Net Express

言語リファレンス

目　　次

第1章：COBOL言語の紹介

第2章：COBOL言語の概念

第3章：翻訳集団の概念

第4章：翻訳集団の定義の概要

第5章：見出し部

第6章：環境部

第7章：データ部

第8章: データ部 - データ記述

第9章: データ部 - 画面節

第10章：手続き部

第11章：手続き部 - 組み込み関数

第12章: 手続き部 - ACCEPT - DIVIDE

第13章: 手続き部 - ENTER - INVOKE

第14章: 手続き部 - MERGE - OPEN

第15章: 手続き部 - PERFORM - ROLLBACK

第16章: 手続き部 - SEARCH - WRITE

第17章: オブジェクトCOBOL言語拡張

第18章: 翻訳指示文

付録A: COBOL標準機能用の一般形式の一覧

第1章：COBOL言語の紹介

1.1 COBOL言語

COBOL（COmmon Business Oriented Language）は、事務処理および管理分野のデータ処理に 最も広く使用されているプログラミング言語である。
Micro Focus COBOLは、 ***ANSI X3.23-1974 (ISO 1989-1978)***、 ***ANSI X3.23-1985 (IS 1989:1985)***、***ANSI X3.23a-1989 (IS 1989:1985/AM1)***、及び***ANSI X3.23b-1993 (IS 1989:1985/AM2)*** に規定されている ANSI及びISO COBOLを含んだ機能を持っている。
主な拡張機能には、下記のものがある。

- 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（すべてのリリース）、IBM DOS/VS COBOL、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
- Micro Focus COBOL製品に固有の拡張機能

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。さらに、コーディングしたCOBOL原始プログラム中の各種の拡張機能がどの言語バージョンのものかを見つけ出して"フラグ "を付ける、任意選択機能が用意されている。この機能を使用すると、指定した言語仕様に合った拡 張機能が使用されているかどうかを、確認できる。
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
- (ANS85)組込み関数（intrinsic function）

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

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

- 画面操作機能単位

  画面節と追加のACCEPT文およびDISPLAY文から構成される。この機能を利用することによって、画面上で項目の位置を正確に指定し、指定した位置から入力されたデータを受け取り、指定した位置に定数字句を表示し、画面属性を定義し、操作卓特性を制御できる。
- 拡張ファイル入出力機能

  原文ファイルを効率よく処理するための追加のファイル編成、原始プログラム内で（データ変数または定数）またはオペレーティングシステムの環境変数から実行時にファイル名を定義する任意選択機能、多数の利用者がファイルを共有する機能、など。
- 言語間の呼出しおよび再帰的COBOLプログラムに関するシステム・プログラミング機能の拡張。

1.2 正書法

原始(ソース)文は
(MF)(XOPEN)自由形式か
固定形式で記述する。

1.2.1 固定形式

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

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

(MF)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。

1.2.2 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文字の大小順序で空白文字よりも小さい文字）を書くと、その行は注記行として扱われ、リスト用のファイルまたは装置には出力されない。この機能を利用して出力したリスト・ファイルを入力用として使用して、コンパイルを行うことができる。
(MF)この機能はMFCOMMENTコンパイラ指令に影響される。
(ANS85)一連番号には計算機の文字集合の中の任意の文字を記述できる。

1.2.3 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)注記行を、見出し部の見出しより前に置ける。
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」の章*を参照）。
(MF)標識領域に"$"を書くと、その行は特殊行となり、指令やコンパイル対象行の選択条件文を記述できる。

1.2.4 A領域とB領域

節名および段落名は A領域から書き始め、終止符（.）と続く空白で止める。 レベル指示語FD、SD、CD及びRDは、A領域から書き始め、 B領域
(ANS85)またはA領域
から対応する記述を書き始める。レベル番号01及び77は、A領域から書き始め、B領域
(ANS85)またはA領域
から対応するデータ記述を書き始める。レベル番号02から49、66及び88は、B領域から書き始める。
(MF)レベル番号78は、A領域またはB領域から書き始めてよい。
(MF)見出し部中の注記項以外には、A領域とB領域に関して規則をいっさい設けない。
(MF)(XOPEN)自由形式の原始コードが使う場合は、A領域とB領域の区別はない。自由形式の原始コードに関する詳細は、この章の次のセクションを参照。
1行のコーディング中に複数の文を記しても構わない。

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. micro focus ltd.
000040 environment division.
000050 configuration section.
000060 source-computer. mds-800.
000070 object-computer. mds-800.
000075 special-names. console is crt.
000080 input-output section.
000090 file-control.
000100  select stock-file assign "stock.it"
000110  organization indexed
000120  access dynamic
000130  record key stock-code.
000140 data division.
000150 file section.
000160 fd   stock-file record 32.
000170 01   stock-item.
000180  02  stock-code   pic x(4).
000190  02  product-desc     pic x(24).
000200  02  unit-size    pic 9(4).
000210 working-storage section.
000220 01   screen-headings.
000230  02  ask-code         pic x(21) value "stock code    <    >".
000240  02  filler pic x(59).
000250  02  ask-desc         pic x(16) value "description   <".
000260  02  si-desc      pic x(25) value "                 >".
000270  02  filler       pic x(39).
000280  02  ask-size         pic x(21) value "unit size <    >".
000290  01 enter-it redefines screen-headings.
000300  02      filler           pic x(12).
000310  02  crt-stock-code pic x(4).
000320  02      filler           pic x(80).
000330  02  crt-prod-desc    pic x(24).
000340  02      filler           pic x(51).
000350  02  crt-unit-size    pic 9(4).
000360  02  filler       pic x.
000370 procedure division.
000380 sr1.
000390  display space.
000400  open i-o stock-file.
000410  display screen-headings.
000420 normal-input.
000430  move space to enter-it.
000440  display enter-it.
000450 correct-error.
000460  accept enter-it.
000470  if crt-stock-code = space go to end-it.
000480  if crt-unit-size not numeric go to correct-error.
000490  move crt-prod-desc to product-desc.
000500  move crt-unit-size to unit-size.
000510  move crt-stock-code to stock-code.
000520  write stock-item invalid key go to correct-error.
000530  go to normal-input. 000540 end-it.
000550  close stock-file.
000560  display space.
000570  display "end of program".
000580  stop run.
<---->|<-><------------------------------------------------------------>
   |   | |     |
   |   | |     +-- カラム 12-72  B領域
   |   | +-- カラム 8-11  A領域
   |   +-- カラム 7 標識領域
   +-- カラム 1-6  一連番号
```

図1-1　正書法を示すプログラム・コーディング例

1.2.5 自由形式

MF XOPEN

自由形式は、SOURCEFORMAT"FREE"指令によって選択される。
始めの6文字は一般行の一部で、COBOL原始文に含まれるものとして扱われる。1カラム目は、 以下のような標識の役目をする。

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

継続行はない。英数字定数の継続は、"A" & "B"のように連結して行う。
A領域とB領域の区別はない。
固定の右マージンはないが、実際上は、行の長さは1バイト文字では、最大250文字、また 2バイト文字では、最大125文字までの制限がある。
注記は段落見出しとして同じ行内におさめ、次の行に続けられない。

第2章：COBOL言語の概念

2.1 文字集合

COBOL言語の最も基本的でそれ以上分割できない単位は、文字である。COBOLの文字列および分離符として使用できる文字集合は、英字、数字、特殊文字から構成される。文字集合を構成する文字を、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
'	(ISO2000)(OSVS)(VSC2)(MF)アポストロフィ
(	左かっこ
)	右かっこ
>	より大きい記号
<	より小さい記号
:	(ANS85)コロン
&	(MF)(XOPEN)アンパサンド
_	(ISO2000)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*文字集合と文字の大小順序*を参照。)

2.2 言語の構造

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

2.2.1 分離符

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

1. 句読文字の空白は、分離符となる。空白を分離符または分離符の一部として使用する場合、1つ以上の空白を続けてもよい。分離符のコンマ、セミコロン、終止符の直後に続く空白はすべて、その分離符の一部であるとみなされ、独立の分離符とはみなされない。
2. 句読文字のコンマまたはセミコロンの直後に空白を続けたものは分離符となる。この分離符は、分離符の空白を使用できる箇所であれば、どこでも使用できる。ただし、PICTURE文字列の中に使用されているコンマは分離符ではない。
3. 句読文字の終止符の直後に空白を続けたものは、分離符となる。この分離符は、完結文の末尾を示すため、または形式に示された場合にだけ使用する。
4. 句読文字の左かっこおよび右かっこは、分離符とする。 左かっこと右かっこは一対として、添字、
   (ANS85)関数の引数リスト、部分参照、
   算術式、条件を囲むときだけ使用する。ただし、仮原文の中で使用するときは、例外である。
5. (MF)句読文字の引用符、'G"'、'H"'、'N"'、'X"'は、
   分離符となる。左側の引用符の直前は、空白か左かっこか仮原文区切り記号とする。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または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. (ISO2000)(OSVS)(VSC2)(MF)句読文字のアポストロフィは分離符となる。この分離符は、 翻訳集団全体を通して引用符の代りに使用できる。
   (MF)同一の原始要素内で引用符とアポストロフィの両方を使用できる。ただし、それぞれ対で使用しなければならない。
7. 句読文字の仮原文区切り記号は、分離符とする。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、終止符のどれかとする。
   (MF)左側の仮原文区切り記号の直前の空白は、省いても差し支えない。
   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*および *COBOL言語拡張*の章を参照。）

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、下記の場合は例外とする。
    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、終止符によってだけ区切られる。
分離符の構成に関する規則は、文字定数、注記項、注記行の中の文字には適用しない。

2.2.2 文字列

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

2.2.2.1 COBOLの語

(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 (ISO2000)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 (ISO2000)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。 (ISO2000)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。

- 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 A 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下記の種類がある。

- 符号系名（alphabet-name）
- 通信記述名（cd-name）
- (ISO2000)(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）
- 指標名（index-name）
- (ISO2000)(MF)インタフェース名(interface-name)
- レベル番号（level-number）
- 登録集名（library-name）
- (ISO2000)(MF)メソッド名(method-name)
- 呼び名（mnemonic-name）
- 段落名（paragraph-name）
- (ISO2000)パラメータ名(parameter-name)
- プログラム名（program-name）
- (ISO2000)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）
- (ISO2000)(MF)(XOPEN)画面名（screen-name）
- 節名（section-name）
- 区分番号（segment-number）
- (ISO2000)(MF)分割キー名（split-key-name）
- (ANS85)記号文字(symbolic-character)
- 原文名（text-name）
- (MF)型定義名(typedef-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
- 通信記述名
- (ISO2000)(MF)オブジェクト指向用の字類名
- (ANS85)真値提案用の字類名
- 条件名、 (MF)定数名、データ名、 (ISO2000)プロパティ名、レコード名、 (ISO2000)(MF)分割キー名、 (MF)型定義名
- ファイル名
- i指標名
- (ISO2000)インタフェース名
- 登録集名
- (ISO2000)(MF)メソッド名
- 呼び名
- 段落名
- (ISO2000)パラメータ名
- プログラム名
- 報告書名
- (MF)手順名
- (ISO2000)(MF)(XOPEN)画面名
- 節名
- (ANS85)記号文字
- 原文名

All 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すべて一意でなければならない。(*一意参照*の節を参照。)
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、 (ISO2000)(MF)クラス名、 (ISO2000)インタフェース名、および (ISO2000)(MF)メソッド名 に含まれていないプログラムのプログラム名
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、インタフェース名、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。

Ⓜ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、
Ⓜまたは1つのハイフンを入れること。

システム名には、下記の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。 予約語は、一般形式に指定されているとおりにしか使用できない。（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. (ISO2002)(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。特殊レジスタには、LINAGE-COUNTER (*データ部 - データ記述*の章を参照)、DEBUG-ITEM (***言語リファレンス - 追加トピック***の*デバッグ*を参照)などがある。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. (ISO2000)(MF)SELF 2. (ISO2000)(MF)SUPER 3. (MF)SELFCLASS

(ISO2000)**コンテキストセンシティブ語:** コンテキストセンシティブ語は、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。コンテキストセンシティブ語および予約されたコンテキストについては、付録の*コンテキストセンシティブ語*を参照。

2.2.3 名前の適用範囲

(ANS85)

ある原始要素の中に別の原始要素が、直接的または間接的に、含まれることがある。各原始要素は、他の原始要素によってまったく同じに名前を付けられた利用者語を使用することができる。（前述の*利用者語*の節を参照。）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、それらが通信節中に宣言されている場合、その通信節を含むプログラム中の文または記述項においてだけ、参照できる。<

- 通信名
- 条件名
- データ名
- レコード名

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。

- (SO2000)(MF)字類名(オブジェクト指向)
- 条件名
- データ名
- ファイル名
- 指標名
- (SO2000)インタフェース名
- (SO2000)(MF)メソッド名
- プログラム名
- レコード名
- 報告書名
- (SO2000)(MF)(XOPEN)画面名
- (MF)型定義名

2.2.3.1 プログラム名に関する表記法

(ANS85)

プログラム名は、見出し部のプログラム名段落に宣言する。プログラム名を参照できるのは、CALL（呼ぶ）文、 (MF)CHAIN（連鎖）文、CANCEL（取消し）文、 (MF)SET(設定)文、プログラム終了見出しからだけである。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。
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。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、共通属性をもつプログラムは参照できない。
3. 他のプログラムに含まれないプログラムのプログラム名は、実行単位中の他のどのプログラム中の文からも参照できる。ただし、このプログラムに直接的または間接的に含まれるプログラムからは例外とする。

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

(ANS85)

原始要素の中に条件名、データ名、ファイル名、レコード名、報告書名、 (MF)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)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、前述の*利用者語*の節を参照。

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

1. 原始要素A中に定義されているすべての大域名、および原始要素A及び原始要素B中に定義されているすべての名前が、参照された名前がどちらの原始要素に属するものかを判定するために使用される。そして、通常の修飾規則および参照の一意性に関するその他の規則が適用されて、該当するものがいくつか見つけ出される。
2. 該当するものが1つしか見つけ出されなかった場合、それが参照されたものである。
3. 該当するものが2つ以上見つけ出された場合、原始要素B用の名前は2つ以上はあり得ない。原始要素B用の名前をもつものがないかまたは1つある場合、下記の規則を適用する。
    a. 参照された名前が原始要素Bの中で宣言されているならば、原始要素B中のものが参照されたとする。

b. これ以外の場合、原始要素Aが他の原始要素に含まれているならば、下記のように判定する。
    1. 参照された名前が原始要素Aの中で宣言されているならば、原始要素A中のものが参照されたとする。
    2. 参照された名前が原始要素Aの中ではなく、原始要素Aを含む原始要素中で宣言されているならば、原始要素Aを含む原始要素中のものが参照されたとする。更に上位の原始要素があるならば、単一の有効な名前が見つかるまで、この規則を順次上に適用する。

2.2.3.3 指標名に関する適用規則

(ANS85)
大域的属性をもつデータ項目が指標名を記述した表を含む場合、その指標名も大域的属性をもつことになる。したがって、指標名の範囲は、その指標名によって名付けられる指標を持つ表に名前を付けるデータ名のものと同じであり、データ名用の名前の範囲規則が適用される。
指標名を修飾することはできない。
(OSVS)指標名を修飾することができる。

2.2.3.4 (ISO2000)(MF)クラス名（オブジェクト指向の場合） および (ISO2000)インタフェース名の適用規則

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 (ISO2000)リポジトリ段落または (MF)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フェースのインタフェース名は、それを含むインタフェー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。
該当のインタフェース名に関して翻訳集団内で定義できるインタフェース定義は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フェース名は、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。

2.2.3.5 メソッド名の適用規則

(ISO2000)(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文とメソッド終了見出しだけである。

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

2.2.4 定数

定数（literal）には下記のものがある。

- 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

定数には文字定数と数字定数がある。

2.2.4.1 文字定数

文字定数（nonnumeric literal）とは、計算機の文字集合中で利用できる任意の文字を並べて、両端を引用符
(ISO2000)(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
(ISO2000)(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。
実行時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句*の節を参照。)
(VSC2)(MF)さらに、16進数を文字定数として扱うことができる。このためには、X"*nn*" という形式で文字定数を書き表わす。ここで、*n* は0-9とA-Fの16進文字を表わす。*nn* は160回まで繰り返すことができる。ただし、nの個数は偶数とする。
(MF)16進桁の個数は奇数でもよい。

2.2.4.2 数字定数

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

2.2.4.2.1 固定小数点数の数字定数

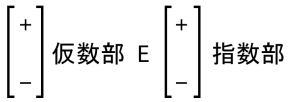
数字定数は、"0"から"9"までの数字と正号、負号、小数点からなる文字列である。この処理系では、数字定数の長さは1文字以上18文字以内である。数字定数を作る際には、下記の規則が適用される。

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

上記の規則に適合するが引用符で囲まれた定数は、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。

2.2.4.3 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	文字の大小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	文字の大小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 定数	指定された文字が何文字か含まれる文字列を表わす。定数は文字定数か表意定数のどちらか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。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)(COB370)または手続きポインタ値 を表わす。USAGE POINTER (MF)(COB370)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)(COB370)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、以下の規則が適用される。

1. 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。
2. 「ALL 定数」以外の表意定数を他のデータ項目と関係付けなかった（たとえば、DISPLAY（表示）、STRING（連結）、STOP（停止）、UNSTRING（分解）文の中で表意定数を使用しなかった）場合、文字列の長さは1文字となる。
3. 表意定数「ALL 定数」を他のデータ項目と関係付けなかった場合、文字列の長さは定数の長さとなる。
   (MF)DISPLAY文の書き方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。

書き方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大小順序によって決まる。(*環境部*の章の*実行用計算機段落*および*特殊名段落*の節を参照。)
表意定数を表わす予約語は、それぞれが独立した文字列である。ただし、「ALL 定数」は例外で、2つの別々の文字列から構成される。
(OSVS)(VSC2)(MF)表意定数の the QUOTE/QUOTES の値は、指令の APOST および QUOTE の影響を受ける。
(ANS85)定数の長さが2桁以上ある表意定数の「ALL 定数」を数字項目または数字編集項目に関係付けることは、ANSI '85標準では廃要素に分類される。これはANSI標準の次の全面改訂時に削除される予定である。
(MF)このCOBOL処理系に組み込まれている方言は、この構文を全面的に使用できる。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。
(XOPEN)標準COBOL定義の一環であるにもかかわらず、この構文はX/OpenのCOBOL言語定義からは明示的に除外されている。したがって、X/Openに準拠するCOBOL原始プログラムの中では、この構文を使用すべきではない。

2.2.4.4 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。書き方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。書き方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。定数名には前方参照が効かない。

2.2.4.5 連結式

(MF) (XOPEN)

連結式（concaternation expression）は、連結演算子で区切られた2つの作用対象をつないだものである。

一般形式

