Micro Focus COBOL™

言語リファレンス

第 26 版
2006 年 5 月

このドキュメントでは、このシステムによりサポートされる基本 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 処理系により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は産業界の言語であり、いかなる企業または組織、およびそれらの集合の占有物でもない。

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ついては、いかなる協力者またはCODASYL プログラミング言語協会も、なんら保証するものではない。また、同様に、いかなる協力者またはCODASYL プログラミング言語協会も、これに関する責任を負わない。

ここで使用された資料の作者および版権の所有者は以下のとおり。

> 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

上記の資料は、一部または全部を問わず、COBOL仕様においてその使用を承認している。これにより、プログラミング資料や同様の出版物におけるCOBOL仕様の引用や使用も承認されている。

このドキュメントは、すでにCOBOL言語に精通しているプログラマーを対象としている。

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。

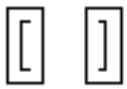
3. 日本語で記してある語（例:ファイル名）。プログラマが指定すべき名前を表わす。

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。

    IBM OS/VS COBOL中の ANS X3.23-1974 に対する拡張機能を表わす。

    IBM VS COBOL II中の ANS X3.23-1985 に対する拡張機能を表わす。

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。

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。

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。

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。

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

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。

15. 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。

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
-  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。

 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。

正書法

原始(ソース)文は

 自由形式か

固定形式で記述する。

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

 COBOL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。

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。

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。

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)(OPEN)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。

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

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。

COBOL言語の概要　翻訳集団の概念

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。

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
'	(SO2002)(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)(OPEN)アンパサンド
_	(SO2002)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。)

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。

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。

   ISO2002 OSVS 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。

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。

COBOLの語

(ANS85)COBOL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）
- (ISO2002)(MF)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）
- (ANS85) 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
- (ANS85) 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
- (ISO2002)関数プロトタイプ名
- 指標名
- (ISO2002)インターフェイス名
- 登録集名
- (ISO2002)(MF)メソッド名
- 呼び名

- 段落名
- (SO2002)パラメータ名
- プログラム名
- (SO2002)(MF) プログラムプロトタイプ名
- 報告書名
- (MF)手順名
- (SO2002)(MF)(OPEN)画面名
- 節名
- (ANS85)記号文字
- 原文名
- (SO2002)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、
   (SO2002)(MF)クラス名、
   (SO2002)関数プロトタイプ名、
   (SO2002)インターフェイス名、
   (SO2002)(MF)メソッド名、
   (SO2002)プログラムプロトタイプ名、
   (SO2002)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. (ISO1002)(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. (ISO2002)(MF)SELF
2. (ISO2002)(MF)SUPER
3. (MF)SELFCLASS

(ISO2002) 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。

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
- (SO2002)インターフェイス名
- (SO2002)(MF) メソッド名
- プログラム名
- (SO2002)(MF) プログラムプロトタイプ名
- レコード名
- 報告書名
- (SO2002)(MF)(XOPEN) 画面名
- (MF)型定義名
- (SO2002)利用者関数名

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)(SO2002)(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)(SQ000)(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。

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。

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。

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

(ISO10) (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。

メソッド名の適用規則

(ISO2002)(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。

関数プロトタイプ名の適用規則

(ISO2002)(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。

プログラムプロトタイプ名の適用規則

(ISO2002)(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。

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。

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

ISO2002 OSVS 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

ISO2002 OSVS 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。

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。

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。

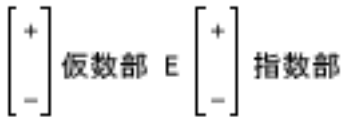
標準データ形式の文字で表した数字定数の大きさは、利用者が指定した数字の桁数に等しいものとする。

(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。

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。

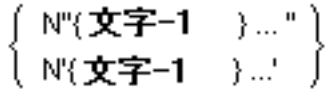
各国語型定数

(ISO2002) (MF)

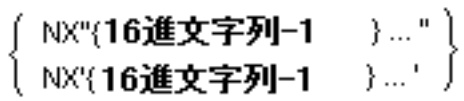
各国語型定数は、計算機内の保存場所で同一サイズの文字で表示される、一連の各国語文字である。詳しくは、使用しているCOBOLシステムの各国語データ（Unicode）に関する文書を参照。

一般形式

形式 **1**



形式 **2**



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。

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 (SO2002)(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。 (SO2002)(MF)(OPEN) 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)(OBJ)または手続きポインタ値 を表わす。USAGE POINTER (MF)(OBJ)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)(OBJ)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. (SO2002)(MF)(OPEN)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)(SC2)(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。

(OPEN)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。

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。

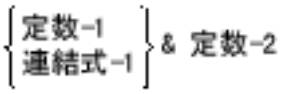
定数名は、定義した後でだけ使用できる。定数名は前方参照の対象とはならない。

連結式

(ISO2002)(MF)(OPEN)

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



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.

   (SOSID) 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。

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。
(SVS) 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-RETURN	S9(4) COMP	SORT手続きを異常終了させるために使用できる。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。

