ、無条件文-1 の実行が終わると、STRING 文の末尾に制御が移される。

10. (ANS85)STRING 文の一般規則に従ってデータの転記が行われた後で、上記の一般規則 9 に示した状況が発生していなければ、ON OVERFLOW 指定はあっても無視されて、STRING 文の末尾に制御が移される。ただし、NOT ON OVERFLOW 指定があれば、無条件文-2 に制御が移される。無条件文-2 に制御が移された場合は、その中に記述されている文に関する規則に従って、プログラムの実行が進められる。手続き分岐文あるいは明示的に制御を移行される条件文が実行されると、その文に関する規則に従って、制御が移される。そうでなければ、無条件文-2 の実行が終わると、STRING 文の末尾に制御が移される。
11. (ANS85)END-STRING 指定は STRING 文の範囲を区切る。(*COBOL 言語の概念*の章の *明示範囲符と暗黙範囲符*節を参照。)

16.8 SUBTRACT(減算)文

SUBTRACT (減算) 文は、いくつかの数字データ項目から 1 つの数字データ項目または 2 つ以上の数字データ項目の和を差し引いて、その結果をいくつかのデータ項目に入れる。

一般形式

形式 1

```
SUBTRACT { 一意名-1 }... FROM [一意名-2 [ROUNDED]]...
         { 定数-1   }
    [ON SIZE ERROR 無条件文-1]
    [NOT ON SIZE ERROR 無条件文-2]   (ANS85)
    [END-SUBTRACT]
```

形式 2

```
SUBTRACT { 一意名-1 }... FROM { 一意名-2 }
         { 定数-1   }        { 定数-2   }
GIVING [一意名-3 [ROUNDED]]...
    [ON SIZE ERROR 無条件文-1]
    [NOT ON SIZE ERROR 無条件文-2]   (ANS85)
    [END-SUBTRACT]
```

形式 3

```
SUBTRACT { CORRESPONDING } 一意名-1 FROM 一意名-2 [ROUNDED]
         { CORR          }
    [ON SIZE ERROR 無条件文-1]
    [NOT ON SIZE ERROR 無条件文-2]   (ANS85)
    [END-SUBTRACT]
```

構文規則

すべての形式

1. 各一意名は、数字の基本項目とする。ただし、形式 2 では、語 GIVING の後ろの一意名は、数字編集の基本項目であってもよい。形式 3 の一意名は、集団項目でなければならない。
2. 作用対象を合成したものは、以下の規則に従って決定される。(*手続き部*の章の*算術文*節を参照。)
    a. 形式 1 では、指定したすべての作用対象が作用対象の合成に用いられる。
    b. 形式 2 では、指定したすべての作用対象のうちの語 GIVING の後ろのデータ項目を除いたものが作用対象の合成に用いられる。

c. 形式 3 では、対応するデータ項目の組ごとに別々に作用対象の合成が行われる。

形式 1 および 2

3. 各定数は、数字定数とする。
4. (OSVS)(VSC2)(MF)数字データ項目および定数を使えるところでは、どこでも浮動小数点数のデータ項目および定数を使用できる。

形式 3

5. CORR は、CORRESPONDING の省略形である。

一般規則

すべての形式

1. ***手続き部***の章の *ROUNDED 指定*、*ON SIZE ERROR 指定*、*算術文*、*作用対象の重なり*、*算術文における複数個の答*の各節を参照。
2. (VSC2)(MF)計算過程で意味のある桁が失われることのないように、COBOL システムによって十分な作業領域が確保される。

形式 1

3. 語 FROM の前にあるすべての定数または一意名の和が取られ、一意名-2 の現在の値からその和が差し引かれ、その結果が直ちに一意名-2 に入れられる。語 FROM の後ろに続く各作用対象に対して、それぞれ上記の処理が繰り返される。

形式 2

4. 形式 2 書き方 2 では、語 FROM の前にあるすべての定数または一意名の和が取られ、定数-2 または一意名-2 の現在の値からその和が差し引かれ、その結果が一意名-3 の各データ項目の新しい値として入れられる。

形式 3

5. 形式 3 では、一意名-1 に属するデータ項目が一意名-2 に属する対応するデータ項目から引かれて、その結果が一意名-2 に属する対応するデータ項目に入れられる。

16.9 TRANSFORM(変形)文

(OSVS)

TRANSFORM (変形) 文は、変形規則に従って文字を変更する。

一般形式

```
TRANSFORM 一意名-3 CHARACTERS
    FROM { 表意定数-1 } TO { 表意定数-2 }
         { 文字定数-1 }    { 文字定数-2 }
         { 一意名-1   }    { 一意名-2   }
```

構文規則

1. 一意名-3 は、基本項目でも集団項目でもよい。ただし、用途が DISPLAY 以外の数字基本項目を除く。<
2. 一意名-1 と一意名-2 は、英字または英数字の基本項目にする。
3. 文字定数-2 または一意名-2 の長さは、1 文字であるか、または文字定数-1 または一意名-1 と同じにする。

一般規則

1. 表意定数-1 または表意定数-2 を使用することは、同じ値の 1 文字の文字定数を使用することと等しい。
2. 一意名-1 または一意名-2 が、一意名-3 と同じ記憶領域を指していると、TRANSFORM 処理の結果はどうなるかわからない。(*データ部 - ファイルおよびデータ記述*の章の *REDEFINES(再定義)句*節を参照。)
3. 文字定数-1 または一意名-1 の中で文字が繰り返されると、TRANSFORM 処理の結果はどうなるかわからない。
4. TRANSFORM 文が実行されると、一意名-3 の内容が調べられて、一意名-1 または文字定数-1 に指定した文字が探される。該当する文字が見つかると、一意名-2 または文字定数-2 の中の対応する文字 (または 1 文字からなる項目の場合はその文字) によって、一意名-3 のその部分が置き換えられる。一意名-1 または文字定数-1 と、一意名-2 または文字定数-2 との対応関係は、データ項目内の文字位置によって付けられる(左から始まる)。

16.10 UNLOCK(解錠)文

(MF)

UNLOCK 文は、実行単位によってロック (施錠) されている、指定されたファイル中のすべてのレコードのロックを解除 (開錠) する。

一般形式

```
UNLOCK ファイル名 { RECORD  }
                  { RECORDS }
```

構文規則

ファイル名は、ファイル管理記述項の SELECT 文中に定義しておく。

一般規則

1. ファイル名のファイルは、OPEN 文によって既に開かれていること。
2. UNLOCK 文は、実行単位によってロックされている、指定されたファイル中のすべてのレコードのロックを解除する。

16.11 UNSTRING(分解)文

UNSTRING (分解) 文は、送出し側項目の連続するデータを分解していくつかの受取り側項目に入れる。

一般形式

```
UNSTRING 一意名-1

  [DELIMITED BY [ALL] { 一意名-2 } [ OR [ALL] { 一意名-3 } ]...]
                      { 定数-1   }            { 定数-2   }

      INTO {一意名-4 [DELIMITER IN 一意名-5] [COUNT IN 一意名-6]}...
      [WITH POINTER 一意名-7]
      [TALLYING IN 一意名-8]
      [ON OVERFLOW 無条件文-1]

      [NOT ON OVERFLOW 無条件文-2]
      [END-UNSTRING]                    (ANS85)
```

構文規則

1. 各定数は文字定数とする。各定数は表意定数であってもよい。ただし、ALL は除く。
2. 一意名-1、一意名-2、一意名-3、一意名-5 は、明示的または暗黙的に、項類が英数字のデータ項目として記述されていなければならない。
3. 一意名-4 は、英字でも英数字でも数字でもよい。ただし、英字の場合、PICTURE 文字列の中に記号文字"B" を含めることはできない。数字の場合は、PICTURE 文字列の中に記号文字"P" を含めることはできない。一意名-4 の用途は、DISPLAY とする。
4. (MF) 構文規則 2 および 3 は適用しない。代わって、下記の規則を適用する。
   a. 一意名-1 は、英数字とする。
   b. 一意名-2 と一意名-3 は、用途を DISPLAY とし、編集されていてはならない。
   c. 一意名-5 の用途は、DISPLAY とする。
   d. 一意名-4 の用途は、DISPLAY にしてもよい。
   e. 一意名-4 は、数字として定義されており正しく転記できるならば、どのような用途のものであってもよい。
5. (OSVS)(VSC2)(MF) 一意名-4 は、浮動小数点数項目であってはならない。
6. 一意名-6 と一意名-8 は、整数項目とする。この PICTURE 文字列の中に、記号文字" P" を含めることはできない。

7. 一意名-7 は整数の基本項目とし、一意名-1 のデータ項目の文字数に 1 を加えた値をもつことができなければならない。この PICTURE 文字列の中に、記号文字"P" を含めることはできない。
8. どの一意名も、88 レベルの記述項を指すことはできない。
9. DELIMITER IN 指定および COUNT IN 指定は、DELIMITED BY 指定を書いたときだけ指定できる。
10. 一意名-1 は部分参照できない。
11. (OS/390)(COBOL)(MF)一意名-1 は部分参照できる。

一般規則

1. 一意名-2 と定数-1 に関する記述はすべて、それぞれ一意名-3 と定数-2 およびそれらを反復指定したものに、等しく当てはまる。
2. 一意名-1 は、送出し側を表わす。
3. 一意名-4 は、受取り側を表わす。一意名-5 は、区切り文字用の受取り側を表わす。
4. 一意名-2 が参照するデータ項目または定数-1 は、区切り文字を指定する。
5. 一意名-6 のデータ項目は、一意名-1 のデータ項目のうちで、区切り文字によって分離されて一意名-4 のデータ項目に転記された文字数を表わす。区切り文字は、この文字数に入らない。
6. 一意名-7 のデータ項目は、一意名-1 のデータ項目中の相対文字位置を表わす。
7. 一意名-8 のデータ項目は、UNSTRING 文の実行中に処理の対象となった受取り側データ項目の数を記録するカウンタである。
8. 区切り文字に表意定数を使用した場合、1 文字の文字定数として扱われる。

   ALL 指定を書くと、定数-1 (文字定数または表意定数) または一意名-2 のデータ項目の内容が一意名-1 の文字列中に連続して何回か現れても、それが 1 つのものとして扱われる。そして、一般規則 13d の規則に従って、その内容が受取り側に転記される。

9. 検査中に区切り文字が 2 つ連続して検出されると、このときの受取り側の項目はそのデータ記述に従って、英字または英数字ならば空白で、数字ならばゼロで満たされる。
10. 一意名-2 が参照するデータ項目の内容、または定数-1 には、計算機文字集合中の任意の文字を含めることができる。
11. 一意名-2 が参照するデータ項目または定数-1 はそれぞれ、1 つの区切り文字を表わす。1 つの区切り文字が何文字かで構成されている場合、送出し側の文字列の中で区切り文字として認識されるためには、まったく同じに並んでいなければならない。
12. DELIMITED BY 指定に複数の区切り文字を書くと、各区切り文字は論理和条件"OR" で結ばれる。各区切り文字が、送出し側項目中の文字列と比較される。両者が一致すると、送出し側項目中の文字列のその部分が、1 つの区切り文字とみなされる。送出し側項目中のどの文字も、2 つ以上の区切り文字の一部とみなされることはない。

    各分離符は UNSTRING 文に指定された順序で送出し側フィールドに適用される。

13. UNSTRING 文の実行が開始されると、下記の規則に従って、送出し側の一意名-1 のデータ項目から受取り側の一意名-4 のデータ項目に、データが転記される。

a. POINTER 指定を書くと、一意名-7 のデータ項目の内容によって示される相対文字位置から、一意名-1 のデータ項目の文字列の検査が開始される。POINTER 指定を書かないと、一意名-1 の左端の文字位置から、データ項目の文字列の検査が開始される。
b. DELIMITED BY 指定を書くと、送出し側の文字列の検査は左から右に進められて、定数-1 または一意名-2 のデータ項目によって示される区切り文字が出てきたところで終了する (一般規則 11 を参照) 。DELIMITED BY 指定を書かないと、検査される文字数はそのときの受取り側の大きさと等しくなる。しかし、受取り側の項目の符号が独立であると定義されている場合は、検査される文字数はそのときの受取り側の大きさよりも 1 つ小さくなる。

   区切り文字が出てくる前に一意名-1 のデータ項目の末尾に達すると、最後の文字が検査された時点で検査は終了する。

c. 検査の結果、対象と判定された文字列 (区切り文字が指定されている場合それは含まれない) が、MOVE 文の規則に従って、英数字の基本項目として、このときの受取り側に転記される。(*手続き部 - MERGE - OPEN* の章の *MOVE(転記)文*節を参照。)
d. DELIMITER IN 指定を書くと、区切り文字 (列) が英数字の基本データ項目として扱われて、MOVE 文の規則に従って一意名-5 のデータ項目に転記される。(*手続き部 - MERGE - OPEN* の章の *MOVE(転記)文*節を参照。) ただし、送出し側の文字列の検査が一意名-1 のデータ項目の末尾に達した場合は、一意名-5 のデータ項目は空白で埋められる。
e. COUNT IN 指定を書くと、検査の対象となった文字の数 (区切り文字が指定されている場合、それは含まれない) が、基本項目転記の規則に従って、一意名-6 のデータ項目に転記される。
f. DELIMITED BY 指定を書くと、文字列の検査は区切り文字のすぐ右側の文字から再開される。DELIMITED BY 指定を書かないと、文字列の検査は転記の済んだ部分のすぐ右側の文字から再開される。
g. 一意名-4 のデータ項目にデータが転記されると、一意名-4 の次の繰り返しデータ項目が新たな受取り側となる。そして、上記の 13b から 13f の処理が繰り返される。一意名-1 のデータ項目中の文字がすべて検査され終わるか、または受取り側の項目がなくなった時点で、この繰り返し処理は終了する。

14. POINTER 指定または TALLYING 指定に関連するデータ項目の内容を初期化しておくことは、プログラマの役割である。
15. 一意名-7 のデータ項目の内容は、一意名-1 のデータ項目中で各文字が検査されるごとに、1 繰り上げられる。POINTER 指定のある UNSTRING 文の実行が終了した時点では、一意名-7 のデータ項目の値は、その初期値に一意名-1 のデータ項目の検査の対象となった文字数を加えたものになっている。
16. TALLYING 指定のある UNSTRING 文の実行が終了した時点では、一意名-8 のデータ項目の値は、その初期値に処理の対象となった受取り側項目の数を加えたものになっている。
17. 下記のどちらかの場合には、あふれ条件が発生する。
    a. UNSTRING 文の実行開始時に、一意名-7 のデータ項目の値が 1 より小さいか、または一意名-1 のデータ項目の大きさよりも大きい場合

b. UNSTRING 文の実行中に、受取り側がすべて使われたのに、一意名-1 のデータ項目中にまだ検査されていない文字が残っている場合

18. あふれ条件が発生したときの UNSTRING 文の終了処理は、次のようになる。

    (ANS85)NOT ON OVERFLOW 指定がある場合は、その指定は無視されて、

    UNSTRING 文の末尾に制御が移される。ON OVERFLOW 指定がある場合は、無条件文-1 に制御が移される。無条件文-1 に制御が移された場合は、その中に記述されている文に関する規則に従って、プログラムの実行が進められる。手続き分岐文あるいは明示的に制御を移行させる条件文が実行されると、その文に関する規則に従って、制御が移される。そうでなければ、無条件文-1 の実行が終わると、UNSTRING 文の末尾に制御が移される。

19. (ANS85)END-UNSTRING 指定は、UNSTRING 文の範囲を区切る。(*COBOL 言語の概念*の章の*明示範囲符と暗黙範囲符*節を参照。)
20. UNSTRING 文の実行中に一般規則 17 に記述した条件が発生しなかった場合、他の一般規則に従ってデータの転記が済んだ後、UNSTRING 文の末尾に制御が移される。ON OVERFLOW 指定は、あっても無視される。

    (ANS85)NOT ON OVERFLOW 指定がある場合は、無条件文-2 に制御が移される。無条件文-2 に制御が移された場合は、その中に記述されている文に関する規則に従って、プログラムの実行が進められる。手続き分岐文あるいは明示的に制御を移行させる条件文が実行されると、その文に関する規則に従って、制御が移される。そうでなければ、無条件文-2 の実行が終わると、UNSTRING の末尾に制御が移される。

21. 一意名に付けられている添字および指標は下記のように評価される。
    a. 一意名-1 と一意名-7 と一意名-8 に関する添字または指標があれば、UNSTRING 文を実行した結果としてデータの転記が行われる直前に、1 回だけ評価される。
    b. 一意名-2、一意名-3、一意名-4、一意名-5、一意名-6 に関する添字または指標があれば、それぞれのデータ項目にデータの転記が行われる直前に評価される。
    c. (ANS85)DELIMITED BY, INTO, DELIMITER IN, COUNT IN の各指定に関連する一意名に添字付けがされている場合、区切り文字を探して送出し側項目を検査する直前に 1 回だけ添字が評価される。
22. 一意名-1、一意名-2、一意名-3 のどれかが、一意名-4、一意名-5、一意名-6、一意名-7、一意名-8 のどれかと同じ記憶領域を共有する場合、または一意名-4、一意名-5、一意名-6 のどれかが、一意名-7、一意名-8 のどれかと同じ記憶領域を共有する場合、または一意名-7 と一意名-8 が同じ記憶領域を共有する場合、UNSTRING 文の実行結果はどうなるかわからない。これらのデータ項目が同じデータ記述に項定義されていたとしても、そのことは変わらない。

16.12 USE(使用)文

USE (使用) 文は、入出力誤りを処理するための手続きを指定する。この手続きは、入出力管理システムによって行われる標準の手続きに、更に追加して行うものである。

一般形式

形式 1 (順ファイル、相対ファイルおよび索引ファイル)

```
USE [GLOBAL] AFTER STANDARD { EXCEPTION }
    (ANSI85)                { ERROR     }
                 PROCEDURE ON { [ファイル名-1]... }
                              { INPUT            }
                              { OUTPUT           }
                              { I-O              }
                              { EXTEND           }
    [GIVING データ名-1 [データ名-2]] .
                (DSWS)
```

形式 2 (レコード順ファイル)

```
USE [GLOBAL] AFTER STANDARD [BEGINNING] [FILE]   (DSWS)
                            [ENDING]    [REEL]   (VSC2)
                            [           [UNIT]   ]
                 LABEL PROCEDURE ON { [ファイル名-1]... }
                                    { OUTPUT           }
                                    { INPUT            }
                                    { I-O              }
                                    { EXTEND           } .
```

形式 3 (相対ファイルおよび索引ファイル)

```
USE [GLOBAL] AFTER STANDARD [BEGINNING] [FILE]   (DSWS)
                            [ENDING]            (VSC2)
                            [           ]
                 LABEL PROCEDURE ON { [ファイル名-1]... }
                                    { OUTPUT           }
                                    { INPUT            }
                                    { I-O              }
                                    { EXTEND           } .
```

構文規則

すべての形式 (すべてのファイル)

1. 形式 1 は、誤りに関する宣言である。

OSVS VSC2 形式 2 および 3 ラベルに関する宣言である。

2. USE 文を記述する場合は、宣言節の見出しの直後から書き始め、終止符 (.) に続く空白で止める。.
3. USE 文自体は、実行されることはなく、USE 手続きの実行を求める条件を定義するだけである。
4. USE 文の中で明示的または暗黙的に参照されるファイルは、すべて同じ編成または呼出し法のものである必要はない。

形式 1 (順ファイル、相対ファイルおよび索引ファイル)

5. この形式に従った別々の形の記述中に、同じファイル名が現れてもよい。USE 文中に指定したファイル名によって、複数の USE 手続きの実行が同時に要求されるようなことがあってはならない。

   ANS85 同じ手続き部内の複数の USE AFTER EXCEPTION 手続き中に、同じファイル名が現れてはならない。

6. ERROR と EXCEPTION は同義語であり、どちらを使用してもよい。

形式 2 および 3 (レコード順ファイル、相対ファイルおよび索引ファイル)

7. OSVS VSC2 BEGINNING と ENDING の指定を両方とも省略すると、BEGINNING と ENDING が共に指定されたものとみなされる。

形式 2 (レコード順ファイル)

8. OSVS VSC2 REEL と UNIT は同義語として扱われる。
9. OSVS VSC2 FILE と REEL/UNIT の指定を両方とも省略すると、FILE と REEL/UNIT が共に指定されたものとみなされる。
10. OSVS VSC2 BEGINNING/ENDING と FILE/REEL の下記の組合わせのそれぞれに対して、任意の 1 つのファイル名および任意の 1 つの OPEN モードを指定できる。

    BEGINNING FILE
    BEGINNING REEL/UNIT
    ENDING FILE
    ENDING REEL/UNIT

一般規則

すべての形式 (すべてのファイル)

1. USE 手続きの実行が終わると、呼出し元の手順に制御が戻される。
2. USE 手続き内では、非宣言手続きを参照してはならない。逆に、非宣言部分では、宣言部分中の手続き名を参照してはならない。ただし、PERFORM 文で USE 文を参照することはできる。

OSVS VSC2 MF この制限は無視してもよい。

3. 呼び出されたがまだ呼び出し元の手順に制御が戻されていない USE 手続きを実行させるような文を、実行してはならない。

形式 1 (順ファイル、相対ファイルおよび索引ファイル)

4. 指定した手続きは、標準入出力誤り手順が終了した後で、または入出力文中に AT END 指定をしなかった場合は AT END 条件が検出された時点で、入出力システムによって実行される。
5. ANS85 ファイル名-1 を明示的に指定した場合は、他の USE 手続きはファイル名-1 に適用されない。
6. GIVING は注記の目的だけに使用する。

形式 2 および 3 (レコード順ファイル、相対ファイルおよび索引ファイル)

7. OSVS VSC2 BEGINNING 指定を明示的または暗黙的に指定すると、適用可能な OPEN 文の実行中に、下記の措置が取られる。

Open 文のモード	措置
INPUT	1. 見出しラベルを読む 2. 開始宣言を実行する
OUTPUT	3. 開始宣言を実行する 4. 見出しラベルを書く
I/O	5. 見出しラベルを読む 6. 開始宣言を実行する 7. 見出しラベルを書く
EXTEND	8. 見出しラベルを読む 9. 開始宣言を実行する (終わりファイル・ラベルは見出しとして扱われる) 10. 見出しラベルを書く

8. OSVS VSC2 ENDING 指定を明示的または暗黙的に指定すると、適用可能な CLOSE 文の実行中に、下記の措置が取られる。

Open 文のモード	措置
INPUT	1. 終わりファイル・ラベルを読む

	2.	終了宣言を実行する
OUTPUT	3.	終了宣言を実行する
	4.	終わりファイル・ラベルを書く
I/O	5.	終わりファイル・ラベルを読む
	6.	終了宣言を実行する
	7.	終わりファイル・ラベルを書く
EXTEND	8.	終了宣言を実行する
	9.	終わりファイル・ラベルを書く

9. OSVS OSC2 GO TO MORE-LABELS 文はそれが現れる宣言手続きの先頭への単純なジャンプとして扱われる。

16.13 WAIT(待機)文

MF

WAIT (待機) 文は、現在のスレッドの実行を、目的のスレッドの実行が終了するまで保留する。また、場合によっては、目的のスレッドの実行により返された値を取得するまで保留する。

WAIT 文は、イベントが真となるまで実行を保留する。

一般形式

形式 1 (スレッド)

```
WAIT FOR スレッドポインター1
   [ RETURNING INTO 一意名-1 ]
   [ STATUS IS 一意名-2 ]
   [ ON EXCEPTION 無条件文-1          ]
   [ NOT ON EXCEPTION 無条件文-2      ]
   [ END-WAIT ]
```

形式 2 (イベント)

WAIT FORイベントポインター1

構文規則

形式 1（スレッド）

1. スレッドポインタ-1 は USAGE THREAD-POINTER として定義しなければならない。
2. 一意名-1 は、USAGE POINTER として定義されるか、サイズが 4 バイトでなければならない。一意名-1 は関数一意名であってはならない。
3. 一意名-2 は、4 桁以上の長さの整数データ項目として定義しなければならない。

形式 2（イベント）

4. イベントポインタ-1 は USAGE EVENT-POINTER として定義しなければならない。

一般規則

形式 1（スレッド）

1. スレッドポインタ-1 は、IDENTIFIED BY 指定を含む START 文または、CBL_THREAD_登録集から返されたスレッドハンドルを含んでいなければならない。
2. WAIT 文が正常に動作すると、スレッドポインタ-1 に含まれるスレッドハンドルが参照するスレッドが終了するまで、現在のスレッドが実行されない。
3. RETURNING 指定を使用すると、正常に動作する WAIT 文により、一意名-1 の値が返される。
4. WAIT 文が正常に動作した後は、スレッドポインタ-1 に含まれるスレッドハンドルは無効となる。
5. STATUS 指定を使用すると、WAIT 文の実行により、CBL_THREAD_登録集ルーチンに対して指定された戻り符号のどれかが、一意名-4 中に配置される。
6. WAIT 文が正常に動作しなかった場合、一意名-2 が指定されていれば、そこに CBL_THREAD_登録集ルーチンに対して指定された戻り符号のどれかが配置される。さらに、以下のいずれかが行われる。
    a. ON EXCEPTION 指定を使用した場合、制御が無条件文-1 に渡される。その後、無条件文-1 で指定された各文の規則に従って実行が継続される。手続きの分岐、または制御の移動を明示的に指示する条件文が実行された場合は、制御が文の規則に従って渡される。そうでない場合は、無条件文-1 の実行が完了すると同時に、WAIT 文の最後に制御が渡され、NOT ON EXCEPTION 指定が使用されていても無視される。
    b. 上記以外の場合で、NOT ON EXCEPTION 指定または STATUS 指定を使用した場合は、WAIT 文の最後に制御が渡され、NOT ON EXCEPTION 指定が使用されていても無視される。
    c. 上記以外の場合は、実行時システムエラーとともに実行単位が終了する。
7. WAIT 文が正常に実行された場合は、以下が記載された順序通りに行われる。

a. STATUS 指定を使用した場合は、ZERO が一意名-2 に転記される。
b. NOT ON EXCEPTION 指定を使用した場合は、制御が無条件文-2 に渡される。 その後、無条件文-2 で指定された各文の規則に従って、実行が続行される。手続きの分岐、または制御の移動を明示的に指示する条件文が実行された場合は、制御が文の規則に従って渡される。そうでない場合は、無条件文-2 の実行が完了すると同時に、WAIT 文の最後に制御が渡され、ON EXCEPTION 指定が使用されていても無視される。

形式 2 (イベント)

8. WAIT 文が正常に動作すると、イベントポインタ-1 が参照するイベントが真となるまで、実行が保留される。

注:文が実行されたとき、イベントポインタ-1 が参照するイベントが真であった場合は、実行がただちに再開される。

---

9. WAIT 文が正常に動作しなかった場合、実行時システムエラーとともに実行単位が終了する。

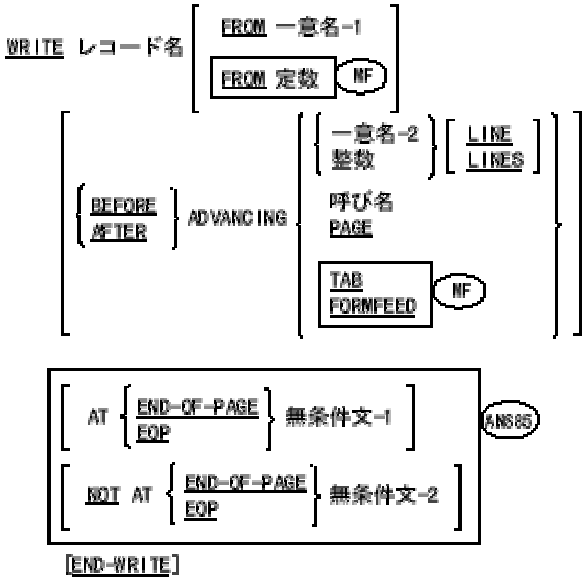
16.14 WRITE(書き出し)文

WRITE (書き出し) 文は、論理レコードを出力ファイルまたは入出力両用ファイルに書き出す。順ファイルに関しては、論理ページ内での縦方向の行の位置付けにも WRITE 文を使用できる。

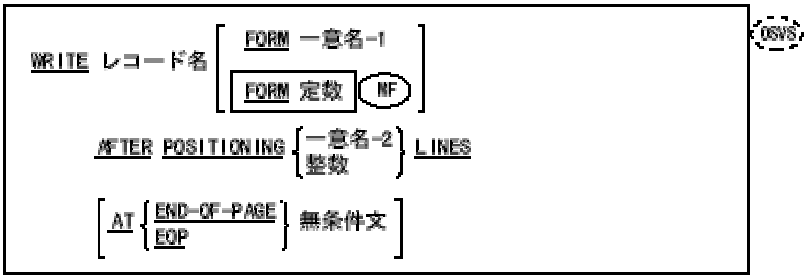
XOPEN 標準 COBOL 定義の一部を構成するにもかかわらず、X/Open の COBOL 言語定義では、ADVANCING 指定中の呼び名は明示的に除外されている。したがって、X/Open の COBOL に準拠する原始プログラム内では呼び名を使用するべきではない。

一般形式

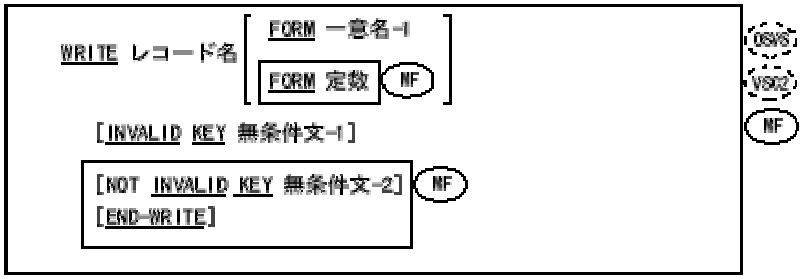
**形式 1 (レコード順ファイル
MF および行順ファイル)**



形式 2 (レコード順ファイル)



形式 3 (レコード順ファイル)



形式 4 (相対ファイルおよび索引ファイル)

```
WRITE レコード名 [ FORM 一意名-1 ]
                 [ FORM 定数 (MF) ]
[INVALID KEY 無条件文-1]
[NOT INVALID KEY 無条件文-2]  (ANS85)
[END-WRITE]
```

指令とランタイム･スイッチ

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - WRITE-LOCK - 複数のレコード･ロックを使用しているときに、WRITE 文にレコード･ロックを取得させる
    - FDCLEAR - WRITE 文の後で、レコード領域の内容を予知できるようにする。
2. 下記のランタイム･スイッチによって、この項に記述した意味が影響を受ける可能性がある。
    - N - 行順レコードを書くときに、制御文字の前への空文字の挿入を制御する。
    - T - 行順レコードを書くときに、タブ文字の挿入を制御する。

構文規則

すべての形式 (すべてのファイル)

1. (ANS85)一意名-1 が関数一意名である場合は、英数字関数を参照すること。一意名-1 が関数一意名でない場合は、レコード名と一意名-1 が同じ記憶領域を指してはならない。
2. レコード名は、データ部のファイル節内の論理レコードの名前であり、修飾してもよい。
3. (OSVS)(VSC2)(MF)レコード名に、浮動小数点数項目を指定してもよい。
4. (OSVS)(VSC2)(MF)一意名-1 は、浮動小数点数項目として定義してもよい。

形式 1 (レコード順ファイル)

5. (MF)特殊名段落に記述すると、TAB の代わりに、利用者が定義した呼び名を使用できる。(*環境部*の章の*特殊名段落*節を参照。)
6. ADVANCING 指定中に一意名-2 を使用する場合、一意名-2 は基本整数データ項目の名前とする。
7. 整数または一意名-2 のデータ項目の値は、ゼロであってもよい。
8. (VSC2)整数は符号付きでもよい。
9. END-OF-PAGE 指定をした場合、該当するファイルのファイル記述項中に LINAGE 句を指定する。
10. 語、END-OF-PAGE と EOP は同義語である。
11. ファイル記述項に LINAGE 句を指定したファイルにレコードを書き出すときは、ADVANCING 呼び名指定をすることはできない。
12. (ANS85)1 つの WRITE 文中に、ADVANCING PAGE と END-OF-PAGE を共に指定してはならない。

(MF)この制限は無視してよい。

(OSVS)(VSC2)(MF)呼び名の代わりに、それに相当する機能名を使用してもよい。

形式 2 (レコード順ファイル)

13. (OSVS)ファイル記述項に LINAGE 句を指定したファイルにレコードを書き出すときは、この形式は使用できない。
14. (OSVS)あるファイルにレコードを書き出すためにこの形式の WRITE 文を使用する場合、そのファイルに適用する WRITE 文はすべてこの形式にする。
15. (OSVS)AFTER POSITIONING 指定に使用する一意名-2 は、1 文字の英数字項目として定義しておく。(指定できる値については、一般規則 18 を参照)

形式 4 (相対ファイルおよび索引ファイル)

16. 該当するファイルに適用できる USE 手続きを指定しなかった場合、INVALID KEY 指定をする。

    (OSVS)(VSC2)(MF)この規則は強制しない。

一般規則

すべての形式 (すべてのファイル)

1. FROM 指定をした WRITE 文を実行することは、下記の順に処理を行うことと同じである
    a. MOVE 文の規則に従って、下記の文を実行する。

        MOVE 一意名-1 TO レコード名

    b. FROM 指定をしない同じ WRITE 文を実行する。

    暗黙の MOVE 文を実行する前のレコード領域の内容は、この WRITE 文の実行には影響を及ぼさない。

    WRITE 文の実行が終了した後で、レコード領域のデータは利用できなくなることがあるが、一意名-1 のデータはそのまま残っている。(一般規則 13 を参照)

2. ファイル位置指示子は、WRITE 文の実行の影響を受けない。
3. WRITE 文を実行すると、その対象のファイルに対応する FILE STATUS データ項目を指定してあれば、それが更新される。(***手続き部***の*入出力状態*節を参照。)
4. ファイル中のレコードの最大の大きさは、ファイルを作成したときに決定される。この大きさを後で変更してはならない。
5. ファイル中の論理レコードを格納する大記憶装置上の文字位置の数は、そのレコードの論理記述に定義した文字位置の数と、等しくてもよいし異なっていてもよい。
6. WRITE 文が実行されると、論理レコードが書き出される。

7. (ANS85)WRITE 文の実行が不成功に終わると、レコード名に対応するファイルの入出力状態が更新される。次いで、そのファイルに適用できる USE AFTER EXCEPTION 手続きが指定してあればその手続きが実行され、その後で USE 文の規則に従って制御が移される。(前述の *USE(使用)文*節を参照。)

形式 1 (レコード順ファイル)

8. (MF)TAB に関連づけられた呼び名が指定されている場合、用紙の標準の垂直タブ位置へと移動する結果となる。
9. END-OF-PAGE 指定をした WRITE 文の実行中に印刷プログラムの論理的な末尾に達すると、END-OF-PAGE 指定中の無条件文-1 が実行される。論理ページの大きさは、該当ファイルに関するファイル記述中の、該当レコードの LINAGE 句に指定する。
10. LINAGE 句を指定してあると、WRITE 文を実行した結果 LINAGE 句の整数-2 またはデータ名-2 のデータ項目に定義した値以上に LINAGE-COUNTER の値が達すると、END-OF-PAGE 指定をした WRITE 文が実行されたときにページ終了条件が発生する。この場合、書き出し操作に続いて、その WRITE 文の END-OF-PAGE 指定中の無条件文が実行される。

    WRITE 文 (END-OF-PAGE 指定があってもなくても) の実行結果が現在のページ本体に収まりきらないときは、自動的にページあふれ条件が発生する。

    ページあふれ条件とは、WRITE 文を実行すると、LINAGE 句の整数-2 に指定した値またはデータ名-2 に指定したデータ項目の値を LINAGE-COUNTER の値が超えることである。ページあふれが発生する場合、レコードの書き出しと改ページのどちらを先にするかは、BEFORE と AFTER のどちらを指定したかによる。改ページされると、装置は LINAGE 句の指定に従って、次の論理ページの最初の行に位置付け直される。END-OF-PAGE 句中に無条件文を指定してあると、レコードの書き出しと装置の位置付け直しが済んだ後で、その無条件文が実行される。

    LINAGE 句に整数-2 もデータ名-2 にも指定していないと、ページあふれ条件とは別にページ終了条件が発生することはない。この場合、ページあふれ条件とページ終了条件は同時に発生する。

    LINAGE 句に整数-2 またはデータ名-2 を指定してある場合で、WRITE 文を実行したときに、LINAGE 句の整数-2 に指定した値またはデータ名-2 に指定したデータ項目の値、および LINAGE 句の整数-1 に指定した値またはデータ名-1 に指定したデータ項目の値の両方を、LINAGE-COUNTER の値が同時に超えるときは、整数-2 またはデータ名-2 は指定されてないように処理は進められる。

形式 1 (行順ファイル)

10. (MF)ADVANCING 指定をしないと、使用しているオペレーティングシステムのテキスト･エディタの方式に従って、自動的に 1 行が行送りされる (通常は BEFORE ADVANCING 1 LINE が想定される) 。

11. (MF)順ファイルの外部的に定義されている境界を超えて書き出しを行おうとすると、例外条件が発生する。このとき、レコード領域の内容は影響を受けず、下記の例外処理が行われる。
    a. 該当するファイルに対応する FILE STATUS データ項目を指定してあると、その値が区域外書き出しを表わすように設定される。(*手続き部*の*入出力状態*節を参照。)
    b. 該当するファイルに適応できる USE AFTER STANDARD EXCEPTION 宣言を明示的または暗黙的に指定してあると、その宣言手続きが実行される。
    c. 該当するファイルに適応できる USE AFTER STANDARD EXCEPTION 宣言を明示的にも暗黙的にも指定してないと、結果はどうなるかわからない。
12. (MF)複数の物理リール/ユニットにまたがる出力ファイルのリール/ユニットの終わりが検出されると、WRITE 文は下記の操作を行う。
    a. 標準のリール/ユニット終了手続き
    b. リール/ユニットの交換手続き
    c. 標準のリール/ユニット開始手続き
13. (MF)(OPEN)レコード長が固定のファイルに長さの異なるレコードを再定義して収めている場合、バッファ領域全体がファイルに書き込まれることを知っている必要がある。したがって、現在のレコードが前のレコードよりも短い場合には、空白を埋める必要がありうる。
14. (ANS85)NOT END-OF-PAGE 指定をした WRITE 文の実行が正常に進み、その間にページ終了条件が発生しないと、入出力処理の後で制御は無条件文-2 に移される。

形式 1 (レコード順ファイル
(MF)および行順ファイル)

15. ADVANCING 指定および END-OF-PAGE 指定は、共に印刷ページ上での各行の縦方向の位置を制御する働きをする。

    ADVANCING 指定をしないと、リスト装置 (PRINTER または PRINTER-1) に向けて出力されたときに、AFTER ADVANCING 1 LINE を指定してあったかのように、自動的に行送りされる。ADVANCING 指定をすると、下記のように行送りされる。

    a. 一意名-2 を指定すると、その現在の値に等しい行数だけ、印刷ページ上で行送りされる。
    b. 整数を指定すると、その値に等しい行数だけ、印刷ページ上で行送りされる。
    c. 呼び名を指定すると、*環境部*の章の*特殊名段落*節に指定した内容に従って、行送りされる。
    d. BEFORE 指定を使用すると、該当する行が出力された後に、上記の a, b, c の規則に従って行送りされる。
    e. AFTER 指定を使用すると、上記の a, b, c の規則に従って行送りされた後で、該当する行が出力される。
    f. PAGE を指定すると、装置が (使用された指定により) 前または次の論理ページに位置付け直される。該当する行の出力と改ページのファイル記述項中に LINAGE 句を指定してあるレコード順ファイルにレコードを書き出すと、LINAGE 句の指定に従って、装置は次の論理ページ上に書き出し可能な最初の行に位置付け直される。

16. 1 つの WRITE 文中に、ADVANCING と END-OF-PAGE を共に指定してはならない。

形式 1、2、および 3 (順ファイル)

17. WRITE 文を実行するときは、対象ファイルを OUTPUT モードまたは EXTEND モードで開いておく。(*手続き部 - MERGE - OPEN* の章の *OPEN(開く)文*節を参照。)
18. WRITE 文の実行によって書き出された論理レコードは、レコード領域内で利用できなくなる。ただし、SAME RECORD AREA 句中に該当するファイルを指定してある場合、または区域外書き出しのために WRITE 文の実行が不成功に終わった場合は、そのレコード領域は元のまま残っている。