```
{定数-1  }
{連結式-1}& 定数-2
```

構文規則

1. 両方の作用対象は同じ字類とする。ただし、表意定数は１つまたは両方の作用対象を構成してもよい。作用対象は数字にはできない。定数-1も定数-2も、語ALLで始まる表意定数にはできない。

構文規則

1. 連結演算の結果としての連結式の字類は、以下のようになるる。
    a. 作用対象の１つが表意定数の場合、他の作用対象を構成する連結式または定数の字類
    b. 作用対象の両方が表意定数の場合、字類は英数字である
    c. 作用対象と同じ字類
2. 連結式の値は、それを構成する定数、表意定数、連結式の値を連結したものである。
3. 連結式は、同じ字類および値をもつ定数とまったく等しい。その字類の定数を使えるところには、どこにでも連結式を使用できる。

2.2.4.6 特殊レジスタ

(OSVS)(VSC2)(MF)

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、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。前述の*COBOLの語*の節を参照。

表 2-2 ： 特殊レジスタ

特殊レジスタ名または式	暗黙のデータ記述 PICTURE	使用法
(VSC2)(MF)ADDRESS OF データ名-1	USAGE IS POINTER	データ名-1の番地を示すポインタ値を生成する。この式は、使用する文で一般形式として明示的に指定する。データ名-1は、連絡節で01レベルおよび77レベルのデータ項目として 宣言される。 (MF)または、データ部の各所で各レベル番号で宣言する。
(OSVS)CURRENT-DATE[1]	X(8)	現在の日付が記録される。(この日付けは、COBOL 実行環境から提供される。) 次の形式をしている。 *MM/DD/YY* *MM* は月を、 *DD*は日を、*YY* は年（1900年からの下2桁）を表わす数字である。 CURRENT-DATEはMOVE文の送出し側としてだけ使用できる。
(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。

特殊レジスタ名または式	暗黙のデータ記述 PICTURE	使用法
(OSVS)(VSC2)(MF)RETURN-CODE[3]	S9(4) COMP (XOPEN) S9(9) COMP	次のことができる。 ・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実行時要素（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。

特殊レジスタ名または式	暗黙のデータ記述 PICTURE	使用法
(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。

注: 特殊レジスタによっては、コンパイラ指令および方言の影響を受けるものがある。

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。

2.2.4.7 定義済オブジェクト一意名

定義済オブジェクト一意名	用途
(ISO2000)(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。）
(ISO2000)(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。
(ISO2000)(MF)NULL	空のオブジェクト参照値を参照する。それはオブジェクトを絶対に参照しないことが保証された固有の値である。NULLは暗黙的にクラス・オブジェクトおよび項類オブジェクト参照として記述されており、一般的オブジェクト参照ではない。受取り側の作用対象にNULLを指定してはならない。

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

2.2.4.9 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、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)または行のA領域にある任意の文字が出たところ
で終了される。標識領域にハイフンを書いて注記項を続けることは許されない。

2.3 形式と規則

2.3.1 一般形式

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

2.3.2 構文規則

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

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

2.3.3 一般規則

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

2.3.4 要素

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

2.4 機種に左右されないデータ記述の概念

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

2.5 レベルの概念

レコードは、レベル構造の概念に従って構成される。この概念は、データを参照するためにレコードを細分化する必要があることから生ずる。いったん細分化したものをさらに細分して、もっと細かいデータとして参照できる。
レコードを最も基本的な単位に細分したもの、つまりそれ以上細分できないものを、基本項目（elementary item）という。したがって、レコードは一連の 基本項目から構成されるといえる。または、レコード自体が1つの基本項目である場合もある。
一組の基本項目を参照するために、それらをまとめて 集団にする。各集団はいくつかの基本項目を並べて名前を付けたものである。さらに、集団をいくつかまとめて集団とすることもできる。したがって、ある基本項目は何階層もの集団に属することがある。

2.5.1 レベル番号

レベル番号（level-number）は、基本項目と集団項目（group item）の構成を表わす。レコードはデータ項目のうちの最も包括的なものであるので、レコードのレベル番号は 01から始まる。レコードに含まれるデータ項目には、01より大きいレベル番号を付ける。ただし、レベル番号は連続している必要はなく、 49を超えないものとする。1レコードに含められるレベル数の最大は49である。この規則の例外である特殊なレベル番号として、66，77，
(MF)78，
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)この規則は強制しない。

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4種類ある。それらは下記のとおり。

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。

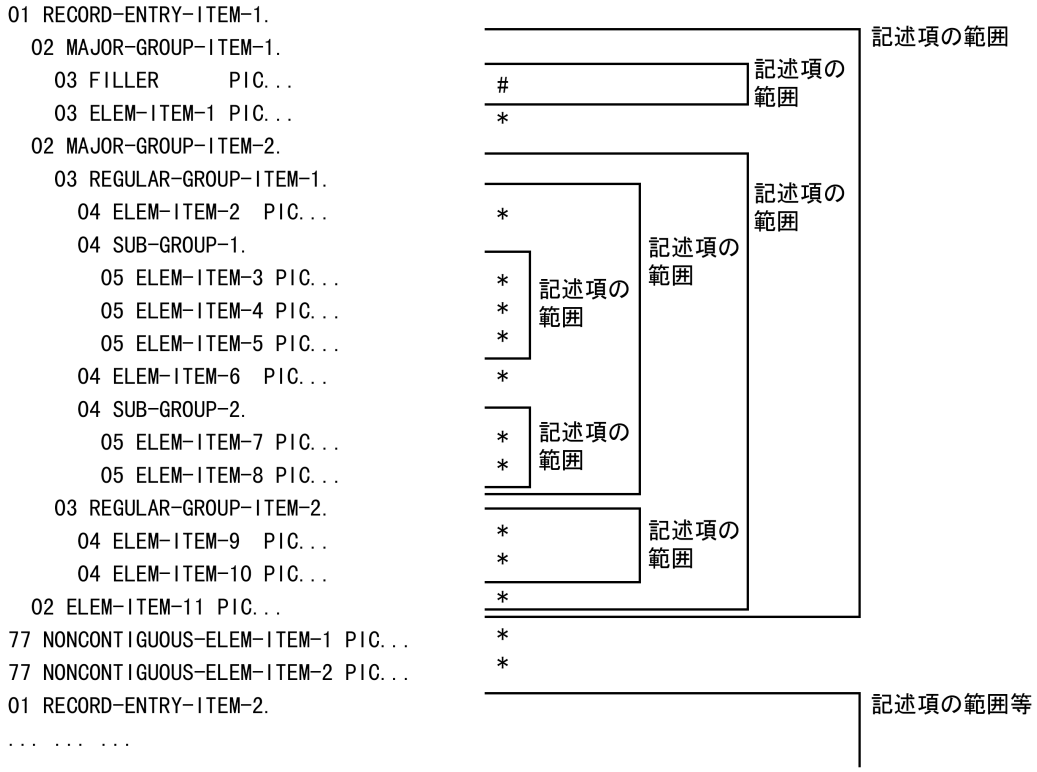


図 2-1: 階層構造をとるレベル番号の例

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句*および*USAGE 句*の節を参照。)
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句*の節を参照。)

では、分かりやすくするために、レベル番号を連続的に付けた。しかし実際には、レベル番号を連続的に付ける必要はない。したがって、01の次の下位レベルのレベル番号が05、さらにその下位レベルのレベル番号が10といった具合いにしてもよい。
図のデータ・レコード用に取られる記憶域は以下のように構成される。

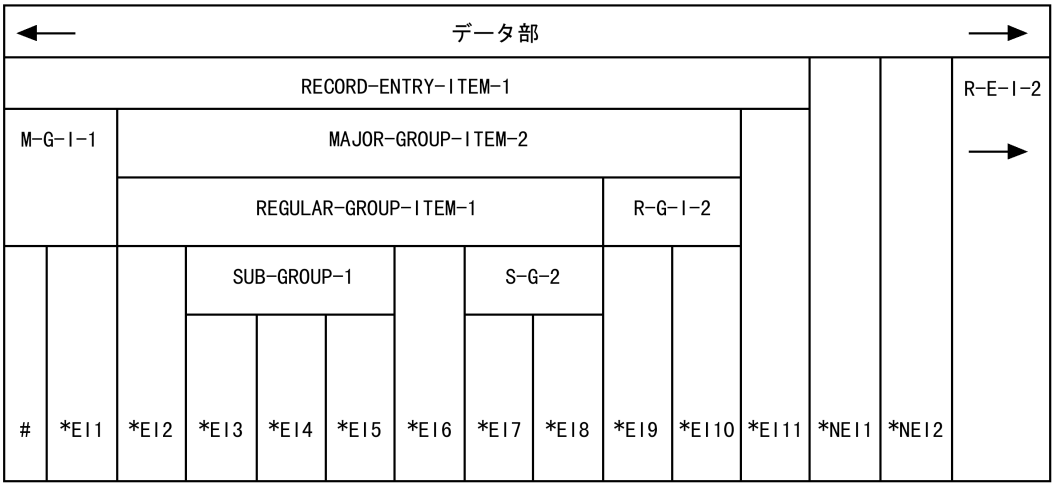


図 2-2: データ・レコード記憶域の割り当て

ここで:

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

2.6 データの字類と項類

基本データ項目、定数、関数は、それぞれ字類と項類を持つ。データ項目の字類と項類は、そのPICTURE文字列か、BLANK WHEN ZERO句か、用途によって定義する。 (ANS85)組込み関数の字類と項類は、組込み関数の定義によって記述する。(*手続き部 - 組込み関数*の章の*組込み関数*の節を参照。) 定数の字類と項類については、前述の*定数*の節を参照。
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	数字編集 英数字編集 英数字 (VSC2)(MF)２バイト文字
指標	指標
数字	数字 (OSVS)(VSC2)(MF)内部浮動小数点数 (OSVS)(VSC2)(MF)外部浮動小数点数
オブジェクト	(ISO2000)(MF)オブジェクト参照
ポインタ	(VSC2)(MF)(ISO2000)データ・ポインタ (MF)(COB370)プログラム・ポインタ

2.6.1 算術符号

算術符号（algebraic sign）は、演算符号（operational sign）と編集用符号（editing sign）に分類される。

1. 演算符号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。指定しなかった場合の 演算符号については、後述の*文字の表現と基数の選定*の節を参照。
編集用符号は、PICTURE句の符号編集用文字を用いて、データ項目に挿入する。

2.6.2 標準桁寄せ規則

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

1. 受取り側データ項目が数字である場合
    a. データは小数点の位置を合わせて、受取り側に収められる。このとき、必要に応じて、端にゼロが補われたり、端が切り捨てられたりする。
    b. 想定小数点（assumed decimal point）を明示的に指定していない場合、データ項目の右端に想定小数点があるものとみなされ、後は上記a.と同様に扱われる。
2. 受取り側データ項目が数字編集データ項目である場合、データは小数点の位置を合わせて、受取り側に収められる。このとき、必要に応じて、端にゼロが補われたり、端が切り捨てられたりする。ただし、編集操作によって先行ゼロ列（leading zeros）が置き換えられた場合は、ゼロは補われない。
3. 受取り側データ項目が英数字（英数字編集を除く）、英数字編集、英字のデータ項目である場合、送出し側データは左側を揃えて受取り側データに転記される。このとき、必要に応じて、右端に空白が補われたり、右端が切り捨てられたりする。
4. (OSVS)(VSC2)(MF)受取り側データ項目が外部浮動小数点数である場合、データは左端の文字位置をそろえられる。それに応じて、指数部が調整される。

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

2.6.3 実行用プログラムの効率を高めるための項目の桁詰め

ある種の計算機の記憶装置は、番地付けのための境界（語の境界、半語の境界、バイトの境界）をもつように構成されている。データの格納では、 これらの境界を意識する必要はない。
しかし、データのある種の使い方（算術演算や添字など）では、データが番地付けのための境界に合わせて記憶されている方が効率のよい場合がある。特に、複数個のデータ項目の部分が隣り合う2つの境界の間に含まれていたり、1つのデータ項目が境界によって分断されていたりすると、これらのデータを呼び出したり記憶したりするための、余分な機械命令が実行時要素の中で必要になる。
余分な機械命令を必要としないように、データ項目を境界に合わせて配置することを、桁詰め（synchronization）という。桁詰めされた項目はその形式に合わせて処理される。桁詰めされた形式への変換は、項目へデータを収める手続き（READとWRITEを除く）を実行するときにだけ行われる。
桁詰めを行う方法には、下記の2通りがある。

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

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

2.6.4 文字の表現と基数の選定

数字項目（PICTUREによって数字と定義されたもの。*データ部 - データ記述*の章の*PICTURE句 - 数字データの規則*の節を参照。）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句*の節を参照。）指定できる 数字の形式には下記のものがある。

- 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
- (MF)COMPUTATIONAL-5, COMP-5, COMPUTATIONAL-X または COMP-X
- (OSVS)(VSC2)(MF)COMPUTATIONAL-1, (OSVS)(VSC2)(MF)COMPUTATIONAL-2, (ISO2000)FLOAT-SHORT または (ISO2000)FLOAT-LONG
- (VSC2)(MF)POINTER
- (MF)(COB370)PROCEDURE-POINTER

(ANS85)英数字関数は常に標準データ形式で表示される。標準データ形式の英数字関数の大きさは、その関数の定義によって定まる。
整数関数および数字関数の表現形式は、作成者が下記のように指定する。

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

整数関数および数字関数は、算術式の中でだけ使用できる。整数関数および数字関数は関数を評価した結果の値を表わすが、その関数の作用対象の構成あるいは受取り側データ項目に制約はない。
ある計算機にデータ表現形式がいくつも備わっているときは、データ記述に指定しないと標準データ形式が使用される。
(ANS85)ただし、整数関数および数字関数は例外とする。

2.6.4.1 DISPLAY 形式

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

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

表2-4 : DISPLAY 形式のSEPARATEでない符号付き数字

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

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

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

これらの数字データ項目は、計算機の記憶域では、純粋な2進数の形で保持される。この形式では、数値は2を基数として表わされ、計算機に保持されている各ビットは右端を最下位の桁として位取りされ、各位の2のべき乗値の有無を表わす。 負の数は絶対値の等しい正の数の補数を取り（すべてのビットの値を逆転する）、その結果に1を加えることによって表される。データ項目を記憶するのに必要な文字数は、PICTURE句の中の"9" の数と符号が付けられているかいないか によって決まる。(*データ部 - データ記述*の章の*PICTURE句*、*COMPUTATIONAL句*、*SIGN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*の節を参照。) 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句*の節を参照)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).
```

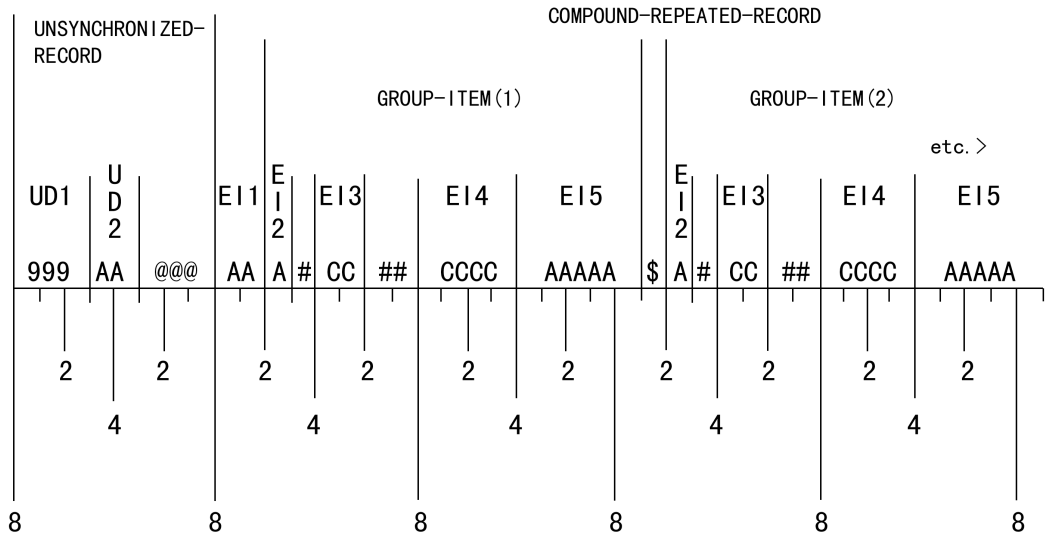


図 2-3: 計算機の記憶域の割当ての例

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。

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

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

操　　作	結　　果		
	TRUNC	NOTRUNC	TRUNC"ANSI"
MOVE 163 TO A	63	163	63
MOVE 263 TO A	63	7	63
MOVE 13 TO A, ADD 150 TO A	63	163	結果は保証されない
MOVE 13 TO A, ADD 250 TO A	63	7	結果は保証されない

注:

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

2.6.4.3 OSVS VSC2 MF COMPUTATIONAL-1, OSVS VSC2 MF COMPUTATIONAL-2, ISO2000 FLOAT-SHORT および ISO2000 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, ISO2000, 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。
(MF)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。
(MF)COBOLシステムでは、COMP-1項目は小数点以下7桁、COMP-2項目は小数点以下16桁になっている。しかし、メインフレームとの互換性により、DISPLAY文は、COMP-1には8桁、COMP-2には18桁の小数を表示する。浮動小数点項目を使用するどの操作もこの制限を考慮する必要がある。この制限を超える小数部については無視するようにすること。

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

(OSVS)(VSC2)(MF)(XOPEN)
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。

COMPUTATIONAL-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

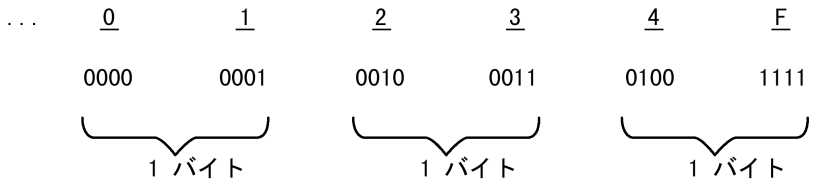
PICTURE句内の符号の表現	データ項目の値の符号	16進法による符号用半バイト
符号なし	該当せず	x"0F"
符号付き	+	x"0C"
符号付き		x"0D"

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

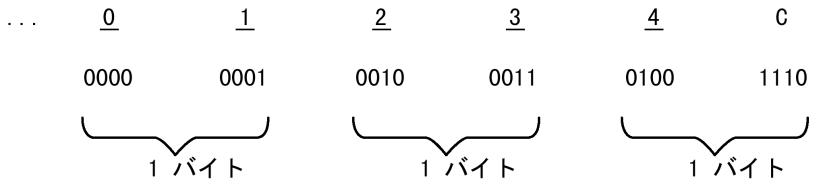
例:

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。



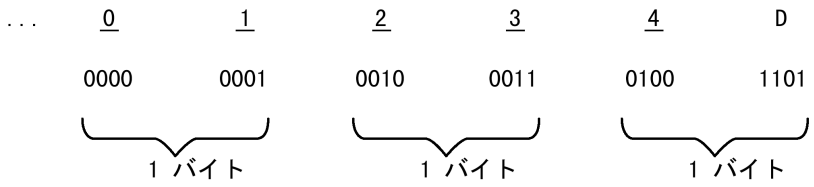
Fは正号を表わす（印刷不能文字）。

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。

2.6.4.5 COMPUTATIONAL-X (XOPEN)および COMPUTATIONAL-5

形式

(MF)

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*の節を参照。

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"の数は実行時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。
    - A COMPUTATIONAL-5データ項目には符号を付けることができるが、COMPUTATIONAL-Xデータ項目には符号を付けてはならない。COMP-5データ項目用のPICTURE句の中に"x" を指定すると、そのデータ項目は符号なしの扱いとなり、符号は付けられない。
    - COMPUTATIONAL-Xデータ項目は、常にBINARYデータ項目と同じ記憶方式に従って記憶される。つまり、最上位のバイトが最下位の番地に収められ、以降順次下位のバイトが上位の番地に収められる。
      COMPUTATIONAL-5データ項目を記憶する方式は、オペレーティングシステムによって異なる。オペレーティングシステムの中には、COMPUTATIONAL-5データ項目の最下位のバイトを最下位の番地に収め、以降順次上位のバイトを上位の番地に収めるものがある。たとえば、数字項目を逆の順序で記憶するオペレーティングシステムでは、
      h "12 34 56 78 9A"
      という内部値を持つ PIC X(5) COMPUTATIONAL-5項目は、
      9A 78 56 34 12
      というように記憶される。 一方、COMPUTATIONAL-X形のデータ項目（あるいは数字項目の記憶順序を逆転しないオペレーティングシステムのもとのCOMPUTATIONAL-5形のデータ項目）は次のように記憶される。
      12 34 56 78 9A
    - COMPUTATIONAL-5形式のデータ項目はIBMCOMPシステム指令とSYNCHRONIZED句の影響を受けるが、COMPUTATIONAL-X形式のデータ項目はその影響を受けない。