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。

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。

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。

形式と規則

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。

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。

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。

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。

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。

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。

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。

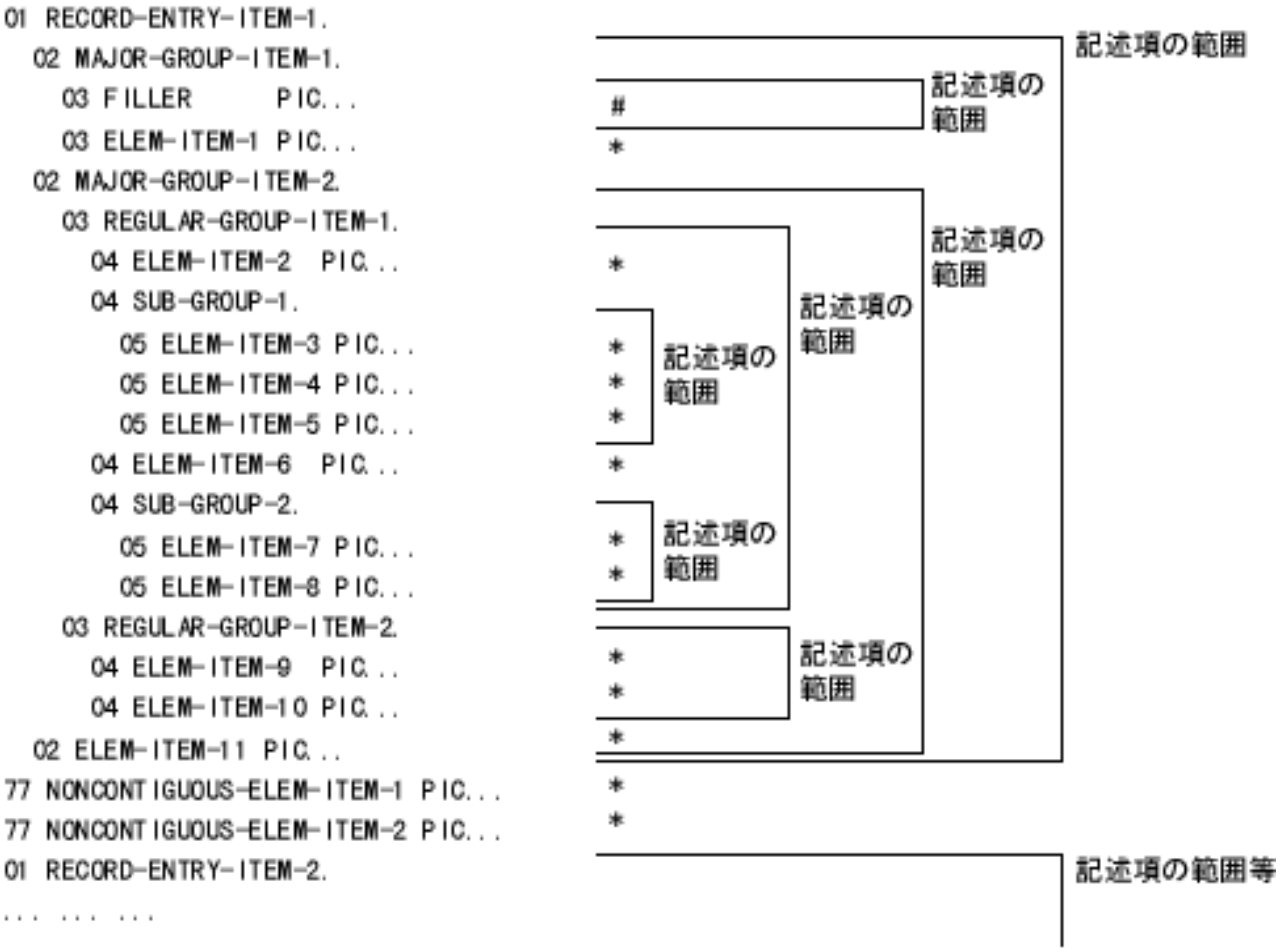


図 2-1: 階層構造をとるレベル番号の例

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。

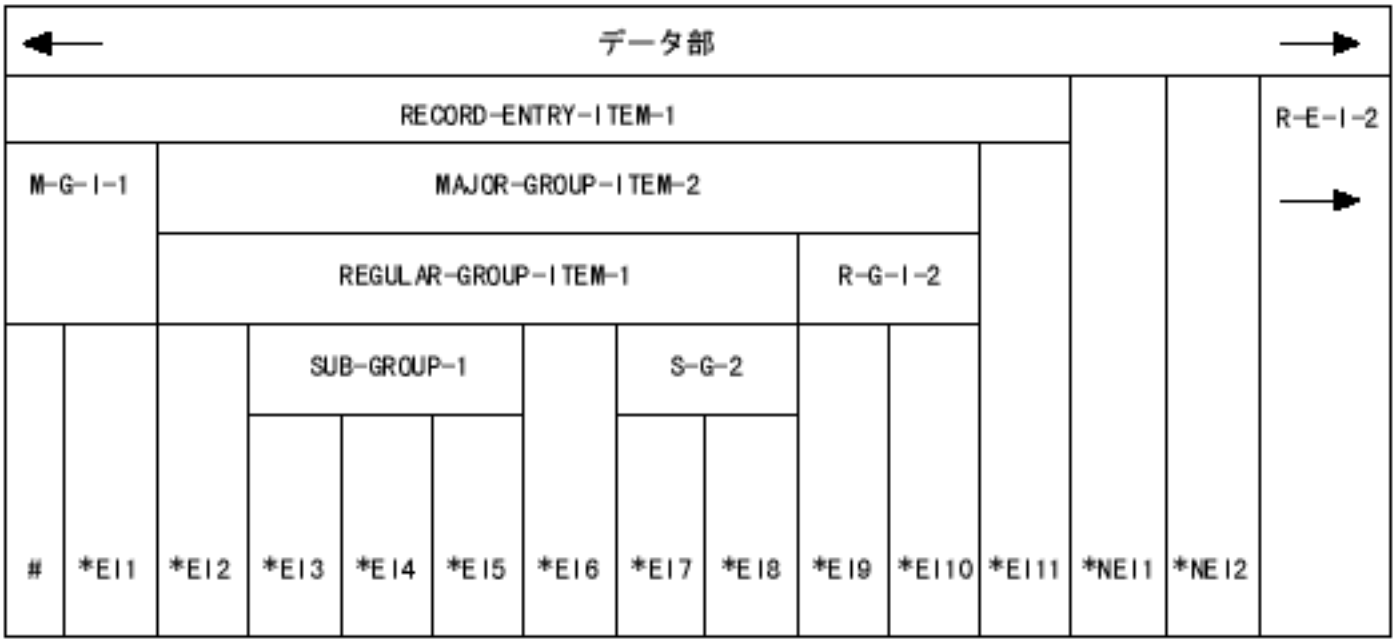


図 2-2: データ・レコード記憶域の割り当て

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)

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 (VSC2)(MF)2バイト文字
指標	指標
各国語	各国語」
数字	数字 (SVS)(VSC2)(MF)内部浮動小数点数 (SVS)(VSC2)(MF)外部浮動小数点数
オブジェクト	(SOJ00)(MF)オブジェクト参照
ポインタ	(VSC2)(MF)(SOJ00)ポインタ (MF)(OBJ370) プログラムポインタ

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。

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.  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 - テータ記述の章のJUSTIFIED(桁寄せ)句節の記述に従って、桁寄せ規則が修正される。

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。

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, ISO1002 FLOAT-SHORTまたはFLOAT-LONG
- VSC2 MF POINTER
- MF OBJ7.0 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。

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。

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).
```

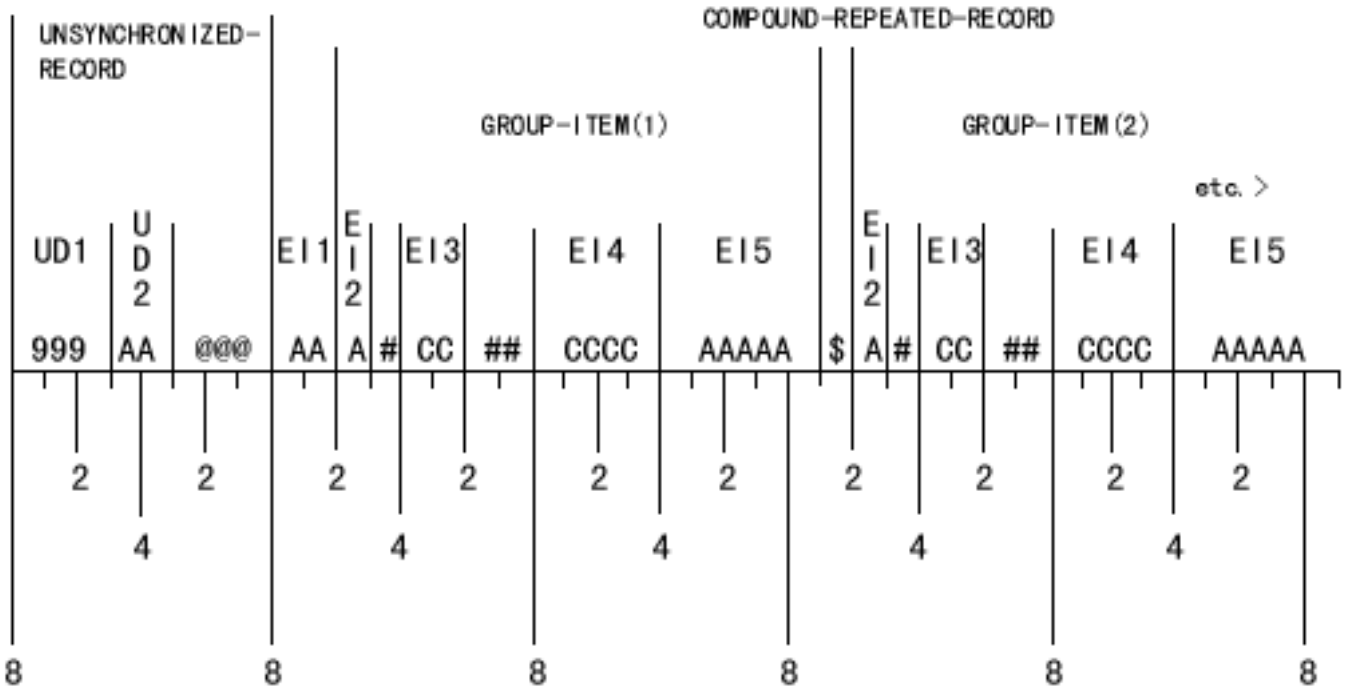


図 2-3: 計算機の記憶域の割当ての例

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。

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。

 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。

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

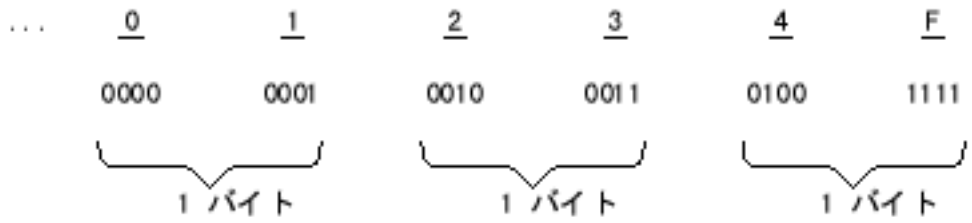
PICTURE句内の符号の表現	データ項目の値の符号	**16**進法による符号用半バイト
符号なし	n/a	x"0F"
符号付き	+	x"0C"
符号付き	-	x"0D"

>表 **2-8**：**COMP(UTATIONAL)-3**
ANS85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

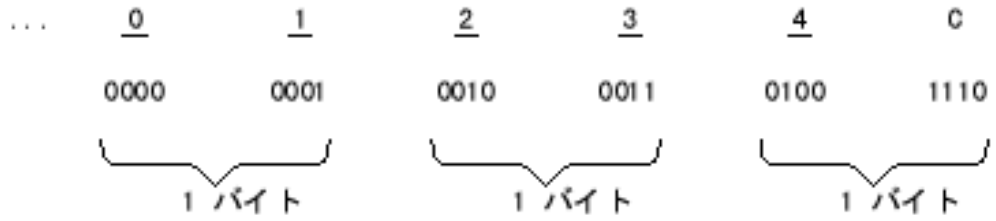
例**:**

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。



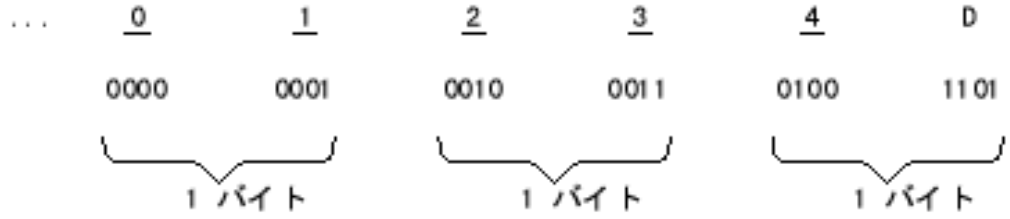
Fは正号を表わす（印刷不能文字）。

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。

MF **COMPUTATIONAL-X**および OS/390 MF OPEN **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。

ポインタ形式

VSC2X 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。

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。

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。

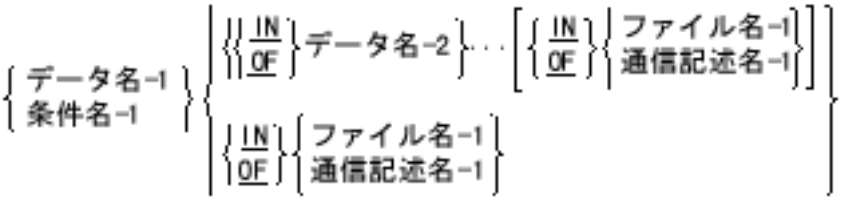
修飾の階層系列では、レベル指示語の後ろに書いた名前を最高位とし、次にレベル番号01に書いた名前、それに続いてレベル番号02以降49の後ろに書いた名前の順とする。段落名については、節名だけを最高位の修飾語として用いることができる。したがって、階層系列の最高位の名前は、一意でなければならず、修飾することはできない。添字または指標の付いたデータ名や条件変数も、修飾によって一意にすることができる。条件変数の名前は、その条件名の修飾に使用できる。修飾の有無にかかわらず、同じつづりの名前をデータ名と手続き名の両方に使用してはならない。

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。

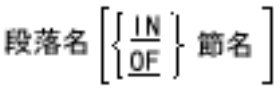
(ANS85)関数を指定する場合関数定義に従って、その関数を参照する記述の中に、いくつかのパラメータに対する値または値の組を指定する必要がある。そのパラメータの値を用いて、参照した関数の値が算出される。パラメータに実際の値を指定することを引数を与えるという。詳細については、この節の関数一意名節で解説する。

修飾の一般書式は、以下の通り。

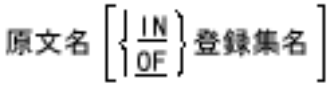
形式 **1**



形式 **2**



形式 **3**



形式 **4**

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。

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。

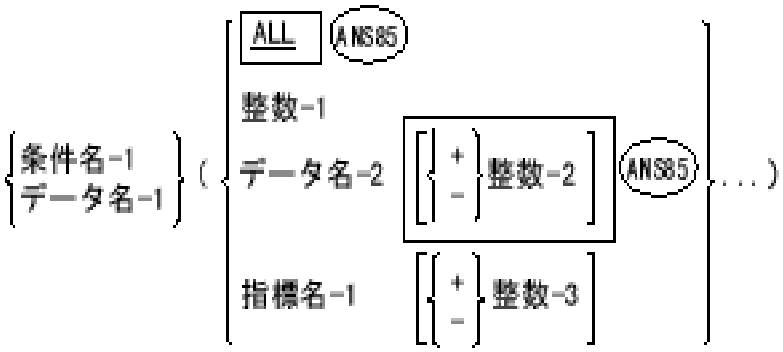
添字に使用するデータ名には、符号が付いていてもよい。この場合、正でなければならない。添字の最小値は1である。この値は表の最初の要素を指す。表の要素の2番目以降を指す添字の値は、2、3、...という具合になる。添字が取る最大値は、OCCURS句で指定した項目の反復回数の最大値である。

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。

(ANS85)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. (ANS85) 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。

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)(VSC2)(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。

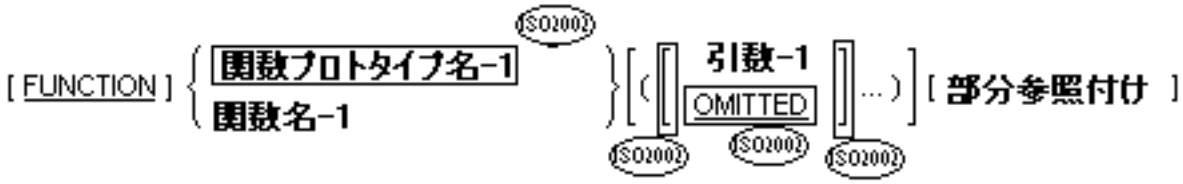
指標付けの一般形式は、添字付け節に記した一般形式に含まれている。

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。

   (SO2002)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. (SO2002)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. (SO2002)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。

   (SO2002)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。

   (2002)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。

   (2002)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. (2002)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. (2002)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。

(2002)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。

(2002)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。

(2002)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。

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、

    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）。

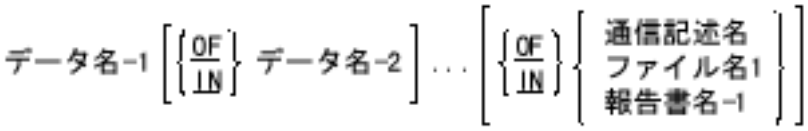
一意名

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
```