    SAME RECORD AREA 句を指定してある場合、論理レコードは レコード名に対応するファイルのものとしてだけではなく、SAME RECORD AREA 句中に指定した他のファイルのレコードとしても使用できる。

形式 2 (レコード順ファイル)

19. OS/VS WRITE 文中に AFTER POSITIONING 指定をすると、システムはレコードの先頭の文字位置に適切な文字を転記してから、レコードをファイルに書き出す。このため、この先頭の文字位置は空けておくこと。AFTER POSITIONING に一意名-2 を指定をすると、その値がレコードの先頭の文字位置に転記される。その値は下記のどれかとする。

一意名-2	意味
(空白)	1 行改行
0	2 行改行
-	3 行改行
+	改行なし
1-9	それぞれ、チャネル 1-9 へ飛ぶ
A, B, C	それぞれ、チャネル 10, 11, 12 へ飛ぶ
V, W	ポケット 1 または 2 を選択

20. AFTER POSITIONING に整数を指定をすると、出力レコードの先頭に転記される文字は下記のようになる。

整数	出力文字	意味
0	1	チャネル 1 へ飛ぶ

1	(空白)	1 行改行
2	0	2 行改行
3	-	3 行改行

21. OSVS END-OF-PAGE 指定は、指定しても注記になり、実行されることはない。

形式 3 (レコード順ファイル)

21. OSVS VSC2 MF 順ファイルの外部的に定義されている境界を超えて書き出しを行おうとすると、無効キー条件が発生する。COBOL システムによって無効キー条件が検出されると、WRITE 文の実行は不成功に終わる。このとき、レコード領域の内容は影響を受けない。該当するファイルに対応する FILE STATUS データ項目を指定してあると、その原因を表わす値がその項目に設定される。実行 は、無効キー条件に関する規則に従って続行される。(*手続き部*の章の*入出力状態*節を参照。)

形式 4 (相対ファイルおよび索引ファイル)

22. WRITE 文を実行するとき、対象ファイルを OUTPUT, I-O, EXTEND のどれかのモードで開いておく。索引ファイルを順呼出し法で処理する場合、 入出力モードで開いてはならない。(*環境部*の章の*ファイル管理記述項*および*手続き部 - MERGE - OPEN の章*の章の *OPEN(開く)文*節を参照。)
23. WRITE 文の実行によって書き出された論理レコードは、SAME RECORD AREA 句中に該当するファイルを指定してある場合、または無効キー条件のために WRITE 文の実行が不成功に終わった場合にだけ、引続きレコード領域内のものが利用できる。

    SAME RECORD AREA 句を指定してある場合、論理レコードは、 レコード名に対応するファイルのものとしてだけではなく、SAME RECORD AREA 句中に指定した他のファイルのレコードとしても使用できる。

24. 無効キー条件が検出されると、WRITE 文の実行は不成功に終わる。このとき、レコード領域の内容は影響を受けない。該当するファイルに対応する FILE STATUS データ項目を指定してあると、その原因を表わす値がその項目に設定される。 実行は、 無効キー条件に関する規則に従って続行される。(*手続き部*の章の*入出力状態*節を参照。)

形式 4 (相対ファイル)

25. 出力モードで開いたファイルに関しては、下記のどちらかの方法でレコードを書き出すことができる。
    a. 順呼出し法を採る場合、WRITE 文によってレコードが書き出される。最初に書き出されるレコードの相対レコード番号は 1 番とされ、以降レコードが書き出される順に相対レコード番号が 1 つずつ繰り上げられる。該当するファイルの

ファイル管理記述項に RELATIVE KEY データ項目を指定してあると、WRITE 文の実行中に書き出されたレコードの相対レコード番号がオペレーティングシステムによって RELATIVE KEY データ項目に収められる。

b. 乱呼出し法または動的呼出し法を採る場合、WRITE 文を実行するのに先立って、プログラム中で RELATIVE KEY データ項目に相対レコード番号を設定しておく。つまり、レコード領域中のレコードを RELATIVE KEY データ項目に関係付けておく。次いで、WRITE 文を実行することによって、そのレコードがファイルに書き出される。

26. 入出力両用モードで開き、乱呼出し法または動的呼出し法で処理するファイルに関しては、レコード領域中のレコードを書き出す先の相対レコード番号をランタイム要素中で RELATIVE KEY データ項目に設定しておく。次いで、WRITE 文を実行すると、そのレコード領域の内容がファイルに書き出される。
27. 下記の場合に、無効キー条件が発生する。
    a. 乱呼出し法または動的呼出し法を採る場合、RELATIVE KEY データ項目が指すレコードが既に存在する。
    b. 外部的に定義されているファイル境界を超えて、レコードの書き出しが試みられた。

形式 4 (索引ファイル)

28. WRITE 文を実行すると、レコード領域の内容が書き出される。オペレーティングシステムはレコードキーの内容を使用して、以降どのレコードキーを通じてでもそのレコードを呼び出すことができるようにする。
29. 主レコードキーの値は、ファイル中の全レコードを通じて一意とする。
30. 主レコードキーに指定したデータ項目は、WRITE 文を実行するのに先立って、ランタイム要素中で希望する値に設定しておく。
31. 順呼出し法でファイルを処理する場合、主レコードキーの値の昇順に順次レコードを書き出すこと。
32. 乱呼出し法または動的呼出し法を採る場合、任意の順序でレコードを書き出せる。
33. 索引ファイルのファイル管理記述項に ALTERNATE RECORD KEY 句を指定した場合、併せて DUPLICATES 指定をしてあれば、副レコードキーの値は一意でなくてもよい。副レコードキーの値が重複する場合、オペレーティングシステムは、それらが書き出された順に呼び出されるようにする。
34. 下記のどれかの場合に、無効キー条件が発生する。
    a. 出力モードで開いたファイルに順呼出し法を適用しているとき、前に書き出したレコードの主レコードキーの値よりも、書き出そうとするレコードの主レコードキーの値が大きくない。
    b. 出力モードまたは入出力両用モードで開いたファイルの場合、書き出そうとするレコードと主レコードキーの値が等しいレコードが、ファイル中に既に存在する。
    c. 出力モードまたは入出力両用モードで開いたファイルの場合、キーの重複を許していないのに、書き出そうとするレコードと副レコードキーの値が等しいレコードが、ファイル中に既に存在する。
    d. 外部的に定義されたファイル境界を超えて、レコードの書き出しが試みられた。

35. (ANS85)WRITE 文の実行が正常にまたは不成功に終わった後、制御がどのように移されるかは、INVALID KEY 指定と NOT INVALID 指定の有無に左右される。(*手続き部*の章の*無効キー条件*節を参照。)

16.9 XML PARSE(XML 解析)文

(MF)

XML PARSE (XML 解析) 文は、COBOL 実行時システムの一部である XML パーサとのインターフェイスに使用される。XML PARSE 文は、XML 文書を各部に分けて解析し、各部を 1 つずつ、利用者が書いた処理手続きへと渡す。

一般形式

```
XML PARSE 一意名-1
PROCESSING PROCEDURE [ IS ] 手続き名-1
    [ { THROUGH } 手続き名-2 ]
      { THRU    }
[ ON EXCEPTION 無条件文-1 ]
[ NOT ON EXCEPTION 無条件文-2 ]
[ END-XML ]
```

構文規則

1. 一意名-1 は英数字または各国語型データ項目でなければならず、関数一意名であってはならない。また、XML 文書文字ストリームを含んでいるものとする。一意名-1 が各国語型データ項目の場合、その内容は、CCSID 1200 (Unicode UTF-16)を使用して符号化されていなければならない。ここには、複数の符号化単位を使用して表示された文字が含まれてはならない。そのような文字を表示するには、"" や""などの文字参照を使用すること。
2. 手続き名-1 は、処理手続き中の最初または唯一の節または段落を指定する。
3. 手続き名-2 は、処理手続き中の最後の節または段落を指定する。

一般規則

1. PROCESSING PROCEDURE 指定は、XML パーサが生成するさまざまなイベントを取り扱う手続きの名前を指定する。この処理手続きは、XML イベントを取り扱う文により構成される。処理手続きの範囲には、この手続き中の CALL、EXIT、GO TO、GOBACK、INVOKE、および PERFORM の各文により実行されるすべての文も含まれる。処理手続きの最後の文の後には、コンパイラにより、戻る仕組みが挿入される。

処理手続きは、STOP RUN 文で実行単位を終了することができるが、GOBACK 文または EXIT PROGRAM 文でパーサへ戻ろうとしてはならない。

2. ON EXCEPTION 指定は、XML PARSE 文が例外条件を挙げたときに、実行される無条件文を指定する。例外条件は、XML 文書の処理中に、XML パーサがエラーを検出したときに発生する。パーサはまず、"EXCEPTION"を含むよう設定された特殊レジスタ XML-EVENT とともに、処理手続きに制御を渡して、例外 XML イベントを知らせる。そして、特殊レジスタ XML-CODE に数字エラーコードを与える。この詳細については、*COBOL 言語の概念*の章の*特殊レジスタ*節の表 2-2 を参照。

   例外条件はまた、処理手続きが、正常な XML イベントからパーサに戻る前に、XML-CODE を -1 に設定して、故意に解析を終了した場合も発生する。この場合、パーサは EXCEPTION XML イベントを知らせず、以下が適用される。

   a. ON EXCEPTION 指定を使用した場合、パーサは制御を無条件文-1 に移す。
   b. ON EXCEPTION 指定を使用しない場合、NOT ON EXCEPTION 指定が使用されていても、無視され、制御は XML PARSE 文の最後に移される。
   c. XML 処理手続きが例外 XML イベントを取り扱い、パーサに制御を返す前に XML-CODE をゼロに設定した場合、例外条件は存在しない。
   d. 取り扱われない例外が他に発生せずにパーサが終了した場合は、制御が NOT ON EXCEPTION 指定の無条件文-2 に移される。ただし、この無条件文-2 を指定した場合に限る。

3. NOT ON EXCEPTION 指定は、XML PARSE 処理の終了時に例外条件がまったく存在しないときに実行される無条件文を指定する。以下の規則が適用される。
   a. 例外条件が XML PARSE 処理の終了時にまったく存在しない場合は、制御が NOT ON EXCEPTION 指定の無条件文-2 に移される。ただし、この無条件文-2 を指定した場合に限る。
   b. NOT ON EXCEPTION 指定を使用しない場合、制御は XML PARSE 文の最後に移される。 この場合、ON EXCEPTION 指定が使用されていても、無視される。特殊レジスタ XML-CODE は、XML PARSE 文の実行後にゼロを含む。

第 17 章 : Micro Focus OO COBOL 言語拡張

MF

本性では、使用可能な Micro Focus OO COBOL 構文および、ISO/IEC 1989:2002 で提供される OO COBOL 構文を解説する。この構文のすべては、Micro Focus COBOL 固有のものであるため、本章の周囲のボックスは削除している。オブジェクト指向機能の詳細な解説については、*オブジェクト指向プログラミング* を参照。

17.1 指令

予約語リストにフラグを付けて修正する機能を提供するコンパイラ指令に加えて、下記の指令が本節の構文または意味のいずれかに影響を与える可能性がある

- ALIGN - 01 レベルまたは 77 レベルのデータ項目を割り付けるメモリの境界を指定する。
- IBMCOMP - 語格納モードをオンにする。
- MAPNAME - メソッド名内のアルファベット以外の文字の取扱いに影響を与える。
- MF-OO - オブジェクト指向の構文を使えるようにする。
- REPOSITORY - クラスに関するリポジトリ記述項を作成するか、無視するか、チェックするかをコンパイラに知らせる
- OOCTRL - オブジェクト指向オプションに関する特別なスイッチを提供する。設定をオンにするには各オプションを表す文字の前に + 記号を付け、オフにするには - 記号を付ける。省略時の解釈は OOCTRL(-G-N-P+Q-W)である。各オプションの意味は下記のとおり。

  +F　小文字に翻訳されるプログラムにより起動されるメソッド名を折り畳む。プログラムが Java メソッドを呼び出す場合は、-F を使用すること。これは、Java メソッドでは大文字と小文字が区別されるためである。Java と COBOL の併用について詳しくは、使用している COBOL システムのマニュアルを参照。

  +G　インスタンスに関してクラスデータをグローバルにする。

  注:このオプションを使用することは推奨しない。この機能は以前のリリースとの互換性を保つために用意されている。

  +P　オブジェクト COBOL ランタイムシステムに対して、パラメータタイプ情報を利用可能にする。

  注意:OLE および SOM に送ったメッセージに関して必要。

+Q　メソッドインターフェイス定義の動詞シグネチャ中のデータ名の後ろに続く可能性のあるどの場所についても、IN および OF を使用できないようにする。

メソッドインターフェイス定義の動詞シグネチャ中のどの動詞も使えないようにする（*翻訳集団の概念*の章の*文の種類*節を参照）。

-Q　メソッドインターフェイス定義の動詞シグネチャ中で IN および OF を使用できるようにするメソッドを呼び出す動詞シグネチャの中で修飾を行えないようにする。

メソッドインターフェイス定義の動詞シグネチャ中で動詞も使えるようにする（*翻訳集団の概念*の章の*文の種類*節を参照）。

+W　オブジェクトおよびファクトリ（クラス）オブジェクトにおいて、オブジェクト記憶の意味で作業場所を使用する。

注：ISO 2002 COBOL 標準との互換性を持たせるために必要。

17.2 クラス定義

クラスはオブジェク COBOL のクラスオブジェクトおよびそのインスタンスオブジェクトを記述する。その中には、クラスメソッドおよびインスタンスメソッドに関するネストされたプログラムが含まれる。

一般形式

```
[IDENTIFICATION DIVISION.]
CLASS-ID. クラス名-1 [ IS ABSTRACT ]
                     [ IS EXTERNAL ]
      [ DATA IS { PRIVATE   } ]
                { PROTECTED }
                { RESTRICTED }
      [INHERITS FROM クラス名-2[WITH DATA]].
クラス本体
オブジェクトプログラム
END CLASS クラス名-1
```

構文規則

1. PROTECTED と RESTRICTED の2つの語は同等である。
2. EXTERNAL 句を指定した場合は、DATA 句も INHERITS 句も使用できない。
3. ABSTRACT 句を指定した場合は、FINAL 句を指定してはならない。
4. INHERITS 句の中に WITH DATA 句を指定した場合、DATA IS PRIVATE 句を明示的にまたはクラス名-2 のソースコード中に指定してはならない。
5. クラス名-2 はクラス名-1 と同じであってはならない。

6. クラス名-2 はクラス名-1 から直接的にまたは間接的に継承してはならない。
7. クラス終了見出し中のクラス名-1 は先行するクラス名段落中に指定されているクラス名-1 と同じでなければならない。

一般規則

1. クラス名-1 は宣言する対象のクラスを識別する。
2. ABSTRACT 句はクラス名-1 が抽象クラスであることを示す。抽象クラスのインスタンスを生成することはできない。
3. EXTERNAL 指定はクラス名-1 が外部クラスであることを示す。コードは生成されない。
4. RESTRICTED 指定はクラス名-1 から継承したデータにサブクラスが直接アクセスできるようにする。
5. PRIVATE 指定はクラス名-1 から継承したデータにサブクラスが直接アクセスできないようにする。
6. INHERITS 指定はクラス-2 がクラス-1 の親クラスであることを指定する。
7. INHERITS 指定を指定しないと、クラス名-1 は分類スキーマ内で新しいルートを形成する。
8. WITH DATA 指定は、クラス名-2 から継承したデータにクラス名-1 が直接アクセスできることを指定する。

17.3 クラス拡張

クラス拡張を行うと、元のソースコードを変更せずに、オブジェクト COBOL に機能を追加できる。

継承を行うのではなく、クラス拡張によってクラスを拡張することの違いは、クラス拡張は既存のすべてサブクラスによって継承されることである。たとえば、クラス A のサブクラスにクラス B があるとする (つまり、 B INHERITS FROM A) 。クラス A のサブクラスとしてクラス C を生成することによって、クラス A にメソッドを追加できる。しかし、その場合はクラス B はクラス C のメソッドを継承しない。クラス拡張 X によってクラス A を拡張した場合は、実行時の効果はクラス A を変更してコンパイルし直したのと同じである。クラス X によって追加されたすべてのメソッドをクラス B も継承する。

一般形式

```
[IDENTIFICATION DIVISION.]

CLASS-ID. 拡張名-1 EXTEND クラス名-1 [WITH DATA].

クラス本体

オブジェクトプログラム

END CLASS 拡張名-1
```

構文規則

1. 拡張名-1 はクラス名-1 と同じでなければならない
2. クラス終了見出し中の拡張名-1 は先行するクラス名段落中に指定されている拡張名-1 と同じでなければならない。
3. クラス名-1 はクラス管理段落に指定されているクラスの名前でなければならない。
4. クラス本体のデータ部に空のオブジェクト記憶節を含めてもよい。その他にクラス拡張のデータ部に指定してもよい節は、作業場所節と連絡節だけである。
5. 下記の両方に該当する場合にのみ、クラス拡張中の文からクラス名-1 内に宣言されているデータを参照できる。
    - クラス名-1 のクラス名段落中に DATA IS PROTECTED 指定または DATA IS RESTRICTED 指定がある。
    - クラス拡張のクラス名段落中に WITH DATA 指定がある。

一般規則

1. EXTEND 句はクラス拡張を指定する。クラス拡張はオブジェクトクラスにメソッドを追加する。拡張名-1 に指定されたメソッドはクラス名-1 の新旧のすべてのサブクラスによって継承される。
2. 実行単位の実行中に、クラス拡張中のメソッドを呼び出すのに先立って、拡張名-1 に対して COBOL の CALL 文を実行しなければならない。それによって、クラス拡張中のメソッドが OO ランタイムシステムに登録される。

17.4 クラス本体

クラス本体には、クラスのデータとメソッドを定義した、すべでのコードが含まれる。

一般形式

形式 1

（コンパイラ指令の OOCTRL(+N)を指定したときに使用される。）

注：この構文を使用することが望ましい

```
OBJECT SECTION.
 CLASS-CONTROL.
        [クラス名-3 IS CLASS "外部名-1"]... .
[データ部]
[クラスオブジェクト]
```

形式 2

（コンパイラ指令の OOCTRL(-N)を指定したときに使用される。）

```
OBJECT SECTION.
 CLASS-CONTROL.
        [クラス名-3 IS CLASS "外部名-1"]... .
[データ部]
[手続き部]
[メソッド-1]...
```

構文規則

すべての形式

1. クラス管理段落中にクラス名-3 を2回以上指定してはならない。
2. クラス名-3 はクラス名段落中に指定されているクラス名と同じであってもよい。
3. データ部に連絡節を含めてはならない。

形式 1

4. データ部にオブジェクト記憶節を含めてはならない。
5. データ部内に宣言したデータ項目をインスタンスメソッドおよびクラスメソッドから参照できる。

形式 2

6. 局所記憶節、報告所節、画面節に宣言されたデータ項目を参照できるのは、該当のクラスの手続き部内の文からだけであって、任意のメソッドからは参照できない。
7. ファイル節または作業場所節に宣言されたデータ項目をインスタンスメソッドとクラスメソッドおよび該当のクラスの手続き部から参照できる。
8. オブジェクト記憶節に宣言されたデータ項目をクラスメソッドから参照できる。

一般規則

すべての形式

1. クラス名-3 は暗黙的に USAGE IS OBJECT REFERENCE と定義される。
2. クラス-3 はクラスの名前であって、それが属する環境部の適用範囲内でそれを使用できる。
3. 外部名-1 は該当のクラスを含むファイルの外部名を指定する。
4. メソッド-1 はクラスメソッドである。
5. オブジェクト記憶節内のデータ項目だけがサブクラスによって継承される。

形式 1

6. データ項目は実行単位の開始時に初期化され、メソッド呼出しの間は最後に使用されたときの状態が保たれる。

形式 2

7. クラス手続き部内の文は、実行単位内で該当のクラスの何らかのクラスメソッドまたはインスタンスメソッドが最初に実行される前に、実行される。
8. クラス手続き部の実行が開始されると、局所記憶節内のデータ項目の内容は保証されない。この記憶節は、クラス手続き部が実行され終わると、直ちに割当を解除される。
9. ファイル節および作業場所節内で定義されたデータ項目は、クラスメソッドおよびインスタンスメソッドの呼出しの間は、最後に使用されたときの状態が保たれる。

注：作業場所節内のデータはクラスまたはインスタンスを初期化するためのデータとして役に立つ。

---

10. オブジェクト記憶節内で定義されたデータ項目は、クラスメソッドの呼出しの間は、最後に使用されたときの状態が保たれる。

17.5 クラスオブジェクト

クラスオブジェクトはオブジェクトを生成する働きをするオブジェクトである。

一般形式

```
[IDENTIFICATION DIVISION.]
 CLASS-OBJECT.
[データ部]
[PROCEDURE DIVISION.]
[メソッド-1]...
 END CLASS-OBJECT.
```

一般規則

1. クラスオブジェクトのオブジェクト記憶節内で宣言されたデータ項目はクラスデータである。クラスデータはサブクラスによって継承されうる。

2. データ項目は実行単位の開始時に初期化され、メソッド呼出しの間は最後に使用されたときの状態が保たれる。
3. メソッド-1 はクラスメソッドである。

17.6 オブジェクトプログラム

オブジェクトプログラムには、クラスのすべてのインスタンスに関するデータとメソッドの定義が含まれる。

一般形式

```
[IDENTIFICATION DIVISION.]
OBJECT.
[データ部]
[PROCEDURE DIVISION.]
[メソッド-1]...
END OBJECT.
```

一般規則

1. オブジェクトプログラムのオブジェクト記憶節内で宣言されたデータ項目はインスタンスデータである。インスタンスデータはインスタンスメソッド内でのみ参照できる。
2. データ項目は実行単位の開始時に初期化され、メソッド呼出しの間は最後に使用されたときの状態が保たれる。
3. メソッド-1 はインスタンスメソッドである。

17.7 メソッド

形式

```
METHOD-ID."メソッド名-1".
[データ部]
[手続き部]
END METHOD"メソッド名-1".
```

構文規則

1. メソッド終了見出し中のメソッド名-1 は先行するメソッド名段落中に指定されているメソッド名-1 と同じでなければならない。
2. メソッドのデータ部内で定義されたデータはそのメソッド内でのみアクセスできる。
3. メソッド名-1 の前後の引用符は指定しても指定しなくてもよい。

注：メソッド名を使用すると、予約語をメソッド名として使用することができ、また COBOL 用以外の文字を使用できるようになる。

---

4. メソッド定義はクラス定義内に置かなければならない。
5. メソッドに関する手続き部の見出しの形式は、プログラムに関する手続き部の見出しの形式 1 と同じである。手続き部の見出しの形式 2 に示されている、GIVING または RETURNING データ名句を指定してもよい。（詳細については、***手続き部***の章の*手続き部の見出し*節を参照）。

一般規則

1. メソッド名-1 は該当のメソッド定義によって宣言されたメソッドの名前である。
2. メソッドの局所記憶節内に宣言されたデータは、メソッドが呼び出されるごとに別の記憶域を割り当てられ、メソッドが終了すると割当を解除される。このデータは、メソッドが呼び出されたときは、未定義の状態にある。

注：メソッド内で使用するデータは局所記憶節内に定義することを推奨する。その理由は、メソッドを別々に呼び出したインスタンスの間で、相互にデータに干渉し合うことが避けられるからである。

---

3. 局所記憶節または連絡節以外の節内のメソッド中で宣言されたデータおよびファイルは、メソッドのすべての呼出しの間で共有され、メソッドが呼び出されたときには最後に使用された状態にある。
4. 該当のメソッドが含まれるオブジェクトを参照するオブジェクト一意名を伴うメソッド呼出しの中で、メソッド名-1 が使用されていてもよい。
5. メソッドの手続き部の見出しの中で RETURNING 指定または GIVING 指定を指定した場合、その中に指定されているデータ項目のメソッドの終了時点での内容が、そのメソッドの結果となる。INVOKE 文の RETURNING 指定または GIVING 指定に指定されている一意名の中に、その結果が入れられる。

17.8 メソッドインターフェイス定義

メソッドインターフェイス定義は、メソッドのパラメータ、パラメータを渡す方法、メソッドを呼び出すために使用できる代替構文を定義する。

形式

```
METHOD-ID."メソッド名-1".
[連絡節]
手続き部見出し
    [INVOKED AS 動詞シグネチャ[OR AS 動詞シグネチャ]...].
END METHOD"メソッド名-1".
```

where verb-signature is:

```
                          ┌<OBJECT>┐
                          │<SELF>  │
                          │<THIS>  │
[FUNCTION]==動詞-1 ┤<データ名-3>├...==
                          │必須語  │
                          │補助語  │
                          │右かっこ│
                          └左かっこ┘
```

構文規則

1. メソッドインターフェイス定義は外部クラスの中にネストされなければならない。
2. 手続き部の見出しは、プログラムに関して指定する手続き部の見出しの形式 2 と同様である。ただし、呼び名も REPEATED 指定も指定できない。上の形式に示した INVOKED 指定は、見出し中の末尾の終止符の直前に置くこともできる (詳細については、***手続き部***の章の*手続き部の見出し*節を参照) 。
3. 動詞シグネチャ内に FUNCTION 指定を指定したときは、手続き部の見出し内に RETURNING 指定を指定しなければならない。
4. 動詞-1 は COBOL の語でなければならない。かつ、予約語でも手続き名でもあってはならない。
5. 語 <OBJECT> と <SELF> と <THIS> の意味は同じである。各動詞シグネチャ内には、それらのどれかひとつが1回だけ現れなければならない。
6. データ名-3 は "<" と ">" で囲まれていなければならない。
7. 必要語は COBOL の語でなければならない。
8. 補助語は COBOL の語を "[" と "]" で囲んだもでなければならない。
9. 開きかっこと閉じかっこはそれぞれ "(" と ")" である。

注：このかっこによって、組み込み関数のように見え、かっこ内にパラメータを指定する、関数を定義できるようになる。

---

10. 動詞シグネチャは他の動詞シグネチャのサブセットであってはならない。

一般規則

1. 動詞シグネチャを使用してメソッド名-1 を呼び出すと、<SELF> が受け手のオブジェクトに対するオブジェクト参照で置き換えられる。
2. 補助語は読みやすくするために使用するもであり、この構文を用いてメソッドを呼び出すときに、指定しても指定しなくてもよい。
3. 動詞-1 がプログラム内でデータ名としても宣言されている場合、そのプログラム内のその語を参照すると、必ずデータ名の方を指すことになる。

第 18 章：翻訳指示文

翻訳指示文 (compiler-directing statement) は、翻訳集団の処理を制御する。以下の翻訳指示機能が用意されている。

- 原始文操作(source text manipulation)を指定するための翻訳指示文
- (ISO2002)標準翻訳オプション(standard compilation options)を指定するための翻訳指示文
- (OSVS)(VSC2)BASIS 機構
- (OSVS)(VSC2)++INCLUDE/ -INC 機構
- (MF)条件付き翻訳(conditional compilation)
- (OSVS)(VSC2)(MF)リスト作成制御文 (listing control statement)
- (MF)実行オプション(implementor options)を指定するコンパイラ指令

18.1 原始文操作

原始文操作(source text manipulation) は、原始利用者登録集 (source user-library) から複写してくる原文を指定する機能を提供する。この原始利用者登録集は、通常、何らかの適切な原始文エディタ (source text editor) を使用して作成する。

COBOL ライブラリ (library) は、COBOL システムで利用するための原始文を含むファイルから構成されている。COPY (複写) 文の働きは、原始プログラム中に原文を挿入して、COBOL システムがそれを原始プログラムの一部として扱うようにすることである。登録集原文 (library text) 中の定数や一意名や語や語の集団を指定して、複写処理中にそれらが出てくると、すべて別の原文に置き換えることができる。 原始文操作では、実行用プログラムを作成するときに、複数の COBOL 登録集を使えるようにすることもできる。

(ANS85)REPLACE (置換) 文の働きは、原始文中に現れる原文を新しい原文に置き換えて、COBOL システムがそれを原始文の一部として扱うようにすることである。

18.1.1 COPY(複写)文

機能

COPY (複写) 文は、COBOL 翻訳集団中に原文を組み入れる。

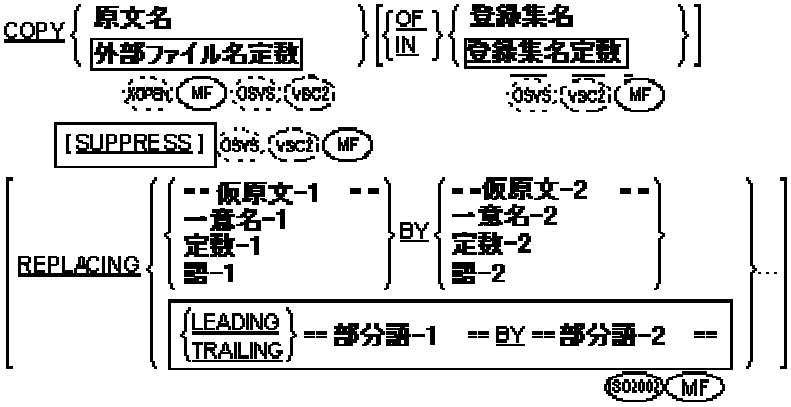
例

以下の例が、*言語リファレンス - 追加トピックの例題*の章に掲げられている。

- ANSI'68 または LANGLVL(1) の動作を実現するための OLDCOPY コンパイラ指令の使用。*COPY (ANSI'68 または LANGLVL(1) 版)* を参照。
- ANSI'85 の規則でサポートされている部分的な語の置換を伴う COPY 文の使用。*特殊文字を含む ANSI'85 の使用例* を参照。

- ISO2002 の規則でサポートされている部分的な語の置換を伴う COPY 文の使用。
  ISO2002 および MF 方言の使用例 を参照。

一般形式



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
   - COPYEXT - コピー･ファイルの所在を突き止めるために使用する、ファイル名の拡張子を指定する。
   - COPYLBR - ライブラリ名が.lbr に相当することを指定する。
   - COPYLIST - コピー文の結果をリスティング･ファイルに含めさせる。
   - FOLD-COPY-NAME - ディスク上のライブラリ名と COPY 文中の原文名の間で登録集メンバー名の大文字と小文字の指定が違っている場合に、コンパイラ･システムがコピー登録集メンバーを見つけ出せるようにする。
   - OLDCOPY - ANS'68 および OS/VS COBOL LANGLVL(1)の規則に合うように、COPY 文の取扱いを変更する。

構文規則

1. 原始プログラムを COBOL システムに渡すときに複数の COBOL 登録集が利用できる場合は、原文名を登録集名で修飾して、原文名の原文が登録されている COBOL 登録集を一意に識別しなければならない。登録集についての詳細は、COBOL システムのマニュアルを参照。

   OSVS VSC2 MF この制限は削除された。

2. COPY 文は、前に空白を 1 つ置き、後ろを分離符の終止符 (.) で止める。
3. 仮原文-1 は、空でも、空白文字だけから成っても、注記行だけから成ってもならない。
4. 仮原文-2 は、空であってもよい。

5. 仮原文-1 および仮原文-2 の中の文字列は、行をまたがって継続させることができる。ただし、仮原文の区切り記号 (psuedo-text delimiter) は、2 つとも同じ行にあること。(*COBOL プログラムの概念*の章の*行のつなぎ*節を参照。)
6. 語-1 または語-2 は、任意の単一の COBOL の語であってよい。
7. 文字列または分離符が書けるところならば、原始プログラム中のどこにでも COPY 文を書くことができる。ただし、COPY 文の中には COPY 文を書けない。
8. 原文名は、一意の外部ファイル名を定義する。このファイル名は、利用者語の作成規則に準拠していること。

   OSVS VSC2 MF COB370 外部ファイル名定数は英数字定数であり、ファイル名に関するオペレーティングシステムの規則に準拠していること。外部ファイル名定数を指定する時は、引用符で囲んでもよいし囲まなくてもよい。

   原文名および引用符で囲んでいない外部ファイル名定数は、常に大文字に変換されることに注意。

9. MF 登録集名定数は文字定数であり、ファイル名に関するオペレーティングシステムの規則、または装置一意名に関するオペレーティングシステムの規則に準拠していること。登録集名定数を指定する時は、引用符で囲んでもよいし囲まなくてもよい。

   引用符で囲んでいない登録集名定数は、常に大文字に変換されることに注意。

10. OSVS VSC2 MF SUPPRESS は、原始リスト上にコピー・メンバの内容を印刷しないようにするために指定する。
11. ANS85 注記項の中または注記項があってもよい場所に語 COPY がある場合、注記項の一部とみなされる。

    OSVS 注記項中にある COPY 文は処理される。

12. OSVS MF 外部ファイル名定数および登録集名定数は、引用符で囲まれている場合、文字$、#、@を含んでもよい。
13. SOM90 MF 部分語-1 は、単一の原文語でなければならない。
14. SOM90 MF 部分語-2 は、ゼロまたは単一の原文語でなければならない。
15. SOM90 MF 文字定数と各国語型定数のいずれも、部分語-1 と部分語-2 のいずれとしても指定してはならない。
16. 登録集原文は、COBOL 参照形式の規則に準拠していなければならない。
17. SOM90 MF COPEN 連結式は、外部ファイル名定数、登録集名定数、定数-1、および定数-2 のどれに対しても指定してはならない。

一般規則

1. COPY 文を含む翻訳集団をコンパイルすることは、論理的には、すべての COPY 文を処理してからその結果の翻訳集団をコンパイルすることと同じである。
2. COPY 文を処理した結果は、予約語 COPY に始まり句読文字の終止符で終わる COPY 文全体を論理的に置き換える形で、原文名によって表わされる登録集原文を原始文中に複写したようになる。

3. REPLACING 指定をしないと、登録集原文はそのままで複写される。

   REPLACING 指定をすると、複写される登録集原文のうちで、仮原文-1、一意名-1、定数-1、語-1 に一致するそれぞれの部分が、対応する仮原文-2、一意名-2、定数-2、語-2 によって置き換えられる。

4. 登録集と一致するかどうか比較する目的では、一意名-1 と定数-1 と語-1 は、それぞれ一意名-1 または定数-1 または語-1 だけを含む仮原文として扱われる。
5. 原文を置換するかどうかを判定するための比較は、以下のように行われる。
   a. 分離符のコンマでもセミコロンでもない登録集原文語の左端の語が、比較の最初の対象となる。この原文語の前にある原文語または空白は、原始文中に複写される。比較対象の最初の原文語以降が、REPLACING の作用対象全体と連続する同じ文字数だけ比較される。ここで、REPLACING の作用対象とは、予約語 BY の前までに指定した仮原文-1、一意名-1、定数-1、語-1 を指す。
   b. 仮原文-1、一意名-1、定数-1、語-1 は、その語を構成する文字の並びが登録集原文語の文字の並びと、逐一同じであるときにだけ、登録集原文と等しいと判定される。
   c. (ISO2002)(MF)LEADING を指定した場合、部分語-1 は、それを構成する文字の並びが、登録集原文語の左端文字位置から続く、同じ数の文字の並びと、逐一同じであるときにだけ登録集原文と等しいと判定される。TRAILING を指定した場合、部分語-1 は、それを構成する文字の並びが、登録集原文語の右端文字位置で終わる、同じ数の文字の並びと、逐一同じであるときにだけ登録集原文と等しいと判定される。
   d. 等しいと判断する際の規則は以下の通り。
      1. 仮原文-1 または登録集原文の中に現れる分離符の各コンマ、セミコロン、または空白は、単一の空白と見なされる。1 個以上連続する空白分離符は、単一の空白と見なされる。
      2. 文字定数または各国語定数で使用されるものを除き、小文字は、COBOL 文字集合に指定されている、対応する大文字と同等に扱われる。
      3. デバッグ行内の原文語は、指示領域に "D" がないものとして、比較に使用される。
      4. (ISO2002)(MF)(COPEN)連結式の各作用対象および各演算子は、別の原文語である。
   e. 照合結果が一致しなかった場合、REPLACING 指定の続きがあれば、次の各仮原文-1、一意名-1、定数-1、語-1 を対象にして比較が繰り返される。このようにして一致するものが見つかるか、REPLACING 指定の作用対象の続きがなくなるまで、比較は繰り返し続けられる。
   f. REPLACING 指定の作用対象のすべてが比較されたが、一致するものが見つからなかった場合、左端の登録集原文語は原始文中に複写される。次いで、その次に続く登録集原文語が左端の登録集原文語であるとみなされて、REPLACING 指定中の最初の仮原文-1、一意名-1、定数-1、語-1 から始まって、比較周期が再び開始される。

g. 仮原文-1、一意名-1、定数-1、語-1 と登録集原文との間で比較結果が一致した場合、対応する仮原文-2、一意名-2、定数-2、語-2 が原始文中に入れられる。

(ISO/IEC) (MF) 語-1 と登録集原文の間で比較結果が一致した場合、登録集原文が、一致した文字を含む実行後の原始文中で、語-1 に置き換えられる。または、語-2 が空の原文である場合は、削除される。

比較の対象とされた登録集原文部分のうちの右端の原文語のすぐ後ろの語が、左端の登録集原文語であるとみなされて、REPLACING 指定中の最初の仮原文-1、一意名-1、定数-1、語-1 から始まって、比較周期が再び開始される。

h. この処理は、登録集原文中の右端の原文語が左端の登録集原文語とみなされて比較され、照合結果が一致するかまたは比較周期を完了したところで、終了する。

6. 比較の目的では、登録集原文および仮原文-1 中に出てくる注記行は、単一の空白とみなされる。仮原文-2 および登録集原文の中に出てくる注記行は、原始文中に元のまま複写される。
7. COPY 文の処理を完了した結果の原文中に、COPY 文が含まれていてはならない。

(VSC2) 結果の原文中に COPY 文が含まれていてもよい。ただし、その COPY 文および既に展開済みの COPY 文に、REPLACING 指定が含まれていないことが条件である。この場合、再帰的な COPY 文 (登録集原文が、その中の COPY 文により参照されている) は含まれていてはならない。

(MF) 任意のレベルまで、COPY 文を入れ子にしてもよい。入れ子になった任意の COPY 文の中に、REPLACING 指定が含まれていてもよい。下位レベルのすべての COPY 文に対して、REPLACING 指定は効力を発揮する。

8. 登録集原文の構文の正しさは、独立して判定することはできない。

(ANS85) ただし、COPY 文は例外である。

すべての COPY 文の処理が完了するまで、COBOL 原始プログラム全体の構文の正しさは判定できない。

9. (MF) REPLACING 指定が含まれる場合は、登録集原文に EXEC HTML 文が含まれていてはならない。この規則に反すると、予期しない動作や構文エラーが発生する可能性がある。
10. 登録集から複写されたが置換はされなかった各原文語は、結果としてできあがるプログラムの中でも、登録集内の行の中でその語が占めていたのと同じ領域から始まるように複写される。しかし、登録集から複写してきた A 領域から始まる原文語の後ろに別のやはり A 領域から始まる原文語があって、前の方の原文語がそれよりも長いものに置換された場合、A 領域に余裕がなければ、後ろの原文語は B 領域から開始されることがある。結果としてできあがる原始文の中に入れられる仮原文-2 の中の

各原文語は、仮原文-2 の中でその語が占めていたのと同じ領域から始まるように配置される。各一意名-2、定数-2、語-2 は、結果としてできあがるプログラムの中で、比較の対象となった左端の登録集原文語が置換されなかったならば、位置していたのと同じ領域から開始される。

COPY 文を処理した結果として原始文中に行が追加される場合、COPY 文がデバッグ行上から始まるか、または複写される原文語が登録集原文中のデバッグ行上にあると、その原文語はデバッグ行上に置かれる。BY 指定の対象の原文語が組み入れられるときに、置換される最初の登録集原文語がデバッグ行上にあると、置き換わる語もデバッグ行上に配置される。上記の場合を除いて、結果としてできあがる原始文中のデバッグ行上に配置されるのは、仮原文-2 の中のデバッグ行上に指定されている原文語だけである。定数-2 としてまたは仮原文-2 か登録集原文の中に指定されている定数が 1 行に収まりきらないほど長く、かつその定数がデバッグ行上にはない場合、あふれる分を収容する継続行が後ろに追加される。置換の結果、定数の続きがデバッグ行にかかる場合は、翻訳集団はエラーとなる。.