- 非算術文によって負の値がCOMPUTATIONAL-X形式のデータ項目または符号なしCOMPUTATIONAL-5項目に収められようとしたときは、絶対値が収められる。
  符号なしのCOMPUTATIONAL-5形式のデータ項目に負の値を収めようとする文の働きは、COMP-5コンパイラ指令 の影響を受ける。

2.6.4.6 ポインタ形式

VSC2 MF

ポインタ形式は利用可能なデータ項目のメモリー番地を表す値を保持する。データ項目が利用できなくなった（たとえば、データ項目が含まれているプログラムがキャンセルされた）場合には、ポインタ形式には形式が合致しない値が保持されているとみなされる。
ポインタ形式に割り当てられる省略時の記憶領域の量は操作環境によって異なりうる。しかし、少なくとも４バイトはある。メモリー番地を表現する方法は環境によって異なるが、一般的にはCOBOL以外の言語で使用されている表現と一貫性がある。
システム指令のIBMCOPMを設定した場合、SYNC指定を伴うポインタ・データの前に充てん文字が生成されることがある。それらの文字はデータ項目の一部ではなく、その項目に収められるデータに影響されることはけっしてない。

2.6.4.7 手続きポインタ形式

MF COB370

手続きポインタ形式は利用可能な手続きのメモリー番地を表す値を保持する。手続きが利用できなくなった（たとえば、手続きが含まれているプログラムがキャンセルされた）場合には、手続きポインタ形式には形式が合致しない値が保持されているとみなされる。
手続きポインタ形式に割り当てられる省略時の記憶領域の量は操作環境によって異なりうる。しかし、少なくとも４バイトはある。COBOL370指令を指定すると、８バイトが割り当てられる。メモリー番地を表現する方法は環境によって異なるが、一般的にはCOBOL以外の言語で使用されている表現と一貫性がある。
システム指令のIBMCOPMを設定した場合、SYNC指定を伴う手続きポインタ・データの前に充てん文字が生成されることがある。それらの文字はデータ項目の一部ではなく、その項目に収められるデータに影響されることはけっしてない。

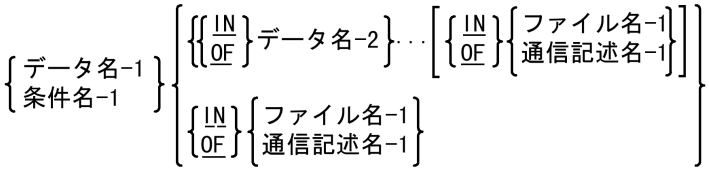
2.6.5 一意参照

2.6.5.1 修飾

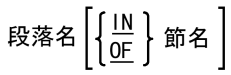
COBOL原始要素中の要素を定義する、利用者が指定したそれぞれの名前。
(ANS85)その 原始要素中で参照する名前は、
すべて一意とする。 名前は、すべて一意とする。そのためには、同じつづりの名前が他にないか、または名前の階層系列の上位にあるいくつかの名前を付け加えて一意にすることができなければならない。このとき、上位の名前を修飾語（qualifier）といい、一意にする手順を修飾（qualification）という。 名前を一意にするには、必要な修飾語を付け加えればよく、上位の名前をすべて書く必要はない。
データ部では、修飾に用いるデータ名はすべて、レベル指示語（level indicator）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ないかぎり、
(ANS85)またはそれらが参照されないかぎり、
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* の節で後述する。

一般形式

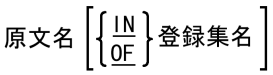
書き方 1



書き方 2



書き方 3



書き方 4

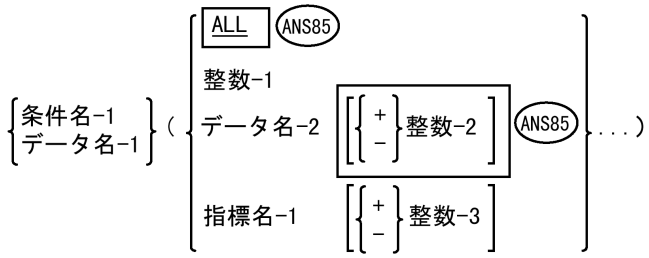
一般規則

1. 各修飾語は、それが修飾する名前と同じ階層系列に属さなければならない。
2. 複数段階にわたって修飾する場合、低いレベルの修飾語から順に書く。
   (ANS85)原始単位の中で明示的に参照される場合、同じ階層系列中の2つのレベルに同じ名前があってはならない。
3. 原始要素中の複数のデータ項目に同じデータ名または条件名を付けた場合、そのデータ名や条件名を手続き部、環境部、データ部で参照するときは、必ず修飾する。ただし、REDEFINES（再定義）句では修飾できない。
4. 1つの節の中に同じ段落名を重複して書いてはならない。段落名を節名で修飾するときは、SECTIONという語を書かない。同じ節内で参照するときは、段落名を修飾する必要はない。
   (ANS85)段落名も節名も明示的に参照されないかぎり、一意であるかまたは一意にする必要はない。
5. データ名を修飾語として使用するときは、添字を付けることはできない。
6. 修飾する必要のない語を修飾しても構わない。一意にする修飾語の組合わせがいくつもある場合、どの組合わせを使用してもよい。あるデータに対する完全な修飾語の系列が、他のデータ名に対する修飾語の系列の一部と同じであってはならない。
   データ名に付けることのできる修飾語の数は、5つまでである。
   (ANS85)50個までの修飾語を付けることができる。
7. 翻訳時に2つ以上の登録集を使えるときには、原文名を参照するたびに修飾しなければならない。
   (OSVS)(VSC2)(MF)この規則は強制しない。

2.6.5.2 添字付け

添字（subscript）は、個別にデータ名を付けていない同種の要素のリストまたは表の中の、個々の要素を参照するために使用する。(*データ部 - データ記述*の章の*OCCURS句*の節を参照。)
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 - 組込み関数*の章の*引数*の節を参照。)
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。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. 下記において表要素を参照する場合は、添字を指定しない。
    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. If
   (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。
   (ANS85)これらは
   指標名-1を用いて指定した値、
   (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。

2.6.5.3 指標付け

指標付け（indexing）を指定して、同種の要素からなる表の中の個々の要素を参照できる。指標（index）を付けるには、表を定義する際に、該当するレベルにINDEXED BY（指標付き）句を指定する。INDEXED BY句を用いて付けた名前を指標名（index-name）といい、付けた指標を参照するために使用できる。指標の値は、対応する表または別の表の中の要素の出現番号を表わす。指標名は、表の参照に使用する前に初期化しておく。指標名に初期値を与えるには、SET（設定）文を指定する。
添字は単なる数字データ項目または定数である。これに対し指標は、表中要素の出現番号を表わす特別な型の項目である。その表現形式は何通りかある。指標の内容は数値とはみなされない。
直接指標付け（direct indexing）は、添字と同じ形式で指標名を使用する方法である。相対指標付け（relative indexing）は、指標名の後ろに演算子"+" または "-" と符号の付かない整数を書き、その全体を一対の左かっこと右かっこで囲んで、表要素のデータ名の後ろに続けた形式で指標名を使用する方法である。相対指標付けによる出現番号は次のようにして決定される。演算子に"+" を指定した場合は、指標の値によって表される出現番号に定数の値が追加される。演算子に"-" を指定した場合は、指標の値によって表される出現番号から定数の値が差し引かれる。複数個の指標を用いるときは、データ構造の階層の上位のものから順に書く。
INDEXED BY句を指定することによってだけ、表に指標を付けることができる。
(OSVS)(VSC2)(MF)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」の節に記した一般形式に含めてある。

2.6.5.4 関数一意名

(ANS85)
関数一意名とは、関数を評価した結果として得られるデータ項目を一意に参照する名前である。この名前は、文字列と分離符を構文的に正しく組み合わせて付ける。

一般形式

FUNCTION 関数名-1 [({引数-1}...)][部分参照]

構文規則

1. 引数-1は、一意名か定数か算術式とする。各関数の定義の中で、引数-1が満たすべき数字、字類、項類に関する具体的な規則が与えられている。(*手続き部*の章の*組込み関数*の節を参照。)
2. 部分参照は、項類が英数字、
   (MF)または各国語型の関数に対して指定できる。
3. 英数字
   (MF)または各国語型
   の関数を参照する関数一意名は、一般形式の中で一意名が使えるところならば、どこにでも指定できる。ただし、一般形式に関する規則によって禁止されている場合は例外である。具体的な禁止条件は下記のとおり。
   a. 文の受取り側作用対象としての場合。
   b. 参照対象のデータ項目が特定の特性（字類と項類、用途、大きさ、符号、許容値など）を取ることが一般形式に関する規則によって求められているが、関数の定義と具体的に指定された 引数に従って関数を評価した結果が、その特性を満たさない場合。
4. 整数または 数字の関数を参照する関数一意名は、算術式
   (MF)またはMOVE（転記）文の送出し側としてだけ、使用できる。

一般規則

1. 参照対象の関数の字類をはじめとする特性は、関数の定義によって定まる。
2. 関数が参照される時点で、指定した引数リスト中の引数は左から右へ1つづつ順に評価される。引数は、それ自体が関数一意名であってもよいし、関数一意名を含む式であってもよい。 評価の対象の引数によって参照される関数は、その引数が指定されている関数と同じであってもよい。言い換えると、関数は再帰型をとることができる。
   (MF)この場合、浮動小数点数が評価の対象になると、作用対象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。

2.6.5.5 部分参照

部分参照（reference modification）とは、データ項目の一部を切り出してデータ項目を定義することである。切り出す範囲はデータ項目中の起点となる文字位置（切り出す文字の再左端文字位置）と長さによって指定する。別途指定がないかぎり、英数字データ項目を参照する 一意名が使えるところでは、どこでも部分参照ができる。

一般形式

```
{データ名-1
{FUNCTION 関数名-1 [({引数-1}...)]} (再左端文字位置:[長さ])
```

注: 上記の一般形式において、データ名-1およびFUNCTION関数名-1（引数-1）は文脈を示すために記したものであり、 部分参照の一部を構成する訳ではない。

構文規則

1. データ名-1は、用途がDISPLAYであるデータ項目を参照しなければならない。
2. 再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、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. データ名-1は修飾してもよいし、添字付けしてもよい。
5. 関数名-1およびその引数（指定されている場合）によって参照される関数は、英数字の関数とする。

一般規則

1. データ名-1または関数名-1およびその引数（ある場合）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。そのデータ名のデータ記述項に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。
2. データ名-1によって参照されるデータ項目の項類が、数字、
   (OSVS)(VSC2)(MF)外部浮動小数点数、
   数字編集、英字、英数字編集のどれかであるとき、 部分参照の目的上、そのデータ項目は同じ大きさの英数字データ項目として再定義されたものとして操作される。
3. 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。
4. 部分参照は、データ名-1または関数名-1およびその引数（ある場合）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. 再左端文字位置を評価した結果、数値が得られる。この値は、データ名-1または関数名-1およびその引数（ある場合）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、再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、再左端文字位置と長さの和から1を引いたものは、データ名-1または関数名-1およびその引数（ある場合）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。

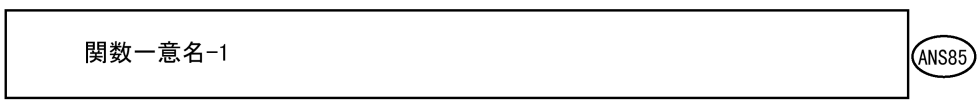
5. 部分参照として取り出されたデータ項目は、JUSTIFIED（桁寄せ）句のない基本データ項目とみなされる。関数を参照した場合は、部分参照として取り出されたデータ項目の字類と項類は英数字となる。データ名-1を指定した場合は、部分参照として取り出されたデータ項目の字類と項類はデータ名-1と同じになる。ただし、項類が数字、数字編集、英数字編集、
(OSVS)(VSC2)(MF)外部浮動小数点数
のものは、字類も項類も英数字とみなされる。
6. (OSVS)OSVSコンパイラ指令を 設定した場合は、 条件式の中では部分参照は指定できない。(詳細については、*手続き部*の章の条件式に関する記述を参照。)

2.6.5.6 一意名

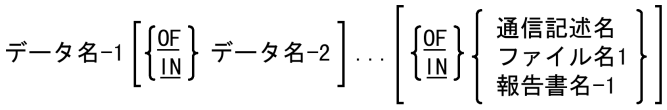
一意名（identifier）とは、データを一意に参照するために、文字列と分離符をいくつか 並べたものである。
(ANS85)関数以外、または (ISO2000)オブジェクト・プロパティのデータ項目を参照するときに、
原始要素の中でデータ名が一意でない場合、そのデータ名の後ろに修飾語か添字、
(ANS85)または 部分参照を指定して、
一意に参照できるようにする。

一般形式

書き方 1



書き方 2



[({添字} ...)]　　[部分参照] (ANS85)

(ISO2000)書き方 3

プロパティ名-1 OF 一意名-1

書き方2の規則

添字-1は添字付け（2.6.5.2「添字付け」を参照）または指標付け（2.6.5.3「指標付け」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。

(ISO2000)書き方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。

2.6.5.7 条件名

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

条件変数を参照するのに指標付けまたは添字付けが必要な場合は、その条件名を参照するときにも、同じ組合わせの指標付けまたは添字付けを使用する。
条件名の修飾、指標付け、添字付けの組合わせの書き方と制限は、一意名の場合のデータ名-1を条件名-1と置き換えたものと同じである。
一般形式において条件名というときは、必要に応じて修飾、指標付け、添字付けしたものを含む。

2.6.6 明示指定と暗黙指定

COBOLの原始要素には、下記の4種類の明示指定と暗黙指定がある。

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

2.6.6.1 手続き部の明示指定と暗黙指定

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）文か、
(ANS85)REPLACE文か、
(OSVS)(VSC2)-INC文か、++INCLUDE文
を実行して参照対象項目の名前を手続き部に複写してきた場合の参照をいう。 暗黙参照（implicit reference）とは、手続き部の文に参照対象項目の名前を書かないでデータ項目を参照することをいう。
PERFORM（実行）文を実行する際にも、暗黙参照が発生する。これは、PERFORM文に関連する制御機構がVARYING句、AFTER句、UNTIL句に指定された指標名または一意名によって参照される、指標またはデータ項目を初期化したり変更したり評価したりした場合である。このような暗黙参照は、データ項目が命令の実行に係わるときだけ行われる。

2.6.6.2 制御の明示移行と暗黙移行

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. ㊎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）文か、
  (ISO2000)(OSVS)(VSC2)(MF)GOBACK(復帰)文か、
  (ISO2000)(MF)EXIT METHOD(メソッド出口)文か、
  EXIT PROGRAM(プログラム出口)文。
- プログラム終了見出し

2.6.6.3 明示属性と暗黙属性

属性は、暗黙的にも明示的にも指定できる。明示的に指定した属性を、明示属性（explicit attribute）という。属性を明示的に指定しないと、省略時の解釈が取られる。このような属性を、暗黙属性（implicit attribute）という。
たとえば、データ項目の用途（usage）は指定しなくてよい。指定のないデータ項目の用途はDISPLAY（表示用）と解釈される。
(VSC2)(MF)(COB370)PICTURE文字列が"G"または"N"を持たないかぎり。この場合、データ項目の用途はDISPLAY-1である。

2.6.6.4 明示範囲符と暗黙範囲符

(ANS85)
範囲符は、手続き部のある種の文（範囲明示文）の範囲を区切る。これには、明示範囲符と暗黙範囲符の2種類がある。
有効な明示範囲符には以下のものがある。

END-ADD
END-PERFORM
(MF)(XOPEN)END-ACCEPT
END-CALL
(MF)END-CHAIN
END-COMPUTE
END-DELETE
(MF)(XOPEN)END-DISPLAY
END-DIVIDE
END-EVALUATE
END-IF

END-MULTIPLY
END-READ
END-RECEIVE
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する。

暗黙範囲符は下記の場合に発生する。

- 完結文の末尾で、それまで続いている前の文のすべての範囲が分離符の終止符によって終わりとされるとき。
- 他の文を含む文の中で、含まれる文のそれまで続いている文の範囲が、含まれる文に続く含む方の文の次の句（たとえば、ELSE, WHEN, AT END）によって終わりとされるとき。

第3章：翻訳集団の概念

3.1 指定しなくてもよい、部、節、段落の見出し

(MF)

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

3.2 予約語

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

3.3 外部リポジトリ

外部リポジトリには、プログラム定義、クラス定義、インタフェー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、インタフェース
- 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「REPOSITORY段落」の節に指定される。

3.4 CALLプロトタイプ

(MF)
CALLプロトタイプは、呼び出されるサブプログラムの特性を定義するのに役立つ、プログラム宣言ないしプログラムの骨組みである。それらの特性には、必要なパラメータの数、それらのパラメータの型、呼出し方式がある。プロトタイプが存在するサブプログラムを参照するCALL文がソース要素中に入っている場合、そのCALL文はそのプロトタイプと照合チェックされる。明確な不一致がある場合には、エラーとされる。CALL文中に定義されていない特性が何かある（たとえば、呼出し方式）場合、それらプロトタイプから引き出される。
CALLプロトタイプは完全なプログラムとして定義され、そのPROGRAM-ID段落中にEXTERNAL句が指定される。そのプログラム構造は、プログラム名段落を伴う見出し部、連絡節を伴うデータ部、手続き部の見出し、およびENTRY文のみから構成されなければならない。ただし、ENTRY文は指定してもしなくてもよい。これらのCALLプロトタイプは他の型の翻訳単位の前に置かれる。その方法はマルチプログラム原始ファイルの場合と同様である。
CALL文をチェックさせるすべてのサブプログラムに関して、そのCALL文が含まれる翻訳単位の見出し部の見出しの前に、CALLプロトタイプのコピーを含めなければならないことに注意。

3.5 ファイル

3.5.1 ファイル結合子

(ANS85)
ファイル結合子はファイルに関する情報が格納されている領域であって、ファイル名と物理ファイルの間およびファイル名とそれに関連するレコード領域との間の連絡に使用される。

3.5.2 順ファイル

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

3.5.2.1 レコード順ファイル

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

3.5.2.2 行順ファイル

(MF)

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

3.5.2.3 (MF)行順ファイルおよびレコード順ファイルの編成

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

3.5.2.4 呼出し法

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

3.5.3 相対ファイル

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

3.5.3.1 相対ファイルの編成

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

3.5.3.2 呼出し法

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

乱 呼出し法（random access method）では、レコードを呼び出す順序はプログラマが制御する。求 めるレコードを呼び出すには、 相対レコード番号を 相対キー・データ項目に設定する。
動的 呼出し法（dynamic access method）では、プログラマが順呼出しと 乱呼出しを 自由に切り替えることができる。このためには、適切なファイル入出力文を使用する。

3.5.4 索引ファイル

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

3.5.4.1 索引ファイルの編成

索引 ファイルは 大記憶ファイルであり、その中のレコードは キーの値によって呼び出される。キーは レコード記述に定義する。 1つ以上の項目をキーに指定できる。各キー項目には索引が付けられる。各索引はレコード中のキー・データ項目の内容に基づいて、そのデータレコードを呼び出す論理的な経路となる。
ファイルのファイル管理記述項のRECORD KEY句に指定したデータ項目が、 そのファイルの主レコードキー（primary record key）となる。 ファイル中のレコードの挿入・ 更新・ 削除の対象となるレコードは主レコードキーによってのみ識別される。したがって、主レコードキーの値は一意でなければならず、レコードを更新する際に変更されてはならない。
ファイルのファイル管理記述項のALTERNATE RECORD KEY句に指定したデータ項目は、そのファイルの副レコードキー（alternate record key）となる。 DUPLICATES指定をしておけば、副レコードキーの値は一意でなくてもよい。 副レコードキーを使用することによって、ファイルからレコードを検索する際の代わりの呼出し経路とすることができる。
(MF)このCOBOLシステムでは、分割キーを使えるように機能が拡張されている。 分割キーとは、いくつかのデータ項目から構成されるキーである。レコード記述の中では、それらのデータ項目は互いに隣接し ていなくても構わないものである。