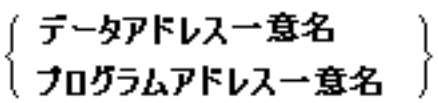
ISO2002 MF形式 **4**

```
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。

(2002)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

(SO2002) (MF)

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

(SO2002) (MF)

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

(SO2002) (MF)

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。

.

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。

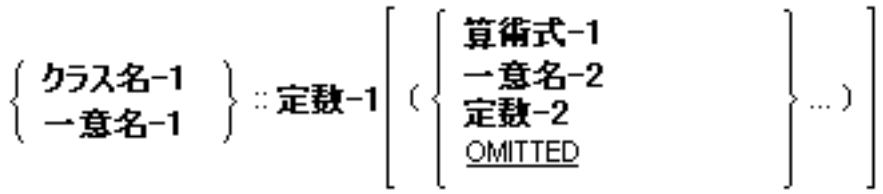
一般形式において条件名というときは、必要に応じて修飾、指標付け、添字付けしたものを含む。

行内メソッド呼出し

(SO2002) 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



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。

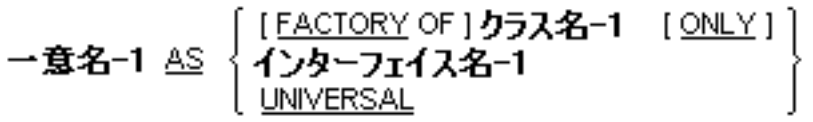
c. 一時一意名は、ここで解説する行内呼出しの機能のためにのみ存在する一時項目で、その他の目的にはいっさい使用されない。

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



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。

データ番地一意名

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。

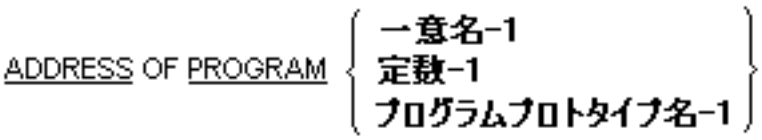
3. この一意名形式を、受取り側作用対象として指定してはならない。

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。

プログラム番地一意名

(ISO2002) (MF) 一般形式



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。

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

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、

ANS85REPLACE文、

OSVSVSC2INC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。

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。

```
USE AFTER ERROR PROCEDURE ON ファイル名.
IF 状態キー-1 >= 3
  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、

   GOBACK(復帰)文、

   EXIT METHOD(メソッド出口)文、

   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

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。

 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。

明示範囲符と暗示範囲符

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

 END-ACCEPT

END-CALL

 END-CHAIN

END-COMPUTE
END-DELETE

 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。

COBOL言語の概念　翻訳集団の定義の概要

第 3 章：翻訳集団の概念

指定しなくてもよい部、節、段落の見出し

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。

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。

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。

CALLプロトタイプ

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。

ファイル

ファイル結合子

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。

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。

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。

行順ファイル

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。行順ファイルの形式は、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形式と同じである。格納されるレコードの末尾の空白は削除される。

レコード順ファイルおよび  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。

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。

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。

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。

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。

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。

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。

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。

共有モード

(ISO2002) (MF)

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。

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。

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。.

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。

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、

MFSELFCLASS、

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。

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。

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。

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。

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。

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。

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。

6. 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。

7. 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。

- 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。

- 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。

-

対応するパラメータでは、OPTIONAL句の有無は同じである。.

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。

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。

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。

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。.

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。

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。.

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。

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。

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。

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。

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。

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。

.NETの概念

カスタム属性

カスタム属性を用いて、プログラム内の要素につき、追加情報を提供できる。 COBOLプログラムで用いられる構文の多くが、メソッドの可視性およびフィールド(PUBLIC、 PRIVATE、 PROTECTED等の識別子)といったプログラム要素を記述するものである。これらすべての記述は、「メタ・データ」としてコンパイラが生成したdll もしくはexeファイルの中に埋め込まれており、また、Reflectionを用いた他のプログラムによって問い合わせを行うことも可能である。カスタム属性は、 拡張可能でしかも無制限にプログラム要素に追加のメタ・データを付加する方法である。カスタム属性は、カスタム属性予約語を用いるCOBOL項目に添付されている。

カスタム属性を用いてメソッド、データ項目、プロパティ、イベント、 デリゲート、メソッドパラメータ、アセンブリ、クラス、メソッド戻り値を記述してもよい。

カスタム属性に対し指定できる引数には、以下の二つの型がある：

- 通常のパラメータ ― コンストラクターへ引数を渡して当該属性を得る
- 指定パラメータ ― カスタム属性クラスで定義されたプロパティに対応する

事例：

```
01 a binary-long custom-attribute is xmlattribute("UpperB").
```

カスタム属性の指定において用いられる通常のパラメータは、当該属性によって定義されたコンストラクターと一致しなければならない。上記の例では、コンストラクターは単一パラメータの型列(type string)を期待する。