11. 比較の目的では、置換後の原文語は、***COBOL プログラムの概念***の章の*参照形式*節に記述されている正書法の規則に従って、原始文中に配置される。
12. 装置一意名を明示的に指定しないと、省略時解釈の装置が使用される (省略時解釈値は、オペレーティングシステムによって異なる) 。
13. (OSVS)OLDCOPY コンパイラ指令を設定すると、COPY 文の内容が原文名の複写ファイルの内容で置き換えられる際に、 COPY 文の前にあるデータ名が複写ファイル中の対応するデータ名の代わりに用いられる。
14. (ANS85)登録集原文中で何らかの適用規則が守られている場合、データ名の接頭辞のような名前の一部を REPLACING 指定を用いて変更できる。この方式の「部分的な語の置換」機能を利用するためには、変更の対象とする部分の前後を下記のように囲まなければならない。
    a. 一組の左かっこと右かっこで、(**ABC**)のように囲む。
    b. コロンで、**:XYZ:**のように囲む。

注：LEADING および TRAILING 指定を、部分的な語の置換に使用できる。LEADING および TRAILING 指定は、MF 方言と次の COBOL 標準でサポートされているが、このサポートは ANSI '85 COBOL 標準には含まれていない。

18.1.2 REPLACE(置換)文

(ANS85)

機能

REPLACE (置換) 文は、原文を翻訳単位に置き換えるために使用する。

一般形式

形式 1

```
REPLACE { == 仮原文-1  == BY == 仮原文-2  ==
          [ { LEADING } == 部分語-1   == BY == 部分語-2   == ]
          { TRAILING }                                        } ...
                                                   (ISO2002)(MF)
```

形式 2

```
REPLACE OFF
```

構文規則

1. REPLACE 文は、翻訳集団中で文字列を書くことができる場所ならば、どこにでも書くことができる。翻訳集団中の先頭の文でない場合は、REPLACE 文の前に分離符の終止符 (.) を打つこと。
2. REPLACE 文は、分離符の終止符 (.) によって止める。
3. 仮原文-1 には、1 語以上の原文語を含める。
4. 仮原文-2 には、原文語を含めなくても、1 語以上の原文語を含めてもよい。
5. 仮原文-1 および仮原文-2 の中の文字列は、次の行へ継続できる。
6. 仮原文中の原文語の長さは、1 文字から 322 文字の間とする。
7. 仮原文-1 は、そのすべてが分離符のコンマまたはセミコロンから構成されていてはならない。
8. 注記項の中または注記項を記してもよい場所に語 REPLACE があると、注記項の一部とみなされる。
9. (ISO2002)(MF)語-1 は、1 つの原文語のみから成る。
10. (ISO2002)(MF)語-2 は、1 つの原文語のみからなる。または、空の語である。
11. (ISO2002)(MF)文字定数および各国語定数のいずれも語-1 または語-2 として指定してはならない。

一般規則

1. 仮原文-1 には、仮原文-2 によって置き換える原文を指定する。
2. (ISO2002)(MF)語-1 は、語-2 により置き換えられる原文を指定する。
3. 形式 22 の REPLACE 文は、現在働いている原文置換機能を停止することを指定する。
4. いったん指定した REPLACE 文の効力は、次の REPLACE 文の指定が出てくるか、そのプログラムの終わりに達するまで持続する。
5. 原始単位中に含まれる REPLACE 文の処理は、その原始単位中に含まれる COPY 文があればその処理の後で行われる。
6. REPLACE 文を処理した結果の原文には、COPY 文も REPLACE 文も含まれていてはならない。

7. 原文を置換するか否かを判定するための比較は REPLACE 文の直後に開始され、以下のように行われる。
   a. 原始文中の左端の原文語と最初の仮原文-1 から始まって、仮原文-1 が原始文中の連続する同じ文字数の原文語と比較される。
   b. 仮原文-1 は、それを構成する原文語の並びが原始文中の原文語の並びと同じであるときにだけ、等しいと判定される。

      (SOWA)(MF)LEADING を指定した場合、部分語-1 は、それを構成する文字の並びが、原文語の左端文字位置から続く、同じ数の文字の並びと、逐一同じであるときにだけ原文語と等しいと判定される。TRAILING を指定した場合、部分語-1 は、それを構成する文字の並びが、原文語の右端文字位置で終わる、同じ数の文字の並びと、逐一同じであるときにだけ原文語と等しいと判定される。

   c. 照合の目的で、以下の規則が適用される。
      1. 仮原文-1 または原文の中に現れる分離符の各コンマ、セミコロン、または空白は、単一の空白と見なされる。1 個以上連続する空白分離符は、単一の空白と見なされる。
      2. 文字定数または各国語定数で使用されるものを除き、小文字は、COBOL 文字集合に指定されている、対応する大文字と同等に扱われる。
      3. デバッグ行内の語は、標識領域に"D"がないものとして、比較に使用される。
      4. (SOWA)(MF)(OPEN)連結式の各作用対象と各演算子は異なる原文語である。
   d. 照合結果が一致しなかった場合、以降の各仮原文-1 または語-1 を対象にして比較が繰り返される。このようにして、一致するものが見つかるか、仮原文-1 または語-1 の続きがなくなるまで、比較は繰り返し続けられる。
   e. 含まれている仮原文-1 または語-1 のすべてが比較され、一致するものがなかった場合、次に続く原文語が左端の原文語と見なされ、 最初の仮原文-1 または語-1 から始まって、比較周期が再び開始される。
   f. 仮原文-1 と原始文中の原文との間で比較結果が一致した場合、原始文中のその原文が対応する仮原文-2 によって置き換えられる。

      (SOWA)(MF)仮原文-1 と原始文中の原文との間で比較結果が一致した場合、原始文中のその文字は、語-2 によって置き換えられる。または、語-2 が空の場合は削除される。

      比較の対象とされた原始文中の原文部分のうちの右端の原文語のすぐ後ろの語が、原始文中の左端の原文語であるとみなされて、最初の仮原文-1 から始まって、比較周期が再び開始される。

   g. この処理は、原始文の原文中の右端の原文語が REPLACE 文の範囲内の左端の原文語とみなされて比較され、照合結果が一致するかまたは比較周期を完了したところで、終了する。

8. 原始文中の原文および仮原文-1 の中に現れる注記行または空白行は、照合の目的上は無視される。原始文中の原文および仮原文-1 の中の原文語の順序は、正書法に関する規則に従って判定される。詳細については、*COBOL プログラムの概念*の章の*正書法の表現*節を参照。原始文中の原文が仮原文-2 によって置換される場合は、仮原文-2 の中の注記行または空白行は結果の原始文にそのまま入れられる。原始文中にあって仮原文-1 に一致する一連の原文語の中に現れる注記行または空白行は、結果の原始文には入れられない。
9. すべての COPY 文と REPLACE 文の処理が完了した後でだけ、残りの原始コードの構文の正しさを判定できる。
10. REPLACE 文を処理した結果として原始文中に挿入される原文語は、正書法に関する規則に従って原始文中に配置される。詳細については、*COBOL プログラムの概念*の章の*参照形式*節を参照。仮原文-2 の原文語を原始文に挿入するときに、追加の空白を組み込むことができるのは、既に空白が空いている原文語の間だけである（原始行の間に想定される空白を含む）。
11. REPLACE 文を処理した結果として原始文中に行が追加される場合、追加される行の標識領域には置換される行の先頭の文字が入れられる。ただし、置換される行の先頭の文字がハイフン（-）である場合は、空白が入れられる。

    仮原文-2 の中の定数が 1 行に収まりきらないほど長く、かつその定数がデバッグ行上にない場合、あふれる分を収容する継続行が後ろに追加される。置換の結果、定数の続きがデバッグ行にかかる場合は、翻訳単位はエラーとなる。

12. MF REPLACING 指定が含まれる場合は、登録集原文に EXEC HTML 文が含まれていてはならない。この規則に反すると、予期しない動作や構文エラーが発生する可能性がある。

18.2 コンパイラ指令

SO2002

コンパイラ指令はコンパイラによって使用される選択機能や変数を指定するためのものである。

一般形式

>>翻訳指示

構文規則

1. コンパイラ指令は1行に指定しなければならない。ただし、EVALUATE と IF は別である。この2つの指令には別の規則が適用される。
2. コンパイラ指令の前に置けるのは1つ以上の空白文字だけである。空白がなくともかまわない。
3. 固定方式の正書法を採る場合、コンパイラ指令はプログラム原文領域に記すものとし、その後ろには空白文字だけを記してよい。

4. 自由方式の正書法を採る場合、コンパイラ指令の後ろに記してよいのは空白文字と書いても書かなくてもよい行内注記だけである。
5. コンパイラ指令は2つの連続した COBOL の文字>>とそれに続く COBOL の空白文字と翻訳指令から構成される。ただし、空白文字は書かなくともかまわない。>>の後ろに空白文字を書かなかった場合、>>の後ろに空白が1つあるように扱われる。
6. 翻訳指令は、いくつかのコンパイラ指令語から構成される。各コンパイラ指令語にはそれなりの構文の定義がある。
7. コンパイラ指令語は該当するコンパイラ指令の文脈内で予約されている。それを、任意の型の COBOL の語として、どこでも使用できる。
8. コンパイラ指令は翻訳集団内のどこにでも指定できる。ただし、下記を除く。
    a. 個々のコンパイラ指令の規則で制限されている場所
    b. 原始文操作文の中
    c. 継続されている文字列の行の間
    d. デバッグ行の上
9. 登録集原文内にコンパイラ指令の行を指定できる。
10. コンパイラ指令内に、連結式または表意定数として、定数を指定してはならない。

一般規則

1. COPY 文または REPLACE 文の照合処理の間は、コンパイラ指令は単一の空白行として扱われる。コンパイラ指令はいかなる仮原文あるいは部分語とも合致しない。したがって、置換処理の影響を受けない。
2. COPY 文および REPLACE 文の処理にさいしては、コンパイラ指令は該当の指令の規則に従って、処理の前、間、後のいずれかの時点で処理される。
3. >>IF 指令および>>EVALUATE 指令の原始文に指定された場合、コンパイラ指令の>>EVALUATE と>>IF は条件付き翻訳の間に出てきたときに効力を発揮する。それ以外のコンパイラ指令はすべて、条件付き翻訳の結果として得られた原始文が処理されるときに処理される。

18.2.1 条件付き翻訳

コンパイラ指令を使用すると、原始文の範囲を選択して対象に含めたり対象から除外したりすることができる。そのような翻訳方法を条件付き翻訳と呼ぶ。そのために使用するコンパイラ指令は EVALUATE と IF である。EVALUATE 指令および IF 指令を使用して、原始文の行を選択し、翻訳の対象に含めたり対象から除外したりする。

18.2.2 コンパイラ指令中の算術式

EVALUATE 指令および定数の条件式の中に算術式を指定できる。これらの算術式の形式、指定順序、および評価規則は、***手続き部***の章の*算術式* 節で解説する。また、以下の規則が適用される。

1. 作用対象はすべて、固定小数点の数字定数、または使用されるすべての作用対象が固定小数点の数字定数である算術式のどちらかでなければならない。

注: この中には固定小数点の数字定数に等しい定数を含む。

---

2. 各算術演算の後で、結果の値の小数部分が切り捨てられて、結果は整数とみなされる。
3. 算術式が評価された後で、結果の値は数字定数とみなされる。
4. 算術演算の結果、桁あふれが発生してはならない。

18.2.3 定数条件式

定数条件式とは、すべての作用対象が定数であるか、定数項だけが含まれる算術式である、条件式である。定義済み条件として知られる特殊な形の条件も、定数条件式の一部として使用できる。

構文規則

1. 定数条件式は下記のどれかに該当しなければならない。
    a. 単純比較条件であって、両方の作用対象が定数、または定数項だけを含む算術式であり、***手続き部***の章の*関係条件*節の規則、および以下の規則に従って、書かれている。
        1. 比較演算子の両側の作用対象の項類が同じでなければならない。算術式の項類は数字である。
        2. 文字定数を指定した場合、比較演算子は"IS [NOT] EQUAL TO"または"IS [NOT] ="でなければならない。
    b. 定義済みの条件
    c. ***手続き部***の章の*複合条件*節で指定された複合条件で、上記の単純条件を組み合わせて構成されるもの。 省略した形の比較条件の組合せを指定してはならない。

一般規則

1. 複合条件は、***手続き部***の章の*複合条件*節の規則に従って評価される。
2. 作用対象が数字ではない単純比較条件に関しては、文字の照合順序は適用されない。1文字ずつ比較して、等しいか否かが判定される。

注: したがって、大文字と小文字は等しくないとみなされる。

18.2.4 定義済み条件

一般形式

翻訳変数名-1 IS [NOT] DEFINED

一般規則

1. IS DEFINED 構文を使用した定義済み条件は、翻訳変数名-1 が現在定義されている場合に TRUE と評価される。
2. IS NOT DEFINED 構文を使用した定義済み条件は、翻訳変数名-1 が現在定義されていない場合に TRUE と評価される。

18.2.5 EVALUATE 指令

EVALUATE 指令を使用すると、複数の分岐のある条件付き翻訳を行える。

一般形式

形式 1

```
>>EVALUATE { 定数-1 }
           { 算術式-1 }
{ >>WHEN { 定数-2 } [ { THROUGH } { 定数-3 } ] [原始テキスト-1] }...
         { 算術式-2 }   { THRU    } { 算術式-3 }
[>>WHEN OTHER [原始テキスト-2]]
>>END-EVALUATE
```

形式 2

```
>> EVALUATE TRUE
{ >>WHEN 定数条件式-1 [原始テキスト-1] }...
[>>WHEN OTHER [原始テキスト-2]]
>> END-EVALUATE
```

構文規則

すべての形式

1. 連続した COBOL の文字>>およびそれに続くコンパイラ指令語および原始テキスト-1 と原始テキスト-2 の直前までの一組の記述を、新しい行にすべて収まるように指定しなければならない。原始テキスト-1 の最初の原文語および原始テキスト-2 の最初の原文語はそれぞれ新しい行の先頭から記入しなければならない。

2. 原始テキスト-1 および原始テキスト-2 はどのような種類のものであってもよい。その中には、コンパイラ指令も含む。原始テキスト-1 および原始テキスト-2 は複数の行にわたっていてもよい。

形式 1

3. EVALUATE 指令の作用対象の項類はすべて同じでなければならない。この規則に関しては、算術式の項類は数字である。
4. THROUGH 指定を書いた場合、選択の左辺および選択の右辺はすべて、項類が数字でなければならない。
5. 語 THROUGH と THRU は同等である。

一般規則

すべての形式

1. EVALUATE 指令は COPY 文および REPLACE 文の処理の間に処理される。

形式 1

2. 選択の左辺である定数-1 または算術式-1 が各 WHEN 指定中の値と比較される。その結果は下記のようになる。
    a. THROUGH 指定がない場合は、選択の左辺が定数-2 または算術式-2 と等しいと、TRUE が返される。
    b. THROUGH 指定がある場合は、選択の左辺が定数-2 または算術式-2 以上であり定数-3 または算術式-3 以下であると、TRUE が返される。

    結果が TRUE となる WHEN 指定が見つかると、対応する原始テキスト-1 が翻訳されて、>>END-EVALUATE に至るまでの残りの行は>>END-EVALUATE も含めて無視される。

形式 2

3. 各 WHEN 指定に関して、順々に、定数条件式が評価される。

    結果が TRUE となる WHEN 指定が見つかった場合、関連する原始テキスト-1 が翻訳されて、>>END-EVALUATE に至るまでの残りの行は>>END-EVALUATE も含めて無視される。結果が TRUE となる WHEN 指定が見つからなかった場合、>>WHEN OTHER に対応する原始テキスト-2 が指定されていれば、それが翻訳される。

18.2.6 IF 指令

IF 指令は、選択肢が1つまたは2つの条件付き翻訳を可能にする。

一般形式

```
>>IF 定数条件式-1 [原始テキスト-1]
  [ >>ELSE [原始テキスト-2] ]
   >>END-IF
```

構文規則

1. 連続した COBOL の文字>>およびそれに続くコンパイラ指令語および原始テキスト-1 と原始テキスト-2 の直前までの一組の記述を、新しい行にすべて収まるように指定しなければならない。原始テキスト-1 の最初の原文語および原始テキスト-2 の最初の原文語はそれぞれ新しい行の先頭から記入しなければならない。
2. 原始テキスト-1 および原始テキスト-2 はどのような種類のものであってもよい。その中には、コンパイラ指令も含む。原始テキスト-1 および原始テキスト-2 は複数の行にわたっていてもよい。

一般規則

1. IF 指令は COPY 文および REPLACE 文の処理の間に処理される。
2. 定数条件式-1 の評価結果が TRUE であると、原始文の一部として原始テキスト-1 が翻訳されて、原始テキスト-2 は無視される。
3. 定数条件式-1 の評価結果が FALSE であると、原始テキスト-1 は無視されて、原始テキスト-2 が指定されていると、原始文の一部として翻訳される。

18.2.7 REPOSITORY 指令

REPOSITORY 指令は外部リポジトリに情報を追加するかどうかを指定する。また、外部リポジトリ中のインターフェイス情報に照らして、プロトタイプおよび定義をチェックするかどうかも指定する。

一般形式

```
>> REPOSITORY UPDATE { ON
                    { OFF [WITH CHECKING] }
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
   - RDFPATH - リポジトリ・ファイル用の登録集の所在を指定する。

構文規則

1. REPOSITORY 指定を書ける場所は、翻訳単位の最初の見出し部の前だけである。翻訳単位の中に REPOSITORY 指定を書いてはならない。
2. REPOSITORY 指令の効力は後続の翻訳集団に及ぶ。他の REPOSITORY 指定が出てくると、新しい指定が発効する。

一般規則

1. 省略時の REPOSITORY 指定は下記のとおり。

```
>>REPOSITORY UPDATE OFF WITH CHECKING.
```

2. ON 指定を明示的または暗黙的に指定すると、翻訳単位に関する情報が外部リポジトリに追加される。それから、次の翻訳単位の翻訳が行われる。次の翻訳単位がない場合には、外部リポジトリに情報を追加した後で、翻訳が終了する。
3. OFF 指定を書くと、外部リポジトリは更新されない。
4. CHECKING 指定を書くと、外部リポジトリ中のソース単位に関する情報と異なるクラス定義またはインターフェイス定義に警告のフラグが付けられる。外部リポジトリ中の情報の詳細については、*翻訳集団の概念*の章の*外部リポジトリ*節に指定されている。

18.3 BASIS 機構

OSVS OSC2

BASIS 機構は、COBOL の原始ファイルを非対話的に (一時的に) 編集し、その結果を COBOL システムに投入できるようにする。 一時的編集の源として参照される COBOL の原始ファイル ("編集元プログラム") は、変更されることはない。また、COBOL システムによって出力されたファイル (リストなど) 以外には、編集結果の記録は残らない。結果として出力されるコード・ファイルは、参照する原始ファイルが存在しないため、アニメートすることはできない。

BASIS 機構は固定形式の原始ファイルにだけ有効である。

COBOL システムに投入されるプログラムは、2 つのファイルから構成される。1 つは編集制御ファイル (edit control file) (BASIS 文と編集情報を記録している) であり、もう 1 つは COBOL 原始ファイル (上記の"編集元プログラム") である。

BASIS 機構の中で特別な働きをする、以下の 3 つの文がある。

1. BASIS
2. INSERT
3. DELETE.

これらの BASIS 機構用の文は、COBOL 言語の一部を構成するものではない。これらの文は 1 行に完全に収まっていなければならず、大文字で記さなければならない。

INSERT 文または DELETE 文を使用して、BASIS 文によって用意された COBOL の原始プログラムを変更する場合、COBOL 原始プログラムの一連の項目は、昇順に一連番号を持っていなくてはならない。

一般規則

1. 原始プログラム行の B 領域に完全に含まれ、その後ろに有効な編集元順序番号または編集元順序番号の範囲が続かない DELETE 文は、COBOL システムによって COBOL の DELETE 文として扱われる。
2. BASIS 機構の DELETE 文で、編集元順序番号または編集元順序番号の範囲は、すべて数字の昇順になっていなければならない。
3. 編集制御ファイル内では、編集元順序番号または編集元順序番号の範囲は、すべて数字の昇順になっていなければならない。

18.3.1 BASIS 文

OSVS OSC2

機能

BASIS 文は、該当するプログラム（編集制御ファイルと COBOL 原始ファイル）を BASIS 機構の規則に従って、COBOL システムに投入することを示す。

一般形式

```
[順序番号] BASIS {原文名
                 外部ファイル名定数}
```

構文規則

1. BASIS 文は、編集制御ファイルの最初の行に置く。
2. 予約語 BASIS は、その文を記述する行のカラム 1 から 66 の間のどこから書き始めてもよい。
3. 順序番号は、その文を記述する行のカラム 1 から 6 の間のどこに書いてもよい。その後ろには空白文字を 1 つ置く。
4. 原文名は、一意の外部ファイル名を定義する。このファイル名は、利用者語の作成規則に準拠していること（小文字は大文字に変換されることに注意）。外部ファイル名定数は、引用符で囲んだ文字定数である。ファイル名に関するオペレーティングシステムの規則に準拠していること。

一般規則

1. 原文名または外部ファイル名定数は、編集制御ファイルによって編集する COBOL 原始ファイルを指名する。
2. COBOL 原始ファイルの編集は、編集制御ファイル中の INSERT 文および DELETE 文によって行われる。

18.3.2 DELETE(削除)文 - BASIS 制御

OSVS OSC2

機能

DELETE (削除) 文は、(BASIS 機構のもとで) COBOL システムに無視させる COBOL 原始ファイル中の行を指定する。DELETE 文の後ろに続く COBOL 文は、(編集制御ファイル中に次の BASIS 機構の INSERT 文または DELETE 文が出てくるまで) 原始ファイル中に残される。

一般形式

```
[順序番号] DELETE

{編集元順序番号-1          }[, {編集元順序番号-2          }]...
{編集元順序番号の範囲-1    }   {編集元順序番号の範囲-2    }

    [COBOL原始プログラムの文]...
```

構文規則

1. 予約語 DELETE は、その文を記述する行のカラム 1 から 6 の間のどこから書き始めてもよい。
2. 順序番号は、その文を記述する行のカラム 1 から 6 の間のどこに書いてもよい。その後ろには空白文字を 1 つ置く。
3. 編集元順序番号-1 や編集元順序番号-2 などは、引用符で囲まない 6 文字の正の整数とする (整数数字定数に関する規則に従う)。
4. 編集元順序番号の範囲-1 や編集元順序番号の範囲-2 などは、2 つの編集元順序番号 (上記) をハイフン (-) でつないだものとする。
5. 編集元順序番号または編集元順序番号の範囲の間のコンマは、必須である。

一般規則

1. 編集元順序番号-1 や編集元順序番号-2 などは、COBOL 原始ファイル中の文のうち COBOL システムに無視させるものの順序番号を表わす。
2. 編集元順序番号の範囲-1 や編集元順序番号の範囲-2 などは、COBOL 原始ファイル中の文のうち中間コードを生成するときに、COBOL システムに無視させるものの順序番号の範囲を表わす。この範囲は両端を含む。
3. 編集制御ファイル中に次の BASIS 機構の INSERT 文または DELETE 文が出てくるまで、DELETE 文の後ろに続く COBOL 文は、COBOL システムに投入される原始ファイル中に残される。それらの COBOL 原始プログラムの文は、BASIS 機構の DELETE 文によって削除された最後の文に代わって、挿入される。

18.3.3 INSERT(挿入)文 - BASIS 制御

OSVS VSC2

機能

INSERT (挿入) 文は、(BASIS 機構のもとで) COBOL システムに投入するプログラム中に挿入する、COBOL 原始プログラムの行をリストする。

一般形式

```
[順序番号] INSERT編集元順序番号
COBOL 文[COBOL 文]
```

構文規則

1. 予約語の INSERT は、その文を記述する行のカラム 1 から 66 の間のどこからでも書き始めることができる。.
2. 編集元順序番号は、引用符で囲まない正の整数とする (整数数字定数に関する規則に従う) 。

一般規則

1. 編集元順序番号は、COBOL 原始ファイル中のどの文の後ろに COBOL 文を挿入するかを示す。
2. 編集制御ファイル中の INSERT 文の後ろに続くすべての COBOL 文は、次の BASIS 機構の DELETE 文または INSERT 文が出てくるまで、COBOL システムに投入される原始プログラム中に含められる。BASIS 機構の INSERT 文の直後には、COBOL 文が少なくとも 1 行はあること。

18.4 ++INCLUDE 機構および-INC 機構

OSVS VSC2

++INCLUDE 機構および-INC 機構は、COBOL プログラムの原始ファイルまたは原始コードの一部をコンパイル対象のファイルに組み込めるようにする、もう 1 つの (IBM メインフレーム互換の) 手段である。

18.4.1 -INC 文

OSVS VSC2

機能

-INC 文は、コンパイル時に、ある原始ファイルのすべてのデータ・レコードを別の原始ファイルに組み込むために使用する。

一般形式

```
-INC 原文名
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - LIBRARIAN - -INC の構文を使用できるようにする。

構文規則

1. -INC はカラム 1 から書き始め、後ろに 1 つ以上の空白文字を置く。
2. 原文名は、一意の外部ファイル名を定義する。このファイル名は利用者語の作成規則に準拠していること。
3. -INC 文は、COBOL 言語の一部を構成するものではない。これらの文は 1 行に完全に収まっていなければならず、大文字で記さなければならない。

一般規則

1. 原文名は、原始コード中でこの文を書いた場所に組み込む、COBOL 原始ファイルを指名する。
2. この行に原文名以外のテキストを書くと、注記として扱われる。

18.4.2 ++INCLUDE 文

OSVS VSC2

機能

++INCLUDE 文は、コンパイル時に、ある原始ファイルのすべてのデータ･レコードを別の原始ファイルに組み込むために使用する。

一般形式

```
++INCLUDE 原文名
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - PANVALET - ++INCLUDE の構文を使用できるようにする。

構文規則

1. ++INCLUDE はカラム 8 から書き始め、 すべて大文字とし、後ろに 1 つ以上の空白文字を置く。
2. 原文名は、一意の外部ファイル名を定義する。このファイル名は、利用者語の作成規則に準拠していること。
3. ++INCLUDE 文は、COBOL 言語の一部を構成するものではない。これらの文は 1 行に完全に収まっていなければならず、大文字で記さなければならない。

構文規則

1. 原文名は、原始コード中でこの文を書いた場所に組み込む、COBOL 原始ファイルを指名する。
2. この行に原文名以外のテキストを書くと、注記として扱われる。

18.5 条件付き翻訳

MF

COBOL システムには、COBOL 原始コードの一部またはすべてを選択してコンパイルする機能が備わっている。条件付き翻訳の利点を十分に活かすためには、78 レベル(*データ部 - ファイルおよびデータ記述*の章の*レベル番号*節を参照) および CONSTANT コンパイラ指令を使用するのがよい。条件付き翻訳を行うには、$IF、$ELSE、$END の構造を用いる。この構造は、COBOL IF 構造と同様の働きをする。

構文規則

1. 条件付き翻訳の文は、固定形式原始コードのカラム 7 にドル記号($)を書き、その後ろに IF、DISPLAY、ELSE、END のどれかを続けたものである。
2. 条件付き翻訳は、COBOL の文字列を分ける形で使用してはならない。つまり、継続する行の間に、条件付き翻訳制御を割り込ませることはできない。

18.5.1 $DISPLAY 文

MF

機能

原始行の処理中に$DISPLAY 文が出て来ると(条件に従って除外された原始行を含まない)、標準の出力装置上にテキスト･データが表示される。

一般形式

形式 1

$DISPLAY テキストデータ

形式 2

$DISPLAY VCS = version-number

構文規則

1. $DISPLAY 文全体を 1 行に収めなければならない。

一般規則

1. $DISPLAY 文が、条件付き翻訳では無視される原始行で検出された場合は、翻訳時間も実行時効果も存在しない。

形式 1

2. 原文データは、翻訳中に標準出力装置で表示される。

形式 2

3. バージョン番号は、"="に続く、原始行全体の内容とする。ただし、前後の空白を除く。
4. 作成されるオブジェクトファイルの種類にかかわらず、"@(#)"の連結により形成される文字列、バージョン番号、および空文字（二進数のゼロ）のいずれかがオブジェクトファイルに含まれる。バージョン番号が"@(#)"という文字の並びで始まっている場合、これらの文字は、文字列の形成時には連結されない。

注：

- バージョン番号はどんな文字列でもよい。ただし、これに含まれるバージョン番号を探して、UNIX SCC の what ユーティリティなどのパターン検出ツールはオブジェクトファイルを検索する。
- 最適化の段階で、到達不可能な手続き符号は削除され、この符号以外には使用されない定数は取り除かれる。これらを含む、手続き部の文中のバージョン番号の、以下の方法を使用した動作は無効となる。
- ファイル名-what 節.
- move "@(#)ファイル名 %I%>" to what-文字列
  perform ファイル名-what

- $DISPLAY 文は原始符号のどこにでも指定できるため、各原文名（copybook）にはそれぞれ、独自のバージョン番号を使用できる。
- 以前のバージョンのコンパイラは、形式 2 $DISPLAY 文を形式 1 $DISPLAY として取り扱う。

18.5.2 $ELSE 文

MF

機能

$ELSE 文は、$IF 文とともに、条件付き翻訳を制御するために使用される。

一般形式

```
$ELSE
```

構文規則

1. 文全体を 1 行に収めなければならない。

一般規則

1. 最も近い$IF 文条件の逆を適用する。最も近い$IF 文が真と評価されると、$ELSE 文の後ろに続く原始行が処理される。最も近い$IF 文が偽と評価されると、次の条件付翻訳行が出て来るまで、COBOL 原始行は無視される。

18.5.3 $END 文

MF

機能

$END 文は、$IF 文とともに、条件付き翻訳を制御するために使用される。

一般形式

```
$END
```

構文規則

1. 文全体を 1 行に収めなければならない。

一般規則

1. 最も内側の$IF 文条件を終了させる。次いで、その時点で働いている$IF 文に従って、処理が行われる。この条件が真と評価されると、$END 文の後ろに続く原始行が処理

される。この条件が偽と評価されると、次の条件翻訳行が出て来るまで、COBOL 原始行は無視される。

18.5.4 $IF 文

(MF)

機能

$IF 文は原始文の選択した部分を翻訳に含めないようにする機能を提供する。

例

1. *言語リファレンス - 追加トピック の 例*の章の*$IF 文 (条件付き翻訳)*節に条件付き翻訳の使用例が掲げられている。

一般形式

形式 1

```
$IF 定数名-1 [NOT] { < } 定数-1
                   { > }
                   { = }
```

形式 2

```
$IF 定数名-2 [NOT] DEFINED
```

形式 3

```
$IF directive-setting SET
```

構文規則

1. 定数名-1 は、レベル 78 の記述項または CONSTANT コンパイラ指令によって定義される。
2. 定数-1 が数字である場合、正の整数またはゼロでなければならない。
3. 指令設定は、$SET 文での指定と同じ形式で指定しなければならない。また、NO の後に続いて指定してもよい。ただし、$IF 文で使用される形式は、$SET で使用される形式とは以下の点で異なる。
    a. NO と指令名の間には空白を入れてはならない
    b. 指令中のハイフンは、正確に指定しなければならない。

c. 指令のパラメータは、省略してもよい

4. 指令設定に対して指定できるのは、構文チェックの段階で使用され、指令設定リスト中に作成される (コンパイラ指令 SETTINGS を参照) 指令のみである。
5. 文全体を 1 行に収めなければならない。
6. $IF 文は、他の$IF 文の中に入れ子にすることができる。

一般規則

1. レベル 78 の記述項または CONSTANT コンパイラ指令の左辺である場合、定数名-2 は DEFINED であり、そうでない場合は NOT DEFINED である。
2. 設定が実際の指令設定と一致すると、指令の設定 SET は真と評価される。このとき、パラメータが指定されていれば、これも一致しなければならない。
3. 指令設定と実際の指令設定の比較においては、大文字と小文字は区別されない。
4. 条件が真と評価されると、$IF 文の後ろに続く原始行が処理される。条件が偽と評価されると、次の条件付き翻訳行が出て来るまで、COBOL 原始行は無視される。

18.6 リスト制御文

OSVS VSC2 MF

リスト制御文 (listing control statement) は、コンパイル処理中に出力ファイルのリストの作成を制御する。

リスト制御文には下記の 3 種類がある。

1. EJECT
2. SKIP1、SKIP2 および SKIP3
3. VSC2 MF TITLE.

18.6.1 EJECT(改ページ)文

OSVS VSC2 MF

機能

EJECT (改ページ) 文は、原始コードの次の行を次のページの冒頭に印刷するように、COBOL システムに指示する。

一般形式

```
EJECT [.]
```

構文規則

1. EJECT 文は、A 領域または B 領域から書き始めてよいが、1 行に単独で書く。最後に終止符を付けなくてもよい。

一般規則

1. EJECT 文自体は印刷されない。

18.6.2 SKIP1 文、SKIP2 文および SKIP3 文

OS/VS VSC2 MF

機能

SKIP1、SKIP2、および SKIP3 の各文は、COBOL システムによって作成される原始コードのリストの縦の間隔を制御する。これらは、原始コードのリスト中で空ける行間隔を指定する。

一般形式