3.5.4.2 呼出し法

順呼出し法では、 レコードキーの値の昇順にレコードを呼び出す。 同じレコードキーの値をもつレコードの組の中では、レコードはその組の中で書き出された順に呼び出される。
乱 呼出し法では、レコードを呼び出す順序はプログラマが制御する。求めるレコードを呼び出すには、レコードキー・データ項目にレコードキーの値を設定する。
動的 呼出し法では、プログラマが順呼出しと乱呼出しを自由に切り替えることができる。そのためには、適切なファイル入出力文を使用する。

3.5.5 共有モード

(ISO2000)(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。(入出力ステータス、OPEN 文、表14-3 、他のファイル結合子によって現在開かれている利用可能な共有ファイルを開く、を参照)
共有モードでは、ファイルへのアクセスは下記のように制御される。

1. SHARING WITH NO OTHERはファイルを排他的にアクセスすることを指定する。その指定のあるファイルが現在他のファイル結合子を通じて開かれているならば、そのファイル結合子をファイルに関連付けようとしても失敗する。OPEN文が成功したならば、そのファイル結合子が閉じられてないうちに、他のファイル結合子を通じてそのファイルを開こうとする以降の要求は不成功に終わる。レコード・ロックは無視される。

2. 読み取り専用モードで共有すると、該当のもの以外のファイル結合子を通じてそのファイルに同時にアクセスするのは、入力モードのみに限定される。入力以外のモードで開かれているファイルをこのファイル結合に関連付けようとすると、エラーとなる。OPEN文が成功したならば、そのファイル結合子が閉じられないうちに、入力以外のモードにある他のファイル結合子を通じてそのファイルを開こうとする以降の要求は不成功に終わる。レコード・ロックは効力を有する。
3. 上記以外のいかなるモードで共有する場合も、INPUT，I-O，EXTENDのどれかを指定した他のファイル結合子を通じて、ファイルに同時アクセスすることができる。ただし、他に制限が適用される場合がある。レコード・ロックは効力を有する。

同じ実行単位または別の実行単位内のランタイム要素や内包されている要素や別のランタイム・モジュールの中に、複数のアクセス経路が存在する可能性がある。開く対象のファイルのファイル・ロックを保持しているファイル結合子のロケーションに関係なく、ファイル共有に矛盾を来す条件が存在する可能性がある。
ファイル・ロックを設定することは、入出力文の不可分な処理の一部である。
あるファイル結合子に用に確立されたファイル・ロックおよびすべてのレコード・ロックは、そのファイル結合子に関して明示的または暗黙的に実行されたCLOSE文によって解除される。

3.6 オブジェクト

(ISO2000) (MF)

3.6.1 オブジェクトとクラス

オブジェクトとは、独自のデータを持ちクラス定義中に定義されているメソッドを共有する、ランタイム要素であるオブジェクトはクラスによって定義される。クラス定義には、データおよびクラスの各オブジェクトに対して呼び出される可能性あるメソッドの特性が記述されている。
各クラスには、ファクトリ・オブジェクトが１つある。ファクトリ・オブジェクトには独自の一連のデータとメソッドが備わっている。ファクトリ・オブジェクトは通常、オブジェクトのインスタンスを生成し、クラスのすべてのインスタンスに共通のデータを維持更新する働きをする。

3.6.1 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。
任意のオブジェクトに複数のオブジェクト参照を持たせることが許されている。ただし、この標準の要件を全面的に満たす必要がある。通常は、１つのオブジェクトにオブジェクト参照が１つあれば十分である。

3.6.3 定義済オブジェクト参照

定義済オブジェクト参照とは、識別子NULL，SELF，(MF)SELFCLASS，SUPERのど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「定義済オブジェクト識別子」に説明がある。

3.6.4 メソッド

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

3.6.5 メソッド呼出し

INVOKE文を用いてメソッドを呼び出すと，その中の手続きコードが実行される。そのさい、(MF)動詞シグネチャまたはオブジェクト・プロパティを参照する。メソッド呼出しに対応するメソッド処理系は、メソッド呼出しの対象のオブジェクトの実行時のクラスによって決まる。それは必ずしもオブジェクト参照の定義中に静的に指定されているクラスであるとは限らない。メソッド呼出しを具体的なメソッド処理系に対応付けるために使用されるのは、実行時に実際に参照されるオブジェクトのクラスである。
オブジェクト参照を用いてオブジェクトを指定してメソッドを呼び出す場合は下記のようになる。

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

クラス名を用いてオブジェクトを指定してメソッドを呼び出す場合、指定したクラスのファクトリ･オブジェクトがメソッドを呼び出す対象のオブジェクトとして使用されて、ファクトリ・メソッドが呼び出される。
メソッドの解明処理は下記のように進められ、該当する最初のものが適用される。

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

3.6.6 準拠性とインタフェース

このドキュメントで使用する「準拠性」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、「準拠性」という用語はオブジェクト・インタフェース間の関連を記述するために使用され、継承やインタフェース定義や準拠性チェックといった基本的な機能の基盤となる。

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

3.6.6.1 オブジェクト指向への準拠

オブジェクトが準拠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フェース以外のインタフェースに従ってクラスを指定することができる。準拠性とは、あるインタフェースから別のインタフェースへの、およびあるオブジェクトからあるインタフェースへの、一方向性の関連である。

3.6.6.1.1 インタフェース

どのオブジェクトにもインタフェースがある。オブジェクトのインタフェースは、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フェースが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フェースとオブジェクト用のインタフェースである。
個々のクラスから独立にインタフェースを定義することもできる。そのためには、インタフェース定義中に、メソッド・プロトタイプを指定する。

3.6.6.1.2 インタフェース間の準拠性

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

1. インタフェース-2中の各メソッドに対応するメソッドがインタフェース-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. インタフェース-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、インタフェース-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. インタフェース-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、インタフェース-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. インタフェース-2中のパラメータがインタフェース名を用いて記述されているならば、インタフェース-1中の対応するパラメータも同じインタフェース名を用いて記述されている。
    c. インタフェース-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、インタフェース-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フェースで同じである。
3. インタフェース-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、インタフェース-1中の対応する仮パラメータに同じ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. すべての通貨記号は同じであるとみなされる。
    b. DECIMAL-POINT IS COMMA句が一方のインタフェース中でのみ有効に使用されている場合、形式編集文字の終止符はコンマに対応し、コンマは終止符に対応する。
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. インタフェース-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、インタフェース-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. インタフェース-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、インタフェース-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. インタフェース-2中の返却する項目がインタフェースint-rを識別するインタフェース名を用いて記述されているならば、インタフェース-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に準拠するインタフェースを識別するインタフェー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のインタフェースはint-rに準拠する。
            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オブジェクトのインタフェースはint-rに準拠する。

c. インタフェース-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、インタフェース-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. インタフェース-2中の返却する項目の記述にONLY指定があるならば、インタフェース-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. インタフェース-2中の返却する項目の記述にONLY指定がないならば、インタフェース-1中の返却する項目の記述には同じクラス名またはそのクラス名のサブクラスが指定されている。
    3. FACTORY指定の有無が同じである
d. インタフェース-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、インタフェース-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

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

6. インタフェース-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. すべての通貨記号は同じであるとみなされる。
    b. DECIMAL-POINT IS COMMA句が一方のインタフェース中でのみ有効に使用されている場合、形式編集文字の終止符はコンマに対応し、コンマは終止符に対応する。
7. 準拠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。

3.6.6.1.3 パラメータ化されたクラスおよびインタフェースのための準拠性

パラメータ化されたクラスまたはインタフェースを使用するときは、クラス定義またはインタフェー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フェースが用いられているかのように、クラスまたはインタフェースは扱われる。

3.6.7 多相性

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

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

3.6.8 クラスの継承

クラスの継承とは、あるクラスのインタフェースと処理系を、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。それらの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。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、そのデータを宣言したクラス中に定義されているファクトリ・メソッドにのみ見える。継承されたオブジェクト・データは、そのデータを宣言したクラス中に定義されているオブジェクト・メソッドにのみ見える。サブクラスにおいて新しいメソッドおよび追加データを定義して、一連の継承したメソッドとデータを補強することができる。
サブクラスのインタフェースは継承元のクラスのインタフェースに必ず準拠する。ただし、継承元のクラスのメソッドの一部をサブクラスにおいて修正して、別の処理系を用意することができる。
継承元のクラス中でユーザが定義した語はサブクラスに継承されない。したがって、それが継承元のクラスに定義されていないかのように、サブクラスの中で定義することができる。

3.6.9 インタフェースの継承

インタフェースの継承とは、インタフェースの定義を、他のインタフェースの基盤として使用する仕組みである。インタフェースを継承すると、継承元のインタフェー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フェー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フェースは継承元のインタフェースに必ず準拠しなければならない。

3.6.10 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フェー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フェー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。
パラメータ化されたクラスのライフ・サイクルは、この章で後述する、「パラメータ化されたクラスのライフ・サイクル」中に定義する。

3.6.11 パラメータ化されたインタフェース

パラメータ化されたインタフェースは一般形をしたインタフェースないしインタフェースの骨組みである。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フェー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フェー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フェースとして機能するインタフェースが生成される。
パラメータ化されたインタフェースのライフ・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フェースのライフ・サイクル」中に定義する。

3.6.12 オブジェクトのライフ・サイクル

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

3.6.12.1 ファクトリ・オブジェクトのライフ・サイクル

ファクトリ・オブジェクトは、実行単位によって最初に参照される前に、生成される。
ファクトリ・オブジェクトは、実行単位によって最後に参照された後で、破棄される。

3.6.12.2 オブジェクトのライフ・サイクル

ファクトリ・オブジェクトに対してNEWメソッドが呼び出された結果として、オブジェクトが生成される。
オブジェクトに対して (MF)FINALIZEメソッドが呼び出されるか、実行単位が修了するかしたときに、オブジェクトが破棄される。

3.6.12.3 パラメータ化されたクラスのライフ・サイクル

パラメータ化されたクラスが展開されると、あらゆる点で、パラメータ化されていないクラスと同様に扱われるようになる。
REPOSITORY段落中にパラメータ化されたクラスが指定されたときに、パラメータ化されたクラスの仕様に基づいて、新しいクラス（パラメータ化されたクラスのインスタンス）が生成される。そのクラスは独自のファクトリ・オブジェクトを持ち、同じパラメータ化されたクラスの他のインスタンスとは完全に別物である。
実行単位内で、パラメータ化された同じクラスを同じ実パラメータで展開して作成された２つのクラスの外部クラス名が同じ場合、その２つは同じクラス・インスタンスである。パラメータ化されたクラスを異なる実パラメータで展開して作成された２つのクラスは同じクラス・インスタンスではなく、同じ外部クラス名を持たない。

3.6.12.4 パラメータ化されたインタフェースのライフ・サイクル

パラメータ化されたインタフェースが展開されると、あらゆる点で、パラメータ化されていないインタフェー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フェースが指定されたときに、パラメータ化されたインタフェースの仕様に基づいて、新しいインタフェース（パラメータ化されたインタフェースのインスタンス）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フェースを同じ実パラメータで展開して作成された２つのインタフェースの外部インタフェース名が同じ場合、その２つは同じインタフェース・インスタンスである。パラメータ化されたインタフェースを異なる実パラメータで展開して作成された２つのインタフェースは同じインタフェース・インスタンスではなく、同じ外部インタフェース名を持たない。

3.7 実行単位における通信

3.7.1 共通プログラムと初期プログラム

(ANS85)
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。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。

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。その具体的な内容については、この章の「メソッド、オブジェクトまたはプログラムの状態」において後述する。 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。

3.7.2 データの共有

(ANS85)

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

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

3.7.3 ファイル結合子の共有

(ANS85)

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

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

3.8 データ部

3.8.1 概要

データ部は下記の節に分割される。

1. ファイル節
2. 作業場所節
3. (MF)スレッド局所記憶節
4. (MF)オブジェクト記憶節
5. (ISO2000)(MF)局所記憶節
6. 連絡節
7. 通信節
8. 報告書節
9. (ISO2000)(MF)(XOPEN)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。
(ISO2000)(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。

通信節には、メッセージ制御システム（MCS）とプログラムとの間のインタフェースの働きをする原始プログラム中のデータ項目を記述する。

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。

(ISO2000)(MF)(XOPEN)画面節では、画面の属性を定義する。それを通じて、画面上に表示するフィールドの位置を正確に指定したり、ACCEPTやDISPLAYの操作の間にある種のコンソール機能を制御したりすることができる。

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

3.8.2 データの種類と状態

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

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

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

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

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

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

3.8.3 メソッド、オブジェクトまたはプログラムの状態

3.8.3.1 メソッド、オブジェクトまたはプログラムの状態

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

3.8.3.1.1 アクティブ状態

メソッドまたはプログラムを再帰的に起動することができる。したがって、ある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ことがありえる。
メソッドのインスタンスは、正常に起動されたときに、アクティブ状態に置かれる。そして、そのメソッドのインスタンス内でSTOP文が実行されるか、EXIT METHOD文が明示的または暗黙的に実行されるまで、アクティブ状態を保つ。
プログラムのインスタンスは、オペレーティング・システムによって正常に起動されるかランタイム要素から正常に呼び出されたときに、アクティブ状態に置かれる。そして、下記のどれかひとつが実行されるまで、アクティブ状態を保つ。

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

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

3.8.3.1.2 データの初期状態と最終使用状態

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

3.8.3.1.2.1 初期状態

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

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

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

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

3.8.3.1.2.2 最終使用状態

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

3.8.3.2 オブジェクトとのみ関連があるデータの初期状態

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」の記述に従って初期化される。

3.8.4 大域名と局所名

(ANS85)
データ名はデータ項目の名前である。ファイル名はファイル結合子の名前である。これらの名前は大域名と局所名のどちらかに分類される。

大域名は、それが宣言されているソース要素内からでも、そのソース要素に含まれる他の任意のソース要素からでも、関連のある項目を参照するために使用することができる。
A それと対照的に、局所名は、それが宣言されているソース要素内から、関連のある項目を参照するためにしか使用することができない。名前の中には常に 大域的なものがある。常に局所的な名前もある。その他に、名前が宣言されているソース要素内の仕様に応じて、大域的または局所的となる名前もある。

レコード名が宣言されているレコード記述項内に GLOBAL句が指定されている場合、そのレコード名は大域的である。あるいは、ファイル節内のレコード記述の場合、そのレコード記述項に関連するファイル名のファイル記述項にGLOBAL句が指定されている場合、そのレコード名は大域的である。
データ名が宣言されているデータ記述項またはその上位のデータ記述項の中にGLOBAL句が指定されている場合、レコード名は大域的である。

データ記述項内に宣言されている条件名は、そのデータ記述項の上位のデータ記述項の中にGLOBAL句が指定されている場合、大域的である。しかし、特定の規則によって、時にはある種のデータ記述項、ファイル記述項、またはレコード記述項にGLOBAL句を指定することが禁止されることがある。
該当ファイルのファイル記述項中にGLOBAL句が指定されている場合、ファイル名は大域的である。
データ記述項中に宣言されたデータ名、ファイル名、条件名が大域的ではない場合、その名前は局所的である。

大域名は他のソース要素内に含まれるソース要素間にわたって推移的である。

3.8.5 外部項目と内部項目

(ANS85)

呼出し可能なデータ項目には通常は何らかの表現のデータが格納されている必要がある。ファイル結合子にはファイルに関する何らかの情報が格納されている必要がある。データ項目またはファイル結合子に関する記憶場所は 外部または 内部でありえる。
外部および内部の項目は大域名または局所名のどちらかを持てる。
作業場所節中に記述されたデータ・レコードは、そのデータ記述項内にEXTERNAL句があると、外部属性を持つ。外部レコードを記述しているデータ記述項の下位のデータ記述項によって記述されているデータ項目も外部属性を取得する。レコードまたはデータ項目が 外部属性を持たない場合、 それは内部データである。
関連するファイル記述項にEXTERNAL句があると、ファイル結合子は外部属性を得る。外部属性を持たないファイル結合子は内部的である。
EXTERANL句が含まれないファイル記述項または整列併合ファイル記述項の下位に記述されたデータ・レコードならびにそれらのレコードのデータ記述項の下位に記述されたデータ項目は、常に内部的である。ファイル記述項にEXTERNAL句が含まれていると、データ・レコードおよびデータ項目は外部属性を得る。
局所記憶節、連絡節、通信節、報告書節、画面節に記述されたデータ・レコード、下位のデータ項目、関連する各種の制御情報は常に内部的である。連絡節に記述されて、記述されたデータ・レコードと他のランタイム要素から呼出し可能な他のデータ項目との間に関連づけがなされているデータに対しては、特別な考慮が当てはまる。
データ項目またはファイル結合子が外部的である場合、その項目に関連する記憶場所は、実行単位内の個々のランタイム・モジュールとではなく、実行単位と関連する。 外部項目は、それを記述する実行単位内の任意のランタイム・モジュールから参照できる。異なるランタイム・モジュールからデータ項目またはファイル結合子の別立ての記述を使用して外部項目を参照した場合、必ず同じ項目が参照される。実行単位内では、外部項目の表現はひとつしか存在しない。
データ項目またはファイル結合が 内部的である場合、それに関連する記憶場所はそれを記述するランタイム・モジュールとのみ関連する。

3.9 手続き部

3.9.1 実行

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

3.9.2 文と完結文

文は、COBOLの動詞で始まる、語と記号の構文的に有効に組み合わせである。
完結文はいくつかの文で構成され、終止符とそれに続く空白で区切られる。
文には下記の４種類がある。

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

完結文には下記の３種類がある。

1. 条件完結文
2. COBOLシステム指示完結文
3. 無条件完結文

3.9.2.1 条件文

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

- (ANS85)EVALUATE,
  IF, SEARCH, RETURNの各文
- AT END句またはINVALID KEY句のあるREAD文
- (OSVS)ON文
- INVALID KEY句またはEND-OF-PAGE句のあるWRITE文
- INVALID KEY句のあるSTART, REWRITE, DELETEの各文
- SIZE ERROR句のある算術文（ADD, COMPUTE, DIVIDE, MULTIPLY, SUBTRACT）
- NO DATA句のあるRECEIVE文
- ON OVERFLOW句のあるSTRING文またはUNSTRING文
- ON OVERFLOW句
  (ANS85)またはON EXCEPTION

句のあるCALL文

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

3.9.2.2 条件完結文

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

3.9.2.3 COBOLシステム指示文

COBOLシステム指示文（COBOL system-directing statement）は、 翻訳指示動詞とその作用対象（operand）からなる。翻訳指示動詞については「翻訳指示文」の章に説明がある。
COBOLシステム指示文は、実行用プログラム・コードを生成する際にCOBOLコンパイラに指定された動作を取らせるものである。

3.9.2.4 COBOLシステム指示完結文

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

3.9.2.5 コンパイラ指令

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

(ISO2000)標準的に用意されているコンパイラ指令については、このマニュアルの「翻訳指示文」の章の「コンパイラ指令」の節に説明がある。
(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が『オンライン参照マニュアル』に記載されている。

3.9.2.6 無条件文

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

ACCEPT[1]	ENABLE	RELEASE
ADD[2]	EXIT	REWRITE[3]
ALTER	GO TO	SEND
CALL[4]	INSPECT	SET
CANCEL	MERGE	SORT
CLOSE	MOVE	START[3]
COMPUTE[2]	MULTIPLY[2]	STOP
DELETE[3]	OPEN	STRING[4]
DISABLE	PERFORM	SUBTRACT[2]
DISPLAY[1]	READ[5]	UNSTRING[4]
DIVIDE[2]	RECEIVE[6]	WRITE[7]

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NO DATA句のないもの
7 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	PURGE