もう一つの事例：

```
custom-attribute is webservice("Description"="My service")
```

指定パラメータは、当該属性によって定義されたプロパティと一致しなければならない。この事例においては、記述がカスタム属性クラスwebserviceに対し定義されたある特性を有する型列である。

デリゲート

デリゲート・システムは、オブジェクト指向の、手続きポインタと同等のものであり、型の安全なソリューションである。手続きもしくは関数のポインタの使用は、多くの言語において非常に一般的に発生しており、.Ｎｅｔにおいても、それらは、一つのソフトウェアのコンポーネントが別のコンポーネントに発生したイベントを通知できるようするためのメカニズムとして一般的に用いられている。

COBOLにおいてデリゲートを使うには、 宣言、インスタンス化、そして最後に呼び出しが必要である。これら三つのステップの概要を以下に示す：

デリゲートの宣言

デリゲートを宣言するには、メソッド名とその署名を宣言しなければならない。COBOLにおいてこれを行うには、当該デリゲートを表す単一のメソッドを用いてある特定化されたクラスを定義する。デリゲートの宣言は、クラスと非常によく似ているが、DELEGATE-IDキーワードを用いて行われる。

デリゲートのインスタンス化

既存のデリゲート、例えば、フレームワークにおいて定義されたSystem.EventHandler を用いるには、メソッド実行に対し当該デリゲートと同じ署名を提供しなければならない。その後、コンストラクターにパラメータとしてのメソッドを供給することによってデリゲート・クラスの新しいインスタンスを生成できる。このメソッドがスタティックなものかクラスのインスタンス・メンバーであるかによって、以下のごとく、コンストラクター内においてこのパラメータをどのように特定するかが決まる。

```
スタティック･メソッド:        クラス名::"メソッド名"
インスタンス･メソッド:     オブジェクト参照::"メソッド名"
```

デリゲートの呼び出し

DELEGATE-IDを宣言するとき、コンパイラは、System.MulticastDelegateクラスから受け継いだクラスを生成する。すべてのデリゲートには"Invoke"というメソッド名があるが、その署名は、DELEGATE-ID宣言にあるメソッドの署名と一致するものである。ゆえに、デリゲート・インスタンスに対しオブジェジェクト参照を行う場合、当該デリゲートが構築されたときに供給されたメソッドの実行を実際に呼び出す、このオブジェクトに関するINVOKEメソッドを呼び出すことができる。

DELEGATE-IDを用いた新しいデリゲートクラスを宣言するサンプル・プログラムを提供し、次にどのようにそのデリゲートを、スタティックもしくはインスタンス・メソッドの実行を用いて呼び出すことができるかを示す。ただし、たいていの場合、デリゲート型はクライアント／サーバーのような役割で用いられており、そこでは、一つのソフトウェアのコンポーネントがデリゲート定義を提供し、別のコンポーネントがその実行を、たとえば、フォーム上のボタンをクリックしたときに実行されるメソッドなどを提供する。

サンプル・プログラムを見るには、Studioを起動させ、Examples\Visual Studio Integration\LRM Samples\Delegates フォルダーにあるデリゲート・ソリューションを開く。

.NET ネイティブ型

COBOLは、最も一般的に用いられている.NET型の多くに対応する、定義済みUSAGEを提供する。これらの名前は、データ項目を宣言するときのUSAGE指定の中で、また、クラス名が期待されるいずれの場所においても用いることができる。

定義済み**COBOL USAGE**	**.NET**クラス	**C#** 等価物	代替**COBOL**
binary-char	System.SByte	sbyte	PIC S9(2) COMP-5
binary-char unsigned	System.Byte	byte	PIC 9(2) COMP-5
binary-short	System.Int16	short	PIC S9(4) COMP-5
binary-short unsigned	System.UInt16	ushort	PIC 9(4) COMP-5
binary-long	System.Int32	int	PIC S9(9) COMP-5
binary-long unsigned	System.UInt32	uint	PIC 9(9) COMP-5
binary-double	System.Int64	long	PIC S9(18) COMP-5
binary-double unsigned	System.UInt64	ulong	PIC 9(18) COMP-5
character	System.Char	char	
float-short	System.Single	float	USAGE COMP-1
float-long	System.Double	double	USAGE COMP-2
condition-value	System.Boolean	bool	
decimal	System.Decimal	decimal	
object	System.Object	object	USAGE OBJECT REFERENCE
string	System.String	string	

さらに、.NET型は、USAGE class-name-1 またはUSAGE OBJECT REFERENCE class-name-1という構文を用いてCOBOLで指定できる。 そのようなすべてのデータ項目ならびにUSAGE DECIMAL、OBJECTもしくはSTRINGのいずれも以下の要件を充たさなければならない：

- 01 (または77) レベルで宣言されなければならない
- RedefinesまたはRenamesを有してはならない
- 部分参照の対象となってはならない
- ADDRESS OF句の対象となってはならない

.NET型は、.NET配列を宣言するためにOCCURS句の形式3を利用してもよい。

さらに、上述のその他の定義済みUSAGEのいずれかを、対応する.NETネイティブ型にマッピングするには、以下の規則に従わなければならない。

これらの規則に従わない、もしくは、その他のカテゴリーに属するCOBOLのデータ項目はすべて、.NETネイティブ型とはみなされないものとし、コンパイラによって内部的に管理されたバイト配列が割り当てられる。COBOLポインタデータ項目は常にこれらのバイト配列(byte arrays)の一つを指示している。

.NETは、オブジェクトの二つの型を区別する：

- 値型：　(e.g. System.Int32、System.Decimal、System.DateTime)このタイプのデータについては、プログラム中で定義するデータ項目は実データを含んでいる。例えば、System.Int32 (BINARY-LONG in COBOL)については、データ項目は32ビットの整数値を含むことになろう。
- 参照型：　オブジェクト・ヒープに対して割り当てられる。このタイプのデータについては、プログラム内のフィールドがオブジェクト・ヒープの中を参照する。参照型を用いると、間接化技法により諸費用がかかり、そのヒープを片付ける必要が生じる(ガーベジコレクション)。

すべての値型は、ボックス化というプロセスによって参照型に変換できる。これは必要に応じて、例えば、値型がパラメータとして、System.Objectをパラメータとして期待するメソッドに渡されたとき、自動的に実行される。それは、値型をtype System.Object (つまり、総称オブジェクト)の一つの項目に与えることによって、明確に実行できる。 ボックス化された値型は、アンボックス化することによって元の値型を回復できる。

USAGE [OBJECT REFERNCE] クラス名を指定すると:

- クラス名が参照型であれば、参照型・オブジェクトが得られる。
- クラス名が値型であれば、値型・オブジェクトが得られる。

.NET配列は、形式 3のOCCURS句を使って定義できる。

実行単位における通信

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。


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。

(AN585)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)(SO1002)(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)(SO1002)(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)(SO1002)(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。

データの共有

(AN585)

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。

ファイル結合子の共有

(AN585)

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。

データ部

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. (ISO2002)(MF)局所記憶節
6. 連絡節
7. 報告書節
8. (ISO2002)(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。

(ISO2002)(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。

(ISO2002) (MF) (OPEN) 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。

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。

メソッド、オブジェクトまたはプログラムの状態

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。

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。

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。

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。

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。

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。 .

大域名と局所名

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。

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。

手続き部

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。

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

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 OPENON EXCEPTION句のあるACCEPT文またはDISPLAY文

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。

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。

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。

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。

2002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。

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

OSVS 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。

OSVS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。

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。

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。

文の分類

分類	動詞
算術	ADD COMPUTE DIVIDE INSPECT (TALLYING) MULTIPLY SUBTRACT OSVS EXAMINE (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) ANS85 EVALUATE OSVS ON
データ転記	ACCEPT (DATE, DAY or TIME) INSPECT (REPLACING) MOVE STRING UNSTRING OSVS EXAMINE 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 (ISO2002)(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 (ANS85) 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。

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。

正書法の表現

行の正書法は、図 3-1のとおり。

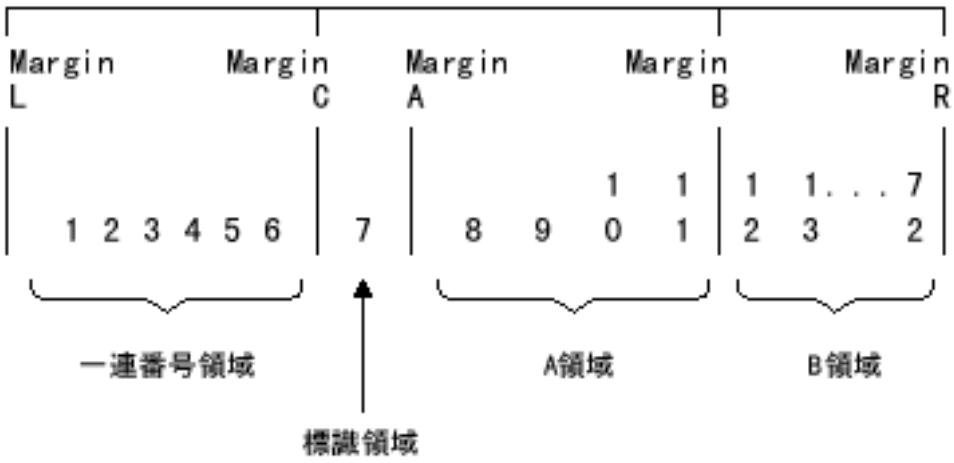


図 3-1：COBOLの原始行の正書法

-
- 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。

(SOJ00)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。

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)(VSC2)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。