```
{SKIP1}
{SKIP2} [ ]
{SKIP3}
```

構文規則

1. これらの文は、A 領域または B 領域から書き始めてよいが、1 行に単独で書かなければならない。

   VSC2 MF 最後に終止符を付けても付けなくてもよい。

一般規則

1. SKIP1 は、COBOL システムに 1 行間隔を空けるように (2 行改行) 指示する。SKIP2 は、COBOL システムに 2 行間隔を空けるように (3 行改行) 指示する。SKIP3 は、COBOL システムに 3 行間隔を空けるように (4 行改行) 指示する。
2. SKIP 文自体は印刷されない。

18.6.3 TITLE(標題)文

VSC2 MF

機能

TITLE (標題) 文は、リストの後続のすべての先頭行に印刷する標題を、COBOL システムに知らせる。

一般形式

TITLE 定数-1 [.]

構文規則

1. 定数-1 は文字とする。後ろに終止符を付けてもよい。表意定数であってはならない。
2. 語 TITLE は、A 領域または B 領域から書き始めてよいが、1 行に単独で書かなければならない。
3. TITLE 文は、翻訳集団中のどこに書いてもよい。

一般規則

1. 定数-1 は、リストの後続のすべてのページに付ける標題として使用される。TITLE 指令が出てくるまでは、省略時解釈の標題が使用される。省略時解釈の標題には、使用されている COBOL システムの識別情報と現在のリリース番号が示される。
2. 指定した標題または省略時解釈の標題は、各ページの先頭行の左側に寄せて印刷される。その行の右側には、中間コードが生成された日付と時刻およびページ番号が表示される。
3. 2 番目の標題行も出力される。この行には、主原始ファイルおよび最新の複写ファイルの名前が表示される。
4. TITLE 文を指定すると、すぐに改ページが行われる。
5. TITLE 文自体は印刷されない。

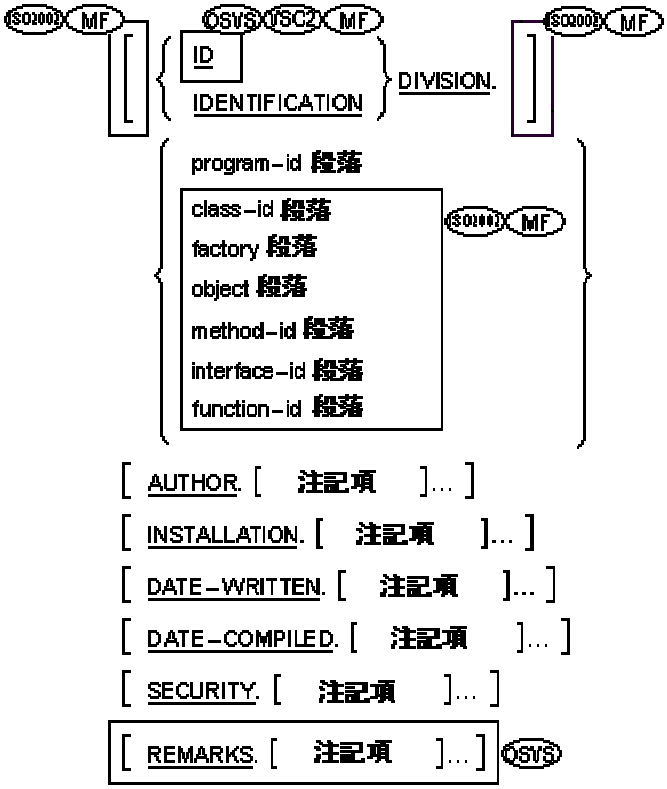
18.7 各 COBOL システム文に固有のコンパイラ指令

Net Express など、各 COBOL システムにより多数のコンパイラ指令が提供されている。詳しくは、各 COBOL システムのマニュアルを参照。

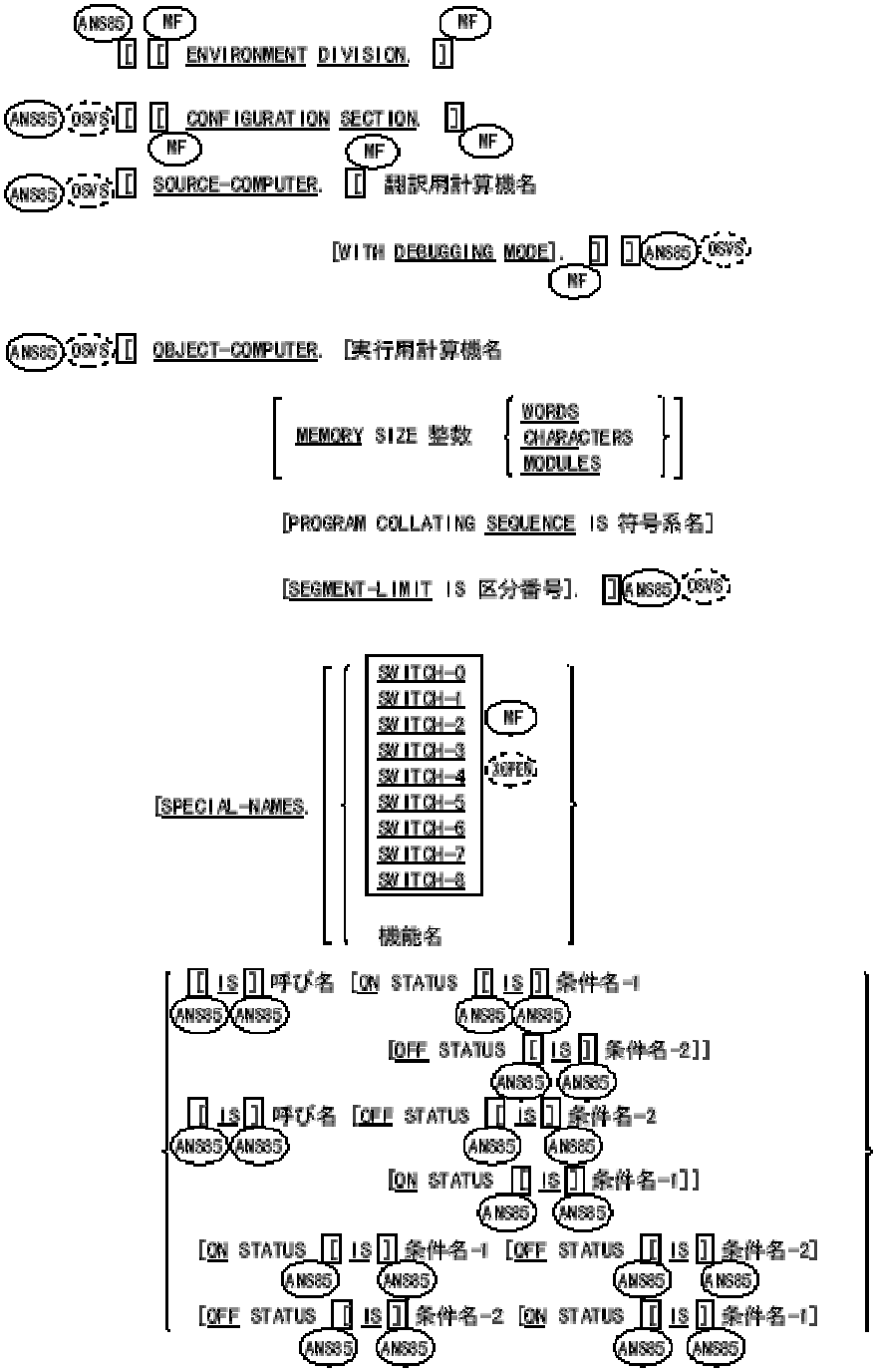
付録 A : COBOL 標準機能用の一般形式の一覧

この付録では、この COBOL システムの構文を列挙する。この中には、Data General Interactive COBOL、Ryan McFarland COBOL、Microsoft COBOL と互換性をもたせることだけを目的として用意されている構文は含まない。これらの情報に関しては、*COBOL 言語リファレンス - 追加トピック*の *Data General Interactive COBOL V1.3 構文支援機能*、*Ryan McFarland COBOL V2.0 構文支援機能*、*Siemens BS2000 COBOL 構文支援機能* および *Microsoft COBOL V1.0 and V2.0 構文支援機能*の各章を参照。

A.1 見出し部の一般形式

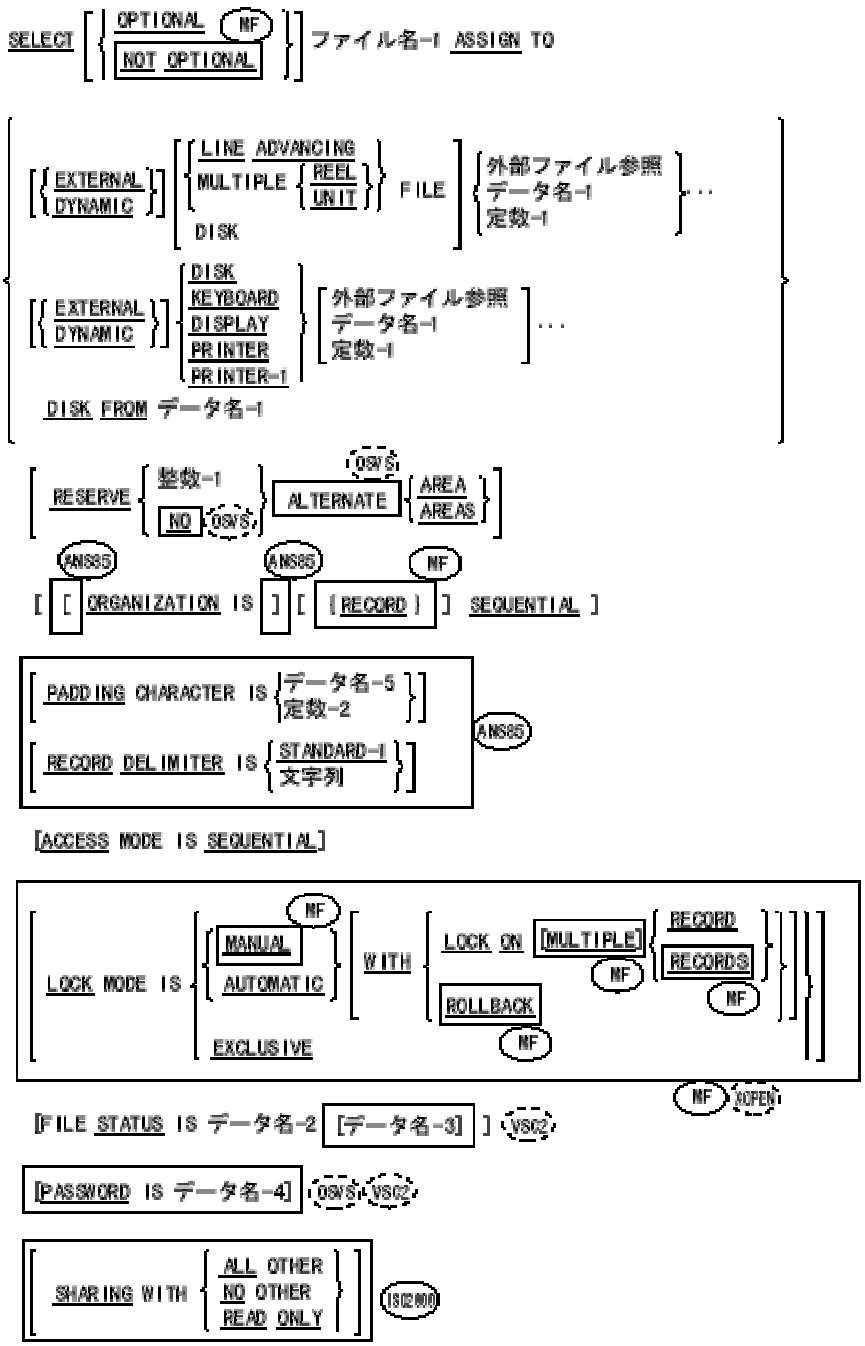


A.2 環境部の一般形式


ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. 翻訳用計算機名
[WITH DEBUGGING MODE].
OBJECT-COMPUTER. [実行用計算機名
MEMORY SIZE 整数 WORDS CHARACTERS MODULES
[PROGRAM COLLATING SEQUENCE IS 符号系名]
[SEGMENT-LIMIT IS 区分番号].
[SPECIAL-NAMES.
SWITCH-0 SWITCH-1 SWITCH-2 SWITCH-3 SWITCH-4 SWITCH-5 SWITCH-6 SWITCH-7 SWITCH-8
機能名
呼び名 [ON STATUS 条件名-1
[OFF STATUS 条件名-2]]
呼び名 [OFF STATUS 条件名-2
[ON STATUS 条件名-1]]
[ON STATUS 条件名-1 [OFF STATUS 条件名-2]
[OFF STATUS 条件名-2 [ON STATUS 条件名-1]
ALPHABET 符号系名-1 IS
STANDARD-1 STANDARD-2 NATIVE ASCII EBCDIC
定数-1 THROUGH THRU 定数-2
[ALSO 定数-3]...
[SYMBOLIC CHARACTERS {記号文字-1}... IS ARE
{整数-1}...}... [IN 符号系名-2]]...

A.2.1 ファイル管理記述項の一般形式

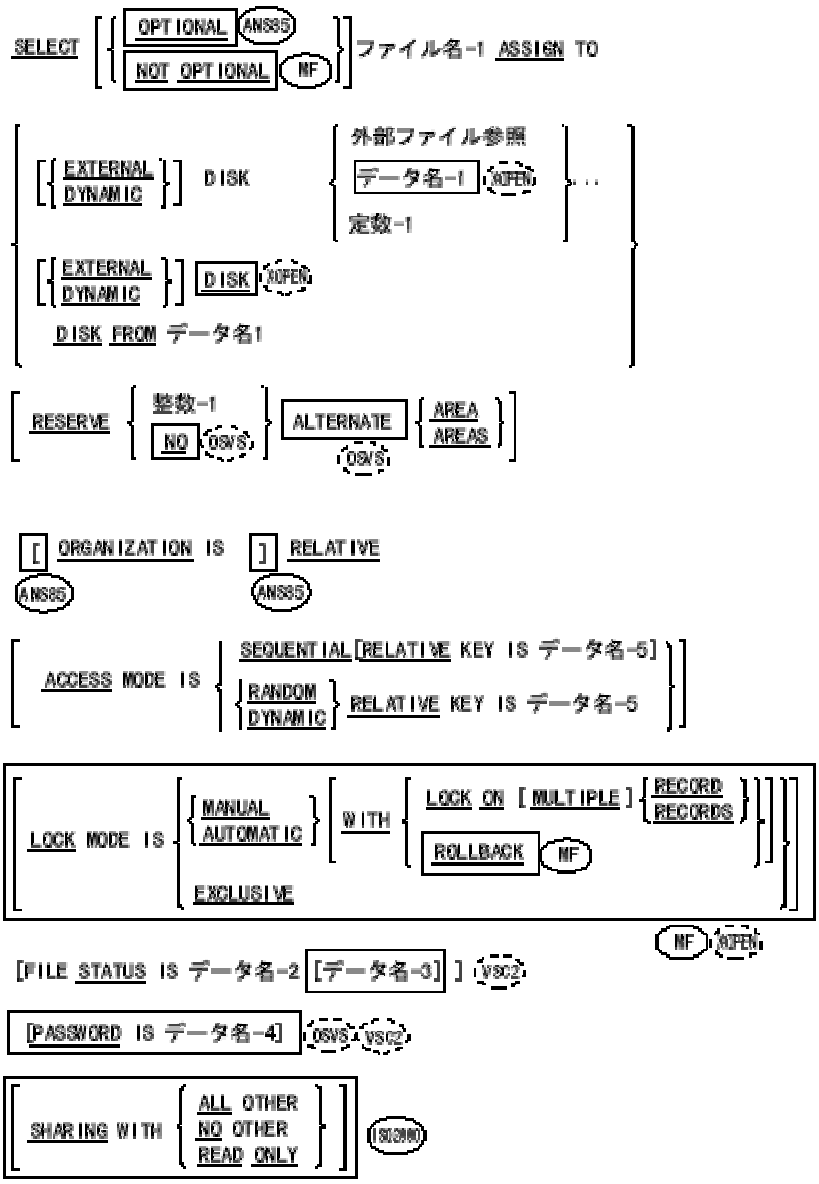
A.2.1.1 レコード順ファイル



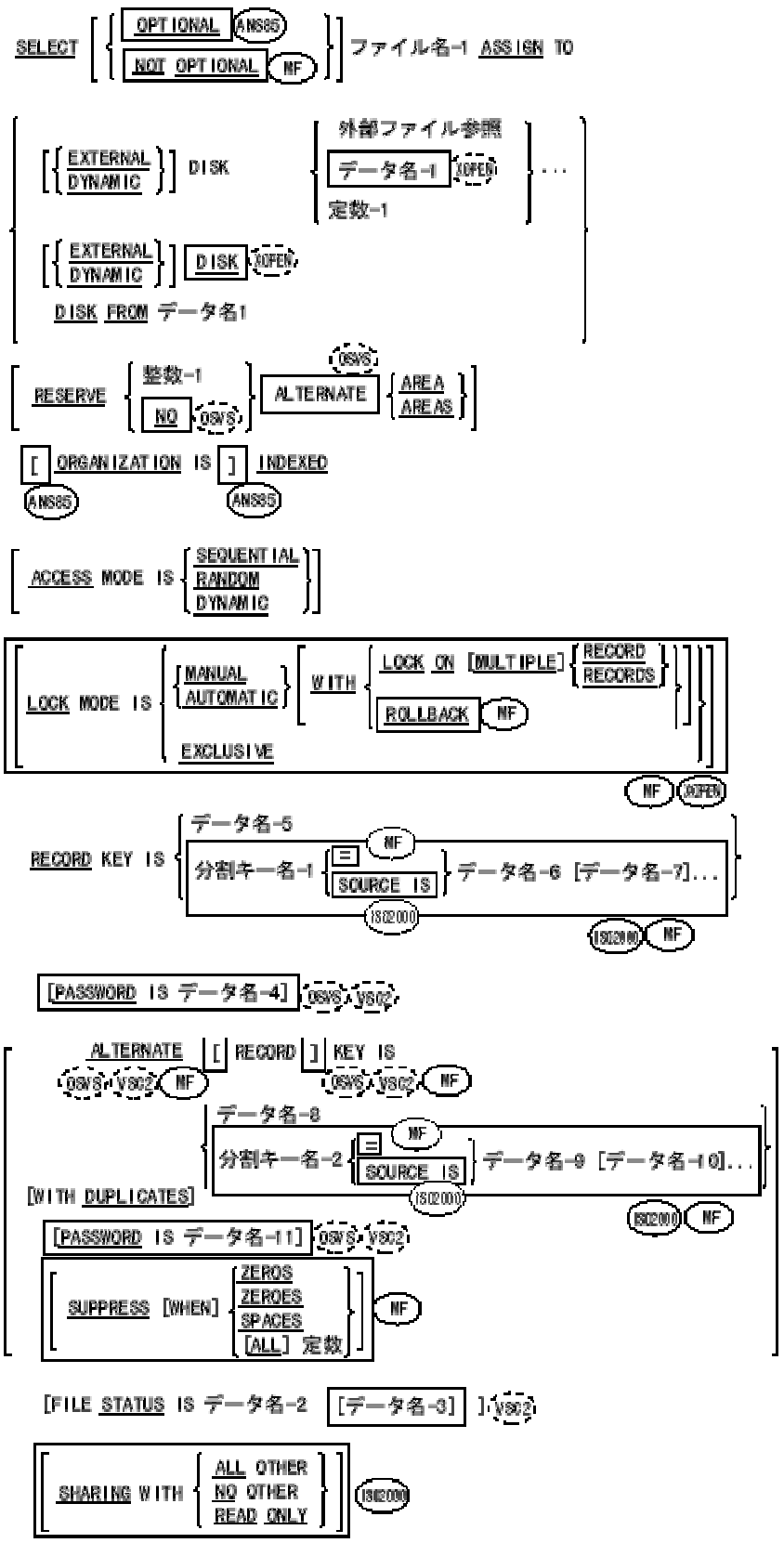
A.2.1.2 行順ファイル

```
SELECT [ { OPTIONAL } ] ファイル名-1 ASSIGN TO                                   (NF)
         { NOT OPTIONAL }                                                        (SOPEN)
                                                                                 (ISO2000)
{ [ { EXTERNAL } ] [DISK FILE] { 外部ファイル参照 } ...
    { DYNAMIC  }               { データ名-1       }
                               { 定数-1           }
  [ { EXTERNAL } ] { DISK     } [ 外部ファイル参照 ] ...
    { DYNAMIC  }   { KEYBOARD } [ データ名-1       ]
                   { DISPLAY  } [ 定数-1           ]
                   { PRINTER  }
                   { PRINTER-1 }
  DISK FROM データ名-1                                                           }

[  [ ORGANIZATION IS ] [ LINE ] SEQUENTIAL ]

[ACCESS MODE IS SEQUENTIAL]

[ LOCK MODE IS { MANUAL   } [ WITH { LOCK ON [MULTIPLE] { RECORD  } } ] ]
               { AUTOMATIC }       { ROLLBACK           { RECORDS } }   ]
               { EXCLUSIVE }                                            ]

[FILE STATUS IS データ名-2]
[PASSWORD IS データ名-4]
[ SHARING WITH { ALL OTHER } ]   (ISO2000)
               { NO OTHER  }
               { READ ONLY }
```

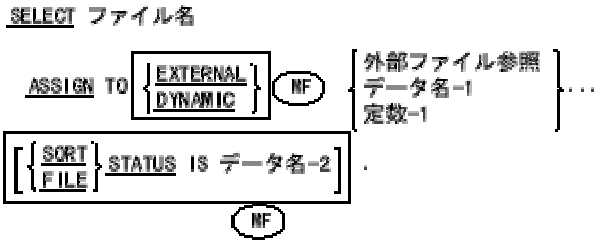
A.2.1.3 相対ファイル


SELECT
OPTIONAL
ANS85
NOT OPTIONAL
NF
ファイル名-1 ASSIGN TO
EXTERNAL
DYNAMIC
DISK
外部ファイル参照
データ名-1
NOPEN
定数-1
DISK FROM データ名1
RESERVE
整数-1
NO
OSVS
ALTERNATE
AREA
AREAS
ORGANIZATION IS
RELATIVE
ACCESS MODE IS
SEQUENTIAL
RELATIVE KEY IS データ名-5
RANDOM
LOCK MODE IS
MANUAL
AUTOMATIC
WITH
LOCK ON
MULTIPLE
RECORD
RECORDS
ROLLBACK
EXCLUSIVE
FILE STATUS IS データ名-2
データ名-3
VSC2
PASSWORD IS データ名-4
SHARING WITH
ALL OTHER
NO OTHER
READ ONLY
ISO2001

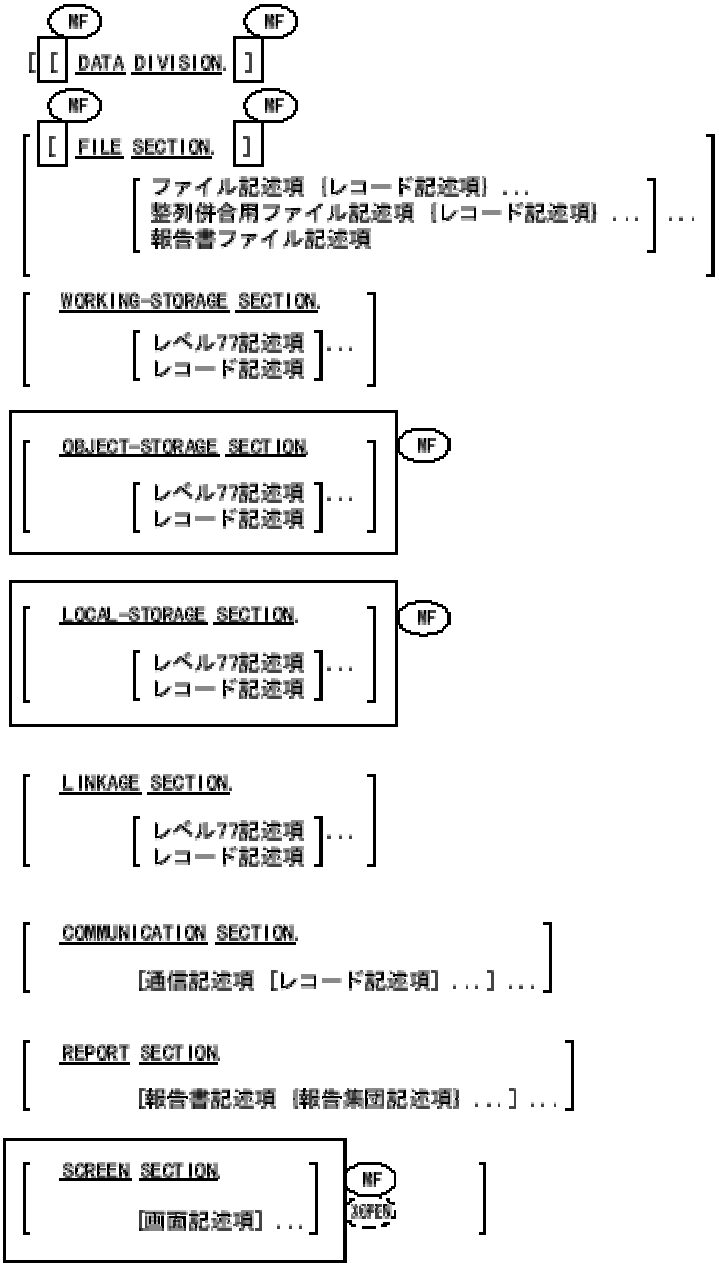
A.2.1.4 索引ファイル



A.2.1.5 整列併合ファイル



A.3 データ部の一般形式



A.3.1 ファイル記述項の一般形式

A.3.1.1 レコード順ファイル

```
FD ファイル名-1
  [IS EXTERNAL]
  [IS GLOBAL]   (ANS85)
[ [BLOCK CONTAINS [整数-1 TO] 整数-2 { RECORDS }] ]   (ANS85)
                                      { CHARACTERS }
[ RECORD { CONTAINS 整数-3 CHARACTERS                                  } ]
         { IS VARYING IN SIZE [[FROM 整数-4] [TO 整数-5]              }
                   CHARACTERS] [DEPENDING ON データ名-1]
         { CONTAINS [整数-6 TO] 整数-7 CHARACTERS                      }
[[ LABEL { RECORD IS } { STANDARD }                                  ]]
  (ANS85)  { RECORDS ARE } { OMITTED }                                (ANS85)
           {             } { {データ名-2}...  }
                             (OSVS)(VSC2)
[ VALUE OF { 作成者語-1 IS { データ名-3 }...                          } ]
           {               { 定数-1     }
           { FILE-ID IS { データ名-6 } (NF)                           }
           {            { 定数-3    }
[[ RECORDING MODE IS { F                                             ]]  (OSVS)
                     { V                                             }  (VSC2)
                     { U                                             }  (NF)
                     { S                                             }
                     { FIXED                                         }
                     { VARIABLE (NF)                                 }
[ DATA { RECORD IS } データ名-7 [データ名-8]... ]
       { RECORDS ARE }
[ LINAGE IS { データ名-9 } LINES [WITH FOOTING AT { データ名-10 }] ]
            { 整数-8     }                        { 整数-9     }
[ LINES AT TOP { データ名-11 }] [[ LINES AT BOTTOM { データ名-12 }]]
               { 整数-10     }                     { 整数-11     }
[CODE-SET IS 符号系名 [FOR {一意名-1}...] ] (NF)
 {レコード記述項}...
```

A.3.1.2 行順ファイル

```
FD ファイル名-1                                                    NF
  [IS EXTERNAL]                                                    NOFEN
  [IS GLOBAL]   ANS85
[ BLOCK CONTAINS [整数-1 TO] 整数-2 { RECORDS }]                   ANS85
                                    { CHARACTERS }
[ RECORD { CONTAINS 整数-3 CHARACTERS                              }]
         { IS VARYING IN SIZE [[FROM 整数-4] [TO 整数-5]           }
         {        CHARACTERS] [DEPENDING ON データ名-1]            }
         {   CONTAINS [整数-6 TO] 整数-7 CHARACTERS                }
[[ LABEL { RECORD IS } { STANDARD }[ ]]
         { RECORDS ARE } { OMITTED }[ ]
         ANS85           { [データ名-2]... }      ANS85
                          OSVS  VS02
[ VALUE OF { 作成者語-1 IS { データ名-3 }...                       }]
           {               { 定数-1     }                         }
           { FILE-ID IS { データ名-6 }                            }
           {            { 定数-3     }                            }
[ RECORDING MODE IS { F }                                          }]
                    { V }
                    {   }
                    { FIXED }
                    { VARIABLE }
[ DATA { RECORD IS } データ名-7 [データ名-8]... ]
       { RECORDS ARE }
[CODE-SET IS 符号系名 [FOR {一意名-1}...]]
  {レコード記述項}...
```

A.3.1.3 相対ファイルおよび索引ファイル

```
FD ファイル名-1
  [IS EXTERNAL]
  [IS GLOBAL]   (ANS85)
  [ BLOCK CONTAINS [整数-1 TO] 整数-2 { RECORDS }  ]   (ANS85)
                                      { CHARACTERS }
  [ RECORD { CONTAINS 整数-3 CHARACTERS                                  }  ]
           { IS VARYING IN SIZE [[FROM 整数-4] [TO 整数-5]               }
           {        CHARACTERS] [DEPENDING ON データ名-1]                }
           { CONTAINS [整数-6 TO] 整数-7 CHARACTERS                      }
  [[ LABEL { RECORD IS } { STANDARD }  ]]
           { RECORDS ARE } { OMITTED }
           { [データ名-2]... }
  (ANS85)                (OSVS) (VSC2)                (ANS85)
  [ VALUE OF { 作成者語-1 IS { データ名-3 }...  }  ]
             {               { 定数-1     }
             { FILE-ID IS { データ名-6 }  (NF)  }
             {            { 定数-3     }
  [ RECORDING MODE IS { F }  ]
                      { V }
                      { U }
                      { S }
                      { FIXED    }  (NF)
                      { VARIABLE }
  [ DATA { RECORD IS } データ名-7 [データ名-8]... ]
         { RECORDS ARE }
      [レコード記述項]...
```

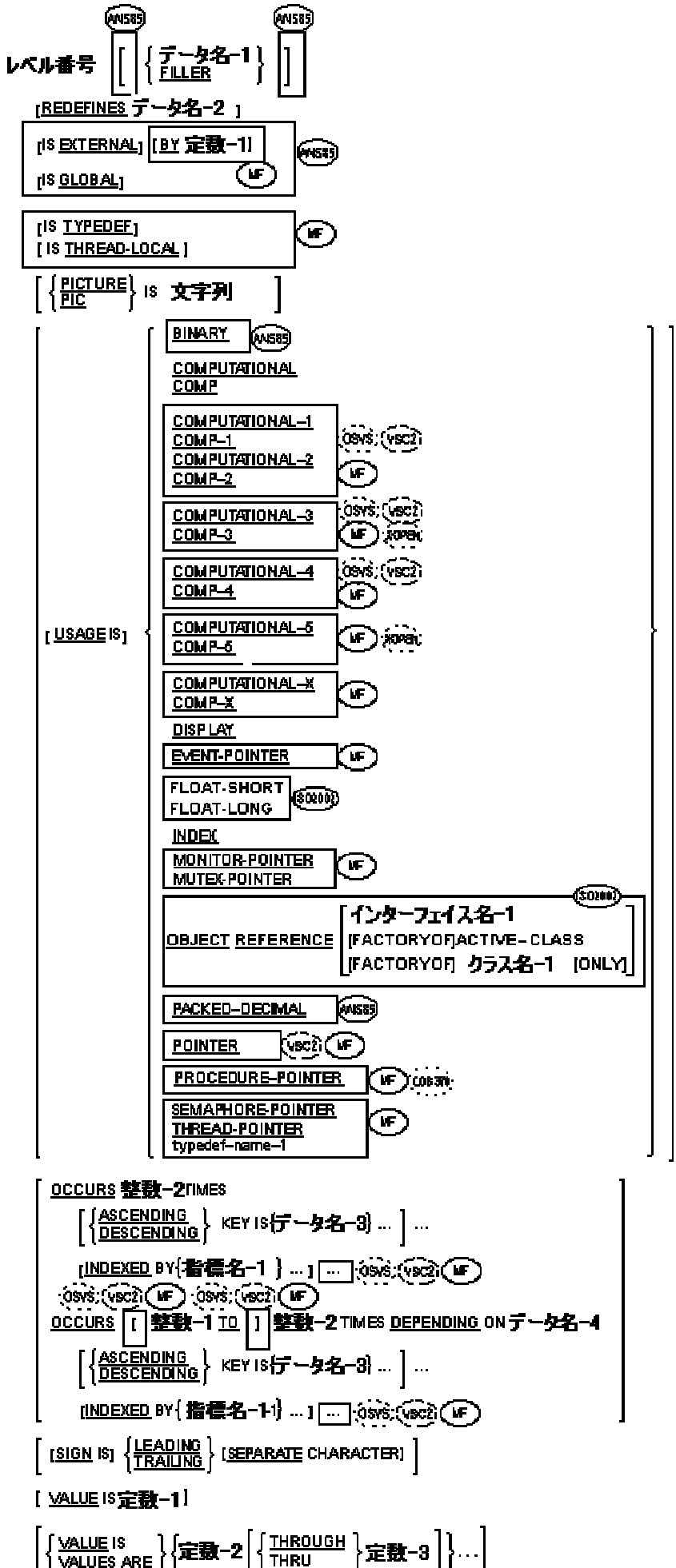
A.3.1.4 整列併合ファイル

```
SD ファイル名-1                                          (OSVS) (VS02)
[BLOCK CONTAINS [整数-1 TO] 整数-2 {RECORDS   }]
                                   {CHARACTERS}
                                                          (ANS85)
[ RECORD { [ { CONTAINS 整数-3 CHARACTERS                 }  ] }  ]
         { [ { IS VARYING IN SIZE [[FROM 整数-4] [TO 整数-5]} ] }
         { [ {    CHARACTERS] [DEPENDING ON データ名-1]   } ] }
         { CONTAINS [整数-6 TO] 整数-7 CHARACTERS            }
[ LABEL {RECORD IS  } { STANDARD  } ]  (OSVS)
        {RECORDS ARE} { OMITTED   }    (VS02)
                      { {データ名-2}... }
[ DATA {RECORD IS  } {データ名-3}... ]
       {RECORDS ARE}
[ VALUE OF FILE-ID IS {データ名-4} ]  (NF)
                      {定数-1     }
```

A.3.1.5 報告書ファイル

A.3.2 データ記述項の一般形式

形式 1


レベル番号
データ名-1
FILLER
ANS85
REDEFINES データ名-2
IS EXTERNAL
BY 定数-1
IS GLOBAL
MF
IS TYPEDEF
IS THREAD-LOCAL
PICTURE
PIC
IS 文字列
USAGE IS
BINARY
COMPUTATIONAL
COMP
COMPUTATIONAL-1
COMP-1
COMPUTATIONAL-2
COMP-2
OSVS
VSC2
COMPUTATIONAL-3
COMP-3
XOPEN
COMPUTATIONAL-4
COMP-4
COMPUTATIONAL-5
COMP-5
COMPUTATIONAL-X
COMP-X
DISPLAY
EVENT-POINTER
FLOAT-SHORT
FLOAT-LONG
SO2000
INDEX
MONITOR-POINTER
MUTEX-POINTER
OBJECT REFERENCE
インターフェイス名-1
FACTORY OF ACTIVE-CLASS
FACTORY OF クラス名-1 ONLY
PACKED-DECIMAL
POINTER
PROCEDURE-POINTER
COBOL2002
SEMAPHORE-POINTER
THREAD-POINTER
typedef-name-1
OCCURS 整数-2 TIMES
ASCENDING
DESCENDING
KEY IS データ名-3 ...
INDEXED BY 指標名-1 ...
OCCURS 整数-1 TO 整数-2 TIMES DEPENDING ON データ名-4
SIGN IS
LEADING
TRAILING
SEPARATE CHARACTER
VALUE IS 定数-1
VALUE IS
VALUES ARE
定数-2
THROUGH
THRU
定数-3

形式 2

```
66 データ名-1 RENAMES データ名-2 [{THROUGH} データ名-3].
                                  {THRU}
```

形式 3

```
88 条件名{VALUE IS }{定数-2 [{THROUGH} 定数-3]}...
         {VALUES ARE}        {THRU}
    [WHEN SET TO FALSE 定数-4] .
                          (NF)
```

形式 4

```
                                                                    (NF)
78  定数名 VALUE IS {整数-5                    }[{+  }{整数-1                    }]
                    {NEXT                      }  {-  }{NEXT                      }
                    {START OF データ名-1       }  {*  }{START OF データ名-3       }
                    {LENGTH OF データ名-2      }  {/  }{LENGTH OF データ名-4      }
                                                  {AND}
                                                  {OR }
```

A.3.3 報告書記述項の一般形式

A.3.4 報告集団記述項の一般形式

形式 1

形式 2

形式 3

A.3.5 画面記述項の一般形式

レベル番号 [画面名 / FILLER]

[BLANK {SCREEN / LINE}]

[{BELL / BEEP}] NF

[BLINK]

[ERASE {EOL / EOS}]

[HIGHLIGHT / LOWLIGHT]

[GRID] [LEFTLINE] [OVERLINE] NF

[REVERSE-VIDEO]

[UNDERLINE]

[SIZE IS {一意名-1 / 整数-1}] NF

[LINE [NUMBER IS [{PLUS / + / -}] NF {一意名-2 / 整数-2}]]

[{COLUMN / COL} NF [NUMBER IS [{PLUS / + / -}] NF {一意名-3 / 整数-3}]]

[{FOREGROUND-COLOR / FOREGROUND-COLOUR} NF IS {一意名-4 / 整数-4}]

[{BACKGROUND-COLOR / BACKGROUND-COLOUR} NF IS {一意名-5 / 整数-5}]

[CONTROL IS 一意名-6] NF

[[VALUE IS] 定数-1]

[{PICTURE / PIC} IS 文字列]

[{[FROM {一意名-6 / 整数-2}] [TO 一意名-7] / USING 一意名-8}]

[[USAGE IS] {DISPLAY / DISPLAY-1}] NF

[BLANK WHEN ZERO]

[{JUSTIFIED / JUST} RIGHT]

[[SIGN IS] {LEADING / TRAILING} [SEPARATE CHARACTER]]

[{AUTO / AUTO-SKIP}] NF

[{SECURE / NO-ECHO}] NF

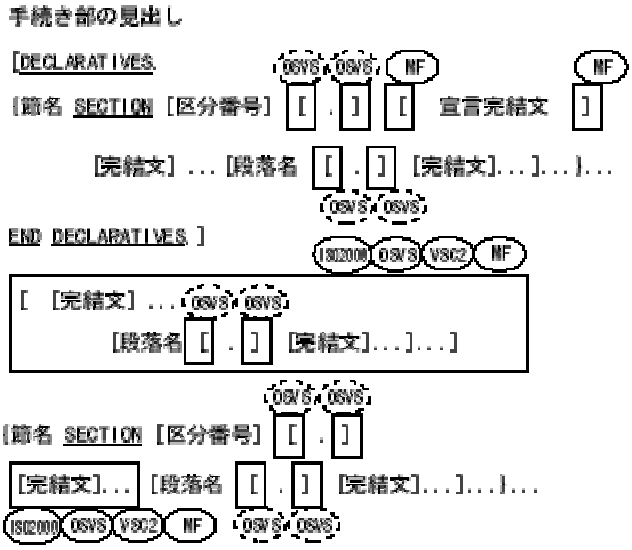
[{REQUIRED / EMPTY-CHECK}] NF

A.4 組み込み関数の一般形式

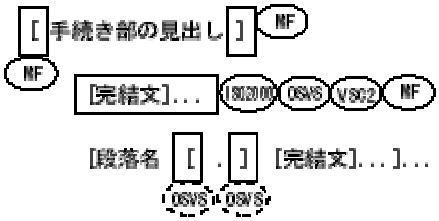
```
FUNCTION ABS (引数-1)                                   NF
FUNCTION ACOS (引数-1)
FUNCTION ANNUITY (引数-1 引数-2)
FUNCTION ASIN (引数-1)                                  AN885
FUNCTION ATAN (引数-1)
FUNCTION CHAR (引数-1)
FUNCTION CHAR-NATIONAL (引数-1)                         NF
FUNCTION COS (引数-1)                                   AN885
FUNCTION CURRENT-DATE
FUNCTION DATE-OF-INTEGER (引数-1)
FUNCTION DAY-OF-INTEGER (引数-1)
FUNCTION DISPLAY-OF (引数-1 [引数-2])                   NF
FUNCTION E
FUNCTION EXP (引数-1)
FUNCTION EXP10 (引数-1)
FUNCTION FACTORIAL (引数-1)                             AN885
FUNCTION FACTION-PART (引数-1)                          NF
FUNCTION INTEGER (引数-1)                               AN885
FUNCTION INTEGER-OF-DATE (引数-1)
FUNCTION INTEGER-OF-DAY (引数-1)
FUNCTION INTEGER-PART (引数-1)
FUNCTION LENGTH (引数-1)
FUNCTION LENGTH-AN (引数-1)                             NF
FUNCTION LOG (引数-1)                                   AN885
FUNCTION LOG10 (引数-1)
FUNCTION LOWER-CASE (引数-1)
FUNCTION MAX ({引数-1}...)
FUNCTION MEAN ({引数-1...})
FUNCTION MEDIAN ({引数-1...})
FUNCTION MIDRANGE ({引数-1...})
FUNCTION MIN ({引数-1}...)
FUNCTION MOD (引数-1 引数-2)
FUNCTION NATIONAL-OF (引数-1 [引数-2])                  NF
FUNCTION NUMVAL (引数-1)                                AN885
FUNCTION NUMVAL-C (引数-1 [引数-2])
FUNCTION ORD (引数-1)
FUNCTION ORD-MAX ({引数-1}...)
FUNCTION ORD-MIN ({引数-1}...)
FUNCTION PI                                             NF
FUNCTION PRESENT-VALUE (引数-1 {引数-2}...)             AN885
FUNCTION RANDOM [(引数-1)]
FUNCTION RANGE ({引数-1}...)
FUNCTION REM (引数-1 引数-2)
```

A.5 手続き部の一般形式

A.5.1 宣言部分の形式



A.5.2 非宣言部分の形式

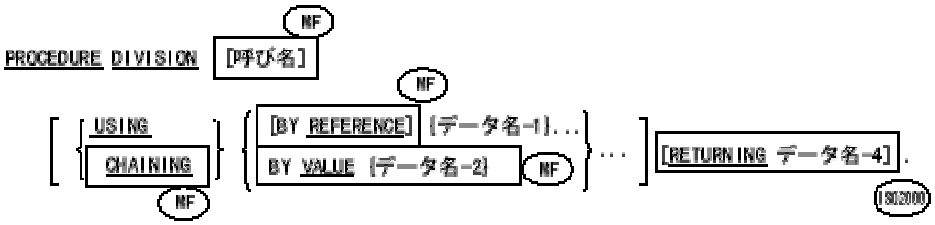


ISO2000 NF

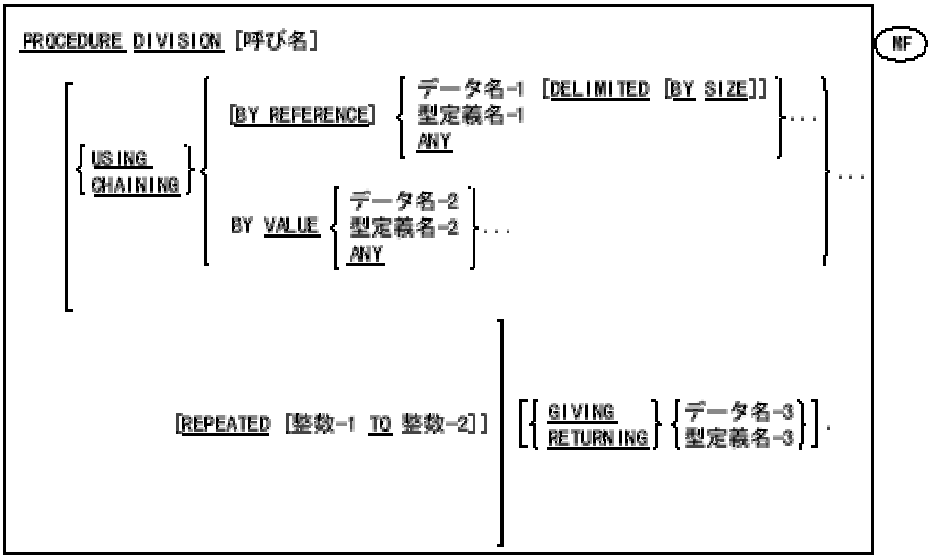
```
PROCEDURE DIVISION.
[ DECLARATIVES.
 { section-name SECTION.
          declarative-sentence.
[ sentence ] ... [ paragraph-name. [ sentence ] ... ] ... } ...
END DECLARATIVES. ]
[ { method-definition } ... ]
```

A.5.3 手続き部の一般形式

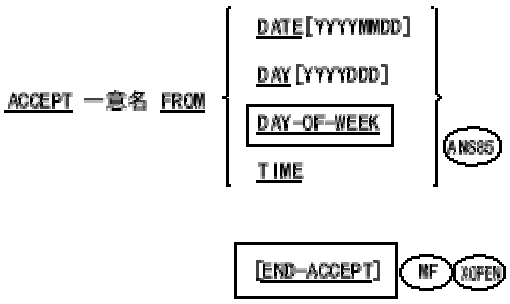
形式 1

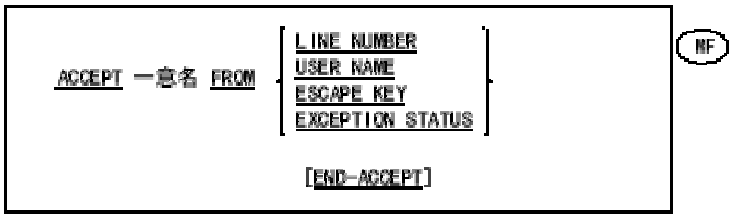


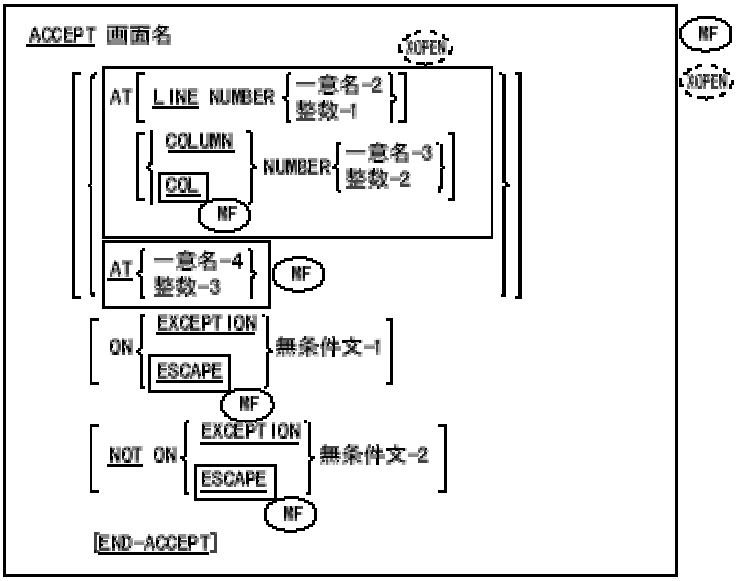
形式 2



A.6 動詞の一般形式


ACCEPT 一意名 FROM
DATE[YYYYMMDD]
DAY[YYYYDDD]
DAY-OF-WEEK
TIME
ANS85
[END-ACCEPT]
NF
XOPEN


ACCEPT 一意名 FROM
LINE NUMBER
USER NAME
ESCAPE KEY
EXCEPTION STATUS
[END-ACCEPT]
NF


ACCEPT 画面名
NF
XOPEN
XOPEN
AT
LINE NUMBER
一意名-2
整数-1
COLUMN
COL
NF
NUMBER
一意名-3
整数-2
AT
一意名-4
整数-3
NF
ON
EXCEPTION
ESCAPE
NF
無条件文-1
NOT ON
EXCEPTION
ESCAPE
NF
無条件文-2
[END-ACCEPT]

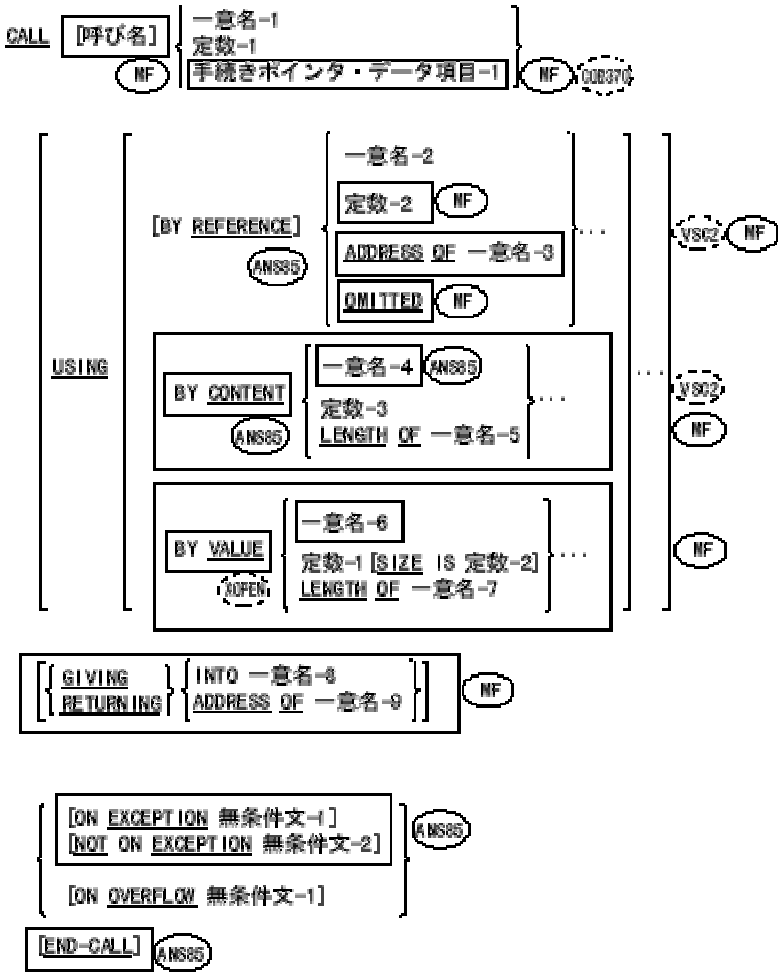
```
ACCEPT 一意名-1                                                    (NF)