(ISO2000)ISO2000の追加の無条件動詞には以下のものがある。

(ISO2000)(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"を挟んでも構わない。

3.9.2.7 無条件完結文

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

3.9.2.8 範囲明示文

(ANS85)
範囲明示文（delimited scope statement）とは、対応する明示的な範囲符（scope terminator）によってその有効範囲を限られる文をいう。たとえば、IF文は範囲明示文であり、それに対応する範囲符はEND-IFである。1組の範囲明示文と 範囲符との間にある文はすべて、その範囲明示文に含まれると みなされる。
範囲明示文は入れ子にすることができる。その場合、 明示的な各範囲符は、その手前にある有限な範囲文のうちでまだ対になっていない最も近いものと対をなすものとみなされる。
範囲明示文を暗黙に終了させることもできる。それには次の２通りがある。手続き完結文の末尾では、分離符の終止符によって、まだ終了していない文がすべて終了される。範囲明示文を含む文が終了すると、含まれる範囲明示文も終了する。

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

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

3.9.2.9 文の分類

分類	動詞
算術	ADD COMPUTE DIVIDE INSPECT (TALLYING) MULTIPLY SUBTRACT (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またはINVALID KEY) RECEIVE (NO DATA) 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またはTIME) ACCEPT MESSAGE COUNT 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 (オブジェクト参照)
終了	(MF)EXIT METHOD EXIT PROGRAM (OSVS)(VSC2)(MF)GOBACK STOP

分類	動詞
入出力	ACCEPT（識別子） CLOSE (MF)COMMIT DELETE DISABLE DISPLAY ENABLE OPEN READ RECEIVE REWRITE (MF)ROLLBACK SEND START STOP（定数） (MF)UNLOCK WRITE (OSVS)(MF)EXHIBIT PURGE 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 (SO2000)(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

分類	動詞
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-RECEIVE 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。

3.10 正書法

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

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

(MF)(XOPEN)「COBOL言語の紹介」の章の「自由方式」の節も参照。

3.10.1 正書法の表現

1行の正書法を図3-1に示す。

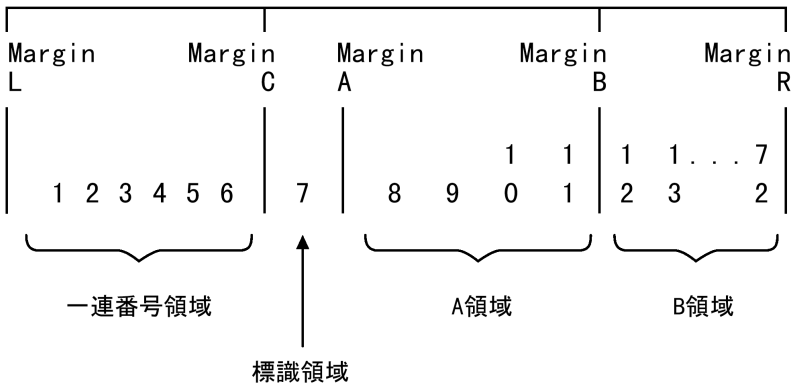


図3-1: COBOLの原始行の正書法

- 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。
(ISO2000)プログラム原文領域はA領域とB領域の両方から構成される。プログラム原文領域のどこから原文語を書き始めてもかまわない。

3.10.1.1 一連番号

一連 番号（sequence number）は、一連番号領域に書く6桁の数字であり、原始 行を識別するために使用する。一連番号は通常、連続する各原始行に対して、昇順に番号付ける。
(OSVS)(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（「翻訳指示文」の章を参照）。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。
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文字の大小順序で空白文字より小さいもの）を指定すると、その行は 注記として扱われ、印刷用のファイルまたは装置に出力されない。 この機能を利用して、出力 印刷用ファイルを次のコンパイルの原始ファイルとして使用できる。この機能は、MFCOMMENTコンパイラ指令の影響を受ける。

3.10.1.2 行の継続

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、任意の語や定数や
(ANS85)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。
(ANS85)この場合、間の空白は存在しないものとみなされる。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)(COB370)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。
(ISO2000)>>は、2文字とも同じ行に書く。

3.10.1.3 空白行

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

3.10.1.4 仮原文

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

3.10.2 部、節、段落の正書法

3.10.2.1 部の見出し

部の見出しは、A領域から書き始める（図3-1を参照）。
(SO2000)(MF)B領域から書き始めてもよい。

3.10.2.2 節の見出し

節の見出しは、A領域から書き始める（図3-1を参照）。
(SO2000)(MF)B領域から書き始めてもよい。
環境部と手続き部では、節はいくつかの段落から構成される。段落がない場合もありえる。データ部では、節はいくつかの記述項から構成される。記述項がない場合もありえる。

3.10.2.3 段落の見出し、段落名、段落

段落には2通りある。1つは、先頭に段落名を書いて終止符（.）に続く空白で止め、その後ろに完結文をいくつか書いたものである。もう1つは、段落の見出しの後ろに記述項をいくつか書いたものである。段落内に注記項を含めることができる。段落の見出しおよび段落名は、部または節の最初の行よりも後ろの任意の行に、A領域から書き始める。
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。
(SO2000)(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を2行以上にわたって書く必要があるときは、前記の「行のつなぎ」の規則に従う。

3.10.3 データ部の記述項

データ部の各記述項は、レベル指示語またはレベル番号で始め、その後ろに空白を置いて、記述項の名前を続け
(ANS85)（存在する場合）、
さらにその記述項について記述する一連の独立な句を続ける。最後の句は常に終止符（.）に続く空白で止める。
データ部の記述項には2種類ある。具体的には、レベル指示語で始まるものと、レベル番号で始まるものである。
レベル指示語（level indicator）には下記のものがある。

- FD （「翻訳集団の定義」の章の「ファイル記述項の全体的な骨組み」の節を参照）
- SD （「翻訳集団の定義」の章の「整列併合ファイル記述項の全体的な骨組み」の節を参照）
- CD （『言語リファレンス - 追加トピック』の「通信」の章の「通信記述項の全体的な骨組み」の節を参照）
- 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」の節を参照）

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領域または
(ANS85)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
，88のどれかとする。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だけ制限される。
(ISO2000)(MF)レベル番号が01と77以外のデータ記述項を、A領域から書き始めてもよい。

3.10.4 宣言部分

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

3.10.5 注記行

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

OSVS 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。

3.10.5.1 行内注記

ISO2000 MF

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

第4章：翻訳集団の定義の概要

4.1 はじめに

この章ではCOBOL言語の基本的な定義について説明する。
まず最初に翻訳集団全体の構造を説明する。翻訳集団の各構成要素は４つの部に分かれている。具体的には、見出し部、環境部、データ部、手続き部がある。
したがって、この資料もそれらの部別に構成されている。各部の中では、COBOLの節ごとに見出しを付けて説明する。項は、参照しやすいように、アルファベット順に配列してある。組み込み関数およびCOBOLの動詞は項のレベルに含める。
追加の言語機能として、DBCS、報告書作成、通信、SQL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 - 追加トピック』を参照。

4.2 翻訳集団の構造

(ANS85)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。

4.2.1 構造

(ANS85)翻訳集団にはいくつかのソース単位を含めることができる。含まれるソース単位が１つもないこともありえる。
ソース単位は見出し部で始まる。 (ANS85)その中に別のソース単位を含みうる。
ソース単位の文、記述項、節は４つの部にグループ化される。各部は下記の順に配列される。(ISO2000)(MF)ただし、コンパイラ指令、原始文操作文、 (ANS85)終了見出しは例外である。

1. 見出し部
2. 環境部
3. データ部
4. 手続き部

ソース単位中の部の開始は適切な部の見出しによって示される。 (ISO2000)(MF)見出し部の開始は、見出し部内で許されている段落の見出しのどれかによっても示すことができる。

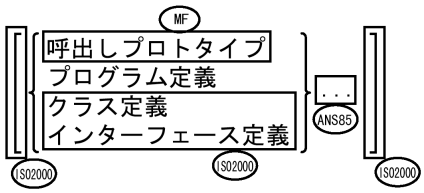
ソース単位中の部の終了は下記のどれかひとつによって示される。

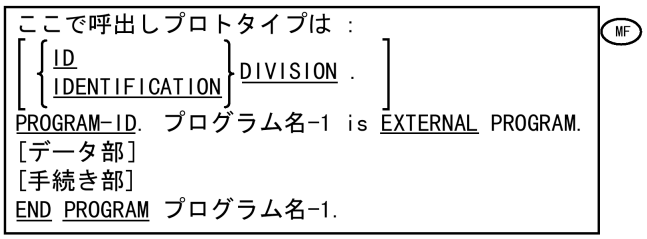
1. そのソース単位中の次の部の開始
2. (ANS85)そのソース単位の終了見出し
3. それ以上原始行を記述できない物理位置の後

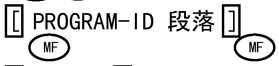
ソース単位の終了は (ANS85)終了見出し（指定された場合）、または翻訳集団内に後続の原始行が存在しないことによって示される
ソース単位Bが (ANS85)他のソース単位Aに含まれるとき、包含関係は直接的でも間接的でもかまわない。Aに含まれBを含む他のソース単位が存在しない場合、ソース単位Bはソース単位Aに直接的に含まれる。Aに含まれBを含む他のソース単位が存在する場合、ソース単位Bはソース単位Aに間接的に含まれる。
他のソース単位に直接的にまたは間接的に含まれるソース単位は、この仕様においては、別のソース単位とみなされる。ただし、入れ子になったソース単位は親のソース単位中に定義されている資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。

4.2.1.1 COBOL翻訳集団

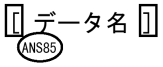
一般形式



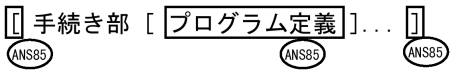


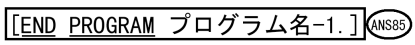


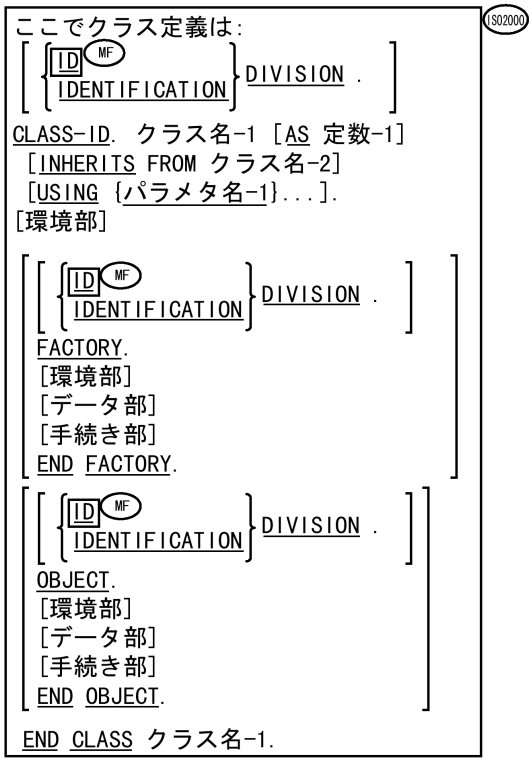
END PROGRAM プログラム名-1.

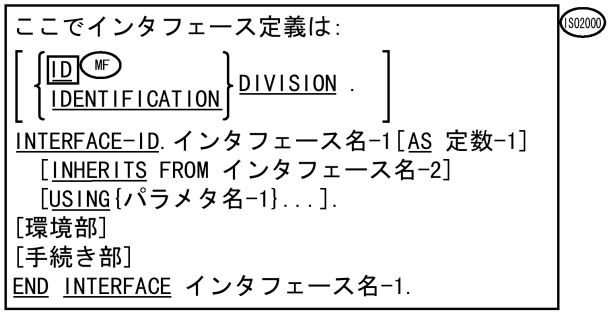


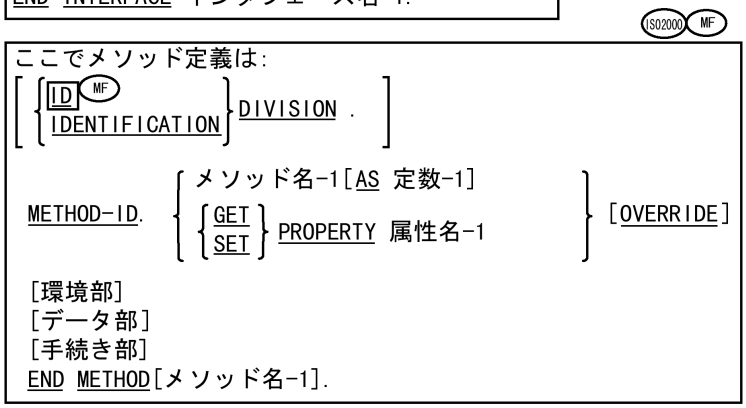
ANS85






ISO2000
ここでクラス定義は:
[{ID MF / IDENTIFICATION} DIVISION .]
CLASS-ID. クラス名-1 [AS 定数-1]
[INHERITS FROM クラス名-2]
[USING {パラメタ名-1}...].
[環境部]
[[{ID MF / IDENTIFICATION} DIVISION .]
FACTORY.
[環境部]
[データ部]
[手続き部]
END FACTORY.]
[[{ID MF / IDENTIFICATION} DIVISION .]
OBJECT.
[環境部]
[データ部]
[手続き部]
END OBJECT.]
END CLASS クラス名-1.


ISO2000
ここでインタフェース定義は:
[{ID MF / IDENTIFICATION} DIVISION .]
INTERFACE-ID.インタフェース名-1[AS 定数-1]
[INHERITS FROM インタフェース名-2]
[USING{パラメタ名-1}...].
[環境部]
[手続き部]
END INTERFACE インタフェース名-1.


ISO2000 MF
ここでメソッド定義は:
[{ID MF / IDENTIFICATION} DIVISION .]
METHOD-ID. { メソッド名-1[AS 定数-1] / {GET / SET} PROPERTY 属性名-1 } [OVERRIDE]
[環境部]
[データ部]
[手続き部]
END METHOD[メソッド名-1].

構文規則

1. (MF)翻訳集団内では、呼び出し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なければならない。
2. (ISO2000)(MF)クラス定義中のメソッドのデータ部には通信節が含まれていてはならない。
3. (ISO2000)インタフェー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. (ISO2000)インタフェー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書き方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文の書き方２の規則に従わなければならない。

一般規則

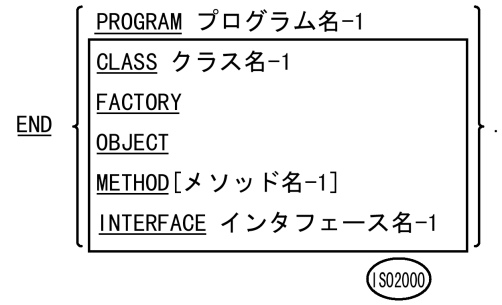
1. (MF)呼出しプロトタイプにおいては、データ部内の節は連絡節以外はすべて無視され、手続き部は手続き部の見出しとENTRY文以外は無視される。

4.2.2 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



一般規則

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. (ISO2000)(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。
5. (ISO2000)(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。 (ISO2000)メソッド名段落中にPROPERTY指定を指定した場合、メソッド名-1を省く。
6. (ISO2000)インタフェース名-1は対応するインタフェース名段落中に宣言されているインタフェース名と同じでなければならない。
7. 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。
8. 終了見出しによって終わりを区切られているソース単位が他のソース単位に含まれていない場合、次の文は別の翻訳単位の最初の文でなければならない。

一般規則

1. 終了見出しは指定したソース単位の終了を示す。

第5章：見出し部

5.1 見出し部

見出し部では、プログラム、 (ISO2000)(MF)クラス、 (ISO2000)(MF)ファクトリ・オブジェクト、 (ISO2000)(MF)オブジェクト、 (ISO2000)(MF)メソッド、 (ISO2000)インタフェースを識別する。 ソース要素を含むファイルのファイル名から得られる基本名からも、そのファイル中に定義されている最初のソース要素が識別される。段落の見出しはその段落に含まれる情報の種類を示す。ユーザーはプログラムの作成日および一般形式に示されている段落のもとで望まれる他の情報を含めることができる。

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

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

(MF)この構文は、ISO2000を除くMicro Focus COBOLに組み込まれているすべての方言で使用できる。FLAGSTDコンパイラ指令を使用すると、この構文が用いられているすべての箇所を見つけ出すことができる。

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

一般形式

構文規則

(ANS85)

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

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

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

(OSVS)注記項の中のB領域の任意の位置に、SKIP1，SKIP2，SKIP3，EJECTの各文を含めることができる。これらの文は、1行に単独で書いた場合にその機能を発揮する。この場合は、これらの文は注記項を終わらせることはない。
(OSVS)注記項は、注記行のA領域またはB領域の任意の位置に書ける。ただし、次の行のA領域に下記のCOBOLの語のどれかを書くと、注記項は終わりとされ、次の段落または部が始まるものとみなされる。

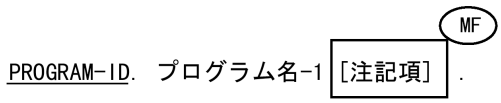
PROGRAM-ID
AUTHOR
INSTALLATION
DATE-WRITTEN
DATE-COMPILED
SECURITY
ENVIRONMENT
DATA
PROCEDURE

5.2 プログラム名段落

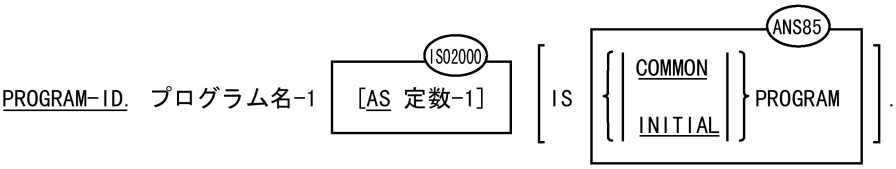
プログラム名（PROGRAM-ID）段落は、プログラムを識別する名前を付けるとともに、指定されたプログラム属性をプログラムに割り当てる。
プログラム名段落は、呼出しプロトタイプを識別する名前を指定する。

一般形式

書き方1



書き方2



書き方3

指令

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

構文規則

すべての書き方

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

書き方1と2

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

   (OSVS)(VSC2)他のプログラムに含まれないプログラムのプログラム名-1の２文字目から８文字目までの間にハイフンがあると、それはゼロに変えられる。
   (VSC2)他のプログラムに含まれるプログラムに関しては、プログラム名-1は長さが30文字まで任意のユーザー定義のCOBOLの語とすることができる。最初の８文字が一意である必要はない。上記のプログラム名の変換は行われない。小文字を使用してもかまわない。ただし、小文字を使用した場合、大文字と小文字は区別される。
   この動作はMAPNAMEコンパイラ指令を用いて制御する。
4. (ANS85)他のプログラムに含まれるプログラムに、その親の翻訳単位内に含まれている他のプログラムと同じ名前を付けてはならない。

書き方2

5. (ANS85)任意指定のCOMMON指定は、プログラムが他のプログラムに含まれる場合だけ指定できる。

6. (ANS85)IS PROGRAM指定を書く場合は、COMMONとINITIALのどちらか一方、または両方を指定する。両方を指定する場合、どちらを先に書いてもよい。
7. (ISO2000)定数-1は英数字の定数でなければならず、表意定数であってはならない。
8. (ISO2000)他のプログラム内に含まれるプログラムの中に定数-1を指定してはならない。

一般規則

書き方1と2

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

書き方2

2. (ANS85)COMMON指定は、プログラムが共通のものであることを指定する。共通のプログラムは他のプログラムの中に含まれるが、親のプログラム以外のプログラムからも呼び出せる。
3. (ANS85)INITIAL指定は、プログラムが初期処理用のものであることを指定する。初期プログラムが呼ばれると、そのプログラムおよびその中に含まれるすべてのプログラムが初期化される。（この章で前述した「プログラムの初期状態の節を参照）

書き方3

4. (MF)プログラム名-1は、CALL定数文の妥当性と性質を判定するための構文チェックの間に使用される、呼出しプロトタイプに名前を付ける。

5.3 クラス名段落

(ISO2000)(MF)

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

一般形式

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

構文規則

1. 定数-1は英数字の定数であり、表意定数であってはならない。
2. クラス名-2はリポジトリ段落 (MF)またはこのソース要素のクラス管理段落の中で指定されたクラスの名前とする。
3. クラス名-2はクラス定義によって定義されたクラスの名前とはしない。

4. クラス名-2はクラス名-1を直接的にも間接的にも継承しない。クラス名-2は、クラス名-1を直接的または間接的に展開する、パラメータ化されたクラスの名前としない。
5. パラメータ名-1は、このクラス定義のリポジトリ段落中のクラス指定子またはインタフェース指定子に指定された名前とする。

一般規則

1. クラス名-1はこのクラス定義に宣言されたクラスに名前を付ける。しかし、定数-1を指定した場合、それは運用環境の外部にあるクラスの名前を指す。
2. INHERITS指定は、「クラス継承」に基づいてクラス名-1によって継承される、クラスの名前を指定する。
3. USING指定は、該当のクラスがパラメータ化されていることを指定する。パラメータ名-1は仮パラメータに付けられた名前である。パラメータ化されたクラスの動作の詳細については、「パラメータ化されたクラス」を参照。
4. パラメータ名-1は、クラス名またはインタフェース名が許されている、このクラス定義中のみで指定する。

5.4 ファクトリ段落

(ISO2000)

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

一般形式

FACTORY.