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 COBOL 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。

ISO2002 >>は、2文字とも同じ行に書く。

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。

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）。

部、節、段落の正書法

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）。

ISO2002 MF B領域から書き始めてもよい。

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）。

ISO2002 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。

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。

ISO2002 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。

(SQ2002)(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。

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

(AN585)（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

(AN585)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。

SO2002 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。

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）。

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 VSC2 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。

行内注記

SO2002 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。

翻訳集団の概念　手続き部

第 4 章：翻訳集団の定義の概要

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。

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。

(ANS85)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。

(ANS85)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、

(ANS85)および終了見出し、

(ISO2002)(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。

(ISO2002)(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. (ANS85)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。

(ANS85)ソース単位の終了は、終了見出しによってのみ示される。

(ANS85) 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。

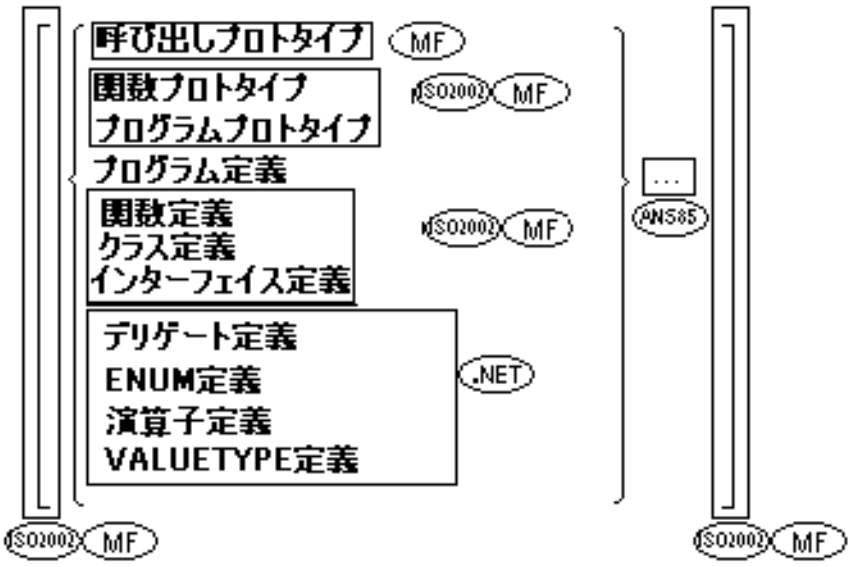
(ANS85)翻訳単位は他のソース単位に含まれていないソース単位である。翻訳集団内で翻訳単位の前または後ろに他の翻訳単位が来ることもある。

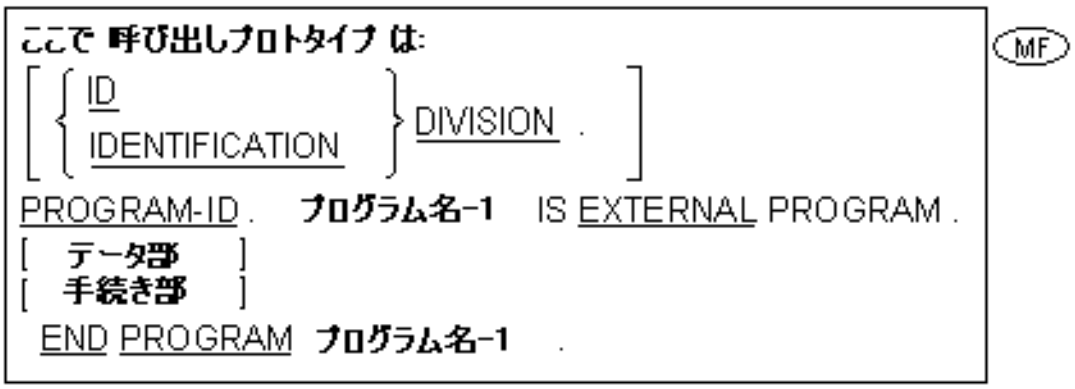
ソース要素はネストされているソース単位を除いたソース単位である。

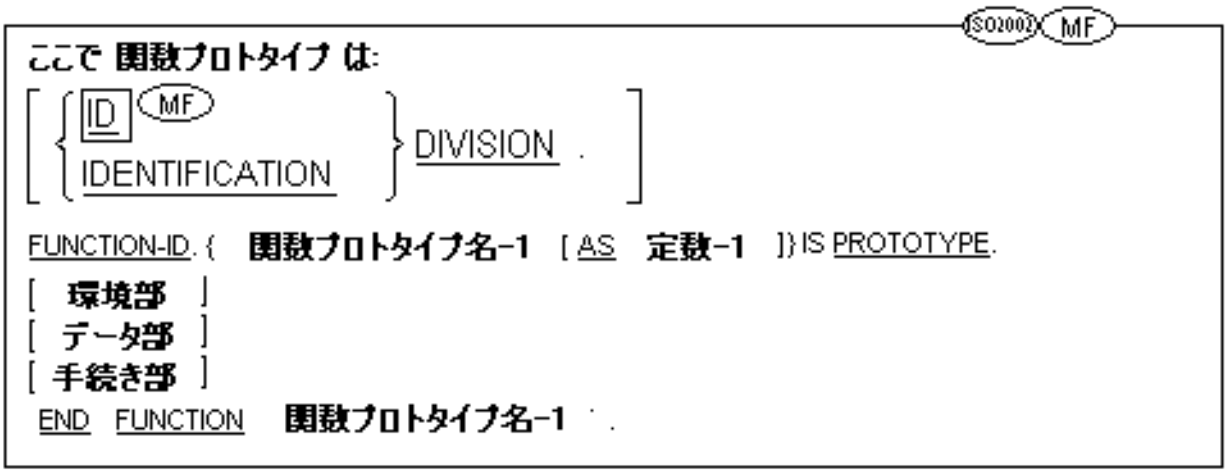
ランタイム要素はソース要素をコンパイルした結果である。

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

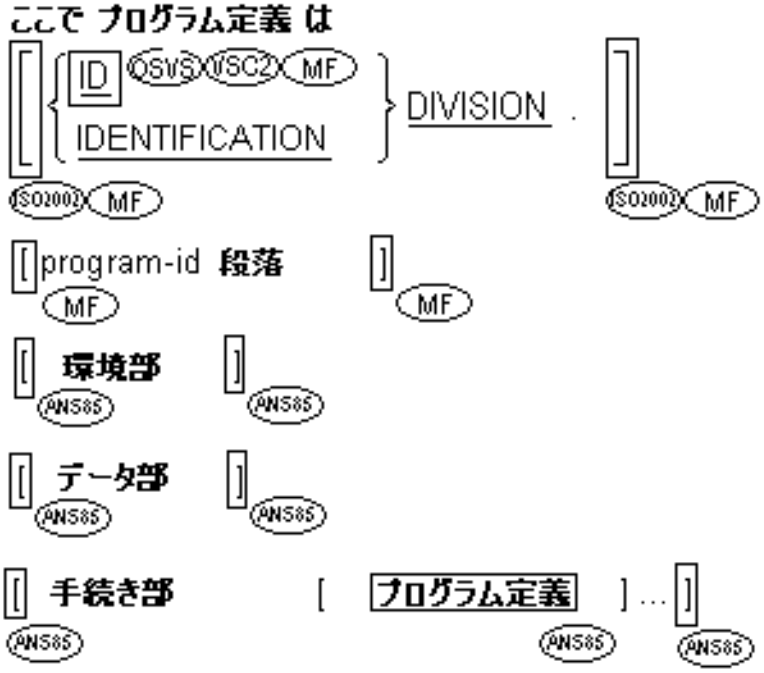



ここで 呼び出しプロトタイプ は:
MF
[{ ID / IDENTIFICATION } DIVISION .]
PROGRAM-ID . プログラム名-1 IS EXTERNAL PROGRAM .
[データ部]
[手続き部]
END PROGRAM プログラム名-1 .