[ { AT [ LINE NUMBER { 一意名-2 } ]
                     { 整数-1   }
        [ { COLUMN } NUMBER { 一意名-3 } ]
          { COL    }        { 整数-2   }
  AT { 一意名-4 }
     { 整数-3   } } ]

[FROM CRT] [MODE IS BLOCK]

[ WITH { { AUTO      }
         { AUTO-SKIP }
         { BELL      }
         { BEEP      }
         BLINK
         { FULL          }
         { LENGTH-CHECK  }
         GRID
         [ HIGHLIGHT ]
         [ LOWLIGHT  ]
         LEFTLINE
         OVERLINE
         PROMPT [CHARACTER IS [定数-1]]
         { REQUIRED      }
         { EMPTY-CHECK   }
         REVERSE-VIDEO
         { SECURE  }
         { NO-ECHO }
         SIZE IS { 一意名-6 }
                 { 整数-4   }
         UNDERLINE
         { FOREGROUND-COLOR  } IS 整数-5
         { FOREGROUND-COLOUR }
         { BACKGROUND-COLOR  } IS 整数-6
         { BACKGROUND-COLOUR }
         CONTROL IS { 一意名-7 }
                    { 定数-2   }
         { TIME-OUT } AFTER { 整数-7 }
         { TIMEOUT  }       { 一意名-8 }
         LEFT-JUSTIFY
         RIGHT-JUSTIFY
         SPACE-FILL
         TRAILING-SIGN
         UPDATE
         UPPER
         LOWER
         ZERO-FILL } ... ]

[ ON { EXCEPTION } 無条件文-1 ]
     { ESCAPE   }

[ NOT ON { EXCEPTION } 無条件文-2 ]
         { ESCAPE   }

[END-ACCEPT]
```

```
ADD {一意名-1} ... TO {一意名-2 [ROUNDED]} ...
    {定数-1}

    [ON SIZE ERROR 無条件文-1]

    [NOT ON SIZE ERROR 無条件文-2]   (ANS85)
    [END-ADD]
```

```
ADD {一意名-1} [... TO] (ANS85) (OSVS)
    {定数-1}

    {一意名-2} GIVING {一意名-3 [ROUNDED]} ...
    {定数-2}

    [ON SIZE ERROR 無条件文-1]

    [NOT ON SIZE ERROR 無条件文-2]   (ANS85)
    [END-ADD]
```

```
ADD {CORRESPONDING} 一意名-1 TO 一意名-2 [ROUNDED]
    {CORR}

    [ON SIZE ERROR 無条件文-1]

    [NOT ON SIZE ERROR 無条件文-2]   (ANS85)
    [END-ADD]
```

```
ALTER 手続き名-1 TO [PROCEED TO] 手続き名-2
      [手続き名-3 TO [PROCEED TO] 手続き名-4] ...
```


CALL
[呼び名]
NF
一意名-1
定数-1
手続きポインタ・データ項目-1
NF
COB370
USING
[BY REFERENCE]
ANS85
一意名-2
定数-2
NF
ADDRESS OF 一意名-3
OMITTED
NF
VSC2
NF
BY CONTENT
ANS85
一意名-4
ANS85
定数-3
LENGTH OF 一意名-5
VSC2
NF
BY VALUE
NOFEN
一意名-6
定数-1 [SIZE IS 定数-2]
LENGTH OF 一意名-7
NF
GIVING
RETURNING
INTO 一意名-8
ADDRESS OF 一意名-9
NF
[ON EXCEPTION 無条件文-1]
[NOT ON EXCEPTION 無条件文-2]
ANS85
[ON OVERFLOW 無条件文-1]
[END-CALL]
ANS85
CANCEL
一意名-1
定数-1

CHAIN { 一意名-1 | 定数-1 }

USING [[BY REFERENCE] (ANS85) { 一意名-2 | 定数-2 (NF) | (VSC2) (NF) | ADDRESS OF 一意名-3 | OMITTED (NF) } ... | (VSC2) (NF) BY CONTENT (ANS85) { 一意名-4 (ANS85) | 定数-3 | LENGTH OF 一意名-5 } ... | BY VALUE (XOPEN) { 一意名-6 (XOPEN) | 定数-1 [SIZE IS 一意名-2] | LENGTH OF 一意名-7 } ...] ... (NF)

[END-CHAIN]

CLOSE { ファイル名-1 [{ REEL | UNIT } WITH LOCK | { REEL | UNIT } [{ WITH NO REWIND | FOR REMOVAL }] | WITH { NO REWIND | LOCK | DISP (OSVS) }] } ...

CLOSE ファイル名-1 [WITH LOCK] [ファイル名-2 [WITH LOCK]]...

COMMIT

COMPUTE {一意名-1 [ROUNDED]} ...

{ EQUAL | = } 算術式 (OSVS) (VSC2) (NF)

[ON SIZE ERROR 無条件文-1]

[NOT ON SIZE ERROR 無条件文-2] (ANS85)
[END-COMPUTE]

CONTINUE

DELETE ファイル名-1 RECORD
[INVALID KEY 無条件文-1]

[NOT INVALID KEY 無条件文-2]
[END-DELETE]
ANS85

DELETE FILE {ファイル名}...

DISPLAY { 一意名-1 | 定数-1 } [{ 一意名-2 | 定数-2 }]...

[UPON { 呼び名 | 機能名 }] [WITH NO ADVANCING]

OSVS VSC2 NF　　ANS85 XOPEN

[ON EXCEPTION 無条件文-1]
[NOT ON EXCEPTION 無条件文-2]
[END-DISPLAY]
NF
XOPEN

DISPLAY 画面名

[{ AT [LINE NUMBER { 一意名-2 | 整数-1 }]
[{ COLUMN | COL } NUMBER { 一意名-3 | 整数-2 }]
AT { 一意名-4 | 整数-3 } NF }]

[END-DISPLAY]
NF
XOPEN

```
DISPLAY {{一意名-1}
         {定数-1  }}
   {AT [LINE NUMBER {一意名-2}]
        {        {整数-1  }}
       [{COLUMN}
        {COL   } NUMBER {一意名-3}]
        {      }        {整数-2  }}
   {AT {一意名-4}
       {整数-3  }}
   [UPON {CRT       }] [MODE IS BLOCK]
         {CRT-UNDER }
   [WITH {{BELL}
          {BEEP}
          BLINK
          GRID
          ERASE {EOL}
                {EOS}
          HIGHLIGHT
          LOWLIGHT
          LEFTLINE
          OVERLINE
          REVERSE-VIDEO
          SIZE IS {一意名-5}
                  {整数-4  }
          UNDERLINE
          {FOREGROUND-COLOR}
          {FOREGROUND-COLOUR} IS 整数-5
          {BACKGROUND-COLOR}
          {BACKGROUND-COLOUR} IS 整数-6
          CONTROL IS {一意名-6}
                     {定数-2  }
          BLANK {SCREEN}
                {LINE  }}...]...
[END-DISPLAY]
```

NF

```
DIVIDE {一意名-1} INTO {一意名-2 [ROUNDED]} ...
       {定数-1  }
  [ON SIZE ERROR 無条件文-1]
  [NOT ON SIZE ERROR 無条件文-2]
  [END-DIVIDE]
```

ANS85

```
DIVIDE {一意名-1} INTO {一意名-2}
       {定数-1  }      {定数-2  }
        GIVING {一意名-3 [ROUNDED]} ...
 [ON SIZE ERROR 無条件文-1]
 [NOT ON SIZE ERROR 無条件文-2]
 [END-DIVIDE]
```

ANS85

```
DIVIDE {一意名-1} BY {一意名-2}
       {定数-1  }    {定数-2  }
        GIVING {一意名-3 [ROUNDED]} ...
 [ON SIZE ERROR 無条件文-1]
 [NOT ON SIZE ERROR 無条件文-2]
 [END-DIVIDE]
```

ANS85

```
DIVIDE {一意名-1} INTO {一意名-2}
       {定数-1  }      {定数-2  }
        GIVING 一意名-3 [ROUNDED]
 REMAINDER 一意名-4
 [ON SIZE ERROR 無条件文-1]
 [NOT ON SIZE ERROR 無条件文-2]
 [END-DIVIDE]
```

ANS85

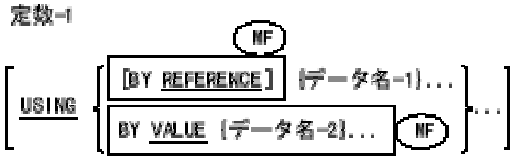
```
DIVIDE {一意名-1} BY {一意名-2}
       {定数-1  }    {定数-2  }
        GIVING 一意名-3 [ROUNDED]
 REMAINDER 一意名-4
 [ON SIZE ERROR 無条件文-1]
 [NOT ON SIZE ERROR 無条件文-2]
 [END-DIVIDE]
```

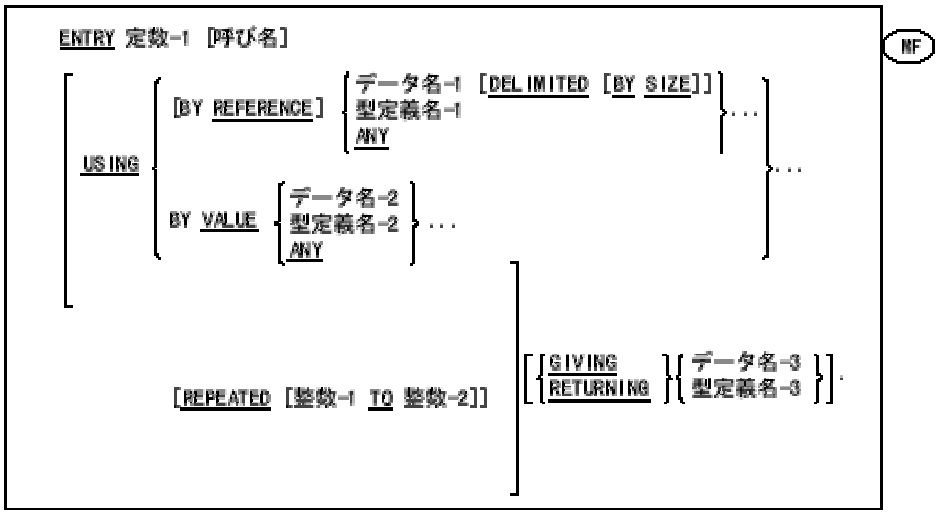
ANS85

```
$ELSE

$END
```

ENTER 言語名 [手順名].

ENTRY 定数-1





```
EVALUATE { 一意名-1 }  [ ALSO { 一意名-2 } ]...
         { 定数-1   }        { 定数-2   }
         { 式-1     }        { 式-2     }
         { TRUE     }        { TRUE     }
         { FALSE    }        { FALSE    }

  {{WHEN

  { ANY                                                          }
  { 条件-1                                                       }
  { TRUE                                                         }
  { FALSE                                                        }
  { [NOT] {{ 一意名-3 }  [{ THROUGH }{ 一意名-4 }]}}             }
  { [NOT] {{ 定数-3   }        { THRU }{ 定数-4   }}             }
  { [NOT] {{ 算術式-1 }                { 算術式-2 }}             }
  { [部分式-1] (NF)                                              }

  [ ALSO { ANY                                                   }]
         { 条件-2                                                }
         { TRUE                                                  }
         { FALSE                                                 }
         { [NOT] {{ 一意名-5 }  [{ THROUGH }{ 一意名-6 }]}}      }
         { [NOT] {{ 定数-5   }        { THRU }{ 定数-6   }}      }
         { [NOT] {{ 算術式-3 }                { 算術式-4 }}      }
         { [部分式-2] (NF)                                       }
                                                                 ]...}...

  無条件文-1}...

  [WHEN OTHER 無条件文-2]

  [END-EVALUATE]
```

```
EXAMINE 一意名 TALLYING
                { UNTIL FIRST }
                { ALL         }  定数-1 [REPLACING BY 定数-2]
                { LEADING     }
```

```
EXAMINE 一意名 REPLACING { ALL         }  定数-1 BY 定数-2
                         { LEADING     }
                         { FIRST       }
                         { UNTIL FIRST }
```

```
EXEC[UTE]原文名 原文データ END-EXEC
```

```
EXHIBIT { NAMED         }  { 一意名-1 }...
        { CHANGED NAMED }  { 定数-1   }
        { CHANGED       }
```

EXIT

EXIT FUNCTION (SO2002)

EXIT METHOD (SO2002) (MF)

EXIT PERFORM [CYCLE] (MF)

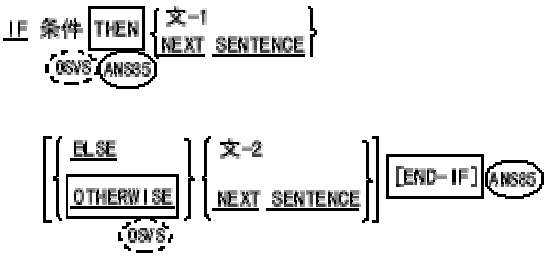
EXIT { PARAGRAPH / SECTION } (MF)

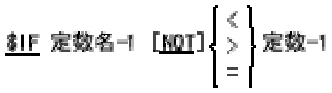
EXIT PROGRAM [{ GIVING / RETURNING } { [ADDRESS OF] 一意名-1 / 整数-1 }] (MF)

GOBACK [{ GIVING / RETURNING } { [ADDRESS OF] 一意名-1 / 整数-1 }] (MF)

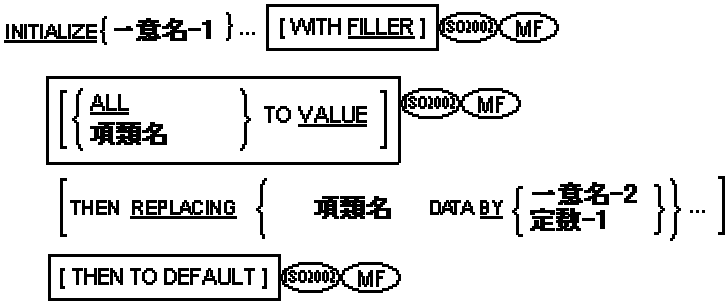
GO TO [手続き名-1]

GO TO 手続き名-1 [手続き名-2]...DEPENDING ON 一意名

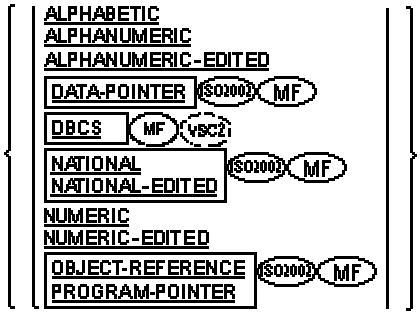

IF 条件 THEN
文-1
NEXT SENTENCE
OSVS
ANS85
ELSE
OTHERWISE
OSVS
文-2
NEXT SENTENCE
[END-IF]
ANS85


$IF 定数名-1 [NOT]
<
>
=
定数-1

$IF 定数名-2 [NOT] DEFINED


INITIALIZE { 一意名-1 } ...
[WITH FILLER]
ISO2002
MF
ALL
項類名
TO VALUE
ISO2002
MF
THEN REPLACING
項類名
DATA BY
一意名-2
定数-1
[THEN TO DEFAULT]
ISO2002
MF

ここで 項類名 は:


ALPHABETIC
ALPHANUMERIC
ALPHANUMERIC-EDITED
DATA-POINTER
ISO2002
MF
DBCS
MF
VSC2
NATIONAL
NATIONAL-EDITED
ISO2002
MF
NUMERIC
NUMERIC-EDITED
OBJECT-REFERENCE
PROGRAM-POINTER
ISO2002
MF

INSPECT 一意名-1 TALLYING

一意名-2 FOR
- CHARACTERS [BEFORE / AFTER] INITIAL {一意名-4 / 定数-2} [...] (ANS85)
- {ALL / LEADING} {一意名-3 / 定数-1} [BEFORE / AFTER] INITIAL {一意名-4 / 定数-2} [...] [...] (ANS85) (ANS85)

[...] (ANS85) [...] (ANS85)

INSPECT 一意名-1 REPLACING

- CHARACTERS BY {一意名-5 / 定数-3} [BEFORE / AFTER] INITIAL {一意名-4 / 定数-2} [...] (ANS85)
- {ALL / LEADING / FIRST} {一意名-3 / 定数-1} BY {一意名-5 / 定数-3} [BEFORE / AFTER] INITIAL {一意名-4 / 定数-2} [...] [...] (ANS85) (ANS85)

[...] (ANS85)

INSPECT 一意名-1 TALLYING

一意名-2 FOR
- CHARACTERS [BEFORE / AFTER] INITIAL {一意名-4 / 定数-2} [...] (ANS85)
- {ALL / LEADING} {一意名-3 / 定数-1} [BEFORE / AFTER] INITIAL {一意名-4 / 定数-2} [...] [...] (ANS85) (ANS85)

(ANS85) [...] [...] (ANS85)

REPLACING

- CHARACTERS BY {一意名-5 / 定数-3} [BEFORE / AFTER] INITIAL {一意名-4 / 定数-2} [...] (ANS85)
- {ALL / LEADING / FIRST} {一意名-3 / 定数-1} BY {一意名-5 / 定数-3} [BEFORE / AFTER] INITIAL {一意名-4 / 定数-2} [...] [...] (ANS85) (ANS85)

[...] (ANS85)

INSPECT 一意名-1 CONVERTING {一意名-6 / 定数-4} TO {一意名-7 / 定数-5}

[{BEFORE / AFTER} INITIAL {一意名-4 / 定数-5}]

(ANS85)

```
INVOKE { オブジェクト一意名-1 | 一意名-1 AS { テンプレート-1 | OBJECT } (MF) } { 定数-1 | 一意名-2 }

[ USING { [BY REFERENCE] { 一意名-3 | OMITTED (ISO2002) (MF) | ADDRESS OF 一意名-4 } ...
          BY CONTENT { 一意名-5 | 定数-2 | 算術式-1 (ISO2002) (MF) | LENGTH OF 一意名-6 (MF) } ...
          BY VALUE { 一意名-7 | 整数-1 [SIZE IS 整数-2] (MF) | 算術式-1 (ISO2002) (MF) | LENGTH OF 一意名-8 (MF) } ... } ... ]

[ { RETURNING | GIVING (MF) } { (MF) INTO 一意名-9 | ADDRESS OF 一意名-10 (MF) } ]
```

```
MERGE ファイル名-1 { ON { ASCENDING | DESCENDING } KEY (OSVS) IS {データ名-1}... }...
      [COLLATING SEQUENCE IS 符号系名]
      USING ファイル名-2 {ファイル名-3}...
{ OUTPUT PROCEDURE IS 手続き名-1 [ { THROUGH | THRU } 手続き名-2 ] }
{ GIVING [ ファイル名-4 ]... (ANS85) (ANS85) }
```

```
MOVE { 一意名-1 | 定数 } TO {一意名-2}...
```

```
MOVE { CORRESPONDING | CORR } 一意名-1 TO {一意名-2} ... (OSVS) (MF)
```

```
MULTIPLY { 一意名-1 } BY {一意名-2 [ROUNDED]}...
         { 定数-1   }

    [ON SIZE ERROR 無条件文-1]
    ┌──────────────────────────────┐
    │[NOT ON SIZE ERROR 無条件文-2]│
    │[END-MULTIPLY]                │
    └──────────────────────────────┘ (ANS85)
```

```
MULTIPLY { 一意名-1 } BY { 一意名-2 }
         { 定数-1   }    { 定数-2   }

                    GIVING {一意名-3 [ROUNDED]}...

    [ON SIZE ERROR 無条件文-1]
    ┌──────────────────────────────┐
    │[NOT ON SIZE ERROR 無条件文-2]│
    │[END-MULTIPLY]                │
    └──────────────────────────────┘ (ANS85)
```

```
NEXT SENTENCE
```

```
NOTE 文字列
```

```
ON { 定数-1   } [AND EVERY { 定数-2   }]
   { 一意名-1 }            { 一意名-2 }

      [UNTIL { 定数-3   }] { 無条件文-1   }
             { 一意名-3 }  { NEXT SENTENCE }

        [{ ELSE      }{ 無条件文-2   }]
         { OTHERWISE }{ NEXT SENTENCE }
```

```
                                                   ...
OPEN { INPUT[共有句] { ファイル名-1 [REVERSED                  ]} ...
     {                             [WITH { NO REWIND }        ]}
     {                                   { LOCK     }          }
     { OUTPUT[共有句] { ファイル名-2 [WITH { NO REWIND }]} ...
     {                                    { LOCK     }        }
     { I-O[共有句] {ファイル名-3 [WITH LOCK]}...
     { EXTEND[共有句] {ファイル名-4 [WITH LOCK]}...
```

```
OPEN { INPUT[共有句]{ファイル名-1 [WITH LOCK]}...
       OUTPUT[共有句]{ファイル名-2 [WITH LOCK]}...
       I-O[共有句]{ファイル名-3 [WITH LOCK]}...
       EXTEND[共有句]{ファイル名-4 [WITH LOCK]}... }...
```

where sharing phrase is:

```
[ SHARING WITH { ALL OTHER
                 NO OTHER
                 READ ONLY } ]
```

OPEN { identifier-1 }...

```
PERFORM [ (ANS85) 手続き名-1 [ { THROUGH } 手続き名-2 ] ] (ANS85)
                               { THRU    }

[無条件文-1 [ END-PERFORM ] ] (ANS85)
            (NF)  (NF)

PERFORM [ (ANS85) 手続き名-1 [ { THROUGH } 手続き名-2 ] ] (ANS85)
                               { THRU    }
{ 一意名-1 } TIMES
{ 整数-1   }
[無条件文-1 [ END-PERFORM ] ] (ANS85)
            (NF)  (NF)
```

```
PERFORM [ 手続き名-1 [ { THROUGH } 手続き名-2 ] ]
        (ANS85)        { THRU    }                (ANS85)

[WITH TEST { BEFORE } ]  (ANS85)
           { AFTER  }

UNTIL { 条件-1 }  (NF)
      { EXIT   }

[無条件文-1 [ END-PERFORM ] ]  (ANS85)
            (NF)  (NF)
```

```
                (ANS85)                                  (ANS85)
PERFORM [ 手続き名-1 [ { THROUGH } 手続き名-2 ] ]
        [              { THRU    }               ]

[WITH TEST { BEFORE } ]  (ANS85)
           { AFTER  }

VARYING { 一意名-2 } FROM { 一意名-3 }
        { 指標名-1 }      { 指標名-2 }
                          { 定数-1   }

BY { 一意名-4 } UNTIL {条件-1}
   { 定数-2   }

[ AFTER { 一意名-5 } FROM { 一意名-6 }
        { 指標名-3 }      { 指標名-4 }
                          { 定数-3   }

BY { 一意名-7 } UNTIL 条件-2 ] ...
   { 定数-4   }

[無条件文-1 [ END-PERFORM ] ]  (ANS85)
            (NF)  (NF)
```

```
READ ファイル名 [NEXT] RECORD [INTO 一意名] [WITH { [ KEPT ] LOCK } ]
                                                  { [ NO   ]      }
                                                  { WAIT          }
                                                  (NF)

[AT END 無条件文-1]

[NOT AT END 無条件文-2]
[END-READ]  (ANS85)
```

```
READ ファイル名 [NEXT] RECORD [INTO 一意名]                    (NF)
     [AT END 無条件文-1]

     [NOT AT END 無条件文-2]
     [END-READ]
```

```
READ ファイル名 [ NEXT ] RECORD [INTO 一意名] [WITH { [ KEPT ] LOCK } ]
                [ PREVIOUS ]                         [ NO   ]        [ IGNORE ]
                (NF)                                 [ WAIT ]
                                                                     (NF)
     [AT END 無条件文-1]

     [NOT AT END 無条件文-2]
     [END-READ]                                      (ANS85)
```

```
READ ファイル名 RECORD [INTO 一意名] [WITH { [ KEPT ] LOCK } ]
                                           [ NO   ]
                                           [ WAIT ]
                                                    (NF)
     [INVALID KEY 無条件文-3]

     [NOT INVALID KEY 無条件文-4]
     [END-READ]                                      (ANS85)
```

```
READ ファイル名 RECORD [INTO 一意名] [WITH { [ KEPT ] LOCK } ]
                                           [ NO   ]
                                           [ IGNORE ]
                                           [ WAIT ]
                                                    (NF)
     [ KEY IS { データ名 }  ]
              { 分割キー名 } (NF)
     [INVALID KEY 無条件文-3]

     [NOT INVALID KEY 無条件文-4]
     [END-READ]                                      (ANS85)
```

RELEASE レコード名 [FROM 一意名]

RETURN ファイル名 RECORD [INTO 一意名-1]
AT END 無条件文-1

[NOT AT END 無条件文-2]
[END-RETURN]
ANS85

REWRITE レコード名 [FROM 一意名]

[END-REWRITE]
NF

REWRITE レコード名 [FROM 一意名]

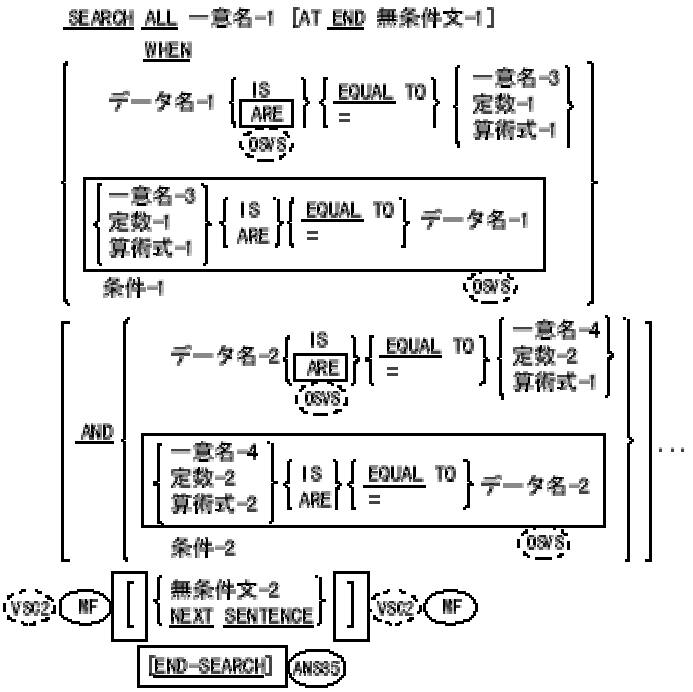
[END-REWRITE]

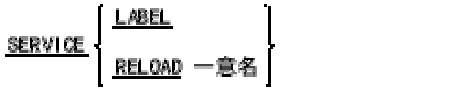
REWRITE レコード名 [FROM 一意名]
[INVALID KEY 無条件文-1]

[NOT INVALID KEY 無条件文-2]
[END-REWRITE]
ANS85

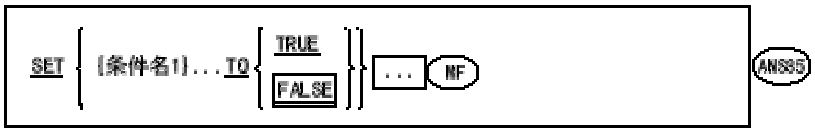
ROLLBLACK

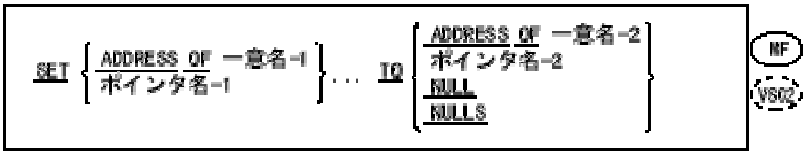
SEARCH 一意名-1 [VARYING { 一意名-2 / 指標名-1 }]
[AT END 無条件文-1]
{ WHEN 条件-1 [{ 無条件文-2 / NEXT SENTENCE }] }...
NF
[END-SEARCH]
ANS85


SEARCH ALL 一意名-1 [AT END 無条件文-1]
WHEN
データ名-1 IS ARE EQUAL TO = 一意名-3 定数-1 算術式-1
OSVS
一意名-3 定数-1 算術式-1 IS ARE EQUAL TO = データ名-1
条件-1
OSVS
AND
データ名-2 IS ARE EQUAL TO = 一意名-4 定数-2 算術式-1
OSVS
一意名-4 定数-2 算術式-2 IS ARE EQUAL TO = データ名-2
条件-2
OSVS
VSC2 NF
無条件文-2
NEXT SENTENCE
VSC2 NF
END-SEARCH
ANS85


SERVICE
LABEL
RELOAD 一意名


SET {呼び名-1}... TO ON OFF
ANS85


SET {条件名1}... TO TRUE FALSE
NF
ANS85


SET ADDRESS OF 一意名-1 ポインタ名-1 TO ADDRESS OF 一意名-2 ポインタ名-2 NULL NULLS
NF
VSC2

```
SET [ポインタ名-3...] { UP | DOWN } BY { 一意名-3 | 整数-1 | LENGTH OF 一意名-4 }
```

(NF)

```
SET { 指標名-1 | 一意名-5 } ... TO { 指標名-2 | 一意名-6 | 整数-2 }
```

```
SET { 指標名-3 } ... { UP | DOWN } BY { 一意名-7 | 整数-3 }
```

```
SET [手続きポインタ名-1] TO { 手続きポインタ名-2 | ENTRY { 一意名-8 | 整数-1 } | NULL | NULLS }
```

(NF)

(SO1999) (MF)

```
SET {一意名-9} ... TO 一意名-10
```

(MF)

```
SET {イベントポインタ-1} ... TO { TRUE | FALSE }
```

(MF)

```
SET {イベントポインタ-1} ... TO イベントポインタ-2
```

```
SET {モニターポインタ-1} ... TO [NOT] { BROWSING | READING | WRITING } [ CONVERTING FROM { BROWSING | WRITING } ]
```

(MF)

(MF)

SET {モニターポインター1}... TO モニターポインター2

(MF)

SET {ミューテックスポインター1}... TO { ON / OFF }

(MF)

SET {ミューテックスポインター1}... TO ミューテックスポインター2

(MF)

SET { セマフォポインター1 }... { UP / DOWN } BY { 整数-4 / 一意名-11 }

(MF)

SET { セマフォポインター1 }... TO セマフォポインター2

SET {オブジェクト一意名-1}... TO { オブジェクト一意名-2 / NULL }

SORT ファイル名-1 ON { { ASCENDING / DESCENDING } KEY [IS] {データ名-1}... } ...

(DBWS)

[WITH DUPLICATES IN ORDER] (ANS85)

[COLLATING SEQUENCE IS 符号系名]

{ INPUT PROCEDURE IS 手続き名-1 [{ THROUGH / THRU } 手続き名-2] / USING {ファイル名-2}... }

{ OUTPUT PROCEDURE IS 手続き名-3 [{ THROUGH / THRU } 手続き名-4] / GIVING [{] ファイル名-3 [}]... }

(ANS85) (ANS85)

```
SORT データ名-2 [ ON { ASCENDING } KEY [データ名-1]... ]...   (NF)
                     { DESCENDING }

    [WITH DUPLICATES IN ORDER]
    [COLLATING SEQUENCE IS 符号系名]
```

```
                                   IS EQUAL TO
                                   IS =
                                   IS GREATER THAN
                                   IS >
                                   IS NOT LESS THAN
                                   IS NOT <
                                                    (ANS85)
START ファイル名-1 [ KEY { IS GREATER THAN OR EQUAL TO   } データ名-1 ]
                         IS >=
                                   IS LESS THAN
                                   IS <                     (NF)
                                   IS NOT GREATER THAN
                                   IS NOT >
                                   IS LESS THAN OR EQUAL TO
                                   IS <=
                                   IS NOT GREATER THAN OR EQUAL TO
                                   IS NOT >=
                                   IS NOT LESS THAN OR EQUAL TO
                                   IS NOT <=
```

```
[INVALID KEY 無条件文-1]
[NOT INVALID KEY 無条件文-2]   (ANS85)
[END-START]
```

START ファイル名-1 [KEY { IS EQUAL TO
IS =
IS GREATER THAN
IS >
IS NOT LESS THAN
IS NOT <
(ANS85)
IS GREATER THAN OR EQUAL TO
IS >=
IS LESS THAN
IS <
IS NOT GREATER THAN
IS NOT >
IS LESS THAN OR EQUAL TO
IS <=
IS NOT GREATER THAN OR EQUAL TO
IS NOT >=
IS NOT LESS THAN OR EQUAL TO
IS NOT <=
(NF) }]

{ データ名-1
分割キー名-1 (NF) } [WITH SIZE { 一意名-1
定数-1 }] (NF)]

[INVALID KEY 無条件文-1]

[NOT INVALID KEY 無条件文-2]
[END-START] (ANS85)

START { 定数-1
一意名-1
手続きポインター1 } ... (NF)

[USING [BY CONTENT] 一意名-2]

[RETURNING INTO 一意名-3
IDENTIFIED BY スレッドポインター1]

[STATUS IS 一意名-4]

[ON EXCEPTION 無条件文-1]

[NOT ON EXCEPTION 無条件文-2]

[END-START]

STOP { RUN
定数-1 }

```
STOP RUN [ { GIVING    } { [ADDRESS OF] 一意名-1 } ]
           { RETURNING } { 整数-1 [SIZE IS 整数-2] }
```

NF

```
STRING { { 一意名-1 } ... DELIMITED BY { 一意名-2 } } ...
         { 定数-1   }                  { 定数-2   }
                                       { SIZE     }
    INTO 一意名-3
    [WITH POINTER 一意名-4]
    [ON OVERFLOW 無条件文-1]
    [NOT ON OVERFLOW 無条件文-2]
    [END-STRING]
```

ANS85

```
SUBTRACT { 一意名-1 } ... FROM {一意名-2 [ROUNDED]}...
         { 定数-1   }
    [ON SIZE ERROR 無条件文-1]
    [NOT ON SIZE ERROR 無条件文-2]
    [END-SUBTRACT]
```

ANS85

```
SUBTRACT { 一意名-1 } ... FROM { 一意名-2 }
         { 定数-1   }          { 定数-2   }
GIVING{一意名-3 [ROUNDED]}...
    [ON SIZE ERROR 無条件文-1]
    [NOT ON SIZE ERROR 無条件文-2]
    [END-SUBTRACT]
```

ANS85

```
SUBTRACT { CORRESPONDING } 一意名-1 FROM 一意名-2 [ROUNDED]
         { CORR          }
    [ON SIZE ERROR 無条件文-1]
    [NOT ON SIZE ERROR 無条件文-2]
    [END-SUBTRACT]
```

ANS85

```
TRANSFORM 一意名-3 CHARACTERS
    FROM { 表意定数-1 } TO { 表意定数-2 }
         { 文字定数-1 }    { 文字定数-2 }
         { 一意名-1   }    { 一意名-2   }
```

```
UNLOCK ファイル名 { RECORD  }
                  { RECORDS }
```

```
UNSTRING 一意名-1
    [ DELIMITED BY [ALL] { 一意名-2 } [ OR [ALL] { 一意名-3 } ]... ]
                         { 定数-1   }            { 定数-2   }
        INTO {一意名-4 [DELIMITER IN 一意名-5] [COUNT IN 一意名-6]}...
        [WITH POINTER 一意名-7]
        [TALLYING IN 一意名-8]
        [ON OVERFLOW 無条件文-1]
        [NOT ON OVERFLOW 無条件文-2]
        [END-UNSTRING]
```

ANS85

```
USE [GLOBAL] AFTER STANDARD { EXCEPTION }
    ANS85                   { ERROR     }
        PROCEDURE ON { [ファイル名-1]... }
                     { INPUT           }
                     { OUTPUT          }
                     { I-O             }
                     { EXTEND          }
    GIVING データ名-1 [データ名-2] .
```

OSVS

```
USE [GLOBAL] AFTER STANDARD [ BEGINNING ] [ FILE ]
                            [ ENDING    ] [ REEL ]
                                          [ UNIT ]
    LABEL PROCEDURE ON { [ファイル名-1]... }
                       { OUTPUT          }
                       { INPUT           }
                       { I-O             }
                       { EXTEND          } .
```

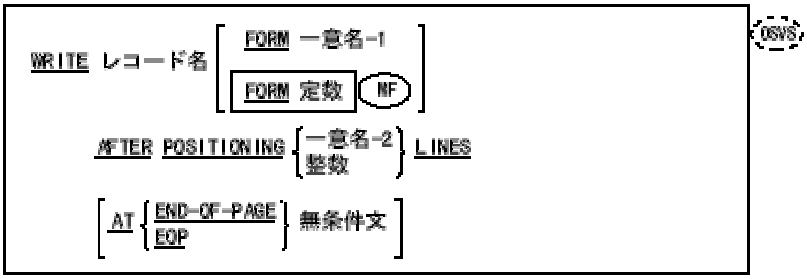
OSVS
VSC2

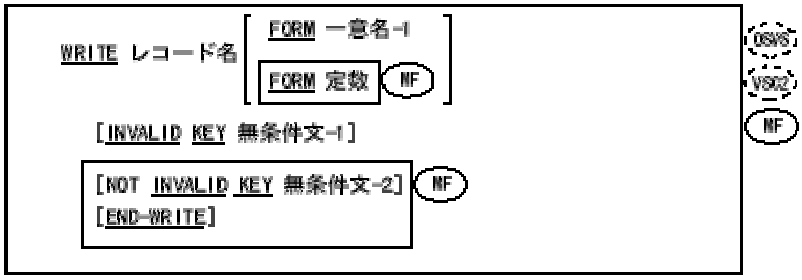
```
USE [GLOBAL] AFTER STANDARD { BEGINNING } [FILE]        (DSVS)
                            { ENDING    }                (VS02)
    LABEL PROCEDURE ON { [ファイル名-1]...
                       { OUTPUT
                       { INPUT
                       { I-O
                       { EXTEND                          }.
```

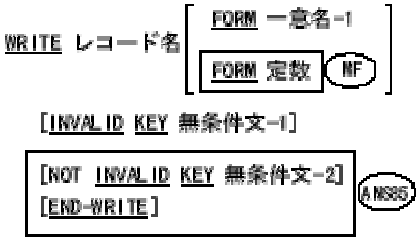
```
WAIT FOR スレッドポインター1
  [ RETURNING INTO 一意名-1 ]
  [ STATUS IS 一意名-2]
  [ ON EXCEPTION 無条件文-1          ]
  [ NOT ON EXCEPTION 無条件文-2      ]
  [ END-WAIT ]
```