5.5 オブジェクト段落

(ISO2000) (MF)

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

一般形式

OBJECT.

5.6 メソッド名段落

(ISO2000) (MF)

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

一般形式

構文規則

1. 定数-1は英数字の定数であり、表意定数であってはならない。
2. 親のオブジェクトまたはファクトリの定義の作業場所節にデータ名としてプロパティ名-1を指定した場合、そのデータ名のデータ記述項にPROPERTY指定を書いてはならない。
3. GET指定を書いた場合、手続き部の見出し中にメソッドのUSING指定パラメータを書いてはならず、メソッドに単一のRETURNING指定を書かなければならない。
4. SET指定を書いた場合、手続き部の見出し中にメソッドのUSING指定パラメータを書かなければならず、RETURNING指定を書いてはならない。
5. メソッド名-1または定数-1が親の定義によって継承されたメソッド名と同じである場合、手続き部の見出し上のパラメータ宣言は準拠規則に従っていなければならない。それは、「パラメータおよび返却する項目への準拠性」に従って継承されたあらゆる定義に、このメソッド定義が含まれる定義が準拠することを保証するためである。

一般規則

1. このメソッド定義によって宣言されるメソッドの名前は下記のように決定される。
    a. PROPERTY句を指定した場合、メソッド名は下記のように設定される。
        1. GET指定を書いた場合、文字"GET"にプロパティ名-1を大文字化したものが結合される。
        2. SET指定を書いた場合、文字"SET"にプロパティ名-1を大文字化したものが結合される。
    b. PROPERTY句を指定しなかった場合、メソッド名-1がメソッド名となる。ただし、定数-1を指定した場合、それは運用環境の外部にあるクラスの名前を指す。
2. メソッドの名前は、このメソッドが定義されているクラスのオブジェクトに関するメソッド呼出しの中で参照できる。
3. 指定した利用者語がこのメソッド定義のデータ部および上位のファクトリ定義またはオブジェクト定義のデータ部の中に定義されている場合、このメソッドにおいて使用した利用者語はこのメソッド内の宣言を参照する。このメソッドから上位のファクトリ定義またはオブジェクト定義中の宣言を呼び出すことはできない。
4. GET指定を指定した場合、このメソッドはプロパティ名-1のプロパティ読み出しメソッドとなる。
5. SET指定を指定した場合、このメソッドはプロパティ名-1のプロパティ設定メソッドとなる。

5.7 インタフェース名段落

(ISO2000)

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

一般形式

```
INTERFACE-ID. インタフェース名-1 [ AS 定数-1 ]
      [ INHERITS FROM インタフェース名-2 ]
      [ USING { パラメータ名-1 } ... ] .
```

構文規則

1. 定数-1は英数字の定数であり、表意定数であってはならない。
2. インタフェース-2このソース要素のリポジトリ段落中に指定されたインタフェースの名前でなければならない。
3. インタフェース名-2はインタフェース-1から直接的にも間接的にも継承してはならない。
4. パラメータ名-1は、このインタフェース定義のリポジトリ段落中のクラス指定子またはインタフェース指定子の中に指定された名前でなければならない。

一般規則

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

5.8 翻訳日付け段落

機能

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

一般形式

DATE-COMPILED.[注記項]...

指令

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

構文規則

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

一般規則

1. 翻訳日付け段落が書いてあると（注記項の有無を問わない）、COBOLシステムは生成される実行用プログラム・コードに日付注記項を挿入する。プログラム・リスト中では、その内容は下記の形に置き換えられる。
   DATE-COMPILED. 現在の日付と時刻。
   ここで、日付と時刻の形式は、DD-MMM-YY hh:mmである。
   The DATEコンパイラ 指令を使用すると、注記項を置き換えるためにCOBOLコンパイラによって使用される文字列を変形できる。
   (ANS85)翻訳日付け段落に挿入される値は、組込み関数の WHEN-COMPILEDで使用される値でもある。ただし、両者の形式に違いがあることもある。

第6章：環境部

6.1 環境部

6.1.1 一般説明

環境部では、計算機機種の物理的な特性によるデータ処理上の問題を取り扱う、標準的な方法を指定する。たとえば、翻訳用計算機と実行用計算機の構成を指定できる。それに加えて、入出力管理、ハードウエアの特殊な仕様、管理技法などに関する情報も指定できる。
(ANS85)COBOL原始要素において、環境部は書いても書かなくてもよい。

一般形式

```
ENVIRONMENT DIVISION.
[構成節]
[入出力節]
```

構文規則

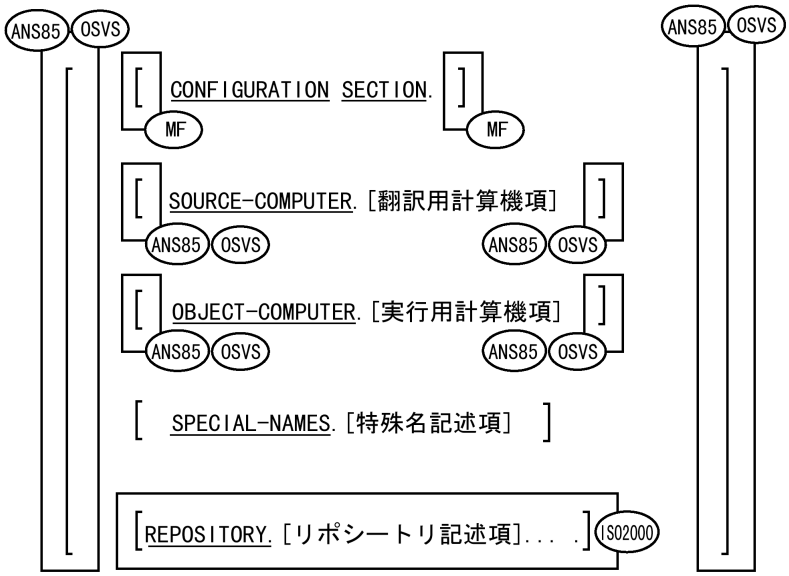
1. (MF)プログラムに環境部が含まれる場合、環境部がプログラムの中の最初の部である場合にだけ、環境部の見出しを書かなくてもよい。

6.1.2 構成節

構成節（configuration section）は、環境部を構成する節の1つである。 構成節で指定する事柄には、特定のシステムに依存するデータ処理システムの様相、特殊な管理技法、局所名を外部名に関連づける方法などがある。この節は翻訳用計算機段落、実行用計算機段落、特殊名段落、リポジトリ段落に分けられる。翻訳用計算機段落では、ソース要素を翻訳する計算機の構成を記述する。実行用計算機段落では、コンパイラによって生成されたランタイム・モジュールを実行する計算機の構成を記述する。特殊名段落では、通貨記号の指定、小数点の選択、記号文字の指定、作成者語とユーザーが指定する呼び名との対応づけ、符号系名と文字集合または文字の大小順序への対応づけ、クラス名と文字集合の関連づけを行う。リポジトリ段落では、局所名を外部資源と関連づける。

(ANS85)(OSVS)構成節は書いても書かなくてもよい。 .

一般形式



構文規則

1. (MF)構成節がプログラムの中の最初の文である場合にだけ、構成節の見出しを書かなくてもよい。
2. (ANS85)他のプログラムに含まれるプログラムの中には、構成節を記述できない。
3. (ISO2000)メソッド定義中に構成節を記述してはならない。
4. (ISO2000)ファクトリ定義またはオブジェクト定義の中に翻訳用計算機、実行用計算機、リポジトリの各段落を記述してはならない。

一般規則

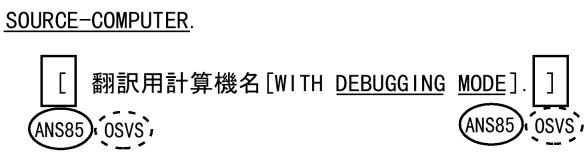
1. (ANS85)他の原始単位を含む原始単位の構成節の中に明示的または暗黙的に記述されている事項は、直接的または間接的に含まれる各原始単位にも適用される。

6.1.2.1 翻訳用計算機段落

機能

翻訳用計算機（source-computer）段落は、翻訳単位を翻訳する計算機を識別する。

一般形式



構文規則

1. 翻訳用計算機段落は注記としてだけ使用する。
2. WITH DEBUGGING MODE句は、標準ANSI COBOLに従ったデバッグ用コードを組み入れるために使用する。(***言語リファレンス - 追加トピック***中の*デバッグの章の*COBOL Debug*の環境部*を参照。)
3. (ISO2000)翻訳用計算機段落のすべての句は、それらが明示的または暗黙的に指定されたソース単位およびそのソース単位に含まれる任意のソース単位に適用される。

6.1.2.2 実行用計算機段落

機能

実行用計算機（object-computer）段落には、コンパイラによって作成された実行時モジュールを実行するのに使う計算機を記述する。<

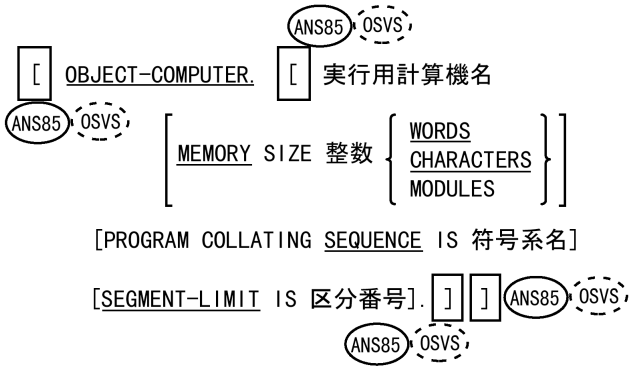
(ANS85)(OSVS)この段落は書いても書かなくてもよい。

(ANS85)MEMORY SIZE（記憶容量）句とSEGMENT-LIMIT(区分制限)句 clauseはANSI'85においては廃要素に分類されており、ANSI標準を次回に全面改訂する際には削除される予定である。

(MF)このCOBOL処理系に組み込まれているすべての方言では、MEMORY SIZE句 とSEGMENT-LIMIT句は注記としてだけ使用する。FLAG STD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。

(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、MEMORY SIZE句と SEGMENT-LIMIT句は明示的に除外されている。したがって、X/Openに準拠するCOBOL原始プログラムにおいては、MEMORY SIZE句と SEGMENT-LIMIT句を使用するべきではない。

一般形式



構文規則

1. 実行用計算機名は、利用者が定義した1語のCOBOLの語である。
2. ANS85 OSVS実行用計算機段落は、OBJECT-COMPUTER 見出しだけから成る。

一般規則

1. 翻訳用計算機名は、機器構成を識別する手段となる。この場合、計算機名およびそれが意味する機器構成は、利用者が指定する。構成の定義には記憶容量に関する情報を含む。計算機名と MEMORY SIZE句は注記としてだけ使用する。
2. PROGRAM COLLATING SEQUENCE プログラムの文字の大小順序）句を指定しないと、計算機に固有の文字の大小順序が適用される。 付録の *文字集合と文字の大小順序* に、ASCIIおよびEBCDICのすべての文字の大小順序の一覧を掲げてある。 NATIVEコンパイラ 指令を使用して、計算機に固有の文字の大小順序を選択することもできる。
3. PROGRAM COLLATING SEQUENCE句を指定すると、プログラムの文字の大小順序は指定した符号系名のものとなる。
4. 実行用計算機段落に指定したプログラムの文字の大小順序は、下記の文字比較における真理値の決定に用いられる。
    a. 比較条件の中で明示(後述の*比較条件*の節を参照。)
    b. 条件名条件の中で明示(後述の*条件名条件(条件変数)*の節を参照。)
5. SORT（整列）文またはMERGE（併合）文にCOLLATING SEQUENCE句を指定しないと、整列または併合用の文字のキーにもPROGRAM COLLATING SEQUENCE句が使用される。
6. 索引ファイルの順序づけには、PROGRAM COLLATING SEQUENCE句は何の効力も発揮しない。
7. PROGRAM COLLATING SEQUENCE句は、それが指定された原始要素の中、ANS85およびその中に含まれる原始要素の中で使用される。
8. SEGMENT-LIMIT（常駐区分の範囲）句は、注記としてだけ使用する。(***言語リファレンス - 追加トピック***中の区分化の章を参照。)

6.1.2.3 特殊名段落

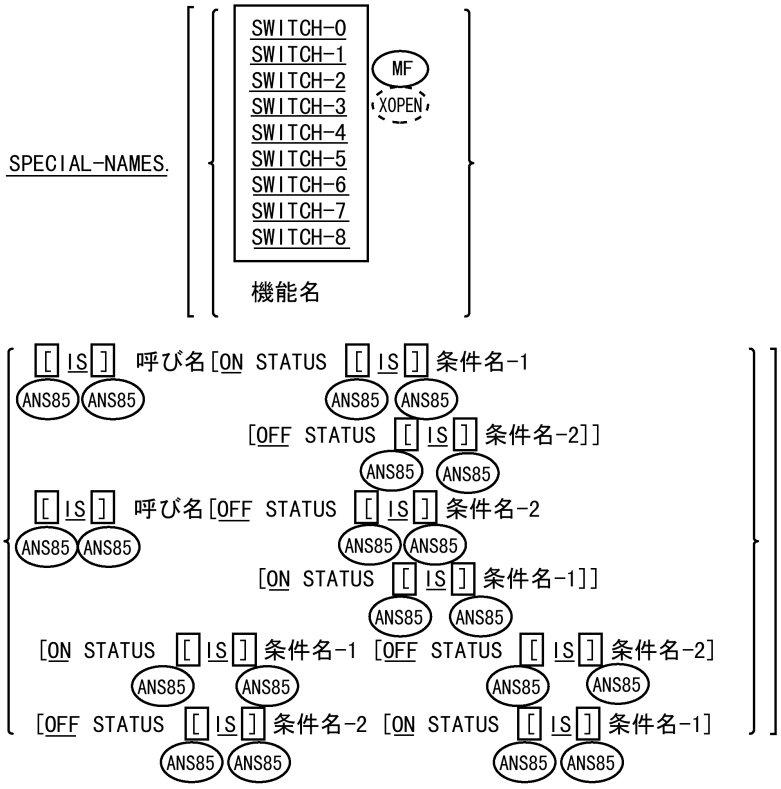
機能

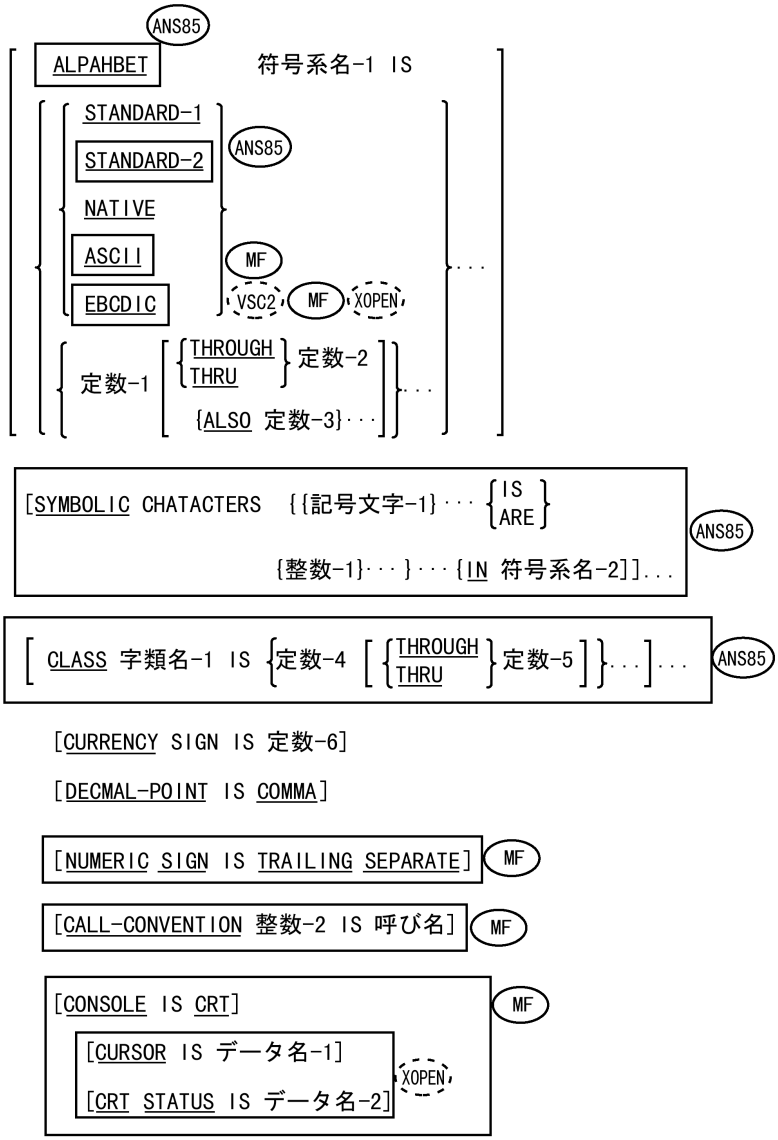
特殊名（special-names）段落は、通貨記号や小数点、
(ANS85)記号文字、
利用者が指定する呼び名に対する作成者語、文字集合や文字の大小順序に対する符号系名、文字の組に対する字類名などの、指定または選択を行う手段を示す。
(MF)パラメータの引渡し方について予め定義してあるいくつかの方式の中から、1つを選択できる。 CALL（呼ぶ）文およびPROCEDURE DIVISIONの見出しの拡張機能によって、これらのパラメータ引渡し方式をプログラム間連絡に使用できるようになった。

例:

1. CRT状態キーの使用方法については、***言語リファレンス - 追加トピック***中の*例*の章を参照。

一般形式





指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - ALPHASTART - ALPHABET句の中で使用する有効な数値定数が０と１のどちらで始まるかを指定する。
    - CHARSET - どのアルファベットを固有文字集合とみなすかに影響する。環境によっては、この指令は無視される。
    - NATIVE - どのアルファベットを固有文字集合とみなすかに影響する。
    - SYMBSTART - SYMBOLIC CHARACTERS句の中で使用する有効な数値定数が０と１のどちらで始まるかを指定する。