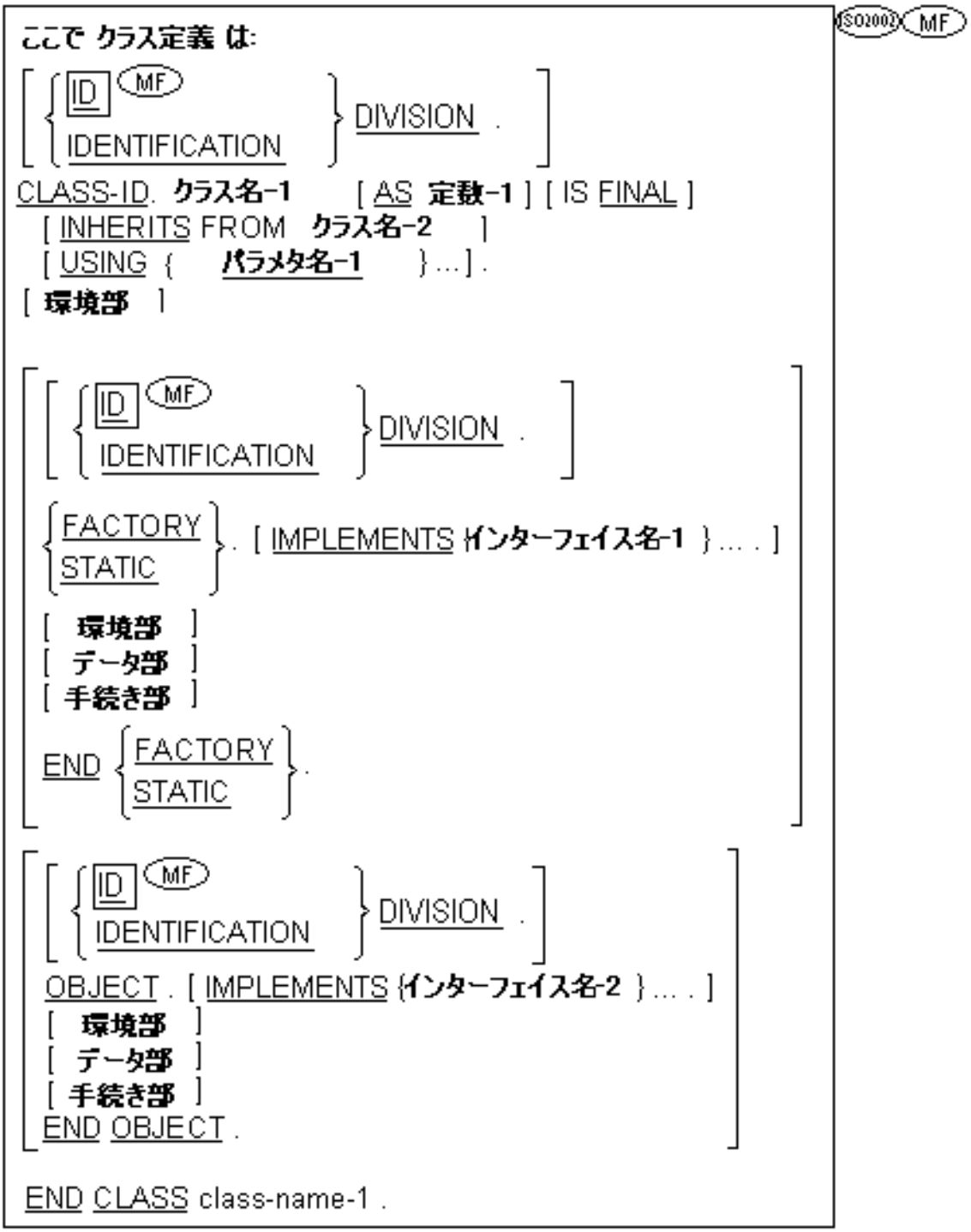

ISO2002 MF
ここで 関数プロトタイプ は:
[{ ID MF / IDENTIFICATION } DIVISION .]
FUNCTION-ID. { 関数プロトタイプ名-1 [AS 定数-1] } IS PROTOTYPE.
[環境部]
[データ部]
[手続き部]
END FUNCTION 関数プロトタイプ名-1 .


ISO2002 MF
ここで プログラムプロトタイプ は:
[{ ID MF / IDENTIFICATION } DIVISION .]
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 .

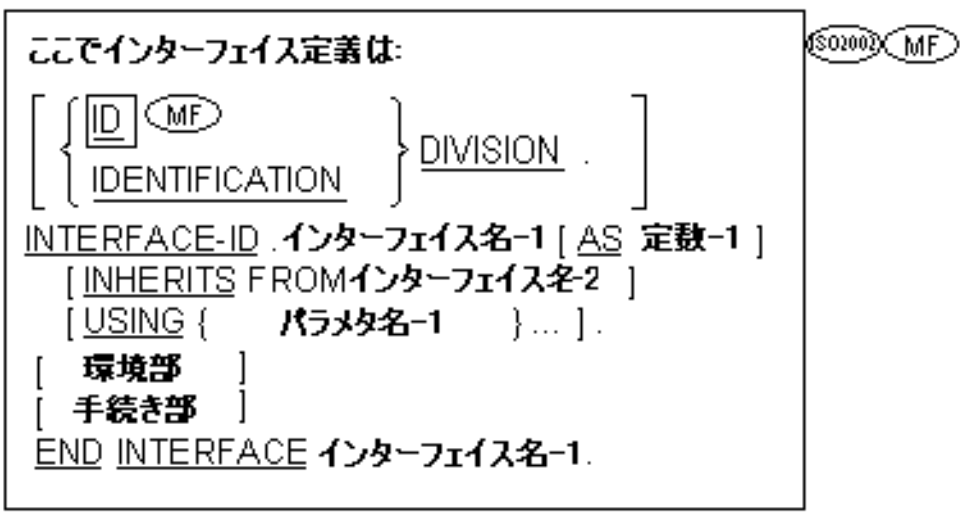
翻訳集団の定義の概要


ここで プログラム定義 は
ID
IDENTIFICATION
DIVISION .
program-id 段落
環境部
データ部
手続き部
プログラム定義
END PROGRAM プログラム名-1
ここで 関数定義 は:
FUNCTION-ID. 利用者関数名-1 [AS 定数-1] .
END FUNCTION 利用者関数名-1 .

ここで クラス定義 は:

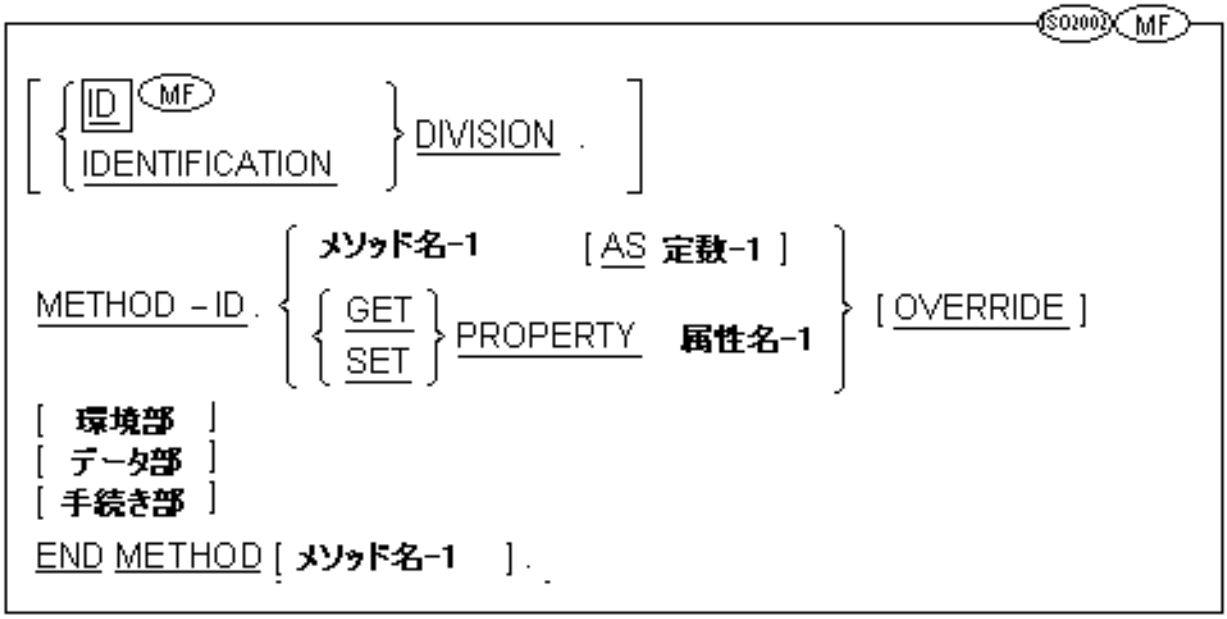


ここでインターフェイス定義は:

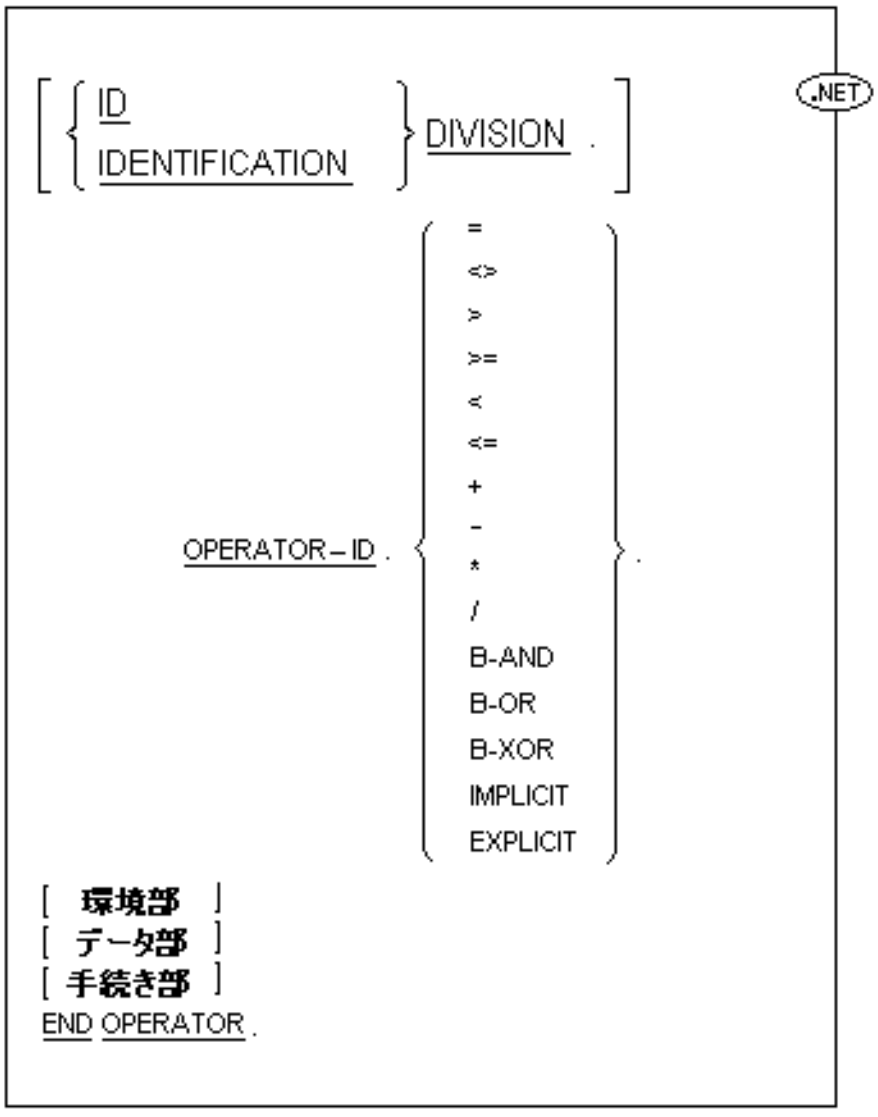


ここで メソッド定義 は:

書き方 1



書き方 2



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

   (ISO2002)(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. (ISO2002)(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. (ISO2002)(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. (ISO2002)(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。

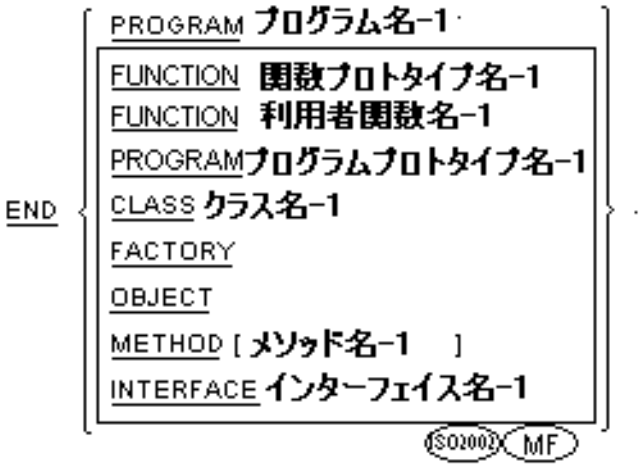
2. (ISO2002)(MF)関数プロトタイプおよびプログラムプロトタイプでは、データ部の節は、連絡節以外は無視される。また、手続き部は、PROCEDURE DIVISIONヘッダを除き、無視される。

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



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。

翻訳集団の定義の概要　手続き部 - 組み込み関数

第 5 章：手続き部

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。

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

ビット演算子

算術式に使用できるビット演算子（bitwise operator）には、4つの2項演算子（binary bitwise operator）と、1つの単項演算子（unary bitwise operator）がある。これらの演算子は特定の予約語で表わし、前後を空白で区切る。

2項ビット演算子	意味
B-AND	2つのデータ項目のビット単位の論理 AND をとる
B-OR	2つのデータ項目のビット単位の論理 OR とる
B-XOR	2つのデータ項目のビット単位の排他的 OR とる
B-EXOR	2つのデータ項目のビット単位の排他的 OR とる

単項ビット演算子	意味
B-NOT	データ項目のビット単位の論理 NOT をとる

B-XOR と B-EXOR は同義語である。

これらの演算子は、COMP-5, COMP-X のデータ項目または数字定数のみから構成される算術式の中でのみ使用することができる。 B-NOT はひとつの被演算子のみを受け付け、その他は２つの被演算子を受け付ける。 演算子の実行結果は、CBL_AND などの論理演算ライブラリルーチンを、被演算子データ項目のうち長い方のサイズをもつ一時領域に対して呼び出した結果と同等となる。 短い長さの被演算子は、事前に長い方のサイズを持つ一時領域に転記される。 被演算子のタイプが COMP-5 と COMP-X とで混在している場合、一時領域のタイプは長い方のタイプに揃えられる。このため実行時性能を考慮すればタイプの混在は避けるべきである。