```
WAIT FOR イベントポインター1
```

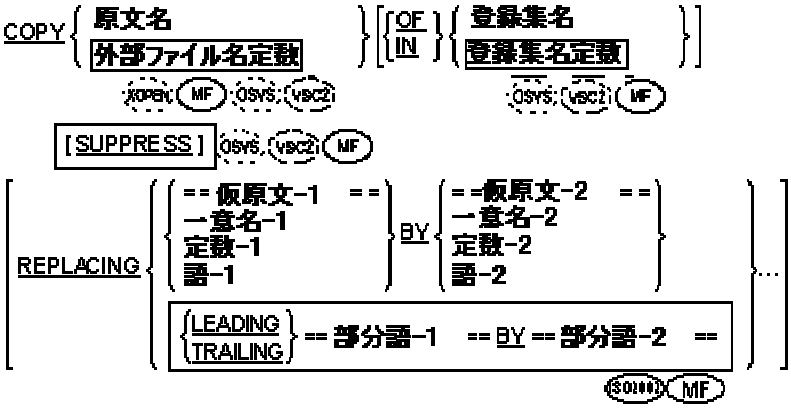
```
WRITE レコード名 [ FROM 一意名-1                    ]
                 [ FROM 定数 (NF)                   ]
  [ { BEFORE } ADVANCING { { 一意名-2 } [ LINE  ] }
    { AFTER  }            { { 整数    } [ LINES ] }
                          { 呼び名                }
                          { PAGE                  }
                          { TAB                   }
                          { FORMFEED (NF)         } ]
  [ AT { END-OF-PAGE } 無条件文-1 ]                    (AN685)
       { EOP         }
  [ NOT AT { END-OF-PAGE } 無条件文-2 ]
           { EOP         }
  [END-WRITE]
```







A.6.1 COPY 文の一般形式



A.6.2 REPLACE 文の一般形式

```
REPLACE { == 仮原文-1 == BY == 仮原文-2 ==
          [ {LEADING} == 部分語-1 == BY == 部分語-2 == ]
          {TRAILING}                                    } ...
                                                  (ISO2002) (MF)
```

```
REPLACE OFF
```

A.7 オブジェクト COBOL 言語拡張の一般形式

A.7.1 クラス拡張

```
[IDENTIFICATION DIVISION.]
CLASS-ID. クラス名-1 [ IS ABSTRACT ]
                     [ IS EXTERNAL ]
    [ DATA IS { PRIVATE   } ]
              { PROTECTED }
              { RESTRICTED }
    [INHERITS FROM クラス名-2[WITH DATA]].
クラス本体
オブジェクトプログラム
END CLASS クラス名-1
```

A.7.2 クラス拡張

一般形式

```
[IDENTIFICATION DIVISION.]
CLASS-ID. 拡張名-1 EXTEND クラス名-1 [WITH DATA].
クラス本体
オブジェクトプログラム
END CLASS 拡張名-1
```

A.7.3 クラス本体

一般形式

形式 1

```
OBJECT SECTION.
 CLASS-CONTROL.
        {クラス名-3 IS CLASS "外部名-1"}... .
 [データ部]
 [クラスオブジェクト]
```

形式 2

```
OBJECT SECTION.
 CLASS-CONTROL.
        {クラス名-3 IS CLASS "外部名-1"}... .
 [データ部]
 [手続き部]
 [メソッド-1]...
```

A.7.4 クラスオブジェクト

一般形式

```
[IDENTIFICATION DIVISION.]
 CLASS-OBJECT.
[データ部]
[PROCEDURE DIVISION.]
[メソッド-1]...
 END CLASS-OBJECT.
```

A.7.5 オブジェクトプログラム

一般形式

```
[IDENTIFICATION DIVISION.]
OBJECT.
[データ部]
[PROCEDURE DIVISION.]
[メソッド-1]...
END OBJECT.
```

A.7.6 メソッド

形式

```
METHOD-ID."メソッド名-1".
[データ部]
[手続き部]
END METHOD"メソッド名-1".
```

A.7.7 メソッドインターフェイス定義

形式

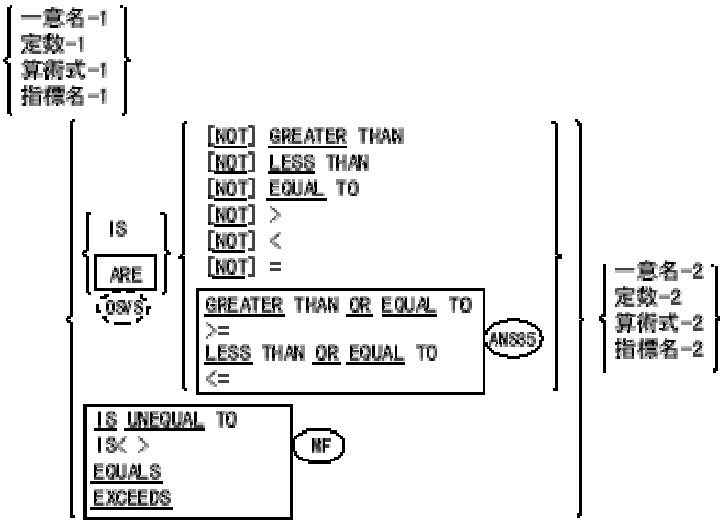
```
METHOD-ID."メソッド名-1".
[連絡節]
手続き部見出し
    [INVOKED AS 動詞シグネチャ[OR AS 動詞シグネチャ]...].
END METHOD"メソッド名-1".
```

ここで動詞シグネチャは以下の通り。

```
                          <OBJECT>
                          <SELF>
                          <THIS>
[FUNCTION] ==動詞-1 { <データ名-3> }... ==
                          必須語
                          補助語
                          右かっこ
                          左かっこ
```

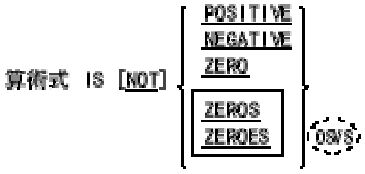
A.8 条件の一般形式

A.8.1 比較条件



A.8.2 字類条件

A.8.3 正負条件



A.8.4 条件名条件

条件名

A.8.5 スイッチ状態条件

条件名

A.8.6 否定単純条件

NOT 単純条件

A.8.7 組合せ条件

条件 { AND | OR } 条件 }...

A.8.8 略記組合せ比較条件

比較条件 { AND | OR } [NOT] [比較演算子] オブジェクト }...

A.8.9 ポインタ条件

{ ADDRESS OF 一意名-1 | 一意名-2 | NULL } { IS [NOT] EQUAL TO | IS [NOT] = | IS UNEQUAL TO | IS <> | EQUALS } { ADDRESS OF 一意名-3 | 一意名-4 | NULL }

NF

A.8.10 手続き部ポインタ(Procedure-Pointer)条件

{ 一意名-1 | NULL } { IS [NOT] EQUAL TO | IS [NOT] = | IS UNEQUAL TO | IS <> | EQUALS } { 一意名-2 | NULL }

NF

A.8.11 オブジェクト比較条件

```
{オブジェクト一意名-1}
{NULL}

{ IS { [NOT]EQUAL TO }
{      [NOT]=        }
{      <>            }
{
{ IS UNEQUAL TO
{ EQUALS

{オブジェクト一意名-2}
{NULL}
```

A.9 その他の形式

A.9.1 修飾

形式 1

```
{データ名-1}
{条件名-1}

{ {{IN}}{データ名-2}}...[{IN}{ファイル名-1}]]
{ {{OF}}{            }   {OF}{通信記述名-1}]
{
{ {IN}{ファイル名-1}
{ {OF}{通信記述名-1}
```

形式 2

```
段落名 [{IN} 節名]
       {OF}
```

形式 3

```
原文名 [{IN}登録集名]
        {OF}
```

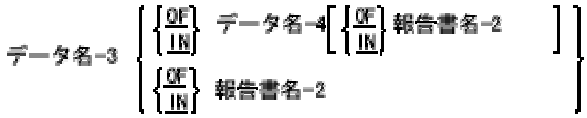
形式 4