構文規則

1. (OSVS)(VSC2)(MF)特殊名段落中の句は、どのような順序で書いても構わない。
2. (ISO2000)クラス定義の外側のレベルでは、CURRENCY，CURSOR，CRT STATUSの各句を指定してはならない。
3. (ISO2000)クラス定義のファクトリまたはオブジェクトにおいては、指定できる句はCURRENCY，CURSOR，CRT STATUSだけである。CURRENCY句のデータ名-1を指定してはならない。
4. (ISO2000)インタフェース定義中に特殊名段落を指定した場合、使用できる句はALPHABET，CURRENCY，DECIMAL-POINTだけである。
5. 呼び名には、任意のCOBOLの利用者語を当てることができる。最低限1文字は英字とする。
6. 機能名は、COBOLコンパイラによって使用されるシステム装置または機能を指す。
   (OSVS)(VSC2)(MF)機能名がUPSI-0からUPSI-7のどれかである場合、機能名は 外部スイッチ を指す。
7. 機能名が 外部スイッチ を参照するか、
   (MF)(XOPEN)またはSWITCH-*n*オプションか SWITCH オプションを使用した場合、
   その呼び名は指定できない。
   (MF)(XOPEN)SET文の中にだけ指定できる。
   その呼び名には、条件名を最低限1つ付ける。
   (ANS85)条件名を付ける必要はない。
8. 外部スイッチを参照しない関数名に対応する呼び名を指定できるのは、ACCEPT（受け取り）、DISPLAY（表示）、SEND（送信）、WRITE（書き出し）の各文だけである。このような作成者語に条件名を関係付けることはできない。
9. 符号系名-1句の定数句の中に指定する定数は、下記のようにする。
   a. 数字である場合は、符号の付かない整数とし、その値は1以上で、計算機に固有の文字集合の文字の個数以下とする。
      ALPHABET（符号系名）句の中の数字定数の有効な最小値は、 ALPHASTART指令 の影響を受ける。
   b. 文字であり、THROUGH（から）指定またはALSO（もまた）指定を指定した場合、すべて長さは1文字とする。
   c. (VSC2)(MF)浮動小数点数または2バイト文字は指定できない。
10. 符号系名-1句の定数句を指定する場合、1つの符号系名の中に同じ文字を2回以上書いてはならない。
11. THRUとTHROUGHは同義語であり、どちらを書いてもよい。
12. (MF)STANDARD-D1とASCIIは同義語であり、どちらを書いてもよい。
13. (ANS85)特殊名段落の中では、予約語のISはまったく必要ない。
14. (ANS85)定数-4句の中に指定する定数は、下記のようにする。
   a. 数字である場合は、符号の付かない整数とし、その値は1以上で、計算機に固有の文字集合の文字の個数以下とする。
   b. 文字でありTHROUGH（から）指定を指定した場合、すべて長さは1文字とする。
   c. (VSC2)(MF)浮動小数点数または2バイト文字は指定できない。

15. (ANS85)定数-1から定数-5には、記号文字の表意定数を指定してはならない。
16. (ANS85)1つの SYMBOLIC CHARACTERS （記号文字）句の中には、同じ記号文字-1は1回だけ使用できる。
17. (ANS85)各記号文字-1とそれに対応する各整数-1とは、SYMBOLIC CHARACTERS句の中の位置によって関係付けられる。最初の記号文字-1が最初の整数-1と対にされ、2番目の記号文字-1が2番目の整数-1と対にされるという具合いに、SYMBOLIC CHARACTERS句全体を通じて対応付けが行われる。ISまたはAREを間に置くのは読みやすくするためだけである。
    整数-1の有効な最小値は、 SYMBSTART指令 の影響を受ける。
18. (ANS85)記号文字-1に指定する一連の文字と整数-1に指定する一連の数字とは、1対1で対応させる。このことは、各IS指定またはARE指定の内部でも、SYMBOLIC CHARACTERS句全体でも、当てはまる。
19. (ANS85)整数-1で指定する文字位置は、計算機固有の文字集合の中に存在させる。 IN指定を書く場合は、符号系名-2で指定する文字集合の中に、整数-1で指定する文字位置を存在させる。符号系名-2は、ALPHABET（符号系）句に指定してあるものとする。
20. 定数-6には、表意定数は使用できない。
21. (MF)整数-2は、0 から 65535 までの符号の付かない整数とする。
22. (MF)(XOPEN)CURSOR IS（カーソルは）句のデータ名-1は、作業場所節に定義しておく。 .
23. (MF)(XOPEN)CRT STATUS（CRT状態）句は、書き方4
    (MF)または5の
    (MF)(XOPEN)各ACCEPT文の後で状態値が転記されるデータ項目を指定する。環境によっては、これは作業場所節の最初の64Kにだけ定義できる。
24. (MF)(XOPEN)データ名-2は、作業場所節に定義しておく。長さは3バイトとする。
25. (MF)(XOPEN)CURSOR IS句は、ACCEPT文によって使用されるカーソルの番地を保持するデータ項目を指定する。

一般規則

1. 外部スイッチは、 実行時に操作員によって設定される。その設定は、 関連する条件名を検査することで判定される。
2. 呼び名が外部スイッチと関係付けられている場合、書き方1のSET（設定）文を実行することによって、その外部スイッチの状態を変更できる。(後述の*SET文*の節を参照。)
3. 符号系名-1句は、文字符号系（character code set）または文字の大小順序に名前を付ける働きをする。PROGRAM COLLATING SEQUENCE句(前述の*実行用計算機段落*を参照)、またはSORT 文かMERGE文(*SORT*文および*MERGE*文を参照)の中のCOLLATING SEQUENCE句の中で符号系名-1が参照される場合、符号系名-1は文字の大小順序を示すことになる。ファイル記述項(*ファイル記述項の全体的な骨組み*を参照) のCODE-SET（符号系）句の中で符号系名-1が参照される場合、符号系名-1は文字符号系を示すことになる。
    a. STANDARD-1句または
       (MF)ASCII句
       を指定すると、文字符号系または文字の大小順序は、米国標準規格X3.4-1968に規定されている情報交換用米国標準コードとなる。

(ANS85)STANDARD-2句を指定すると、文字符号系は、国際標準646の情報処理交換用7ビット・コード化文字集合に規定されている、ISO7ビット・コードの国際参照版となる。
(VSC2)(MF)(XOPEN)EBCDIC句を指定すると、文字符号系および文字の大小順序はEBCDICとなる。
NATIVE（計算機固有）句を指定すると、計算機に固有の文字符号系または文字の大小順序が使用される。計算機に固有の文字の大小順序は、 NATIVE指令 を使用して、ASCIIまたはEBCDICのどちらかに設定できる。
ASCII符号系およびASCIIとEBCDICの文字の大小順序と両者の対応関係については、*文字符号系*と*文字の大小順序*の章を参照。

b. 定数句を指定すると、符号系名-1はCODE-SET句の中では参照できない。(*CODE-SET句*の節を参照。）文字符号系または文字の大小順序は、下記の規則に従って定義される。
    1. 各定数の値は、下記のことを表わす。
        a. 定数が数字であるならば、固有文字集合内の文字の順序番号。この値は、固有文字集合内の文字の数を超えてはならない。
        b. 定数が文字であるならば、固有文字集合内の実際の文字。文字定数の値が複数の文字から構成される場合、その文字定数内の各文字に、左端の文字から開始して連続した昇順に、文字の大小順序が設定される。
    2. ALPHABET句内に定数が現れる順番によって、文字の大小順序の文字の順序番号を昇順に定める。
    3. 固有の文字の大小順序のうち、この定数句で書かれなかった文字は、書かれたどの文字よりも大きい順序位置に置かれる。書かれていない文字相互間の大小順序は、固有の文字の大小順序に従う。
    4. 文字符号系を指定する際に、固有の文字集合の中の定数-1句に指定されなかった各文字に関しては、作成者が指定する文字集合内の順序番号を定義する。
    5. THROUGH指定を書くと、固有文字集合の定数-1の値によって指定される文字から定数-2の値によって指定される文字までの一連の文字が、指定される文字の大小順序で昇順に連続した位置を割り当てられる。THROUGH指定を書く際には、一連の文字を昇順に指定しても降順に指定してもよい。
    6. ALSO指定を書くと、定数-1と定数-3の値によって指定される固有文字集合の文字は、文字の大小順序
    (ANS85)またはデータを表わすために使用される文字符号系の中で、同じ順序を割り当てられる。

4. プログラムの文字の大小順序で順序位置が最大のものが、表意定数のHIGH-VALUEとなる。 ただし、表意定数HIGH-VALUEを
(ANS85)特殊名段落の中で定数として指定した場合は例外である。

プログラムの文字の大小順序で順序位置が最大のものが2つ以上ある場合は、最後に指定した方が表意定数のHIGH-VALUEとなる。

5. プログラムの文字の大小順序で順序位置が最小のものが、表意定数のLOW-VALUEとなる。 ただし、表意定数LOW-VALUEを
   (ANS85)特殊名段落の中で定数として指定した場合は例外である。
   プログラムの文字の大小順序で順序位置が最小のものが2つ以上ある場合は、最後に指定した方が表意定数のLOW-VALUEとなる。
6. (OSVS)(VSC2)(MF)浮動小数点数定数は、利用者が定義した文字の大小順序の中では使用できない。
7. (ANS85)特殊名段落の中で定数として指定した場合、表意定数のHIGH-VALUEと LOW-VALUE はそれぞれ、固有の文字の大小順序の中の順序位置が最大のものと最小のものとなる。
8. (ANS85)SYMBOLIC CHARACTERS句は、表意定数のように使用できる利用者語を定義するために使用する。IN指定を指定しないと、記号文字-1は固有の文字集合の中で順序位置が整数-1によって指定される文字を表わす。IN指定を書くと、整数-1は符号系名-2によって指定される文字集合の中の文字の順序位置を表わす。
9. (ANS85)記号文字-1の内部表現は、固有の文字集合の中の該当する文字の内部表現となる。
10. CURRENCY SIGN IS句に指定する定数-6は、PICTURE句において通貨記号を表わすために使用するものである。これに指定する文字は1文字とする。ただし、下記の文字は指定できない。
    - 数字の0から9
    - 大文字の英字A, B, C, D, L, P, R, S, V, X, Z, または空白文字
      (MF)CとRを使用してもよい。
      (ANS85)Lを使用してもよい。
      (OSVS)(VSC2)(MF)Eは使用できない。
      (MF)(COB370)N は使用できない。
      (VSC2)(MF)G は使用できない。
      (ANS85)小文字の英字aからzまで
      (MF)(e, f, g, h, i, j, k, m, n, o, q, t, u, w, yを除く)
    - 特殊文字の *, +, -, ,, ., ;, (,), ", / or, =

    この句を指定しないと、COBOLの文字集合に定義されている通貨記号だけが、PICTURE句の中で使用できる。 詳細については、*COBOL言語の概念*の章の*文字集合*の節を参照。
11. DECIMAL POINT IS COMMA句は、PICTURE句の文字列と数字定数においてはコンマ（,）と終止符（.）の機能を逆転させることを意味する。
12. (MF)NUMERIC SIGN句を指定すると、符号付き数字項目の符号の位置を個々に指定しなかった場合、末尾に独立の符号が付けられる。
13. (OSVS)(VSC2)(MF)呼び名を使用できるところでは、代わりに機能名を使用できる。
14. 機能名に外部スイッチを指定しない場合、下記のものの中から選ぶことができる。

(MF)TAB	縦のタブ位置へ飛ぶ（必要に応じて WRITE ADVANCING文出力レコード中にASCII X"0B"を挿入)	
(MF)PRINTER	プリンタ	DISPLAY文

(MF)FORMFEED	改ページする（必要に応じて出力レコード中にASCII X"0C" を挿入）	WRITE ADVANCING文
(MF)(XOPEN)COMMAND-LINE	コマンド転送	ACCEPT文とDISPLAY文
(MF)(XOPEN)ARGUMENT-NUMBER	コマンド行の引数番号	ACCEPT文とDISPLAY文
(MF)(XOPEN)ARGUMENT-VALUE	コマンド行の引数値	ACCEPT文
(MF)(XOPEN)ENVIRONMENT-NAME	環境変数名	DISPLAY文
(MF)(XOPEN)ENVIRONMENT-VALUE	環境変数値	ACCEPT文とDISPLAY文
(MF)(XOPEN)SYSERR	標準エラー装置	DISPLAY文

(OSVS)(VSC2)このCOBOLシステムに組み込まれている OS/VS COBOL,VS COBOLIIおよびSAAにおいて使用できる機能名を下の表6-1に示す。

表6-1: 使用できる機能名

	OSVS	VSC2 Rel (2)	COBOL/370 または VSC2 Rel (3)/(4)	SAA L1
SYSIN	y	y	y	y
SYSIPT	U	U	y	
SYSOUT	y	y	y	y
SYSLIST			y	
SYSLST	U	U	y	
SYSPCH	U	U	y	
SYSPUNCH	U	y	y	
CONSOLE	y	y	y	y
C01	y	y	y	y
C02	y	y	y	
C03	y	y	y	
C04	y	y	y	
C05	y	y	y	
C06	y	y	y	
C07	y	y	y	
C08	y	y	y	
C09	y	y	y	
C10	y	y	y	
C11	y	y	y	
C12	y	y	y	
S01	y	y	y	
S02	y	y	y	
S03		U	y	
S04		U	y	
S05		U	y	
CSP	y	y	y	y
1 文字 rw 定数	y			
AFP-5A			y	

y　メインフレームのコンパイラおよびCOBOLシステムに明記されており、受け付けられる。
U　　明記されていないが、メインフレームのコンパイラに受け付けられる。

15. (ANS85)CLASS句は、指定した文字の組に名前を付ける働きをする。字類名-1は字類条件の中でだけ参照できる。字類名-1には、この句に指定する定数の値によって表わされる文字の組だけが含まれる。各定数の値は下記の内容を表わす。
    a. 定数が数字の場合、固有の文字集合の中の文字の順序番号。 この値は、固有の文字集合の文字の数を表わす値を超えてはならない。

b. 定数が文字の場合、固有の文字集合の中の実際の文字。 文字定数が複数の文字から構成される場合、その文字定数の各文字が字類名-1によって表わされる文字の組に含まれる。

(ANS85)THROUGH指定を書くと、固有文字集合における定数-4から定数-5までのすべての文字が字類名-1によって表わされる文字の組に含まれる。THROUGH指定に書く一連の文字は、固有文字集合の中で昇順であっても降順であってもよい。

16. 原始要素の特殊名段落の中に指定したすべての句は、その原始要素 に含まれる原始要素にも適用される。含む原始要素の特殊名段落の中に指定した条件名は、含まれるどの原始要素からも参照できる。
17. (MF)CALL-CONVERSION句を使用すると、パラメータ受渡しに関して予め定義されているいくつかの方式の中から1つを選択できる。これによって、COBOL以外の言語で書かれ、パラメータ受渡し方式の異なる副プログラムを呼び出せる。
18. (MF)CONSOLE IS CRT句を指定すると、ACCEPT文またはDISPLAY文のうち、作用対象が画面でなく特定の書き方に固有の句を含まないものが、書き方5として扱われるようになる。CONSOLE IS CRT句を指定しないと、それらの文は標準のANSI ACCEPT文またはDISPLAY文として扱われる。
19. (MF)(XOPEN)CURSOR IS句は、ACCEPT文によって使用され、カーソルの番地を保持するデータ項目を指定する。
    a. ACCEPT文の開始時に、データ名-1に画面上の文字位置として有効な値が入っていると、その文字位置がカーソルの初期位置として使用される。データ名-1に有効な値が入っていない場合は、データ名-1は無視されて、画面上の最初の入力項目の先頭がカーソルの初期位置となる。データ名-1に入っている文字位置をACCEPT文の中で使用すると、ACCEPT文の終了時にデータ名-1の値が更新される。その結果、ACCEPT文が終了した時点でのカーソルの位置がデータ名-1に入れられる。
    b. CURSOR IS句は、画面上の項目への位置付けには効力をもたない。
    c. データ名-1の長さは4文字または6文字とする。データ名-1の長さが4文字である場合、上2文字が行番号として解釈され、下2文字がカラム番号として解釈される。データ名-1の長さが6文字である場合、上3文字が行番号として解釈され、下3文字がカラム番号として解釈される。
    d. データ名-1に入っている値が有効でない（たとえば、ゼロや文字や画面の下辺から外れる値）場合は、この句は効力をもたない。
    e. データ名-1に含まれている値が有効な文字位置ではあるが、実行されようとしているACCEPT文の入力項目に対応しない場合、カーソルは次の入力項目に位置付けられる。このとき、次の入力項目が存在しないと、最初の入力項目に位置付けられる。入力項目の順番はデータ部にデータ記述を書いた順番である。
20. (MF)(XOPEN)CRT STATUS句は、書き方4または書き方5のACCEPT文が実行されるたびに、状態値が転記されるデータ項目を指定する。

(MF)(XOPEN)特殊名段落にCRT STATUS句を指定すると、書き方4または書き方5のACCEPT文が実行されるたびに（詳細については後述する）、その結果を示す値がデータ名-2に入れられる。データ名-2は 状態キーから構成される。そのそれぞれに、ACCEPT文を実行した結果の状態が記録される。各状態キーの意味を以下に説明する。

CRT状態キー

(MF)(XOPEN)

状態キー1: データ名-1の最初のバイトは、CRT状態キー1である。このデータ記述には、PICTURE 9 USAGE DISPLAYと指定しておく。この状態キーはACCEPT操作の終了を引き起こした条件を表わす。このキーがとる具体的な値は下記のとおり。

"0"　終了キーまたは最後の項目の外に出る自動スキップを表わす
"1"　利用者が定義したファンクション・キーを表わす
"2"　COBOLシステムで定義されているファンクション・キーを表わす
"9"　エラーが発生したことを表わす

終了キーとは、ACCEPT操作を終わらせるためのキーである（たとえば、実行キー）。ACCEPTの最後の項目でフィールド・タブ・キーを押すと、終了キーとして働くように設定する構成オプションもある。ファンクション・キーを定義することも構成オプションである。
この状態値に"0" が返されて終了することを、正常終了という。
ACCEPT文の中にON EXCEPTION指定を書くと、CRT状態キーの値が" 0" 以外であった場合、指定した無条件命令が実行される。

CRT 状態キー2: データ名-1の2番目のバイトは、状態キー2である。ここには、ACCEPT操作を終了させた条件のさらに詳細を示すコードが記録される。状態キー2の形式と値はCRT状態キー1の値によって変わる。その具体的な値を下の表6-2に示す。

表 6-2: CRT状態キー1と2の有効な組合せ

キー1	キー2		意　味
	形　式	値	
0	PIC 9 DISPLAY	0	操作員が終了キーを押した
0	PIC 9 DISPLAY	1	最後の項目の外に出る自動スキップ
1	PIC 99 COMP	0-127	ファンクション・キー番号
2	PIC 99 COMP	0-26	ファンクション・キー番号
9	PIC 99 COMP	0	画面内に項目が何もない

Sファンクション・キー番号については、COBOLシステムのマニュアル中の画面処理およびユーザインタフェースに関する記述を参照。

CRT 状態キー3: データ名-1の3番目のバイトは、状態キー3である。これはPICTURE 99 COMP-X または PICTURE 99 COMPとして定義する。 (NOIBMCOMP指令を指定して)。 CRT状態キー1とCRT状態キー2がともにゼロであるならば、ACCEPT操作を終わらせたキーそのままのキーボード・コードがCRT状態キー3に記録される。これ以外では、CRT状態キー3の内容はどうなるかわからない。
単一のキーではなく、いくつかのキーを連続して打つことによって1つの機能が実行されるように設定されている場合、最初に打たれたキーに対するコードだけが返される。