以下に例題プログラムを示す:

```
data division.
working-storage section.
01 n1 pic 9(9).
01 n2 pic xx comp-5.
01 n3 pic xxxx comp-5.
procedure division.
    move 2 to n2
    move 4 to n3
    compute n1 = n2 b-or n3         *> n1 = 6
    if n2 b-and n3 = n2
        display 'true'
    end-if
    compute n1 = b-not n2           *> n1 = 65533
    compute n1 = n2 b-xor n3 + 1    *> n1 = 7
    compute n1 = n2 b-and 2         *> n1 = 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	* / ** + - B-AND B-OR B-XOR B-EXOR	単項 + - B-NOT	(	)
変数	-	P	-	-	P
* / ** + - B-AND B-OR B-XOR B-EXOR	P	-	P	P	-
単項 + - B-NOT	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

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-, B-NOT
   2番目 べき乗
   3番目 乗算と除算
   4番目 加算と減算
   5番目 B-AND
   6番目 B-XOR, B-EXOR
   7番目 B-OR

   以下に式の評価の例を示す。

   ```
   1 + 2 * 3
   ```

   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。

2. 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。

3. 算術式の先頭には、記号の"("と "+"と " -" と " B-NOT" 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。

4. 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。

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。

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。

単純条件

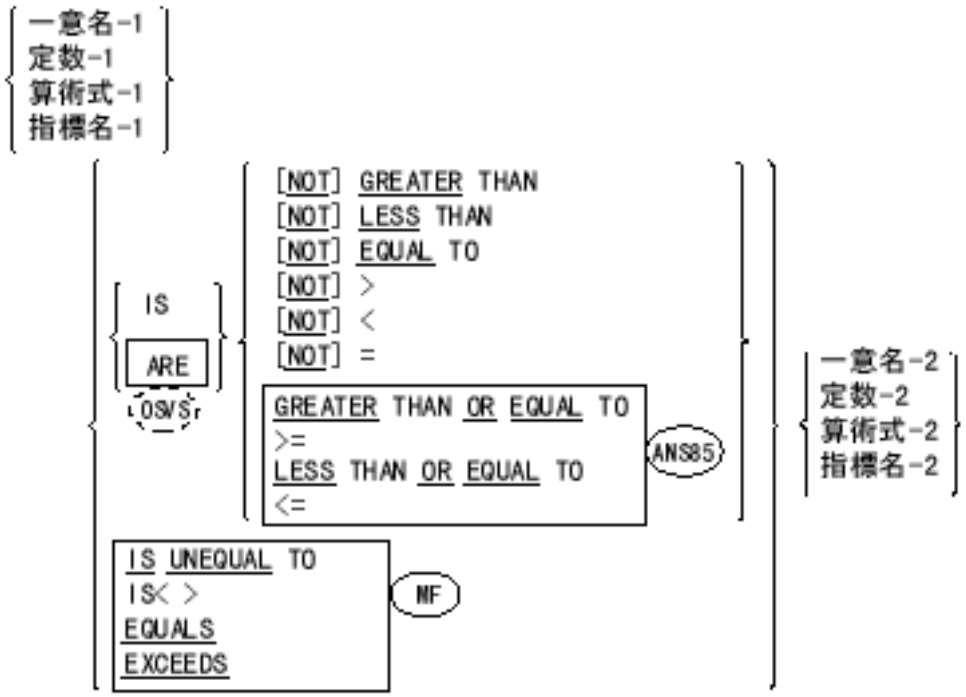
単純条件（simple condition）には、比較条件、字類条件、条件名条件、スイッチ状態条件、符号条件がある。単純条件は真理値として、「真」または「偽」のどちらかをとる。単純条件をかっこで囲んでも、真理値は変わらない。

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 XVSC2 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。

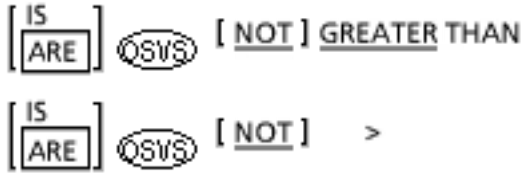
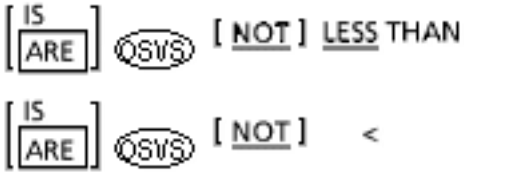
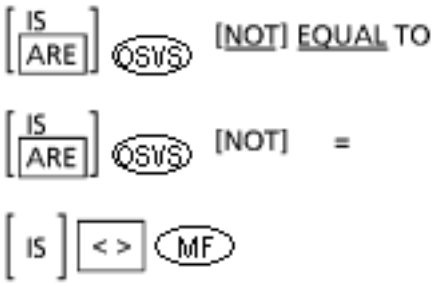
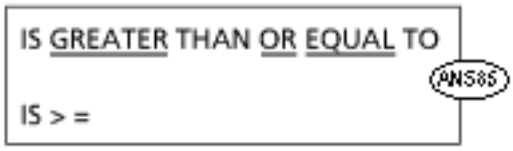
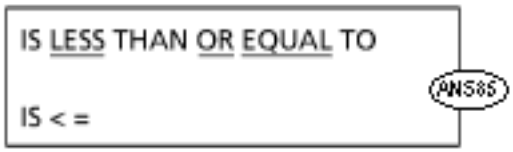
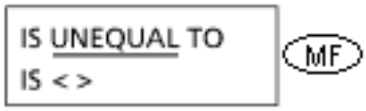
比較演算子（relational operator）は、比較条件の中で行う比較の種類を指定する。比較演算子としての予約語は、前後を空白で区切らなければならない。 必要語または比較文字の前に"NOT" を付けたものは、1つの比較演算子である。たとえば、"NOT EQUAL" は「等しくない」ことを「真」とすることを表わし、"NOT GREATER" は「以下である」ことを「真」とすることを表わす。各比較演算子の意味を表 10-2に示す。

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」	
「より小さい」または「以上」	
「等しい」または「等しくない」	
「より大きい」または「等しい」	
「より小さい」または「等しい」	
「等しい」	
「等しくない」	
「より大きい」	
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。	

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。

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。

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。

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。

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。

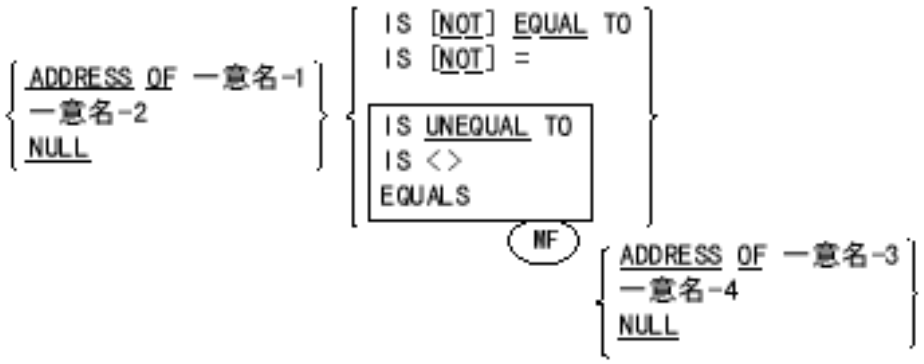
1. 2つの指標名。結果は、指標名に対応する出現番号を比較したのと等しい。
2. 指標名と数字データ項目または数字定数。指標名の値に対応する出現番号が、数字データ項目または数字定数と比較される。
3. 指標データ名と指標名または他の指標データ項目。実際の値が、変換されることなしに比較される。
4. (OSVS)(VSC2)指標名と算術式。指標名の値に対応する出現番号が、式を計算した値と比較される。
5. 指標データ項目を上記以外のデータ項目または定数と比較すると、結果はどうなるかわからない。

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。

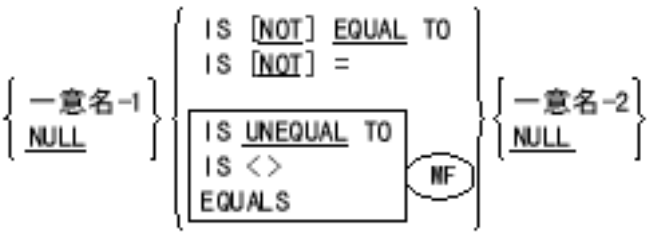
2. この種の比較条件は、IF、PERFORM、EVALUATE、およびSEARCH(形式 1)の各文の中で指定できる。形式 2のSEARCH文（SEARCH ALL）の中では、この種の比較条件は使用できない。ポインタ・データ項目は、意味のある順序付けをすることができないからである。

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 COBOL

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

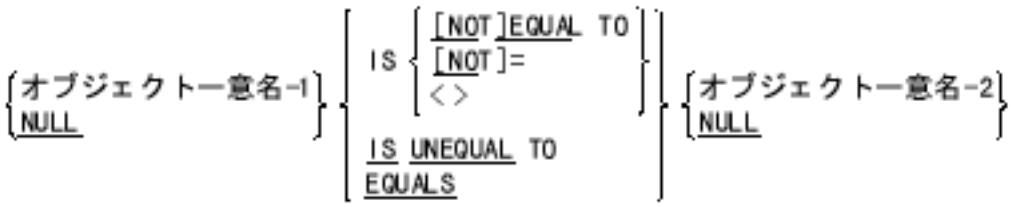
1. 2つの番地が等しいときに、作用対象は等しいとみなされる。それ以外の場合は、作用対象は等しくないとみなされる。

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。

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



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。

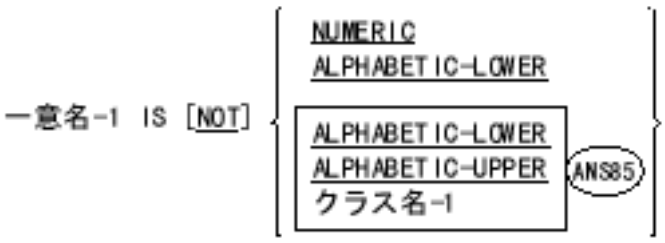
字類条件

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



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、

OSVSISC2MFXOPENCOMPUTATIONAL-3、

MFXOPENCOMPUTATIONAL-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。

   OSVS MFNUMERIC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。

   OSVS ISC2 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. ANS85ALPHABETIC-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. ANS85ALPHABETIC-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。

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。

スイッチ状態条件

スイッチ状態条件（switch-status condition）は、

MF9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。

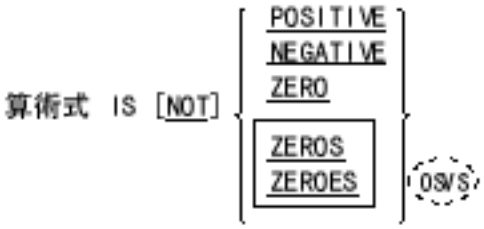
コンパイラ指令 SWITCH-TYPE を初期値の 1に設定すると、スイッチを設定するランタイム要素の内部からCOBOLスイッチが設定されているかどうかをテストする以外に、スイッチ状態条件を使用することはできない。

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。

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。

省略引数条件

ISO2002 MF

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。

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。

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。

否定単純条件

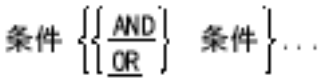
否定単純条件（negated simple condition）は、単純条件に論理演算子の"NOT" を付けたものである。否定単純条件は、単純条件の真理値を逆転させる働きをする。したがって、単純条件の真理値が「偽」のときに否定単純条件の真理値は「真」となり、単純条件の真理値が「真」のときに否定単純条件の真理値は「偽」となる。否定単純条件は、単純条件の真理値を逆転させる働きをする。したがって、単純条件の真理値が「偽」のときに否定単純条件の真理値は「真」となり、単純条件の真理値が「真」のときに否定単純条件の真理値は「偽」となる。否定単純条件をかっこで囲んでも真理値は変わらない。

一般形式

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



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,)

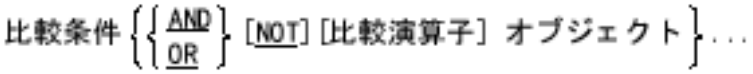
表 からわかるとおり、"OR NOT" は許される組合せであるが、"NOT OR" は許されない組合せである。また、"NOT (" は許されるが、 "NOT NOT " は許されない）。

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



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. ANS85NOT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)
x > a OR (y AND z)	x > a OR (x > y AND x > z)
x > (a OR y) AND z	(x > a OR x > y) AND x > z
x (= a OR > b)	x = a OR x > b
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)

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. 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。

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。

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。

(OSVS)(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。

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。

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。

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。

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。

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。

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。

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。

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。

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。

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。

ファイル入出力の概要

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。

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。

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"  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。

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"  （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"  （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"  （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"  （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。

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、

"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)									

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。

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。

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。

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

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

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。

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

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。

(SO2002X) (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。

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。

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。

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。

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. 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。

b. 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。

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。

    a. 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。

    b. 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。

    c. 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。

    d. 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。

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。

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。

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、

    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 6 章 : 手続き部 - 組み込み関数

ANS85

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。)

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。

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。)

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。

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。

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、

 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、

       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。

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。

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

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
(ISO2002)(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 (MF)Nat1 ... または Num1 ...	引数による*	引数の最小値
MOD	Int1, Int2	Int	Int1と Int2のモジュロー（法）
(ISO2002)(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 (MF)または Nat1	引数による*	引数中のすべての文字を大文字にする
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。

手続き部 - 組み込み関数　　文字集合と文字の照合順序

第 7 章：Micro Focus OO COBOL 言語拡張

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。

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。

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。

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。

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、すべて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。

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。

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。

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。

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:

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。

Micro Focus OO COBOL言語拡張　ANSIファイル状態コード

付録 A 章：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

Micro Focus OO COBOL言語拡張 ANSIファイル状態コード

文字集合と文字の照合順序　予約語

付録 B 章：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。

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

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。

ANSIファイル状態コード

付録 C 章：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。

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	…	…	…	…

予約語

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

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