```
LINAGE-COUNTER {OF} ファイル名-2
               {IN}
```

形式 5

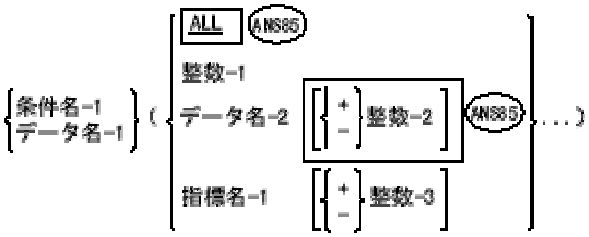
```
{PAGE-COUNTER} {OF} 報告書名-1
{LINE-COUNTER} {IN}
```

形式 6

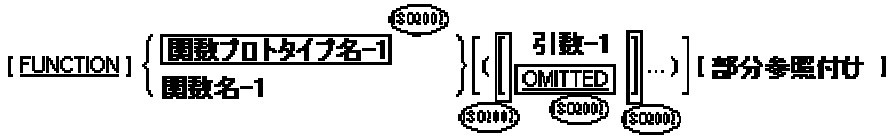


形式 7

A.9.2 添字付けと指標付け



A.9.3 関数一意名



A.9.4 一意名

形式 1

形式 2

データ名-1 [{OF|IN} データ名-2] ... [{OF|IN} {通信記述名|ファイル名1|報告書名-1}]

[({添字} ...)]　　[部分参照] (ANS85)

A.9.5 部分参照

{データ名-1 | FUNCTION 関数名-1 [({引数-1}...)]} (再左端文字位置:[長さ])

付録 B：文字集合と文字の照合順序

表 B-1 に、ASCII および EBCDIC の文字集合と文字の照合順序を示す。あわせて、その 16 進値 も示す。COBOL の欄に×印が付けてある場合は、その文字を COBOL 構文中で使用できないこと を意味する。

ALPHASTART 指令および SYMBSTART 指令を使用すると、コンパイラが文字の照合順序の中の 位置を数えるときの始点を設定できる。これらの指令の詳細については、COBOL システムリファ レンス』を参照。

EBCDIC の欄は IBM の米国での標準ビット･パターンの割当てを示す。文字によっては、言語が異なると そのビット･パターンが異なるものがある。それらの文字は表の中で四角で囲んである。

表 B-1: 文字集合と文字の照合順序

Hex	ASCII		EBCDIC	
	Character	COBOL	Character	COBOL
00	NUL	X	NUL	X
01	SOH	X	SOH	X
02	STX	X	STX	X
03	ETX	X	ETX	X
04	EOT	X	SEL	X
05	ENQ	X	HT	X
06	ACK	X	RNL	X
07	BEL	X	DEL	X
08	BS	X	GE	X
09	HT	X	SPS	X
0A	LF	X	RPT	X
0B	VT	X	VT	X
0C	FF	X	FF	X
0D	CR	X	CR	X
0E	SO	X	SO	X
0F	SI	X	SI	X
10	DLE	X	DLE	X
11	DCI	X	DC1	X
12	DC2	X	DC2	X
13	DC3	X	DC3	X

14	DC4	X	RES/ENP	X
15	NAK	X	NL	X
16	SYN	X	BS	X
17	ETB	X	POC	X
18	CAN	X	CAN	X
19	EM	X	EM	X
1A	SUB	X	UBS	X
1B	ESC	X	CU1	X
1C	FS	X	IFS	X
1D	GS	X	IGS	X
1E	RS	X	IRS	X
1F	US	X	ITB/IUS	X
20	SPACE		DS	X
21	!	X	SOS	X
22	"		FS	X
23	#	X	WUS	X
24	$		BYP/INP	X
25	%	X	LF	X
26	&		ETB	X
27	'		ESC	X
28	(		SA	X
29	)		SFE	X
2A	*		SM/SW	X
2B	+		CSP	X
2C	,		MFA	X
2D	-		ENQ	X
2E	.		ACK	X
2F	/		BEL	X
30	0			X
31	1			X
32	2		SYN	X
33	3		IR	X
34	4		PP	X

35	5		TRN	X
36	6		NBS	X
37	7		EOT	X
38	8		SBS	X
39	9		IT	X
3A	:		RFF	X
3B	;		CU3	X
3C	<		DC4	X
3D	=		NAK	X
3E	>			X
3F	?	X	SUB	X
40	@	X	SP	X
41	A		RSP	X
42	B			X
43	C			X
44	D			X
45	E			X
46	F			X
47	G			X
48	H			X
49	I			X
4A	J		¢	X
4B	K		.	X
4C	L		<	X
4D	M		(	X
4E	N		+	X
4F	O		¦	X
50	P		&	
51	Q			X
52	R			X
53	S			X
54	T			X

55	U			X
56	V			X
57	W			X
58	X			X
59	Y			X
5A	Z		!	X
5B		X	$	
5C		X	*	
5D		X	)	
5E		X	;	
5F		X	¬	X
60		X	·	
61	a		/	
62	b			X
63	c			X
64	d			X
65	e			X
66	f			X
67	g			X
68	h			X
69	i			X
6A	j		¦	X
6B	k		,	
6C	l		%	X
6D	m		_	X
6E	n		>	
6F	o		?	X
70	p		.	X
71	q			X
72	r			X
73	s			X
74	t			X

75	u			X
76	v			X
77	w			X
78	x			X
79	y		`	X
7A	z		:	
7B		X	#	X
7C		X	@	X
7D		X	'	
7E		X	=	
7F	DEL	X	"	
80				X
81			a	
82			b	
83			c	
84			d	
85			e	
86			f	
87			g	
88			h	
89			i	
8A				X
8B				X
8C				X
8D				X
8E				X
8F				X
90				X
91			j	
92			k	
93			l	
94			m	

95			n	
96			o	
97			p	
98			q	
99			r	
9A				X
9B				X
9C				X
9D				X
9E				X
9F				X
A0				X
A1			~	X
A2			s	
A3			t	
A4			u	
A5			v	
A6			w	
A7			x	
A8			y	
A9			z	
AA				X
AB				X
AC				X
AD				X
AE				X
AF				X
B0				X
B1				X
B2				X
B3				X
B4				X

B5				X
B6				X
B7				X
B8				X
B9				X
BA				X
BB				X
BC				X
BD				X
BE				X
BF				X
C0			(	X
C1			A	
C2			B	
C3			C	
C4			D	
C5			E	
C6			F	
C7			G	
C8			H	
C9			I	
CA			SHY	X
CB				X
CC				X
CD				X
CE				X
CF				X
D0			)	X
D1			J	
D2			K	
D3			L	
D4			M	

D5			N	
D6			O	
D7			P	
D8			Q	
D9			R	
DA				X
DB				X
DC				X
DD				X
DE				X
DF				X
E0			\	X
E1			NSP	X
E2			S	
E3			T	
E4			U	
E5			V	
E6			W	
E7			X	
E8			Y	
E9			Z	
EA				X
EB				X
EC				X
ED				X
EE				X
EF				X
F0			0	
F1			1	
F2			2	
F3			3	
F4			4	

F5			5	
F6			6	
F7			7	
F8			8	
F9			9	
FA			¦	X
FB				X
FC				X
FD				X
FE				X
FF			E0	X

付録 C：ANSI ファイル状態コード

ここでは、ANSI ファイル状態コードのリストを掲載する。詳しくは、***環境部***の章の*ファイル管理記述項*節、***手続き部***の章の*入出力状態*節、および COBOL システムのマニュアルを参照。

C.1 ANSI'74 ファイル状態コード

ANSI'74 ファイル状態コードは以下の通り。

状態キー 1	状態キー 2	説明
0	0	正常終了。
	2	索引ファイルの場合のみ設定される。以下のいずれかが考えられる。 a. READ 文の実行で、現在の参照キーの値が次のレコードの参照キーの値と等しく なっている。 b. WRITE、または REWRITE 文の実行で、書き出されたレコードの、少なくとも 1 つの副レコードキーの値が重複している。副レコードキーは重複していてもよい。
1	0	次の論理レコードがない。ファイルの終わりに達したことを意味する。
2	1	順呼出しのファイルの場合にのみ設定される。順序の誤りを表す。一連のレコードキー値は昇順でなければならないのに、違反 しているか、または主レコードキー値が、そのファイルに対する、正常終了した READ 文と次の REWRITE 文の実行との間で、COBOL プログラムによって変更されている。
	2	索引および相対ファイルの場合だけ設定される。重複キーの状態を示す。索引ファイルまたは相対ファイルに重複キーを作成するこ とになるレコードを格納、または再格納しようとした。
	3	レコードが見つからなかったことを示す。キーで識別されるレコードにアクセスしようとしたが、そのレコードはファ イル中に存在しない。
3	0	順ファイルに対する区域外書出し、またはデータチェック･パリティエラーや伝送エラーなどの入出力エラーが発生した。
	4	入出力文の実行が、区域外書出しのため失敗しました。この状態は、順ファイルについて、外部で定義された区域を越えて書き出しが実行されようとしたことを示す。
9		拡張されたファイル状態コード

C.2 ANSI'85 ファイル状態コード

ANSI'85 ファイル状態コードには、すべての ANSI'74 ファイル状態コードと以下のコードが含まれる。

状態キー 1	状態キー 2	説明
0	4	処理しているレコードの長さが、このファイルのファイル固有属性に従っていない。
	5	OPEN 文が実行されたときに、参照された不定ファイルが存在しない。
	7	順ファイルの場合にのみ設定される。REEL/UNIT 指定のある CLOSE 文、または OPEN 文で参照されたファイルが、リール/ユニット媒体ではない。
1	4	相対ファイルの場合だけ設定される。相対レコード番号の有効桁数が、そのファイルに対して記述された相対キーデータ項目の大きさより大きい。
2	2	索引および相対ファイルの場合だけ設定される。重複キーの状態を示す。索引ファイル、相対ファイルに重複キーを作成することになるレコードを格納しようとした。または、重複指定を行っていないのに、索引ファイルに、重複した副レコードキーを作成することになるレコードを格納しようとした。
	3	ANSI'74 コード"23"と同じ状態を示す。さらに、START または READ 文を、存在しない不定入力ファイルに対して実行しようとした。
	4	相対および索引ファイルの場合だけ設定される。次の状態の中のどれかによって起きる、区域外書き出しを示す。 a. 外部で定義されたファイルの区域を越えて書き出しをしようとした。 b. 順次 WRITE 文を、相対ファイルに実行しようとしたが、 相対レコード番号の有効桁数が、そのファイルに対して記述された相対キーデータ項目の大きさより大きかった。
3	5	I-O、INPUT、または EXTEND 句のある OPEN 文が、不定ファイル以外の存在していないファイルに実行されようとした。
	7	OPEN 文で指定されたオープンモードをサポートしないファイルに、OPEN 文を実行しようとした。
	8	前にロックをして閉じたファイルに、OPEN 文を実行しようとした。
	9	ファイル固有属性と、プログラム中でそのファイルに対して指定した属性との間に矛盾が検出された。
4	1	すでに開いているファイルに OPEN 文を実行しようとした。
	2	すでに閉じているファイルに CLOSE 文を実行しようとした。
	3	順呼出しのファイルの場合に設定される。DELETE、または REWRITE 文の実行より前に、このファイルに対して最後に実行された入出力文が、READ 文ではなかった。
	4	区域外書出しがある。

6　順呼出しの READ 文が、INPUT または I-O モードで開いているファイルに実行されようとしたが、次の有効なレコードが確立されていない。

7　INPUT、または I-O モードで開かれていないファイルに READ または START 文を実行しようとした。

8　OUTPUT、I-O、もしくは EXTEND モードで開かれていないファイル、または順呼出し法の I-O モードで開いているファイルに WRITE 文を実行しようとした。

9　I-O モードで開いていないファイルに DELETE、または REWRITE 文を実行しようとした。

付録 D：予約語

この付録には、この COBOL 言語において予約されている語をすべて列挙する。

D.1 予約語

以下に示す表に は、どのコンパイラ指令を使用するとどの予約語が使用可能となるか、を示してある。

A

予約語	方言コード (後述の説明を参照)										
ABSENT	…	…	I2	…	…	…	…	…	…	…	…
ABSTRACT	…	…	…	…	…	…	MF11	…	…	…	…
ACCEPT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ACCESS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ACQUIRE	…	…		…	…	…	MF7	…	…	…	…
ACTIVE-CLASS	…	…	I2	…	…	…	MF12	…	…	…	BS
ACTUAL	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
ADD	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ADDRESS	…	…	I2	OS	VS(2, 3, 4)	…	MF3	…	…	DVS	BS
ADVANCING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
AFP-5A	…	…	…	OS	…	…	…	…	…	DVS	…
AFTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ALIGNED	…	…	I2	…	…	…	…	…	…	…	…
ALL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ALLOCATE	…	…	I2	…	…	…	…	…	…	…	…
ALLOW	…	…	I2	…	…	…	…	…	…	…	…
ALPHABET	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
ALPHABETIC	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ALPHABETIC-LOWER	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
ALPHABETIC-UPPER	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
ALPHANUMERIC	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
ALPHANUMERIC-EDITED	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS

ALSO	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ALTER	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ALTERNATE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
AND	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ANY	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
APPLY	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	DVS	…
ARE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
AREA	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
AREAS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
AS	…	…	I2	…	…	…	MFOO	…	…	…	BS
ASCENDING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ASSIGN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
AT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
AUTHOR	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
AUTO-HYPHEN-SKIP	…	…	…	…	…	…	MF7	…	…	…	…
AUTO-SKIP	…	…	…	…	…	…	MF3	MS2	…	…	…
AUTOMATIC	…	…	…	…	…	…	MF1	MS2	…	…	…

B

B-AND	…	…	I2	…	…	…	MF11	…	…	…	…
B-EXOR	…	…	…	…	…	…	MF11	…	…	…	…
B-LEFT	…	…	…	…	…	…	MF11	…	…	…	…
B-NOT	…	…	I2	…	…	…	MF11	…	…	…	…
B-OR	…	…	I2	…	…	…	MF11	…	…	…	…
B-RIGHT	…	…	…	…	…	…	MF11	…	…	…	…
B-XOR	…	…	I2	…	…	…	MF11	…	…	…	…
BACKWARD	…	…	…	…	…	…	MF3	…	…	…	…
BASED	…	…	I2	…	…	…	…	…	…	…	…
BASIS	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	DVS	…
BEEP	…	…	…	…	…	…	MF3	MS2	RM	…	…
BEFORE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
BEGINNING	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	DVS	BS
BINARY	…	85	I2	…	VS(3, 4)	XO	MF7	…	RM	…	BS

BINARY-CHAR	…	…	I2	…	…	…	MF12	…	…	…	…
BINARY-DOUBLE	…	…	I2	…	…	…	MF12	…	…	…	…
BINARY-LONG	…	…	I2	…		…	MF12	…	…	…	…
BINARY-SHORT	…	…	I2	…	…	…	MF12	…	…	…	…
BIT	…	…	I2	…	…	…	…	…	…	…	…
BLANK	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
BLOCK	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
BOOLEAN	…	…	I2	…	…	…	…	…	…	…	…
BOTTOM	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
BROWSING	…	…	…	…	…	…	MF11	…	…	…	…
BY	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

C

C01	…	…	…	OS	…	…	…	…	…	DVS	…
C02	…	…	…	OS	…	…	…	…	…	DVS	…
C03	…	…	…	OS	…	…	…	…	…	DVS	…
C04	…	…	…	OS	…	…	…	…	…	DVS	…
C05	…	…	…	OS	…	…	…	…	…	DVS	
C06	…	…		OS	…	…	…	…	…	DVS	
C07	…	…		OS	…	…	…	…	…	DVS	
C08	…	…		OS	…	…	…	…	…	DVS	
C09	…	…		OS	…	…	…	…	…	DVS	
C10	…	…		OS	…	…	…	…	…	DVS	
C11	…	…		OS	…	…	…	…	…	DVS	
C12	…	…		OS	…	…	…	…	…	DVS	
CALL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CALLED	…	…	…	…	…	…	MF11	…	…	…	…
CANCEL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
CBL	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	…	…
CBL-CTR	…	…	…	…	…	…	…	…	…	…	BS
CD	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
CF	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CH	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CHAIN	…	…	…	…	…	…	MF3	MS2	…	…	…

CHAINING	…	…	…	…	…	…	MF3	MS2	…	…	…
CHANGED	…	…	…	OS	VS(2)	…	MF3	…	…	DVS	…
CHARACTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CHARACTERS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CHECKING	…	…	…	…	…	…	…	…	…	…	BS
CLASS	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
CLASS-CONTROL	…	…	…	…	…	…	MFOO	…	…	…	…
CLASS-ID	…	…	I2	…	C370	…	MF11	…	…	…	BS
CLASS-OBJECT	…	…	…	…	…	…	MFOO	…	…	…	…
CLOCK-UNITS	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CLOSE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
COBOL	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CODE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CODE-SET	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
COERCION	…	…	…	…	…	…	MF11	…	…	…	…
COL	…	…	I2	…	…	…	MF3	MS2	…	…	…
COLLATING	74	85	I2	OS	VS(2, 3, 4)	XO	MF3	MS2	RM	DVS	BS
COLS	…	…	I2	…	…	…	…	…	…	…	…
COLUMN	74	85	I2	OS	VS(2, 3, 4)	XO	MF3	MS2	RM	DVS	BS
COLUMNS	…	…	I2	…	…	…	…	…	…	…	…
COM-REG	…	…	…	…	VS(2, 3, 4)	…	…	…	…	DVS	…
COMMA	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
COMMIT	…	…	…	…	…	…	MF1	…	…	…	BS
COMMITMENT	…	…	…	…	…	…	MF7	…	…	…	
COMMON	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
COMMUNICATION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
COMP	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
COMP-0	…	…	…	…	…	…	MF3	MS2	…	…	…
COMP-1	…	…	…	OS	VS(2, 3, 4)	…	MF7	…	RM	DVS	BS
COMP-2	…	…	…	OS	VS(2, 3, 4)	…	MF7	…	…	DVS	BS
COMP-3	…	…	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
COMP-4	…	…	…	OS	VS(2, 3, 4)	…	MF7	MS2	…	DVS	…
COMP-5	…	…	…	…	OS390	XO	MF3	…	…	…	BS
COMP-6	…	…	…	…	…	…	MF10	…	RM	…	…
COMP-X	…	…	…	…	…	…	MF2	…	…	…	…

COMPUTATIONAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
COMPUTATIONAL-0	…	…	…	…	…	…	MF3	MS2	…	…	…
COMPUTATIONAL-1	…	…	…	OS	VS(2, 3, 4)	…	MF7	…	RM	DVS	BS
COMPUTATIONAL-2	…	…	…	OS	VS(2, 3, 4)	…	MF7	…	…	DVS	BS
COMPUTATIONAL-3	…	…	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
COMPUTATIONAL-4	…	…	…	OS	VS(2, 3, 4)	…	MF7	MS2	…	DVS	…
COMPUTATIONAL-5	…	…	…	…	OS390	XO	MF3	…	…	…	BS
COMPUTATIONAL-6	…	…	…	…	…	…	MF10	…	RM	…	…
COMPUTATIONAL-X	…	…	…	…	…	…	MF2	…	…	…	…
COMPUTE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CONFIGURATION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CONSOLE	…	…	…	OS	…	…	MF1	MS2	…	DVS	…
CONSTANT	…	…	I2	…	…	…	…	…	…	…	…
CONTAINS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CONTENT	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
CONTINUE	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
CONTROL	74	85	I2	OS	VS(2, 3, 4)	XO	MF5	MS2	RM	DVS	BS
CONTROL-AREA	…	…	…	…	…	…	MF7	…	…	…	…
CONTROLS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CONVERT	…	…	…	…	…	…	…	…	RM	…	…
CONVERTING	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
COPY	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CORE-INDEX	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
CORR	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CORRESPONDING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
COUNT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CREATING	…	…	…	…	…	…	…	…	…	…	BS
CRT	…	…	I2	…	…	XO	MF1	…	…	…	…
CRT-UNDER	…	…	…	…	…	…	MF1	…	…	…	…
CSP	…	…	…	OS	…	…	…	…	…	DVS	BS
CURRENCY	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
CURRENT-DATE	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
CURSOR	…	…	I2	…	…	XO	MF1	…	…	…	…
CYL-INDEX	…	…	…	…	…	…	…	…	…	DVS	…
CYL-OVERFLOW	…	…	…	…	…	…	…	…	…	DVS	…

D

DATA	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DATABASE-KEY	…	…	…	…	…	…	…	…	…	…	BS
DATABASE-KEY-LONG	…	…	…	…	…	…	…	…	…	…	BS
DATE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DATE-COMPILED	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DATE-WRITTEN	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DAY	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DAY-OF-WEEK	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
DBCS	…	…	…	…	VS(3, 4)	…	MF7	…	…	…	…
DE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DEBUG	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
DEBUG-CONTENTS	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DEBUG-ITEM	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DEBUG-LINE	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DEBUG-NAME	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DEBUG-SUB-1	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DEBUG-SUB-2	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DEBUG-SUB-3	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DEBUGGING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DECIMAL-POINT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DECLARATIVES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DEFAULT	…	…	I2	…	…	…	MF11	…	…	…	…
DEFINITION	…	…	…	…	…	…	MF11	…	…	…	…
DELETE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DELIMITED	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DELIMITER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DEPENDING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DESCENDING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DESTINATION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
DETAIL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DISABLE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
DISC	…	…	…	…	…	…	…	…	…	…	BS

DISK	…	…	…	…	…	…	MF3	MS2	…	…	…
DISP	…	…	…	OS	VS(2)	…	…	…	…	…	…
DISPLAY	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DISPLAY-1	…	…	…	…	VS(2, 3, 4)	…	MF7	…	…	…	…
DISPLAY-ST	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
DIVIDE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DIVISION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DOWN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DROP	…	…	…	…	…	…	MF7	…	…	…	…
DUPLICATES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
DYNAMIC	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

E

EBCDIC	…	…	…	…	…	…	…	…	…	…	BS
ECHO	…	…	…	…	…	…	…	…	RM	…	…
EGCS	…	…	…	…	VS(2, 3, 4)	…	…	…	…	…	…
EGI	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
EJECT	…	…	…	OS	VS(2, 3, 4)	…	MF7	MS2	…	DVS	BS
ELSE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
EMI	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
EMPTY-CHECK	…	…	…	…	…	…	MF3	MS2	…	…	…
ENABLE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
END	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
END-ACCEPT	…	…	I2	…	…	XO	MF4	…	…	…	BS
END-ADD	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-CALL	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-CHAIN	…	…	…	…	…	…	MF5	…	…	…	…
END-COMPUTE	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-DELETE	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-DISPLAY	…	…	I2	…	…	XO	MF7	…	…	…	BS
END-DIVIDE	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-EVALUATE	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-IF	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-INVOKE	…	…	…	…	C370	…	MFOO	…	…	…	BS

END-MULTIPLY	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-OF-PAGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
END-PERFORM	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-READ	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-RECEIVE	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
END-RETURN	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-REWRITE	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-SEARCH	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-START	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-STRING	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-SUBTRACT	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-UNSTRING	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
END-WAIT	…	…	…	…	…	…	MF11	…	…	…	…
END-WRITE	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
ENDING	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	DVS	BS
ENTER	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ENTRY	…	…	…	OS	VS(2, 3, 4)	…	MF5	…	…	DVS	BS
ENVIRONMENT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
EOP	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
EQUAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
EQUALS	…	…	…	…	…	…	MF7	…	…	…	…
ERROR	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ESCAPE	…	…	…	…	…	…	MF3	MS2	…	…	…
ESI	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
EVALUATE	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
EVENT-POINTER	…	…	…	…	…	…	MF11	…	…	…	…
EVERY	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
EXAMINE	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
EXCEEDS	…	…	…	…	…	…	MF7	…	…	…	…
EXCEPTION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
EXCEPTION-OBJECT	…	…	I2	…	…	…	…	…	…	…	…
EXCESS-3	…	…	…	…	…	…	MF1	…	…	…	…
EXCLUSIVE	…	…	…	…	…	…	MF1	MS2	…	…	…
EXEC	…	…	…	…	…	…	MF2	…	…	…	…

EXECUTE	…	…	…	…	…	…	MF2	…	…	…	…
EXHIBIT	…	…	…	OS	VS(2)	…	MF3	MS2	…	DVS	…
EXIT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
EXTEND	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
EXTENDED	…	…	…	…	…	…	…	…	…	…	BS
EXTENDED-SEARCH	…	…	…	…	…	…	…	…	…	DVS	…
EXTERNAL	…	85	I2	…	VS(3, 4)	XO	MF2	…	…	…	BS
EXTERNALLY-DESCRIBED-KEY	…	…	…	…	…		…	MF7	…	…	…

F

FACTORY	…	…	I2	…	…	…	MF11	…	…	…	BS
FALSE	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
FD	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
FH--FCD	…	…	…	…	…	…	MF5	…	…	…	…
FH--KEYDEF	…	…	…	…	…	…	MF5	…	…	…	…
FILE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
FILE-CONTROL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
FILE-ID	…	…	…	…	…	…	MF3	MS2	…	…	…
FILE-LIMIT	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
FILE-LIMITS	…	…	…	OS	VS(2)	…	…	…	…	DVS	BS
FILLER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
FINAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
FIRST	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
FIXED	…	…	…	…	…	…	MF3	…	…	…	…
FLOAT-EXTENDED	…	…	I2	…	…	…	MF12	…	…	…	…
FLOAT-LONG	…	…	I2	…	…	…	MF12	…	…	…	…
FLOAT-SHORT	…	…	I2	…	…	…	MF12	…	…	…	…
FOOTING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
FOR	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
FORMAT	…	…	I2	…	…	…	MF7	…	…	…	…
FREE	…	…	I2	…	…	…	…	…	…	…	…
FROM	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
FUNCTION	…	85	I2	…	…	XO	MF7	…	…	…	BS

FUNCTION-ID	…	…	I2	…	…	…	MF12	…	…	…	…

G

GENERATE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
GIVING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
GLOBAL	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
GO	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
GOBACK	…	…	I2	OS	VS(2, 3, 4)	…	MF5	…	…	DVS	BS
GREATER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
GRID	…	…	…	…	…	…	MF4	…	…	…	…
GROUP	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

H

HEADING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
HIGH-VALUE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
HIGH-VALUES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

I

I-O	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
I-O-CONTROL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ID	…	…	…	OS	VS(2, 3, 4)	…	MF7	…	…	DVS	BS
IDENTIFICATION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
IDENTIFIED	…	…	…	…	…	…	MF11	…	…	…	…
IF	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
IGNORE	…	…	…	…	…	…	MF8	…	…	…	…
IN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
INDEX	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
INDEXED	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
INDIC	…	…	…	…	…	…	MF7	…	…	…	…
INDICATE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
INDICATOR	…	…	…	…	…	…	MF7	…	…	…	…
INDICATORS	…	…	…	…	…	…	MF7	…	…	…	…
INHERITING	…	…	…	…	…	…	MFOO	…	…	…	BS

INHERITS	…	…	I2	…	C370	…	MF11	…	…	…	BS
INITIAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
INITIALIZE	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
INITIATE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
INPUT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
INPUT-OUTPUT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
INSERT	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	DVS	…
INSPECT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
INSTALLATION	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
INSTANCE	…	…	…	…	…	…	MF12	…	…	…	…
INTERFACE	…	…	I2	…	…	…	MF12	…	…	…	BS
INTERFACE-ID	…	…	I2	…	…	…	MF12	…	…	…	BS
INTO	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
INVALID	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
INVOKE	…	…	I2	…	C370	…	MF11	…	…	…	BS
INVOKED	…	…	…	…	…	…	MFOO	…	…	…	…
IS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

J

JAPANESE	…	…	…	…	…	…	MF1	…	…	…	…
JUST	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
JUSTIFIED	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

K

KANJI	…	…	…	…	VS(2, 3, 4)	…	MF8	…	…	…	…
KEPT	…	…	…	…	…	…	MF1	…	…	…	…
KEY	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
KEY-YY	…	…	…	…	…	…	…	…	…	…	BS
KEYBOARD	…	…	…	…	…	…	MF3	…	…	…	…

L

LABEL	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LAST	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

LEADING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LEAVE	…	…	…	OS	VS(2)	…	…	…	…	…	…
LEFT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LEFT-JUSTIFY	…	…	…	…	…	…	MF3	MS2	…	…	…
LEFTLINE	…	…	…	…	…	…	MF4	…	…	…	…
LENGTH	74	85	I2	OS	VS(2, 3, 4)	XO	MF6	MS2	RM	…	BS
LENGTH-CHECK	…	…	…	…	…	…	MF3	MS2	…	…	…
LESS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LIMIT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LIMITS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LIN	…	…	…	…	…	…	…	MS2	…	…	…
LINAGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LINAGE-COUNTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
LINE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LINE-COUNTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LINES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LINKAGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LOCAL-STORAGE	…	…	I2	…	C370	…	MF5	…	…	…	BS
LOCALE	…	…	I2	…	…	…	…	…	…	…	…
LOCK	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LOCKING	…	…	…	…	…	…	…	MS2	…	…	…
LOW-VALUE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LOW-VALUES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
LOWER	…	…	…	…	…	…	MF8	…	…	…	…

M

MASTER-INDEX	…	…	…	…	…	…	…	…	…	DVS	…
MEMORY	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
MERGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
MESSAGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
METACLASS	…	…	…	…	C370	…	…	…	…	…	…
METHOD	…	…	I2	…	C370	…	MF11	…	…	…	BS
METHOD-ID	…	…	I2	…	C370	…	MF11	…	…	…	BS
MODE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

MODIFIED	…	…	…	…	…	…	MF7	…	…	…	…
MODULES	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
MONITOR-POINTER	…	…	…	…	…	…	MF11	…	…	…	…
MORE-LABELS	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	DVS	BS
MOVE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
MULTIPLE	…	…	…	…	…	…	…	…	…	…	BS
MULTIPLY	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
MUTEX-POINTER	…	…	…	…	…	…	MF11	…	…	…	…

N

NAME	…	…	…	…	…	…	MF3	…	…	…	…
NAMED	…	…	…	OS	VS(2)	…	MF3	MS2	…	DVS	…
NATIONAL	…	…	I2	…	…	…	MF8	…	…	…	…
NATIONAL-EDITED	…	…	I2	…	…	…	MF8	…	…	…	…
NATIVE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
NCHAR	…	…	…	…	…	…	MF5	…	…	…	…
NEGATIVE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
NESTED	…	…	I2	…	…	…	12	…	…	…	BS
NEXT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
NO	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
NO-ECHO	…	…	…	…	…	…	MF3	MS2	…	…	…
NOMINAL	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
NOT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
NOTE	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
NSTD-REELS	…	…	…	…	…	…	…	…	…	DVS	…
NULL	…	…	I2	…	VS(2, 3, 4)	…	MF6	…	…	…	BS
NULLS	…	…	…	…	VS(2, 3, 4)	…	MF6	…	…	…	BS
NUMBER	74	85	I2	OS	VS(2, 3, 4)	XO	MF3	MS2	RM	DVS	BS
NUMBERS	…	…	I2	…	…	…	…	…	…	…	…
NUMERIC	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
NUMERIC-EDITED	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS

O

O-FILL	…	…	…	…	…	…	MF7	…	…	…	…

OBJECT	…	…	I2	…	C370	…	MF11	…	…	…	BS
OBJECT-COMPUTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
OBJECT-ID	…	…	…	…	…	…	MFOO	…	…	…	…
OBJECT-STORAGE	…	…	…	…	…	…	MFOO	…	…	…	…
OCCURS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
OF	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
OFF	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
OMITTED	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ON	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
OOSTACKPTR	…	…	…	…	…	…	MFOO	…	…	…	…
OPEN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
OPTIONAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
OPTIONS	…	…	I2	…	…	…	…	…	…	…	…
OR	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ORDER	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
ORGANIZATION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
OTHER	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
OTHERWISE	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
OUTPUT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
OVERFLOW	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
OVERLINE	…	…	…	…	…	…	MF4	…	…	…	…
OVERRIDE	…	…	I2	…	C370	…	MF12	…	…	…	BS

P

PACKED-DECIMAL	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
PADDING	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
PAGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
PAGE-COUNTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
PARSE	…	…	…	…	…	…	MF12	…	…	…	…
PASSWORD	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	DVS	…
PERFORM	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
PF	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
PH	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

PIC	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
PICTURE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
PLUS	74	85	I2	OS	VS(2, 3, 4)	XO	MF3	MS2	RM	DVS	BS
POINTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
POS	…	…	…	…	…	…	…	…	RM	…	…
POSITION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
POSITIONING	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
POSITIVE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
PRESENT	…	…	I2	…	…	…	…	…	…	…	…
PRINT	…	…	…	…	…	…	…	…	RM	…	…
PRINT-SWITCH	…	…	…	OS	…	…	…	…	…	DVS	BS
PRINTER	…	…	…	…	…	…	MF3	MS2	…	…	…
PRINTER-1	…	…	…	…	…	…	MF3	…	…	…	…
PRINTING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
PRIOR	…	…	…	…	…	…	MF7	…	…	…	…
PRIVATE	…	…	…	…	…	…	MFOO	…	…	…	…
PROCEDURE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
PROCEDURE-POINTER	…	…	…	…	C370	…	MF5	…	…	…	…
PROCEDURES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
PROCEED	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
PROCESS	…	…	…	…	…	…	MF7	…	…	…	…
PROCESSING	…	…	…	OS	VS(2, 3, 4)	…	MF12	…	…	DVS	…
PROGRAM	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
PROGRAM-ID	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
PROGRAM-POINTER	…	…	I2	…	…	…	MF12	…	…	…	…
PROMPT	…	…	…	…	…	…	MF3	MS2	RM	…	…
PROPERTY	…	…	I2	…	…	…	MF12	…	…	…	BS
PROTECTED	…	…	…	…	…	…	MF3	…	…	…	…
PROTOTYPE	…	…	IS	…	…	…	MF12	…	…	…	…
PUBLIC	…	…	…	…	…	…	MFOO	…	…	…	…
PURGE	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS

Q

QUEUE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	···	BS
QUOTE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
QUOTES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

R

RAISE	···	···	I2	···	···	···	···	···	···	···	···
RAISING	···	···	I2	···	···	···	···	···	···	···	···
RANDOM	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RANGE	···	···	···	···	···	···	MF3	···	···	···	···
RD	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
READ	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
READING	···	···	···	···	···	···	MF11	···	···	···	···
READY	···	···	···	OS	VS(2, 3, 4)	···	MF3	MS2	···	DVS	···
RECEIVE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	···	BS
RECORD	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RECORD-OVERFLOW	···	···	···	OS	VS(2)	···	···	···	···	···	···
RECORDING	···	···	···	OS	VS(2, 3, 4)	···	MF3	···	···	DVS	BS
RECORDS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RECURSIVE	···	···	I2	···	C370	···	MF12	···	···	···	···
REDEFINES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
REDEFINITION	···	···	···	···	···	···	MF11	···	···	···	···
REEL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
REFERENCE	···	85	I2	···	VS(2, 3, 4)	XO	MF7	···	···	···	BS
REFERENCES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RELATIVE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RELEASE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RELOAD	···	···	···	OS	VS(2, 3, 4)	···	···	···	···	DVS	···
REMAINDER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
REMARKS	···	···	···	OS	VS(2)	···	···	···	···	DVS	···
REMOVAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RENAMES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

REORG-CRITERIA	…	…	…	OS	VS(2)	…	…	…	…	…	…
REPEATED	…	…	…	…	…	…	MF10	…	…	…	BS
REPLACE	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
REPLACING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
REPORT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
REPORTING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
REPORTS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
REPOSITORY	…	…	I2	…	C370	…	MF12	…	…	…	BS
REREAD	…	…	…	OS	VS(2)	…	…	…	…	…	…
RERUN	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RESERVE	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RESET	…	…	I2	OS	VS(2, 3, 4)	…	MF3	MS2	…	DVS	BS
RESTRICTED	…	…	…	…	…	…	MF11	…	…	…	…
RESUME	…	…	I2	…	…	…	…	…	…	…	…
RETRY	…	…	I2	…	…	…	…	…	…	…	…
RETURN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RETURN-CODE	…	…	…	OS	VS(2, 3, 4)	XO	MF5	…	…	…	BS
RETURNING	…	…	I2	…	C370	…	MF5	…	…	…	BS
REVERSED	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
REWIND	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
REWRITE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RF	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RH	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RIGHT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RIGHT-JUSTIFY	…	…	…	…	…	…	MF3	MS2	…	…	…
ROLLBACK	…	…	…	…	…	…	MF1	…	…	…	BS
ROLLING	…	…	…	…	…	…	MF7	…	…	…	…
ROUNDED	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
RUN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

S

S01	…	…	…	OS	…	…	…	…	…	DVS	…
S02	…	…	…	OS	…	…	…	…	…	DVS	…
S03	…	…	…	…	…	…	…	…	…	DVS	…

S04	…	…	…	…	…	…	…	…	…	DVS	…
S05	…	…	…	…	…	…	…	…	…	DVS	…
SAME	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SCREEN	…	…	I2	…	…	XO	MF3	MS2	…	…	…
SD	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SEARCH	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SECTION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SECURITY	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SEEK	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
SEGMENT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
SEGMENT-LIMIT	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SELECT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SELECTIVE	…	…	…	OS	VS(2)	…	…	…	…	…	…
SELF	…	…	I2	…	C370	…	MFOO	…	…	…	…
SELFCLASS	…	…	…	…	…	…	MFOO	…	…	…	…
SEMAPHORE-POINTER	…	…	…	…	…	…	MF11	…	…	…	…
SEND	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
SENTENCE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SEPARATE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SEQUENCE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SEQUENTIAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SERVICE	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	DVS	…
SET	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SHARING	…	…	I2	…	…	…	MF11	…	…	…	…
SHIFT-IN	…	…	…	…	VS(2, 3, 4)	…	…	…	…	…	…
SHIFT-OUT	…	…	…	…	VS(2, 3, 4)	…	…	…	…	…	…
SIGN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SIZE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SKIP1	…	…	…	OS	VS(2, 3, 4)	…	MF7	…	…	DVS	BS
SKIP2	…	…	…	OS	VS(2, 3, 4)	…	MF7	…	…	DVS	BS
SKIP3	…	…	…	OS	VS(2, 3, 4)	…	MF7	…	…	DVS	BS
SORT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SORT-CONTROL	…	…	…	…	VS(2, 3, 4)	…	…	…	…	…	…
SORT-CORE-SIZE	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	DVS	BS

SORT-FILE-SIZE	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	DVS	BS
SORT-MERGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SORT-MESSAGE	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	…	…
SORT-MODE-SIZE	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	DVS	BS
SORT-OPTION	…	…	…	…	…	…	…	…	…	DVS	…
SORT-RETURN	…	…	…	OS	VS(2, 3, 4)	…	MF7	…	…	DVS	BS
SORT-TAPE	…	…	…	…	…	…	…	…	…	…	BS
SORT-TAPES	…	…	…	…	…	…	…	…	…	…	BS
SOURCE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SOURCE-COMPUTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SOURCES	…	…	I2	…	…	…	…	…	…	…	…
SPACE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SPACE-FILL	…	…	…	…	…	…	MF3	MS2	…	…	…
SPACES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SPECIAL-NAMES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
STANDARD	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
STANDARD-1	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
STANDARD-2	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…	BS
STANDARD-3	…	…	I2	…	…	…	…	…	…	…	…
START	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
STARTING	…	…	…	…	…	…	MF7	…	…	…	…
STATUS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
STOP	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
STRING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SUB-QUEUE-1	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
SUB-QUEUE-2	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
SUB-QUEUE-3	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
SUBFILE	…	…	…	…	…	…	MF7	…	…	…	…
SUBTRACT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SUM	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SUPER	…	…	I2	…	C370	…	MFOO	…	…	…	…
SUPPRESS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SYMBOLIC	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
SYNC	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

SYNCHRONIZED	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
SYSIN	…	…	…	OS	…	…	…	…	…	…	…
SYSIPT	…	…	…	OS	…	…	…	…	…	DVS	…
SYSLST	…	…	…	OS	…	…	…	…	…	DVS	…
SYSOUT	…	…	…	OS	…	…	…	…	…	…	…
SYSPCH	…	…	…	…	…	…	…	…	…	DVS	…
SYSPUNCH	…	…	…	OS	…	…	…	…	…	DVS	…
SYSTEM-DEFAULT	…	…	I2	…	…	…	…	…	…	…	…

T

TAB	…	…	…	…	…	…	…	…	RM	…	…
TABLE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
TALLY	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	DVS	BS
TALLYING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
TAPE	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
TAPES	…	…	…	…	…	…	…	…	…	…	BS
TERMINAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
TERMINATE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
TEST	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
TEXT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…	BS
THAN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
THEN	…	85	I2	OS	VS(2, 3, 4)	XO	MF1	…	…	DVS	BS
THREAD-LOCAL	…	…	…	…	…	…	MF11	…	…	…	…
THREAD-LOCAL-STORAGE	…	…	…	…	…	…	MF10	…	…	…	…
THREAD-POINTER	…	…	…	…	…	…	MF11	…	…	…	…
THROUGH	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
THRU	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
TIME	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
TIME-OF-DAY	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
TIME-OUT	…	…	…	…	…	…	MF7	…	…	…	…
TIMEOUT	…	…	I2	…	…	…	MF7	…	…	…	…
TIMES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
TITLE	…	…	…	…	VS(2, 3, 4)	…	MF7	…	…	…	BS

TO	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
TOP	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
TOTALED	…	…	…	OS	VS(2)	…	…	…	…	…	…
TOTALING	…	…	…	OS	VS(2)	…	…	…	…	…	…
TRACE	…	…	…	OS	VS(2, 3, 4)	…	MF3	MS2	…	DVS	…
TRACK-AREA	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
TRACK-LIMIT	…	…	…	OS	VS(2)	…	…	…	…	…	…
TRACKS	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
TRAILING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
TRAILING-SIGN	…	…	…	…	…	…	MF3	MS2	…	…	…
TRANSACTION	…	…	…	…	…	…	MF7	…	…	…	…
TRANSFORM	…	…	…	OS	VS(2)	…	…	…	…	DVS	…
TRUE	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…	BS
TRY	…	…	…	…	…	…	…	…	…	…	BS
TYPE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
TYPEDEF	…	…	I2	…	…	…	MF10		…	…	…

U

UNEQUAL	…	…	…	…	…	…	MF7	…	…	…	
UNIT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
UNITS	…	…	…	…	…	…	…	…	…	…	BS
UNIVERSAL	…	…	I2	…	…	…	MF12	…	…	…	BS
UNLOCK	…	…	I2	…	…	…	MF1	MS2	RM	…	…
UNSTRING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
UNTIL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
UP	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
UPDATE	…	…	…	…	…	…	MF3	MS2	RM	…	…
UPON	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
UPPER	…	…	…	…	…	…	MF8	…	…	…	…
UPSI-0	…	…	…	OS	…	…	…	…	…	DVS	…
UPSI-1	…	…	…	OS	…	…	…	…	…	DVS	…
UPSI-2	…	…	…	OS	…	…	…	…	…	DVS	…
UPSI-3	…	…	…	OS	…	…	…	…	…	DVS	…
UPSI-4	…	…	…	OS	…	…	…	…	…	DVS	…

UPSI-5	…	…	…	OS	…	…	…	…	…	DVS	…
UPSI-6	…	…	…	OS	…	…	…	…	…	DVS	…
UPSI-7	…	…	…	OS	…	…	…	…	…	DVS	…
USAGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
USE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
USER	…	…	…	…	…	…	MF3	…	…	…	…
USER-DEFAULT	…	…	I2	…	…	…	…	…	…	…	…
USING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

V

VALID	…	…	I2	…	…	…	…	…	…	…	…
VALIDATE	…	…	I2	…	…	…	…	…	…	…	…
VALUE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
VALUES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
VARIABLE	…	…	…	…	…	…	MF3	…	…	…	…
VARYING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

W

WAIT	…	…	…	…	…	…	MF8	MS2	…	…	…
WHEN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
WHEN-COMPILED	…	…	…	OS	VS(2, 3, 4)	…	MF7	…	…	DVS	…
WITH	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
WORDS	74	85	…	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
WORKING-STORAGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
WRITE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
WRITE-ONLY	…	…	…	OS	VS(2, 3, 4)	…	…	…	…	DVS	…
WRITE-VERIFY	…	…	…	…	…	…	…	…	…	DVS	…
WRITING	…	…	…	…	…	…	MF11	…	…	…	…

X

XML	…	…	…	…	…	…	MF12	…	…	…	…
XML-CODE	…	…	…	…	…	…	MF12	…	…	…	…
XML-EVENT	…	…	…	…	…	…	MF12	…	…	…	…
XML-NTEXT	…	…	…	…	…	…	MF12	…	…	…	…
XML-TEXT	…	…	…	…	…	…	MF12	…	…	…	…

Z

ZERO	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ZERO-FILL	…	…	…	…	…	…	MF3	MS2	…	…	…
ZEROES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
ZEROS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS

!

+	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
-	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
*	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
/	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
**	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
>	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
<	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
=	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
>=	…	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
<=	…	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS	BS
…	…	…	I2	…	…	…	MF	…	…	…	…
*>	…	…	I2	…	…	…	MF	…	…	…	…
::	…	…	I2	…	…	…	MF	…	…	…	…
>>	…	…	I2	…	…	…	MF	…	…	…	…

説明：

上記の表の方言コードの定義は以下の通り。

方言コード	コンパイラ指令	意味
74	なし	ANSI 74 標準の予約語
85	ANS85	ANSI 85 標準の予約語
I2	ISO2002	ISO 2002 標準の予約語
OS	OSVS	OSVS COBOL の予約語
VS(2)	VSC2"2"	VS COBOL II リビジョン 2 の予約語
VS(3)	VSC2"3"	特に注記されていない限り、VS COBOL II rev 3、IBM LE COBOL/370 および IBM COBOL for MVS & VM の予約語
VS(4)	VSC2"4"	特に注記されていない限り、VS COBOL II rev 4、IBM LE COBOL/370 及び IBM COBOL for MVS & VM の予約語
C370	COBOL370"2"	Word reserved in IBM LE COBOL/370 リビジョン 2 および IBM COBOL for MVS … VM の予約語だが、VS COBOL II の予約語ではない。
OS390	OS390	COBOL for OS/390 … VM V2R2 の予約語だが、IBM LE COBOL/370 リビジョン 2 および IBM COBOL for MVS … VM の予約語ではない。
XO	XOPEN	X/Open 定義の一部としての予約語
MFER	EARLY-RELEASE	Early Release 構文の一部としての予約語
MF1	MF"1"	Micro Focus LEVEL II COBOL, LEVEL II COBOL/ET および Professional COBOL の予約語
MF2	MF"2"	MF1 に Micro Focus VS COBOL Workbench V1.2 での追加機能を加えたもの
MF3	MF"3"	MF2 に Micro Focus VS COBOL Workbench V1.3 および V2.0, Professional COBOL V2.0 および Micro Focus V1.5 での追加機能を加えたもの
MF4	MF"4"	MF3 に Micro Focus COBOL/2 V1.1, Microsoft COBOL" V3.0 および IBM COBOL V2 での追加機能を加えたもの
MF5	MF"5"	MF4 に Micro Focus COBOL/2 V1.2 および Micro Focus COBOL/2 Workbench V2.3 での追加機能を加えたもの
MF6	MF"6"	MF5 に Micro Focus COBOL/2 V2.4 および Micro Focus COBOL/2 Workbench V2.4 での追加機能を加えたもの
MF7	MF"7"	MF6 に Micro Focus COBOL/2 V2.5 および Micro Focus COBOL/2 Workbench V2.5 での追加機能を加えたもの
MF8	MF"8"	MF7 に Micro Focus COBOL V3.0 および Micro Focus COBOL Workbench V3.0 での追加機能を加えたもの
MF9	MF"9"	MF8 に Micro Focus COBOL V3.1 および Micro Focus COBOL Workbench V3.1 での追加機能を加えたもの

MF10	MF"10"	MF9 に Micro Focus COBOL V3.2 および Micro Focus COBOL Workbench V3.2 での追加機能を加えたもの
MF11	MF"11"	MF10 に Micro Focus Net Express 3.0 での追加機能を加えたもの
MF12	MF"12"	MF11 に Micro Focus Net Express 4.0 での追加機能を加えたもの
MFOO	MFOO	MFOO 指令を使用した時の OO 構文の一部としての予約語
MS1	MS"1"	Microsoft COBOL バージョン 1 の予約語
MS2	MS"2"	Microsoft COBOL バージョン 2 の予約語
RM	RM	Ryan-McFarland COBOL V2.0 の予約語
DVS	DOSVS	DOSVS COBOL の予約語
BS	BS2000	Siemens BS2000 COBOL の予約語

D.2 文脈依存語

以下は文脈依存語で、これらは、指定の方言が選択される時に、指定の言語構造の予約語として扱われる。 文脈依存語が、一般形式の中で使用可能な場所で使われる場合は、キーワードとして扱われる。 そうでない場合は、利用者語として扱われる。ここで、方言コードは*予約語*節のものと同じ意味を持つ。

文脈依存語	言語の文脈または構造	方言コード
ARITHMETIC	OPTIONS 段落	I2
ATTRIBUTE	SET 文	I2
AUTO	画面記述項	I2, XO, MF3, MS2
BACKGROUND-COLOR	画面記述項	I2, XO, MF3, MS2
BACKGROUND-COLOUR	画面記述項	I2, MF3
BELL	画面記述項および SET 属性文	I2, XO, MF3, MS2
BLINK	画面記述項および SET 属性文	I2, XO, MF3, RM, MS2
CENTER	COLUMN 句	I2
CYCLE	EXIT 文	I2, MF7
EOL	画面記述項の ERASE 句	I2, XO, MF7, RM
EOS	画面記述項の ERASE 句	I2, XO, MF7, RM
ERASE	画面記述項	I2, XO, MF3, RM, MS2
EXPANDS	REPOSITORY 段落の CLASS 指定および INTERFACE 指定	I2, MF12

FOREGROUND-COLOR	画面記述項	I2, XO, MF3, MS2
FOREGROUND-COLOUR	画面記述項	I2, MF3
FOREVER	RETRY 指定	I2
FULL	画面記述項	I2, XO, MF3, MS2
HIGH	ACCEPT 文および DISPLAY 文	RM
HIGHLIGHT	画面記述項および SET 属性文	I2, XO, MF3, MS2
IGNORING	READ 文	I2
INITIALIZED	ALLOCATE 文	I2
INTRINSIC	REPOSITORY 段落の関数指定	I2
LC_ALL	SET 文	I2
LC_COLLATE	SET 文	I2
LC_CTYPE	SET 文	I2
LC_MESSAGES	SET 文	I2
LC_MONETARY	SET 文	I2
LC_NUMERIC	SET 文	I2
LC_TIME	SET 文	I2
LOW	ACCEPT 文および DISPLAY 文	RM
LOWLIGHT	画面記述項および SET 属性文	I2, XO, MF7
MANUAL	LOCK MODE 句	I2, MF1, MS2
MULTIPLE	LOCK ON 句	すべての方言
NONE	DEFAULT 句	I2
NORMAL	STOP 文	I2
NUMBERS	COLUMN 句および LINE 句	I2
ONLY	オブジェクトビュー、SHARING 句、SHARING 指定、および USAGE 句	I2, MF11, BS
PARAGRAPH	EXIT 文	I2, MF7
PREVIOUS	READ 文	I2, MF3
RECURSIVE	PROGRAM-ID 段落	I2, MF12, C370
RELATION	VALIDATE-STATUS 句	I2
REQUIRED	画面記述項	I2, XO, MF3, MS2
REVERSE	ACCEPT 文および DISPLAY 文	RM
REVERSE-VIDEO	画面記述項および SET 属性文	I2, XO, MF3, MS2
SECONDS	RETRY 指定	I2

SECURE	画面記述項	I2, XO, MF3, MS2
SIGNED	USAGE 句	I2, MF12
STEP	OCCURS 句	I2
STRONG	TYPEDEF 句	I2
SYMBOL	CURRENCY 句	I2, MF12, OS390
UCS-4	ALPHABET 句	I2
UNDERLINE	画面記述項および SET 属性文	I2, XO, MF3, MS2
UNSIGNED	USAGE 句	I2, MF12
UTF-8	ALPHABET 句	I2
UTF-16	ALPHABET 句	I2
YYYYDDD	ACCEPT 文	すべての方言
YYYYMMDD	ACCEPT 文	すべての方言

用 語 集

この用語集は、本書で使用される用語の意味を定義したものである。用語の定義はおおむね、簡潔な説明にとどめてあり、詳細な構文規則、一般的でない方言は含んでいない。したがって、方言によっては、ここに記載された情報が該当しない場合がある。

(あ行)

脚書き領域 (footing area)

印刷ページの下端部の余白に隣接したページ本体の位置。

暗黙区分 (implicit segment)

コード区分の大きさを調整するために、COBOL システムによって作成される区分。

暗黙属性 (implicit attribute)

明示的に指定されていない属性。

暗黙範囲符 (implicit scope terminator)

先行する文の有効範囲の終わりを区切る、分離符の終止符 (.) 。または、文の中の指定で、それが出て来ることにより、先行する指定が含まれる文の有効範囲を区切るもの。

一意名 (identifier)

データ項目に名前を付ける、構文的に正しい文字列と分離符との組合わせ。

一般的オブジェクト参照 (universal object reference)

特定のクラスまたはインターフェイスに限定されないオブジェクト参照。

インスタンスメソッド (instance method)

インスタンスオブジェクトのメソッド (クラス・メソッドと対峙するもの)

インターフェイス (interface) (メソッドの)

メソッドを正しく起動するために必要な情報。

インターフェイス (interface) (言語構成の)

メソッド・プロトタイプのグループ分け。

インターフェイス定義 (interface definition)

インターフェイスを定義する原始単位。

受取り側項目 (receiving item)

画面節中の PICTURE 句において、TO 指定または USING 指定が対象とするデータ項目。

英字 (alphabetic character)

下記の文字集合に属する文字。

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,

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、

空白。

英数字 (alphanumeric character)

計算機の文字集合中の、任意の文字または数字。

英数字関数 (alphanumeric function)

関数のうち、得られる値が、計算機の文字集合の中の 1 文字以上の文字から構成されるもの。

演算符号 (operational sign)

数字データ項目または数字定数に付けて、値が正であるか負であるかを示す算術符号。

送出し側項目 (sending item)

画面節中の PICTURE 句において、FROM 指定または USING 指定が対象とするデータ項目。

オブジェクト (インスタンス) (object (instance))

データ、およびそのデータに対して動作を実行するメソッドにより構成される単位。

オブジェクト参照 (object reference)

オブジェクトへの参照を含む、明示的または暗黙的に定義されたデータ項目。

オブジェクト・データ (object data)

オブジェクト定義のデータ部中で記述されたデータ。ただし、そのメソッド中で記述されたデータを除く。

オブジェクト定義 (object definition)

オブジェクトを定義する原始単位。

オブジェクトビュー (object view)

オブジェクト参照が、翻訳時に、すでに記述が指定されているとして扱われるようにするための構文。実行時に、オブジェクトに対し、この記述への適合チェックが行われる。

オブジェクトプロパティ (object property)

オブジェクト参照が、オブジェクトとの値の受け渡しが可能かどうかを決定するために使用される名前。

オブジェクトメソッド (object method)

オブジェクトのメソッド (ファクトリオブジェクトのメソッドであるファクトリメソッドに対する) 。

(か行)

カーソル (cursor)

CRT 画面上の指標で、キーボードから次に入力されるデータ文字が画面上に表示される位置を示す。

カーソル制御キー (cursor control keys)

文字入力を行う端末のキーボードのキーで、画面上のカーソルの位置を制御するもの。典型的なものは、カーソルを上下、左右、または、次の画面項目や前の画面項目に移動させるキー。

外部スイッチ (external switch)

ハードウェアまたはソフトウェアの装置で、作成者によって定義および命名され、二者択一の状態のいずれかであ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したデータ。

外部ファイル結合子 (external file connector)

実行単位の中で、1 つ以上の実行用プログラムによって呼び出すことのできるファイル結合子。

外部レポジトリ (external repository)

レポジトリ (repository) の項を参照。

カウンタ (counter)

他の数値に基づいて、値を増減したり、ゼロまたは任意の正か負の値に変更したり、再設定したりするために使用する、データ項目。

拡張モード (extend mode)

EXTEND 指定とともに OPEN 文を実行した後の、ファイル結合子の状態。このファイル結合子に対して、REEL または UNIT 指定を伴わない CLOSE 文を実行するまで、拡張モードのままである。

仮数部 (significand)

浮動小数点の数字の中で、有効な桁の数となる固定小数点数字。

型定義 (type definition)

プログラマが定義した型、またはその型のデータ項目の定義に使用される用途。型定義は、TYPEDEF 句を含むデータ記述項で定義される。型には、データ項目およびその下位の項目の性質すべてが含まれる。

型定義名 (typedef-name)

型定義を表す、利用者定義の語。

可変反復データ項目 (variable-occurrence data item)

反復回数が変えられる、表の中のデータ項目。そのデータ記述項には OCCURS DEPENDING ON 句が含まれるか、またはそのような項目の下位に属する。

画面記述項 (screen description entry)

データ部の画面節中の記述項。レベル番号と、それに続く画面名と、一連の画面句から構成される。画面名は指定しなくてもよく、画面句は必要なものを指定する。この記述項は、形の上ではデータ記述項とよく似ている。ただし、データ記述項は記憶領域を定義するのに対して、画面記述項は画面領域を定義する。

画面項目 (screen item)

画面記述項がその属性を定義する、画面上の項目。

画面節 (Screen Section)

データ部中の最後の節。この節で、ACCEPT 文の形式 4 および DISPLAY 文の形式 2 を通じて呼び出す、画面領域のレイアウトを定義する。

画面名 (screen-name)

画面記述項で記述された画面項目の名を表す、利用者定義の語。

画面レコード (screen record)

レベル番号 1 の画面記述項。

カラム (column)

印刷行または画面表示行の中の文字位置。左端の文字位置を 1 として、順次右側に 1 ずつ繰り上げて番号付けされる。

仮原文 (pseudo-text)

原始文中の一連の文字列や分離符。または、仮原文区切り記号 (==) によって囲まれた (ただし、仮原文区切り記号は含まない) COBOL 登録集。

仮原文区切り記号 (psuedo-text delimiter)

等号 (=) を 2 つ並べたもの。仮原文を区切るために使用する。

完結文 (sentence)

1 つ以上の文を連ね、その最後の文を終止符 (.) に続く空白で止めたもの。

関数 (function)

一時データ項目で、その値は、文の実行中に組み込み関数または利用者定義の関数が参照された時点で決定される。

関数一意名 (function-identifier)

関数を参照する、構文的に正しい文字列と分離符の組合わせ。関数によって表わされるデータ項目は、関数名と、引数がある場合にはその引数とによって、一意に識別される。関数一意名には、部分参照を適用することができる。英数字関数を指す関数一意名は、一般形式中の一意名を指定できる箇所であればどこにでも指定できる。ただし、何らかの制限がある場合がある。整数関数または数字関数を指す関数一意名は、一般形式中の算術式を指定できる箇所であればどこにでも指定できる。

関数プロトタイプ定義 (function prototype definition)

特定の関数の評価に必要な引数、関数の評価結果であるデータ項目、およびその関数の評価に必要なその他の条件すべてに適用される規則を指定する定義。

関数名 (function-name)

関数の値を決定するために、作成者が提供する機構を命名する語。

キー (key)

レコードの位置を識別するためのデータ項目、またはデータの整列順を決定するデータ項目の組。

記述項 (entry)

COBOL 原始プログラムの見出し部、環境部、データ部に書く一連の句。分離符の終止符 (.) で終了され (ただし、注記行は除く) 、見出し部、環境部、またはデータ部の中に書かれる。

記述項の左辺 (subject of entry)

データ記述項により定義されるデータ項目。

記述項名 (entry-name)

利用者定義の語。これにより、初期設定点に代わる、プログラムの実行の開始点が識別される。初期設定点は、原始プログラム中の最初の非宣言手続き文である。

基数 (radix)

位置を表す数値における正の整数で、これに桁位置の意味を乗算すると、次に大きい桁位置の意味が得られる。

起動 (invocation)

メソッドの起動 (method invocation) の項を参照。

起動状態 (active state)

起動したが、起動ランタイム要素に戻されていない、関数、メソッド、またはプログラムの状態。

起動文 (activating statement)

関数、メソッド、またはプログラムを実行する文。

起動ランタイム要素 (activating runtime element)

起動文を含む関数、メソッド、またはプログラム。

機能キー (function key)

文字を入力する端末のキーボード上のキーで、有効なときに押すと、画面レコードの入力が完了し、関数が要求されたという合図を送る。

基本項目 (elementary item)

論理的にさらに分割されることのない形で記述されている、データ項目。

逆編集 (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除去して、編集前のそのデータ項目の値に戻すこと。

行順ファイル編成 (line sequential file organization)

順ファイルの一種で、ホスト・オペレーティングシステムによって生成される、テキスト・ファイル形式の可変長レコードを含む。

共通プログラム (common program)

あるプログラム中に含まれているプログラムであるが、その親プログラム中に直接的または間接的に含まれる任意のプログラムからも呼出せるプログラム。

行内注記 (in-line comment)

原始行で、1 つ以上の COBOL の語または文字列の後に続く注記。

共有ファイル (shared file)

ファイル共有が指定されたファイル。

共有モード (sharing mode)

ファイル共有のモードを示す、開かれたファイル結合子の状態。共有モードには、他のすべてとの共有、どれとも共有しない、および読み込み専用で共有の 3 つがある。

句 (clause)

記述項の属性を指定するために並べた、一連の COBOL の文字列。

句読文字 (punctuation character)

以下の表に含まれる文字。

文字	意味
,	コンマ
;	セミコロン
.	終止符
"	引用符
(	左かっこ
)	右かっこ
	空白
=	等号 (仮原文区切り記号に含まれる)
'	アポストロフィ
:	コロン
&	アンパサンド

区分番号 (segment-number)

手続き部内の節を、区分化の目的で分類するための利用者語。区分番号には、0 から 9 までの数字だけが使用できる。区分番号は、1 桁または 2 桁の数字で表わすことができる。

組合せ条件 (combined condition)

2 つ以上の条件を、論理演算子の"AND"または"OR"で結んだ条件。

組み込み関数名 (intrinsic-function-name)

作成者により、関数の値を決定するために提供された機構の名を現す語。

クラス (class)

0 個、1 個以上のオブジェクトに対する共通の動作および使用を定義するもの。同じ使用法のオブジェクトは、同じクラスのオブジェクトと見なされる。

クラス定義 (class definition)

クラスを定義する原始単位。

クラス名 (class-name) (オブジェクト指向の場合)

クラスを識別する、利用者定義の語。

クラスメソッド (class method)

クラスオブジェクトのメソッド。

計算機名 (computer-name)

原始プログラムを翻訳 (コンパイル) したり、実行用プログラムを実行したりする計算機を識別するためのシステム名。

形式 (format)

データを整える形。

形式パラメータ (format parameter)

手続き部見出しの USING 指定の中、または関数、メソッド、プログラムでパラメータに対して使用される名を付ける ENTRY 文の中で指定されたデータ名。

継承 (inheritance) (クラスの場合)

インターフェイスを使用する、および他のクラスの基礎として 1 つ以上のクラスを使用するための仕組み。サブクラス (下位のクラス) が、1 つ以上のスーパークラス (上位のクラス) から継承する。継承するクラスのインターフェイスは、継承されたクラスのインターフェイスに準拠する。

継承 (inheritance) (インターフェイスの場合)

1 つ以上のインターフェイスの仕様を他のインターフェイスの基礎として使用する仕組み。継承するインターフェイスは、継承されたインターフェイスに準拠する。

桁位置 (digit position)

1 桁の数字を収めるのに必要な、物理的記憶単位。その大きさは、該当するデータ項目の用途によって変わる。

結果の一意名 (resultant identifier)

算術演算の結果を収めるための、利用者が定義するデータ項目。

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。

現在のレコード (current record)

ファイルのレコード領域中で、利用できる状態にあるレコード。

原始単位 (source unit)

見出し部で始まり、終了符号または翻訳集団の最後 (それに含まれる原始単位を含む) で終わる一連の文。

原始文操作文 (source text manipulation statement)

語 COPY または REPLACE で始まる文。この REPLACE により挿入機能が付与され、翻訳中に原文が置換される。

原始要素 (source element)

入れ子になった原始単位以外の原始単位。

原文語 (text-word)

COBOL 登録集、原始プログラム、仮原文の中の、境界 A から境界 R の間の 1 文字または一連の文字のうち、下記のもの。

1. 分離符。ただし、空白、仮原文区切り記号、文字定数の両端の区切り記号を除く。左かっこと右かっこは、登録集、原始プログラム、仮原文の文中のどこにあっても常に原文語とみなされる。
2. 定数。文字定数の場合、その両端を囲む引用符を含む。
3. 上記以外の、連続する COBOL の文字。ただし、複数の分離符 (1 つの分離符および定数を除く) によって区切られた注記行および語 COPY を除く。分離符のいずれか、または両方は仮原文区切り記号でもよい。

原文名 (text-name)

登録集原文 (ライブラリ・テキスト) を識別する、利用者語。

語 (word)

30 字以下の文字列。利用者語、システム名、予約語、関数名として使用される。

降順キー (descending key)

データ項目の比較規則に従って、値が最大のものから最小のものの順にデータを整列させるために使用するキー。

更新項目 (update field)

画面項目のうち、その記述中に USING が指定されているもの。

構成節 (Configuration Section)

環境部内にあって、翻訳用計算機および実行用計算機の全般的な仕様を記述する節。

構文 (syntax)

翻訳集団を形成するために、各要素を組み合わせて並べる順序。

項目 (field) (画面の場合)

基本画面項目を表す、端末画面の隣接領域。

固定形式原文 (fixed format source)

原文の形式の一つ。ここでは、各 COBOL 原始行が 80 文字で構成され、それがさまざまな固定領域に分かれている。各領域に出現可能な構文には制限がある。もう一つの形式は自由形式原文である。

固定小数点数 (fixed-point representation)

位置を表す数字で、各数字は単一の連続する桁により表される。また、その基数点の位置は、明示的または暗黙的な基数点の位置に従って、桁の右端に固定されている。

固定小数点数字項目 (fixed-point numeric item)

固定小数点数を使用する数字データ項目。

固定小数点数字定数 (fixed-point numeric literal)

固定小数点数で表される、基数が 10 の数。この数は、1 個以上の数字から成る定数として表される。また、小数点や代数記号のいずれか、または両方を含む場合もある。

固定長レコード (fixed-length record)

ファイルに関連づけられたレコードの一つ。関連づけられたファイルのファイル記述または整列併合記述項により、すべてのレコードが同じ数の文字位置を含むことが要求される。

固定方式 (fixed format mode)

画面上の数字項目および数字編集項目にデータを入力するときの、省略時解釈の方式。この方式では、入力されたデータは形式を整えられて画面に表示される。また、キーボードからデータを入力するにつれて、該当する項目に指定されている PICTURE の内容に応じて、カーソルが移動される。"+"と"-"と小数点以外の文字は受け付けられない。編集項目中の挿入文字は、カーソルが前後に移動する際に、飛ばされる。符号を表わすものは正規の仕様に従って変更される。浮動記号は項目中を左右に移動される。数字が挿入されたり削除されたりするのに応じて、挿入記号が表示されたり消去されたりする。

固有の照合順序 (native collating sequence)

実行用計算機段落に指定した、計算機に固有の省略時解釈の文字の照合順序。

固有文字集合 (native character set)

実行用計算機段落に指定した、計算機に作成者が定義した文字集合。

混合定数 (mixed literal)

2 バイト文字を含む、文字定数。

コンパイラ指令 (compiler directive)

COBOL コンパイラへの指示。これにより、コンパイラが翻訳集団の翻訳時に特定の動作を取る。

コンパイラ指令行 (compiler directive line)

2 つの連続する COBOL 文字 >> と、その後のコンパイラ指令語で始まる原文の行。

コンパイラ指令語 (compiler-directive word)

コンパイラ指令の構文で使用される COBOL の語。

(さ行)

最右端 (low order end)

文字列の右端の文字。

最左端 (high order end)

文字列の左端の文字。

作業場所節 (Working-Storage Section)

データ部内にあって、作業場所データ項目を記述する節。作業場所節は、独立項目または作業場所レコードのどちらか一方または両方から構成される。

索引ファイル (indexed file)

索引編成のファイル。

索引編成 (indexed organization)

ファイル内の各レコードを、レコード中の 1 つ以上のキーの値によって識別する、永続的な論理ファイル構造。

サブクラス (subclass)

他のクラスから継承するクラス。2 つのクラスが継承関係にある場合、継承する側のクラスがサブクラス、継承される側のクラスがスーパークラスと見なされる。

業界の文献では、「派生クラス」という用語がしばしば、「サブクラス」という用語の代わりに使用される。この 2 つの用語は同義である。

作用対象 (operand)

操作の対象となる要素。本書では、一般形式の中に日本語で示した語が、作用対象を表わす。これは、作用対象によって示されたデータへの暗黙的な参照である。

算術演算子 (arithmetic operator)

下記の 1 文字または 2 文字の組合わせ。

文字	意味
+	加算
-	減算
*	乗算
/	除算
**	べき乗

算術式 (arithmetic expression)

数字基本項目の一意名、数字定数、それらの一意名や定数を算術演算子でつないだもの。または、算術式を算術演算子でつないだもの。または、かっこで囲んだ算術式。

参照キー (key of reference)

索引ファイル中のレコードを呼び出すために現在、ファイル結合子により使用されている主キーまたは代替キー。

式 (expression)

算術または条件を表したもの。

シグニチャ (signature)

原始単位について、外部リポジトリに保管されている情報。

指数部 (exponent)

浮動小数点数において、その基数から得られる累乗。

システム指示文 (system-directing statement)

指示動詞で始まる文。この動詞により、COBOL システムが、中間符号の作成中に特定の動作を起こす。

システム名 (system-name)

操作環境に情報を伝えるために使用する、COBOL の語。

実行時 (execution time)

関数、メソッド、またはプログラムが実行される時点。

実行時 (run-time)

COBOL システムにより作成されたコードが実行される時点。

実行単位 (run unit)

実行時に、問題を解決する要素として相互に動作する、 1 つ以上の実行モジュール。

実行の不成功 (unsuccessful execution)

文を実行しようとしたが、そこで指定された動作が全部は完了しなかったという結果。

実小数点 (actual decimal point)

データ項目中の小数点の位置を、終止符 (.) またはコンマ (,) を使用して、実際に表現したもの。

指定 (phrase)

記述項または文の属性を指定する、一連の COBOL 文字列。手続き部以外のすべての部の中では、句または段落の下位に位置し、手続き部の中では文の下位に位置する。

自動データ (automatic data)

局所記憶節で記述されたデータ。

自動データ項目 (automatic data item)

自動データレコードの一部として記述されたデータ項目。

自動データレコード (automatic data record)

局所記憶節で記述された論理的レコード。

指標 (index)

表の中の特定の要素を識別するために使用する、計算機の記憶領域またはレジスタ。

指標付きデータ名 (indexed data-name)

データ名とその後ろにかっこで囲んで付けた、1 つ以上の指標名からなる一意名。

指標データ項目 (index data item)

指標名に関連する値を格納するデータ項目。

指標名 (index-name)

特定の表に付けた指標の名前を表わす利用者語。

修飾語 (qualifier)

1. 一意名を指定するときに、同じデータ階層中の下位のレベルのデータ名と共に使用するデータ名。

2. 段落名を指定するときに共に指定する、その段落を含む節名。

3. 原文名を指定するときに共に指定する、その原文を含む登録集（ライブラリ）名。

修飾されたデータ名 (qualified data-name)

一意名のうち、連結語の OF または IN の後にデータ名を書いた修飾語の組を、データ名の後ろにいくつか続けたもの。

集団項目 (group item)

一連の基本項目の組に名前を付けたもの。

自由方式 (free format mode)

画面上の数字項目および数字編集項目にデータを入力する方式の一つ。初期設定では固定方式が使用される。この方式は設定を変更して、適切な大きさの PIC X 領域にデータをキー入力でき、カーソルが入力項目内にある間は、キーボードから入力したデータはそのまま入力項目内に保持される。カーソルが入出力項目の外に出された時点ではじめて、PICTURE 仕様に従ってデータが編集される。このとき、数字と符号と小数点以外のすべての文字は無視される。その後、画面または作業場所項目と同じ形式の項目へと転記（MOVE）するときの通常の COBOL の規則に従って、数字が抽出、編集、または格納される。得られた数値は、画面上に表示されるのが普通である。

自由方式原文 (free format source)

250 バイト以下の文字から成る各 COBOL 原始行の原始形式。構文は、行内のどこに置かれていてもよい。この形式の他に、固定方式現部という形式がある。

終了キー (termination key)

端末の機能キーのいずれか、または改行キー。

終了見出し (end marker)

原始単位の最後を示す見出し。

出力項目 (output field)

画面項目のうち、FROM 指定を含むもの。

出力手続き (output procedure)

SORT 文の実行中に整列が完了した後で、または MERGE 文の実行中に、併合順で次のレコードが選択された後で制御が移される、一連の手続き。

出力ファイル (output file)

出力モードまたは拡張モードで開かれたファイル。

出力モード (output mode)

OUTPUT 指定とともに OPEN 文を実行した後から、REEL および UNIT 指定を伴わない CLOSE 文が実行されるまでの、ファイル結合子の状態。

主レコードキー (prime record key)

索引ファイル中のレコードを、一意に識別するキー。

準拠 (conformance) (オブジェクト指向の場合)

あるインターフェイスを持つオブジェクトが使用されるべき箇所で、別のインターフェイスを含むオブジェクトの使用を許可するプロパティ。準拠により、準拠されたインターフェイスに指定されたあらゆる操作が、準拠するインターフェイスによりサポートされることが保証される。

順ファイル (sequential file)

順編成のファイル。

順編成 (sequential organization)

ファイル中にレコードを格納したときに確立される先行･後行関係に基づいてレコードを識別する、永続的な論理ファイル構造。

順呼出し (sequential access)

論理レコードを、ファイル中のレコードの順序によって決まる連続した先行レコードと後行レコードの順に、読み込んだり書き出したりする呼出し法。

条件 (condition)

真理値がプログラムの実行時に決定され状態。一般形式の「条件」(条件-1、条件-2、など) の中または条件を参照する中に、「条件」(条件-1、条件-2、など) という用語があるとき、これは真理値を定める条件式を表わす。条件式を構成するものには、単純条件 (かっこで囲んでもよい) 、組合せ条件 (構文的に正しい単純条件の組合わせからなる) 、論理演算子、かっこがある。

条件式 (conditional expression)

IF 文、PERFORM 文、EVALUATE 文、SEARCH 文の中に指定された、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)* の項を参照。

条件文 (conditional statement)

条件の真理値を判定して、その値によって、実行用プログラムのその後の処理が決定される文。

条件変数 (conditional variable)

1 つ以上の値に条件名を割り当てられたデータ項目。

条件名 (condition name)

条件変数がとることのできるすべての値の中の、ある特定の値や、値の組や、値の範囲に付けた利用者語。または、作成者が定義したスイッチまたは装置に付けた利用者語。

条件名条件 (condition-name condition)

ある条件変数の値が、その条件変数に関連する条件名に割り当てられている値の組に含まれるかどうかを問う命題。この結果によって真理値が定まる。

条件名データ記述項 (88 level-description-entry)

条件名を記述するデータ記述項。レベル番号に 88 を使用することになっている。

昇順キー (ascending key)

データ項目の比較規則に従って、値が最小のものから最大のものの順にデータを整列させるために使用するキー。

初期データ (initial data)

初期プログラムの作業場所節またはファイル節に記述されたデータ。

初期データ項目 (initial data item)

初期データレコードの一部として記述されたデータ項目。

初期データレコード (initial data record)

初期プログラムの作業場所節またはファイル節に記述された論理的レコードで、このプログラムが呼ばれるたびに初期化される。

初期ファイル結合子 (initial file connector)

初期プログラムに記述されたファイル結合子で、このプログラムが呼ばれたときにOPEN モードにならない。

初期プログラム (initial program)

実行要素中でプログラムが呼ばれるたびに、初期状態となるプログラム。

初期状態 (initial state)

実行要素中で最初に起動されたときの、関数、メソッド、またはプログラムの状態。

字類条件 (class condition)

作用対象 (オペランド) がすべて英字、数字、小文字の英字、大文字の英字であるか、さらに、環境部の特殊名段落中で CLASS 句によって定義されている文字集合中の文字だけを含むのかどうかを問う命題。この結果によって真理値が定まる。

字類名 (class-name) (**真理値の命題の場合**)

環境部の特殊名段落で定義された利用者定義の語で、データ項目の内容が字類名の定義で並べた文字だけで構成されているかどうかにより、真理値が決定される命題に付けられた名。

真理値 (truth value)

ある条件を評価した結果の値。「真」または「偽」のどちらかで表わされる。

スイッチ状態条件 (switch-status condition)

作成者が定義した「オン」または「オフ」の状態に設定できるスイッチが、指定された状態になっているかどうかを問う命題。この結果によって真理値が定まる。

数字 (numeric character)

以下のいずれかの文字。

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

数字関数 (numeric function)

字類および項類が数字に属する関数であるが、関数評価上の何らかの理由によって、整数関数の条件を満たさないもの。

数字項目 (numeric item)

0 から 9 の数字によってだけ表わされる値を取る、データ項目。符号付きの場合は、"+"と"-"またはそれ以外の演算符号を含むことができる。

数字定数 (numeric literal)

固定小数点数字定数 (fix-point numeric literal) および *浮動小数点数字定数 (floating-point numeric literal)* の項を参照。

正書法 (reference-format)

原始文および登録集文を記述する標準的な方法とされる形式。

整数 (integer)

小数点の右側に有効数字のない数字定数または数字データ項目。

整数関数 (integer function)

項類が数字の関数で、どのような場合にも、その関数を評価した結果として返される値の小数点の右側がゼロであると定義されるもの。

静的データ (static data)

初期プログラム以外の原始要素のファイル節または作業場所節に記述されたデータ。

静的データ項目 (static data item)

静的データレコードの一部として記述されたデータ項目。

静的データ・レコード (static data record)

初期プログラム以外の原始要素のファイル節または作業場所節に記述された論理的レコード。

正負条件 (sign condition)

データ項目または算術式の代数値が、正であるか負であるかゼロであるかを問う命題。この結果によって真理値が定まる。

整列併合用ファイル記述項 (sort-merge file description entry)

整列ファイルまたは併合ファイルの性質を指定する、データ部の記述項。

整列用ファイル (sort file)

SORT 文によって整列 (ソート) される、レコードの集合。整列用ファイルは、整列関数によってだけ作成され、使用される。

節 (section)

節の本体とよばれる、0 個以上の段落または記述項の集合。節の本体の前には節の見出しが置かれる。各節は、節の見出しとそれに関連する節の本体から成る。

節の見出し (section header)

環境部、データ部、手続き部において、終止符 (.) および空白で終わる、語の組み合わせ。節の始まりを示す。

節名 (section name)

手続き部内の節に付ける名前として使用する、利用者語。

宣言完結文 (declarative-sentence)

単一の USE 文を、分離符の終止符 (.) で止めた翻訳指示完結文。

宣言部分 (declaratives)

手続き部の冒頭に続けて書いた、特殊な目的用のいくつかの節。最初の宣言節の冒頭には必要語 DECLARATIVES を置き、最後の宣言節の末尾には必要語 END DECLARATIVES を置く。宣言部分には、まず節の見出しを書き、続けて USE 翻訳指示完結文と必要な段落を書く。段落は、必要ない場合もある。

相対キー (relative key)

相対ファイル中の論理レコードを識別するためのキー。

相対ファイル (relative file)

相対編成のファイル。

相対編成 (relative organization)

ファイル中のレコードを 1 以上の整数値によって一意に識別する、永続的な論理ファイル構造。この整数値は、ファイル中のレコードの論理的な順序を表わす。

相対レコード番号 (relative record number)

ファイル中のレコードの順序番号で、相対的に編成されているもの。

想定小数点 (assumed decimal point)

データ項目中に実小数点を含まないで指定された、小数点位置。想定小数点は、論理的には意味をもつが、物理的な表現はなされない。

添字 (subscript)

表の中の、個々の要素を識別するための出現番号。この番号は、整数またはデータ名または指標名によって表わす。データ名または指標名を使用する場合は、その後ろに、演算子の正号 (+) または負号 (-) に続けて整数を記述してもよい。関数の引数に添字付けした一意名を使用するときは、語 ALL を添字として使用できる。

添字付きデータ名 (subscripted data-name)

データ名の後ろに、いくつかの添字をかっこで囲んで付けたデータ名。

(た行)

大域名 (global name)

1 つのプログラム中でのみ宣言されている名で、そのプログラム自体、およびそれに含まれるプログラムにより参照される。

大記憶 (mass storage)

データを、順にも乱にも構成し、維持できる媒体。

大記憶ファイル (mass storage file)

大記憶媒体に収められたレコードの集まり。

単位 (unit)

リール (reel) の項を参照。

単純条件 (simple condition)

下記の条件の 1 つ。

a. 比較条件
b. 字類条件

c. スイッチ状態条件
d. 条件名条件
e. 正負条件
f. 上記の単純条件をかっこで囲んだもの

端末 (terminal)

表示画面とキーボードを含む、文字の入力が可能な装置。

段落 (paragraph)

見出し部および環境部においては、段落の見出しに続く、いくつかの記述項 (記述項がない場合もある) 。手続き部においては、段落名 (終止符に続く空白で止める) に続く、いくつかの文 (文がない場合もある) 。

段落の見出し (paragraph header)

見出し部および環境部で、段落の始まりを示す予約語。終止符 (.) に続く空白で止める。

段落名 (paragraph-name)

手続き部で、段落を識別し、その始まりを示す利用者語。

注記行 (comment line)

原始プログラム中の標識領域に、星印 (*) を付けた行。その行の A 領域および B 領域に、計算機の文字集合中の任意の文字を記述できる。注記行は、プログラム中で説明のためだけに使用する。もう 1 つ、特別の注記行を表わすものとして、標識領域に斜線 (/) を書くことができる。この場合、その注記行が印字される前に、改ページされる。

注記項 (comment entry)

見出し部中の記述項で、計算機の文字集合中の任意の文字を含めることができる。注記のみを目的とする注記項は、複数の行にわたることもあり、部名、節名、または段落名が出現した時点で終わりとなる。

注記指示語 (comment indicator)

隣接した 2 つの COBOL 文字 '*>' のことで、注記行または行内注記であることを示す。

抽象クラス (abstract class)

インスタンス･オブジェクトを作成しないクラス。抽象クラスで、その下位クラスの動作を使用できる。

通貨記号 (currency sign)

COBOL 文字 '$' のことで、PICTURE 文字列中の通貨記号の初期値として使用される。また、データ項目の編集形式中に現れる通貨文字列の初期値としても使用される。

注：他の通貨文字列および通貨記号を選択するための機能が存在する。

通貨編集用文字 (currency symbol)

形式文字列の中で、通貨記号の存在を示すのに使用される文字。***環境部***の章の*特殊名段落*節を参照。

通貨文字列 (currency string)

数字編集データ項目がその PICTURE 文字列中に通貨記号を含む場合に、そのデータ項目中に編集操作の結果として配置される 1 組の文字列。***環境部***の章の*特殊名段落*節を参照。

次の実行完結文 (next executable sentence)

現在の文の実行が完了した後で制御が移される、次の完結文。

次の実行文 (next executable statement)

現在の文の実行が完了した後で制御が移される、次の文。

次のレコード (next record)

ファイル中の現在レコードの、論理的に次のレコード。

定義済みオブジェクト参照 (predefined object reference)

一意名、NULL、SELF、SELFCLASS、および SUPER のどれかにより参照される、暗黙的に生成されたデータ項目。

定数 (literal)

記述したとおりの値が、暗黙に設定される文字列。

定数名 (constant-name)

固定値（定数）の名前として割り当てられた利用者語。

定数名データ記述項（78 level-description-entry）

定数名を記述するデータ記述項。レベル番号に 78 を使用することになっている。

定数項目（literal field）

画面上の基本項目で、その記述項中に PICTURE 句が含まれないもの。

データ・キー（data keys）

文字を入力する端末のキーボード上のキーで、個々の印刷可能なデータ文字を表すもの。

データ記述項（data description entry）

データ部中の記述項。レベル番号と、それに続くデータ名と、必要な一連のデータ句とから構成される。

データ項目（data item）

データ記述項により定義された、データの単位。または、一意名の評価の結果である、データの単位。

データ名（data-name）

データ部中のデータ記述項に記述されたデータ項目の名前である利用者語。一般形式においては、添字も指標も付けていないものを指す。ただし、形式に関する規則に、添字または指標を使えると明記してある場合は例外である。

手順名（routine-name）

COBOL 以外の言語で書かれた手続き（ルーチン）を識別するための利用者語。

手続き（procedure）

手続き部の中の、1 つまたは論理的に連続したいくつかの段落。または、1 つまたは論理的に連続したいくつかの節。

手続き部の終わり（end of procedure division）

COBOL プログラム中の、物理的な終端。これ以降には、プログラム手続きは現れない。

手続き分岐文（procedure branching statement）

原始プログラム中に書かれた順序に従わず、次の実行文ではない他の文への制御の明示的な移行を起こす文。手続き分岐文には、CALL、EXIT、EXIT METHOD、EXIT PROGRAM、GO TO、GOBACK、INVOKE、MERGE (OUTPUT PROCEDURE 指定を伴う) 、PERFORM、および SORT (INPUT PROCEDURE 指定または OUTPUT PROCEDURE 指定を伴う) がある。

手続きポインタデータ項目 (procedure-pointer data item)

プログラムの番地を含むデータ項目。

手続き名 (procedure-name)

手続き部内の段落または節に、名前を付けるために使用する利用者語。

デバッグ行 (debugging line)

固定形式の原文で、行の標識領域に"D"または"d"が含まれる行。または、自由形式の原文で、最初の文字が"D"または"d"で、その次が空白である行。

デバッグ節 (Debugging Section)

USE FOR DEBUGGING 文を含む節。

動詞 (verb)

固有コード (機械語) の生成中または実行時に、COBOL プログラムが取る動作を記述する語。

動的呼出し (dynamic access)

呼出し方式で、OPEN 文を実行してから、論理レコードを順序に従わずにディスク・ファイルから取り出したり、格納できる (*乱呼出し (Random Access)* の項を参照) 。また、論理レコードを順序に従ってディスク・ファイルから取り出すこともできる (*順呼出し (sequential access)* の項を参照) 。

登録集原文 (library-text)

翻訳時に、COPY 文により原始文中に取り入れられる、COBOL 登録集 (ライブラリ) 中の一連の文字列および区切り文字。

登録集名 (library-name)

目的コードを生成中に、COBOL システムによって使用される COBOL 登録集原始ファイルに名前を付ける利用者語。

特殊名 (Special-Names)

環境部の中の段落の名。この段落中で、作成者語が利用者が付けた呼び名に関連づけられる。

特殊文字 (special character)

以下の表にある文字の一つ。

文字	意味
+	正号
-	負号
*	星印 (アスタリスク)
/	斜線 (スラッシュ)
=	等号
\	通貨記号
,	コンマまたは小数点
;	セミコロン
.	終止符 (ピリオド) または小数点
"	引用符
(	左かっこ
)	右かっこ
>	より大きい記号
<	より小さい記号
'	アポストロフィ
:	コロン
&	アンパサンド

特殊文字語 (special-character word)

特殊文字のみから成る予約語。

特殊レジスタ (special registers)

COBOL システムによって作成される記憶領域。主として、特定の COBOL 機能を使う利用者用に作成される情報を記録するために使用される。

独立項目 (noncontiguous item)

作業場所節、局所記憶節、連絡節中の基本データ項目で、他のデータ項目と階層関係をもたないもの。

独立データ記述項 (77 level-description-entry)

独立データ項目を記述するデータ記述項。レベル番号に 77 を使用することになっている。

独立のプログラム (separate program)

その中に含んでいるプログラムを含め、他のプログラムには含まれないプログラム。

(な行)

内部データ (internal data)

プログラム中に記述されている、外部データ項目および外部ファイル結合子以外のデータ。連絡節中に記述したデータは、内部データとして扱われる。

内部ファイル結合子 (internal file connector)

実行単位中の 1 つの実行用プログラムからだけ呼び出すことのできる、ファイル結合子。

長さゼロの集団項目 (zero-length group item)

最小サイズがゼロで、実行時のサイズがゼロの項目。長さゼロの集団項目は、以下のいずれかの結果として作成される。

- 最小反復回数がゼロの変数反復データ項目のみを含む集団データ項目。
- 下位に長さゼロの集団項目のみを含む集団データ項目。
- 文字位置の最小数がゼロである、RECORD 句の形式 2 または形式 3 を使用して指定されたレコード。
- 長さがゼロの項目を返す関数。

入出力管理 (i-o control)

環境部内にあって、入出力に関する実行用プログラムの要件を記述する段落の名前。この段落中に、入出力技法、再開始点、複数のデータ･ファイル間での同じ領域の共有、および単一の入出力装置上への複数のファイルの格納を指定する。

入出力制御システム (input-output control system) (IOCS)

ファイルの指示、制御、または処理を行う作成者により提供されたシステム。

入出力節 (Input-Output Section)

環境部中にあって、ファイルの入出力に関する記述を書く節。この節で、プログラムで使用するファイルおよび外部媒体に名前を付け、プログラム実行中のデータの伝送や操作に必要な情報を記述する。

入出力両用ファイル (input-output file)

入出力両用モードで開いたファイル。

入出力両用モード (i-o mode)

ファイルに対し、I-O 指定をして OPEN 文を実行した後から、このファイルに対し、REEL 指定も UNIT 指定も伴わない CLOSE 文を実行するまでの、ファイル結合子の状態。

入力キー (Enter key)

文字を入力する端末のキーボード上のキーで、画面項目または画面レコードの入力が完了したという合図を送るもの。

入力項目 (input field)

記述中に TO 指定が含まれる画面項目。

入力手続き (input procedure)

整列されるよう指定したレコードの引き渡しが制御されるよう、SORT 文の実行中に制御を獲得する一組の文。

入力ファイル (input file)

入力モードで開かれるファイル。

入出力文 (input-output statement)

個々のレコードまたはファイルを単位とした操作の実行による、ファイルの処理を起こす文。入出力文には、CLOSE、DELETE、OPEN、READ、REWRITE、START、および WRITE がある。.

入力モード (input mode)

ファイルに対し、INPUT 指定をして OPEN 文を実行した後から、このファイルに対し、REEL 指定も UNIT 指定も伴わない CLOSE 文を実行するまでの、ファイル結合子の状態。

ヌル (null)

1. データ項目により、どの記憶場所の番地も示されないことを保証するポインタの状態 - 定義済みのヌルデータ番地値により示される。
2. データ項目により、どのプログラムの番地も指示されないことを保証する手続きポインタの状態 - 定義済みのヌルプログラム番地値により示される。
3. データ項目により、どのオブジェクトも参照されないことを保証するオブジェクト参照の状態 - 定義済みのオブジェクト参照の値 NULL により示される。

(は行)

廃要素 (obsolete element)

現在の版の COBOL 標準にはあるが、次の版からは削除される予定の言語要素。

パラメータ化されたインターフェイス (parameterized interface)

1 つ以上の依存クラスまたは依存インターフェイスの情報が、インターフェイス定義で完全には指定されていないインターフェイス。依存クラスまたは依存インターフェイスの情報が完全に指定された後に、パラメータ化されたインターフェイスが使用される。

パラメータ化されたクラス (parameterized class)

1 つ以上の依存クラスまたは依存インターフェイスの情報が、クラス定義で完全には指定されていないクラス。依存クラスまたは依存インターフェイスの情報が完全に指定された後に、パラメータ化されたクラスが使用される。

範囲明示文 (delimited scope statement)

範囲符を明示した (記述した) 文。

比較 (relation)

比較演算子 (relational operator) の項を参照。

比較演算子 (relational operator)

比較条件を記述するために使用する、予約語や比較文字の組。下記のものがある。

比較演算子	意味
IS [NOT] GREATER THAN IS [NOT] >	より大きい、または、[より大きくない]
IS [NOT] LESS THAN IS [NOT] <	より小さい、または、[より小さくない]
IS [NOT] EQUAL TO IS [NOT] =	等しい、または、[等しくない]
IS GREATER THAN OR EQUAL TO IS >=	以上
IS LESS THAN OR EQUAL TO IS <=	以下
IS UNEQUAL TO IS <>	等しくない
EQUALS	等しい
EXCEEDS	より大きい

比較条件 (relation condition)

ある算術式またはデータ項目の値が、他の算術式またはデータ項目の値と、特定の関係にあるかどうかを問う命題。この結果によって、真理値が定まる。

比較文字 (relation character)

以下の文字のいずれか。

文字	意味
>	より大きい
<	より小さい
=	等しい
>=	以上
<=	以下

<>	等しくない

引数 (argument)

関数、メソッド、またはプログラムの中の、実行中の文の中で指定された作用対象。引数により、組み込み関数またはランタイム要素の起動により渡されるデータが指定される。

必要語 (key word)

語が出現する一般形式が使用されるときに、必要とされる予約語または組み込み関数名。

否定組合せ条件 (negated combined condition)

かっこで囲んだ組合せ条件の直前に、論理演算子の NOT を付けたもの。

否定単純条件 (negated simple condition)

単純条件の直前に、論理演算子の NOT を付けたもの。

表 (table)

データ部で OCCURS 句を使用して定義する、論理的に連続したデータ項目の集合。

表意定数 (figurative constant)

特定の予約語の使用により参照される値。COBOL システムにより生成される。

標識領域 (indicator area)

COBOL 原始コードの、左端の文字位置。ここに記入した値によって、その用途が示される。

標準データ形式 (standard data format)

COBOL プログラムのデータ部内でデータの特性を記述するために用いる概念である。この概念では、無限の長さと幅をもつページにデータを印字することを想定して、データの性質や特性を表現する。計算機の内部や何らかの外部媒体にデータを格納することは考慮されない。

表要素 (table element)

表を構成する一組の項目の中の、個々のデータ項目。

部 (division)

COBOL プログラムには、見出し部、環境部、データ部、手続き部の 4 つがある。各部は、部の見出しと、一定の規則に従って構成される 0 個以上の節または段落から構成される。

ファイル (file)

論理的レコードの物理的な集合。

ファイル位置指示子 (file position indicator)

概念上の指示子であり、次のいずれかを示す。

a. 索引ファイルに対する参照キー中の現在のキーの値
b. 順ファイルに対する現在のレコードのレコード番号
c. 相対ファイルに対する現在のレコードの相対レコード番号
d. 次の論理的レコードが存在しないこと
e. 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桁数より大きいこと
f. 入力の不定ファイルが存在しないこと
g. ファイル終了条件がすでに存在していること
h. 有効な次のレコードが作成されていないこと

ファイル管理記述項 (file control entry)

ファイルの有効な物理的属性を宣言する、SELECT 句およびその下位の句。

ファイル記述項 (file description entry)

データ部のファイル節中にあって、ファイルに関する情報を記述する部分。レベル指示子の FD から始まって、それに続くファイル名と、必要な一連のファイル句から構成される。

ファイル共有 (file sharing)

同じ物理的ファイルへの同時アクセスを制御する、協力的な環境。

ファイル結合子 (file connector)

ファイルに関する情報を収録した記憶領域。ファイル名と物理ファイルの間、およびファイル名とそのレコード領域の間をつなぐために使用される。

ファイル固有属性 (fixed file attribute)

ファイルを作成したときに決定され、以降そのファイルが存在している間は変更できない、ファイルの属性情報。具体的には、ファイルの編成（順編成、相対編成、索引編成）、主レコードキー、副レコードキー、符号系、レコードの最大の大きさと最小の大きさ、レコードの型（固定長か可変長か）、索引ファイル用のキーの文字の照合順序、物理レコードの最大の大きさと最小の大きさ、充てん文字、レコードの区切り文字がある。

ファイル終了条件（at end condition）

下記のどれかの場合に発生する条件。

1. 順呼出しファイルに対する READ 文を実行中に、次のレコードがなくなったとき。
2. RETURN 文を実行中に、整列併合用ファイルの次のレコードがなくなったとき。
3. SEARCH 文を実行中に、関連する WHEN 指定の条件を 1 つも満たすことなく表引き操作が終了したとき。

ファイル状態（file status）

入出力操作の状態を示す 2 文字の値で表される、概念的な状態。

ファイル節（File Section）

データ部内にあって、ファイル記述項と整列併合ファイル記述項、およびそれらのレコード記述項を含む節。

ファイル編成（file organization）

ファイルを作成するときに永続的に決定される、ファイルの論理構造。

ファイル名（file name）

ファイルの名前として付けた利用者語。データ部のファイル節のファイル記述項、または整列併合ファイル記述項中に指定する。

ファクトリ定義（factory definition）

ファクトリオブジェクトを定義する原始単位。

ファクトリメソッド（factory method）

ファクトリオブジェクトに対するメソッド。

ファクトリオブジェクト (factory object)

クラスの単一のオブジェクトで、クラスのファクトリ定義により定義される。各クラスには、1 つのファクトリオブジェクトが存在する。ファクトリオブジェクトは、クラスのオブジェクトを作成する。

ファクトリ (オブジェクト) データ (factory (object) data)

ファクトリオブジェクトに対するデータで、ファクトリ定義のデータ部で宣言され、そのクラスの全オブジェクトに共通である。

複合条件 (complex condition)

1 つ以上の論理演算子が、1 つ以上の条件に作用する条件。*否定単純条件 (nagated simple condition)*、*組合せ条件 (combined condition)*、*否定組合せ条件 (nagated combined condition)* の各項を参照。

副プログラム (subprogram)

呼ばれるプログラム (called program) の項を参照。

副レコードキー (alternate record key)

索引ファイル中のレコードを識別するための、主レコードキー以外のキー。

符号系名 (alphabet-name)

使用する文字集合や、文字の照合順序に付けた名前。環境部の特殊名段落中で指定する、利用者語。

物理ページ (physical page)

作成者により定義された、装置に依存する概念。

物理レコード (physical record)

ブロック (block) の項を参照。

不定ファイル (optional file)

ランタイムモジュールが実行されるときに必ずしも存在しなくてもよいように宣言されたファイル。

浮動小数点数 (floating-point representation)

仮数部と指数部という、連続した数字により表される数字の表現方法。

浮動小数点数定数 (floating-point literal)

浮動小数点形式で表わした数量。仮数部と指数部に分けて表現される。仮数部は、10 を基数にした固定小数点定数であり、小数点を含まなければならない。指数部は、文字"E"の後ろに符号付きで小数点を含まない固定小数点定数を続けたものである。

部の見出し (division header)

部の始まりを示すいくつかの語を、終止符 (.) と続く空白で止めたもの。下記のものがある。

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.

[{USING   }                          ]
PROCEDURE DIVISION [{        } データ名-1 [データ名-
2] ...]

[{CHAINING}                          ]
```

部分参照 (reference modification)

データ項目の一部を、データ項目として定義すること。このためには、元のデータ項目の文字位置の中で、新しいデータ項目の左端とする位置 (オフセット) と、取り出す文字の数 (長さ) を指定する。

部分参照名 (reference-modifier)

一意のデータ項目を表す一意名。一意名、開始位置、および長さの指定により作成される。

プログラム終了見出し (end program marker)

分離符の終止符 (.) で終わる一組の語。COBOL 原始プログラムの終わりを示す。下記の形式をしている。

END PROGRAM プログラム名 .

プログラム宣言 (program declaration)

プログラムがどのように呼び出されるか、つまり呼出しプロトタイプの宣言のみを行う原始プログラム。プログラム定義とは異なり、実行可能な手続きはまったく含まない。*呼出しプロトタイプ*の項を参照。

プログラム定義 (program definition)

プログラムが、手続き部で指定された手続きで実行する方法定義する原始プログラム。プログラム宣言とは区別される、従来の COBOL プログラムである。

プログラム・プロトタイプ定義 (program prototype definition)

特定のサブプログラムが受け取るべきパラメータのクラスに対する規則を指定する定義。また、このサブプログラムに制御を移したり、このサブプログラムから制御を受け取ったり、情報を返すのに必要なその他のあらゆる条件をいう。

プログラム名 (program-name)

COBOL の原始プログラムを識別する利用者語。

ブロック (block)

データの物理的な単位。通常は 1 つ以上の論理レコードから成る。大記憶ファイルの場合は、複数のブロックにまたがって 1 論理レコードを構成することもできる。ファイル中のブロックの大きさとそのファイルの大きさとの間には、直接的な関係はない。また、1 ブロックに複数の論理レコードが含まれる場合でも、1 論理レコードが複数のブロックにわたる場合でも、ブロックの大きさとファイルの大きさとの間には、直接的な関係はない。ブロックは、物理レコードの同義語である。

プロンプト文字 (prompt character)

画面上で、空の文字位置を示して入力を要求する文字。

文 (statement)

手続き部内で、動詞に続けて語や記号を文法的に正しく組み合わせて書いたもの。命令文。

分割キー (split key)

ファイル名に関連づけられたレコード内の 1 つ以上のデータ項目を、論理的にいくつかつなげて 1 つのキーとすること。この形のキーは、START 文と READ 文においてだけ使用できる。分割キーの例は、*言語リファレンス - 追加トピック*の*例*の章に掲載。

分離符 (delimiter)

文字列の最後を示し、それと後続の文字列とを分ける文字 (または隣接する連続した文字) 。分離符 (delimiter) は、分離する文字列の一部ではない。

分離符 (separator)

後続の文字列との間を分離する、1 つの文字または連続する 2 つの文字。

併合用ファイル (merge file)

MERGE 文によって併合 (マージ) されるレコードの集合。併合用ファイルは、併合機能によってだけ作成され、使用される。

編集用文字 (editing character)

PICTURE 文字列で指定された 1 文字または 2 つの文字の組み合わせ。データ項目の編集時に挿入される。

変数 (variable)

実行用プログラムを実行することによって、値が変わり得るデータ項目。

変数長レコード (variable-length record)

ファイルに関連づけられたファイルで、このファイルのファイル記述または整列併合記述項により、レコードに、変数である文字位置を含めることが可能となる。

ポインタ・データ項目 (pointer data item)

データ項目の位置を保管するデータ項目。

補助語 (optional word)

記述を読みやすくするためだけに使用する、予約語。

ボリューム (volume)

リール (reel) の項を参照。

翻訳時 (compile time)

翻訳集団が、1 つ以上の実行可能なランタイム要素から成るランタイムモジュールに翻訳 (コンパイル) される時点。

翻訳指示文 (compiler directing statement)

コンパイラ指令、または原始文操作文。

翻訳集団 (compilation group)

ともに翻訳されるよう提出された、連続する複数の原始単位。

翻訳単位 (compilation unit)

他の原始単位内に入れ子となっていない原始単位。

(ま行)

無効キー条件 (invalid key condition)

索引ファイルまたは相対ファイルのキーに指定された値が、実行時に無効であると判定されたことによって引き起こされる条件。

無条件文 (imperative statement)

無条件に動作を行うことを指定する文。または、明示範囲終了語で終わる条件文 (範囲明示文) 。無条件文は、一連の無条件文で構成することもできる。

明示属性 (explicit attribute)

明示的に定義されている属性。

明示範囲終了語 (explicit scope terminator)

特定の手続き部文の範囲の終わりとなる予約語。

メソッド (method)

メソッド定義の手続き部中で宣言された手続き符号。メソッド呼出しによって、そのオブジェクトに対して実行される。

メソッド定義 (method definition)

メソッドを定義する原始単位。

メソッドプロトタイプ (method prototype)

メソッドのメソッド名およびパラメータ型 (戻す項目が指定されている場合は、それを含む) 。インターフェイス中のメソッド定義により指定される。

メソッド名 (method-name)

メソッドを識別する、利用者定義の語。

メソッド呼出し (method invocation)

あるオブジェクトに対する、指定されたメソッドの実行要求。メソッド呼出しで、オブジェクト、メソッド名、メソッド定義により要求されたパラメータが識別される。

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)

用途が DISPLAY として記述された標準データ形式の 1 文字を収めるために必要な記憶単位。

文字境界 (character boundary)

計算機の記憶領域で、位置の境目である、左端のビット。

文字項目 (nonnumeric item)

計算機の文字集合に属する、任意の文字の組合わせを値に取ることのできる、データ項目。文字項目の項類によっては、使用できる文字が制限される。

文字集合 (character set)

COBOL 言語の文字集合。下記のすべての文字から成る。

文字	意味
0, 1, ..., 9	数字
A, B, ..., Z	大文字の英字
a, b, ..., z	小文字の英字
	空白 (ブランク)
+	正号
-	負号
*	星印 (アスタリスク)
/	斜線 (スラッシュ)
=	等号
\	通貨記号
,	コンマまたは小数点
;	セミコロン
.	終止符または小数点
"	引用符
(	左かっこ

)	右かっこ
>	より大きい記号
<	より小さい記号
:	コロン
'	アポストロフィ
&	アンパサンド

注: 計算機の文字集合に小文字が含まれるときは、これらを文字列や原文語に使用できる。各小文字は、対応する大文字と等しいものとする。ただし、文字定数として使用した場合は例外である。

文字定数 (nonnumeric literal)

引用符で両端を囲んだ文字列。この文字列の中には、計算機の文字集合に属する任意の文字を入れることができる。文字定数の中に、1 つの引用符を入れる場合は、引用符を 2 つ並べて記述する。

文字の照合順序 (collating sequence)

計算機で処理できる文字を並べる順序。整列、併合、比較、および索引ファイルを順に処理するときに使用される。

文字列 (character-string)

文字をいくつか並べたもの。COBOL の語、定数、PICTURE 文字列、注記項を形成する。

(や行)

有効なランタイム要素 (activated runtime element)

有効な状態の文の中に位置する関数、メソッド、またはプログラム。

呼ばれるプログラム (called program)

CALL 文実行の結果として、制御を受け取るプログラム。

呼出しプロトタイプ (call prototype)

EXTERNAL 句を伴う、プログラム名段落中のプログラム。呼出しプロトタイプは、実行されないプログラム宣言で、翻訳システムはこれを使用して、同じ原始ファイル中にあるプログラム定義の CALL 文を有効にする。

呼出し法 (access mode)

ファイル中のレコードを操作する方式。(アクセス法)。

呼出しランタイム要素 (calling runtime element)

CALL 文の実行により、プログラムに制御を移す関数、メソッド、またはプログラム。

呼び名 (mnemonic-name)

環境部において、指定された作成者語に付けた利用者語。(ニモニック)。

予約語 (reserved word)

COBOL 言語の一環として、あらかじめ決めてある語。予約語は、COBOL の原始プログラム中で使用できるが、利用者語またはシステム名として使用してはならない。

(ら行)

ランタイムシステム (run-time system)

COBOL システムによって生成されたコード (実行用プログラム) を解釈して、オペレーティングシステムと CRT へのインターフェイスをとりながら、そのプログラムを実行できる状態にするソフトウエア。(RTS)。

ランタイムモジュール (runtime module)

翻訳単位の翻訳結果。

ランタイム要素 (runtime element)

原始要素の翻訳により得られる実行可能な単位。関数、メソッド、またはプログラムのいずれかである。

乱呼出し (random access)

相対ファイルまたは索引ファイルに対して、論理レコードを読込み、削除、挿入するときに、対象とするレコードを、プログラム中のキー・データ項目の値によって指定できる呼出し法。

リール (単位、ボリューム) (reel (unit, volume))

ファイルの一部、1 ファイル全体、または任意の数のファイルを含む、記憶媒体の独立した単位。

リポジトリ (repository)

クラス名、メソッド名、メソッドパラメータに関する情報や、作成者が要求するその他の情報のための、作成者が提供する記憶場所。

リポジトリ段落 (Repository paragraph)

環境部の構成節中の段落で、ここに、この環境部の範囲内で使用されるプログラムプロトタイプ名、関数プロトタイプ名、プロパティ名、クラス名、およびインターフェイス名を指定できる。 また、語 FUNCTION を指定せずに使用できる組み込み関数名を宣言することもできる。

略記組合せ比較条件 (abbreviated combined relation condition)

一連の比較条件を続けて記述するとき、共通する左辺、右辺または比較演算子を記述上省略した組合わせ条件。

利用者語 (user-defined word)

句または文の形式に適合させるために、利用者が指定しなければならない COBOL の語。

ルートクラス (root class)

それ自体の上位クラスを持たないクラス。

レコード (record)

論理レコード (logical record) の項を参照。

レコードキー (record key)

索引ファイル中のレコードを識別するために使用するキー。主レコードキーと副レコードキーがある。

レコード記述項 (record description entry)

特定のレコードに関連づけられたデータ記述項の全体。

レコードのロック (record locking)

共有ファイルへのレコードアクセス制御で、このファイルには、レコードロックが設定され、他のファイル結合子から関連レコードへのアクセスが拒否されている。

レコード番号 (record number)

順編成のファイル中のレコードの順序番号。

レコード名 (record name)

データ部中のレコード記述項に記述した、レコードに付ける名前として使用する利用者語。

レコード領域 (record area)

ファイル節中のレコード記述項に記述したレコードを処理するために割り当てられた、記憶領域。

レコードロック (record lock)

ファイル中の特定のレコードに関連づけられた指示子で、ロック機能により、設定および解除される。レコードロックを使用して、複数のファイル結合子が同時にレコードにアクセスできるかどうかが決定される。

レベル指示語 (level indicator)

特定のファイルの型を示す 2 文字の英字。データ部のレベル指示語は、FD、RD、および SD である。

レベル番号 (level-number)

論理レコードの階層構造中のデータ項目の位置を示すか、またはデータ記述項の特殊な性質を示すための、利用者語。レベル番号は、1 桁または 2 桁の数字で表わす。

1 から 49 の範囲内のレベル番号は、論理レコードの階層構造中のデータ項目の位置を示す。1 から 9 までのレベル番号は、1 桁の数字として書いてもよいし、前に 0 を付けて書いてもよい。

レベル番号 66, 77, 78, 88 は、特殊な性質のデータ記述項を示す。

連鎖されるプログラム (chained program)

CHAIN 文の右辺に指定されたプログラム。

連鎖するプログラム (chaining program)

他のプログラムに対して CHAIN 文を実行し、そのプログラムへ連鎖（チェイン）するプログラム。

連続項目 (contiguous item)

データ部内で、連続的に記述されている項目。これらの項目間には、階層関係が作られる。

連絡節 (Linkage Section)

呼ばれるプログラムのデータ部中にあって、呼ぶプログラム側で使用するデータ項目を記述する節。この節に記述するデータ項目は、呼ぶプログラムからも呼ばれるプログラムからも参照できる。

ロックモード (lock mode)

レコードのロックが有効なときの、開かれているファイル結合子の状態。レコードのロックが手動か自動かを示す。

論理演算子 (logical operator)

予約語の AND、OR、または NOT。条件を記述するとき、AND と OR は、論理連結子として使用できる。NOT は、論理否定子として使用できる。

論理ページ (logical page)

最上端、ページ本体、最下端からなる概念的な要素。

論理レコード (logical record)

最も包括的なデータ項目。レコード用のレベル番号には、01 を当てる。

(英数字)

COBOL の語 (COBOL word)

語 (word) の項を参照。

CRT の状態 (CRT status)

概念的な要素で、その値が、ACCEPT 画面文の実行時に、端末の入出力操作の状態を示すよう設定されている。

DBCS

2バイト文字セット (double-byte character set) の項を参照。

INSTANCE-OF 条件 (instance-of condition)

オブジェクト参照が特定のクラスまたはインターフェイスのインスタンスであるという命題。これに対し、真理値を決定することができる。

IOCS

入出力制御システム (input-output control system) の項を参照。

OPEN 文のモード (open mode)

あるファイルに対して OPEN 文を実行してから、REEL 指定も UNIT 指定も伴わない CLOSE 文を実行するまでの間の、ファイルの状態。どのモードを使用するかは、OPEN 文中に指定する。INPUT (入力) , OUTPUT (出力) , I-O (入出力両用) , EXTEND (拡張) がある。

2 バイト文字セット (double-byte character set)

各文字につき、2 バイトが使用される文字セット。