6.1.2.4 リポジトリ段落

(ISO2000)
リポジトリ段落では、環境部の範囲内で使用できるクラス名、インタフェース名、属性名を指定できる。

一般形式

```
REPOSITORY.
[ { クラス指定
  { インタフェース指定
  { 属性指定                } .... ]

ここでクラス指定は:
 CLASS クラス名-1[AS定数-1][EXPANDS クラス名-2 USING {クラス名-3
                                                     {インタフェース名-1}]

ここでインタフェース指定は:
 INTERFACE インタフェース名-2[AS 定数-2][EXPANDS インタフェース名-3 USING {クラス名-4
                                                                          {インタフェース名-4}]

ここで属性指定は:
 PROPERTY 属性名-1[AS 定数-3]
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - RDFPATH - リポジトリ・ファイル用の登録集の所在を指定する。

構文規則

1. リポジトリ段落中にクラス名-1、インタフェース名-2、属性名-1のいずれかを２回以上指定する場合、その名前での指定はすべて同じでなければならない。
2. 定数-1、定数-2、定数-3は文字定数でなければならず、表意定数であってはならない。
3. CLASS指定を書く場合、クラス名-2、クラス名-3、インタフェース名-1は下記のどちらかでなければならない。

   a. それを含むクラス定義またはインタフェース定義のパラメータ
   b. クラス名－１が定義されているのと同じリポジトリ段落中に定義されている
4. このリポジトリ段落を含むクラス定義のクラス名段落のUSING指定またはこのリポジトリ段落を含むインタフェース定義のインタフェース名段落のUSING指定の中にクラス名-1を指定した場合、EXPANDS指定を書いてはならない。
5. インタフェース指定を書く場合、クラス名-4、インタフェース名-3、インタフェース名-4は下記のどちらかでなければならない。
   a. それを含むクラス定義またはインタフェース定義のパラメータ
   b. インタフェース名-2が定義されているのと同じリポジトリ段落中に定義されている
6. このリポジトリ段落を含むクラス定義のクラス名段落のUSING指定またはこのリポジトリ段落を含むインタフェース定義のインタフェース名段落のUSING指定の中にインタフェース名-2指定した場合、EXPANDS指定を書いてはならない。
7. 指定したクラス名-1がこのリポジトリ段落を含むクラス定義の名前である場合、クラス名-1を参照するとそのクラス定義を参照することになり、このクラス指定子は無視される。
8. 指定したインタフェース名-2がこのリポジトリ段落を含むインタフェース定義の名前である場合、インタフェース名-2を参照するとそのインタフェース定義を参照することになり、このインタフェース指定子は無視される。
9. クラス指定を書いた場合、
   a. 定数-1を指定したならば、クラスである定数-1に関する情報が外部リポジトリ中に存在しなければならない。
   b. 定数-1を指定しなかったならば、クラスであるクラス名-1に関する情報が外部リポジトリ中に存在しなければならない。
10. インタフェース指定を書いた場合、
   a. 定数-2を指定したならば、インタフェースである定数-2に関する情報が外部リポジトリ中に存在しなければならない。
   b. 定数-2を指定しなかったならば、インタフェースであるインタフェース名-2に関する情報が外部リポジトリ中に存在しなければならない。
11. PROPERTY指定を書いた場合、
   a. 定数-3を指定したならば、属性である定数-3に関する情報が外部リポジトリ中に存在しなければならない。定数-3はこの段洛中に宣言されたクラスまたはインタフェースのどれかの一部である。
   b. 定数-3を指定しなかったならば、属性である属性名-1に関する情報が外部リポジトリ中に存在しなければならない。属性名-1はこの段洛中に宣言されたクラスまたはインタフェースのどれかの一部である。

一般規則

1. クラス名-1はそれが指定されている環境部の範囲内で使用できるクラスの名前である。

2. AS指定を書いた場合、定数-1または定数-2はそれぞれクラスまたはインタフェースが操作環境によって認識される名前である。定数-3は、指名された属性を実現するメソッドとして、操作環境によって認識される名前である。AS指定を書く必要があるのは、クラスやインタフェースや属性の名前が利用者語を形成するための規則に従っていないか、または名前の大文字と小文字を区別する場合である。
3. クラス名-3とインタフェース名-1は、クラス名-2によって参照されるパラメータ化されたクラスのための、実パラメータである。
4. クラス名-4とインタフェース名-4は、インタフェース名-3によって参照されるパラメータ化されたインタフェースのための、実パラメータである。
5. クラス指定子にEXPANDS指定を書いた場合、パラメータ化されているクラスであるクラス名-2から、クラス名-1がクラスとして作成される。クラス指定子のEXPANDS指定のUSING指定の中のパラメータの数は、クラス名-2のクラス名段落のUSING指定の中のパラメータの数と同じであるものとする。クラス名-1用のインタフェースは、クラス名-2用に指定されたインタフェースのパラメータをクラス指定子に指定されたパラメータで置き換えたものである。

   クラスであるクラス名-1はパラメータ化されたクラスであるクラス名-2から作成される。そのさい、仮パラメータの各仕様が対応する実パラメータで置き換えられる。その置換はCOPY文とREPLACE文の処理が終わった後で行われる。
6. コンパイラはクラス名-1に指定されている情報を外部リポジトリと一緒に使用して、使用するクラスの詳細を決定する。そのクラスに関するリポジトリ情報は、ファイルに格納されていなければならない。そのファイル名はそのクラスを外部化した名前に拡張子rdfを付けたものでなければならない。RDFPATH指令を指定した場合には、そのファイルは指定されたディレクトリのもとに置かれていなければならない。そうでない場合には、.intファイルおよび.idyファイルが作成されるローカル・ディレクトリのもとにそのファイルが置かれていなければならない。
7. インタフェース名-2はそれが指定されている環境部の範囲内で使用できるインタフェースの名前である。
8. インタフェース指定子にEXPANDS指定を書いた場合、パラメータ化されているインタフェースであるインタフェース名-3から、インタフェース名-2がインタフェースとして作成される。インタフェース指定子のEXPANDS指定のUSING指定の中のパラメータの数は、インタフェース名-3のインタフェース名段落のUSING指定の中のパラメータの数と同じであるものとする。インタフェース名-2用のインタフェースは、インタフェース名-3用に指定されたインタフェースのパラメータをインタフェース指定子に指定されたパラメータで置き換えたものである。 <P*f*C*f*“*f*^フェースであるインタフェース名-2はパラメータ化されたインタフェースであるインタフェース名-3から作成される。そのさい、仮パラメータの各仕様が対応する実パラメータで置き換えられる。その置換はCOPY文とREPLACE文の処理が終わった後で行われる。

9. コンパイラはインタフェース名-2に指定されている情報を外部リポジトリと一緒に使用して、使用するインタフェースの詳細を決定する。そのクラスに関するリポジトリ情報は、ファイルに格納されていなければならない。そのファイル名はそのインタフェースを外部化した名前に拡張子rdfを付けたものでなければならない。RDFPATH指令を指定した場合には、そのファイルは指定されたディレクトリのもとに置かれていなければならない。そうでない場合には、.intファイルおよび.idyファイルが作成されるローカル・ディレクトリのもとにそのファイルが置かれていなければならない。
10. 属性名-1はそれが指定されている環境部の範囲内で使用できるオブジェクト属性の名前である。

6.1.3 入出力節

機能

入出力節では、外部媒体とランタイム要素との間での、データの転送と処理を制御するために必要な情報を取り扱う。

一般形式

```
INPUT-OUTPUT SECTION.
[ ファイル管理段落 ]
[ 入出力管理段落 ]
```

構文規則

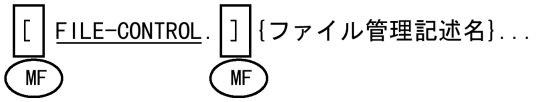
1. (ISO2000)入出力節はプログラム定義中に指定できる。クラス定義においては、入出力節を指定できる箇所は、ファクトリ定義、オブジェクト定義、メソッド定義の中だけである。インタフェース定義中に入出力節を指定してはならない。

ファイル管理段落

機能

ファイル管理段落では、各ファイルに名前を付けるとともに、ファイルに関するその他の情報を指定する。

一般形式



6.1.3.1 ファイル管理記述項

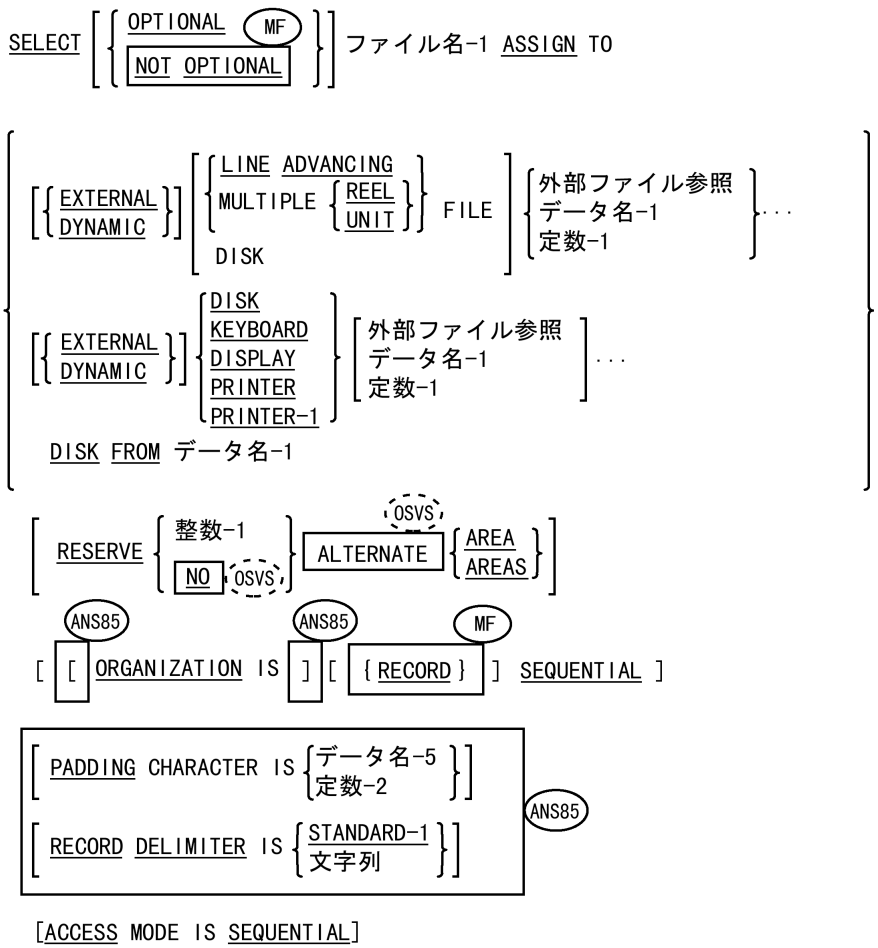
機能

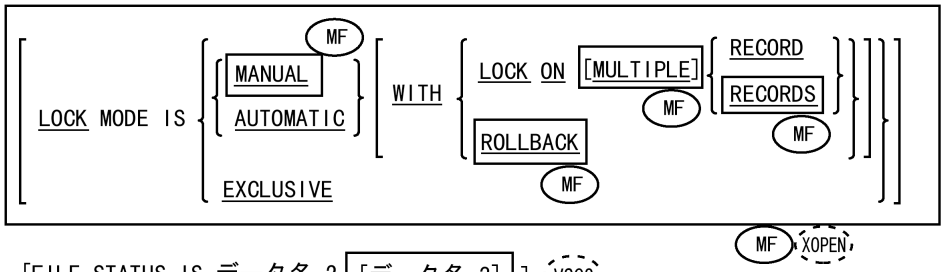
The ファイル管理記述項では、ファイルに名前を付けるとともに、ファイルに関するその他の情報を指定する。

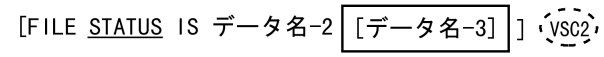
(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、ASSIGN句中のRECORD DELIMITER指定とRESERVE指定と反復記号は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内では、これらの指定および記号を使用するべきではない。

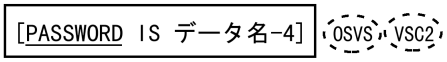
一般形式

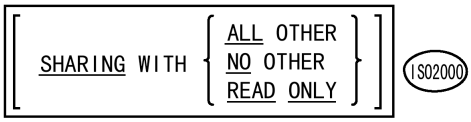
書き方 1 (レコード順ファイル)



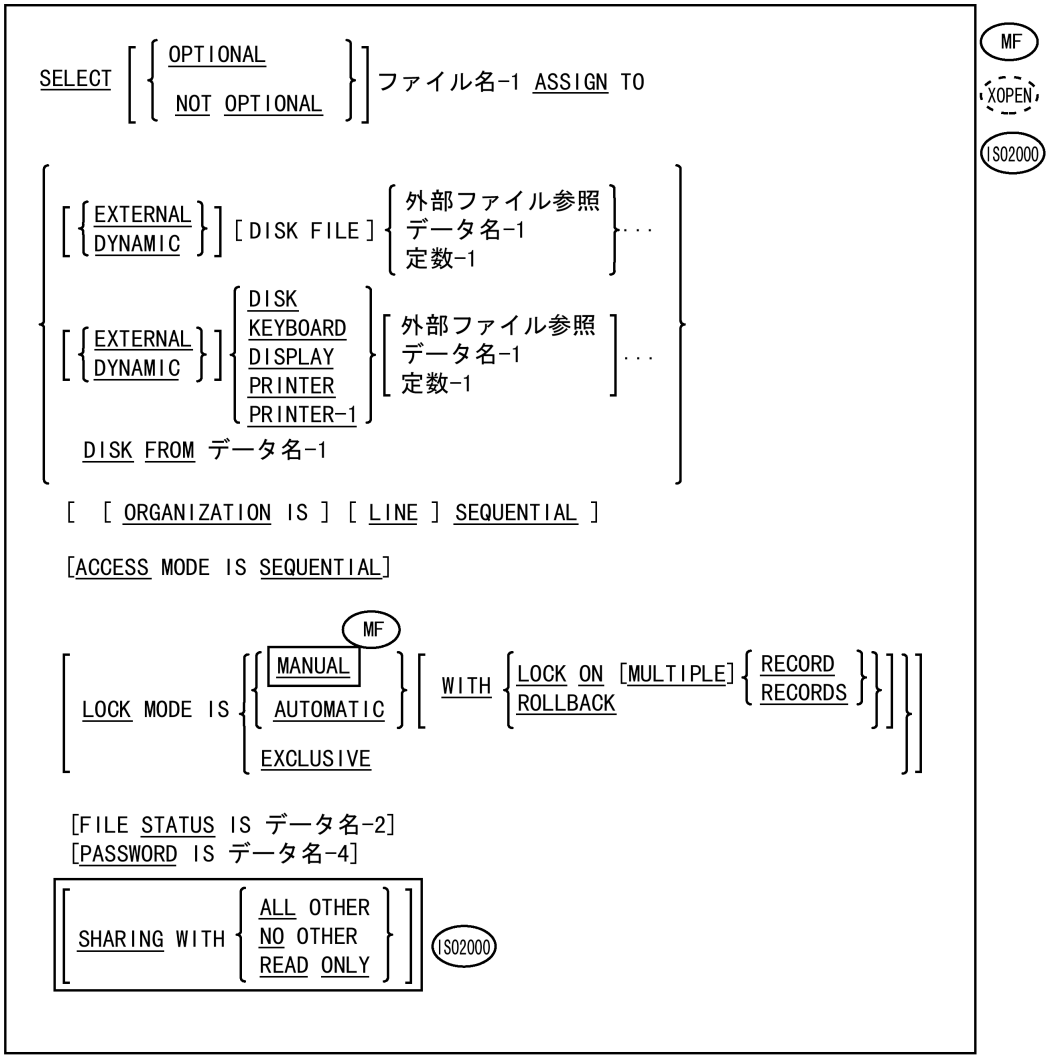

LOCK MODE IS
MANUAL
AUTOMATIC
EXCLUSIVE
MF
WITH
LOCK ON
[MULTIPLE]
RECORD
RECORDS
ROLLBACK
MF
MF
MF
XOPEN


[FILE STATUS IS データ名-2 [データ名-3]]
VSC2

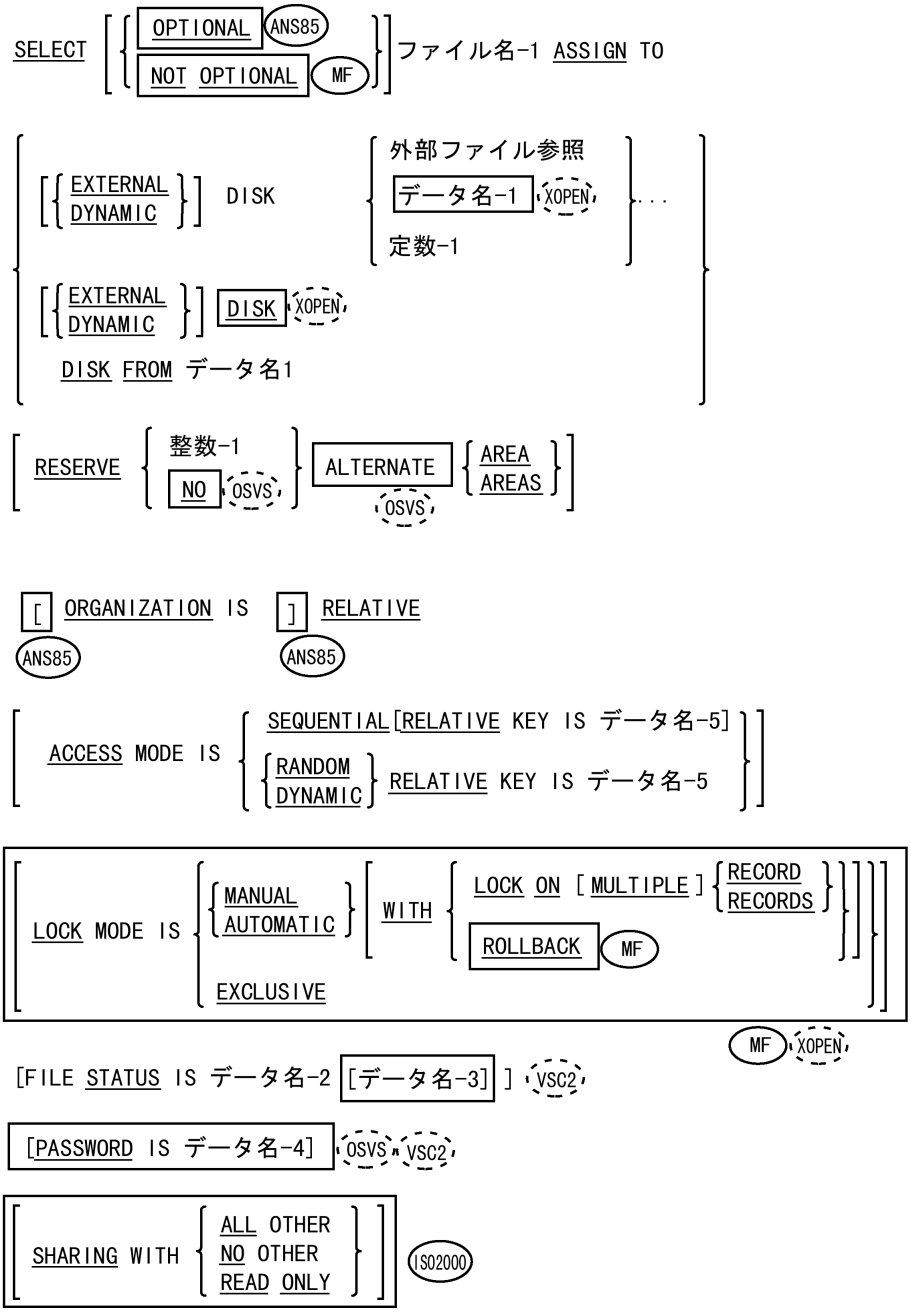

[PASSWORD IS データ名-4]
OSVS
VSC2


SHARING WITH
ALL OTHER
NO OTHER
READ ONLY
ISO2000

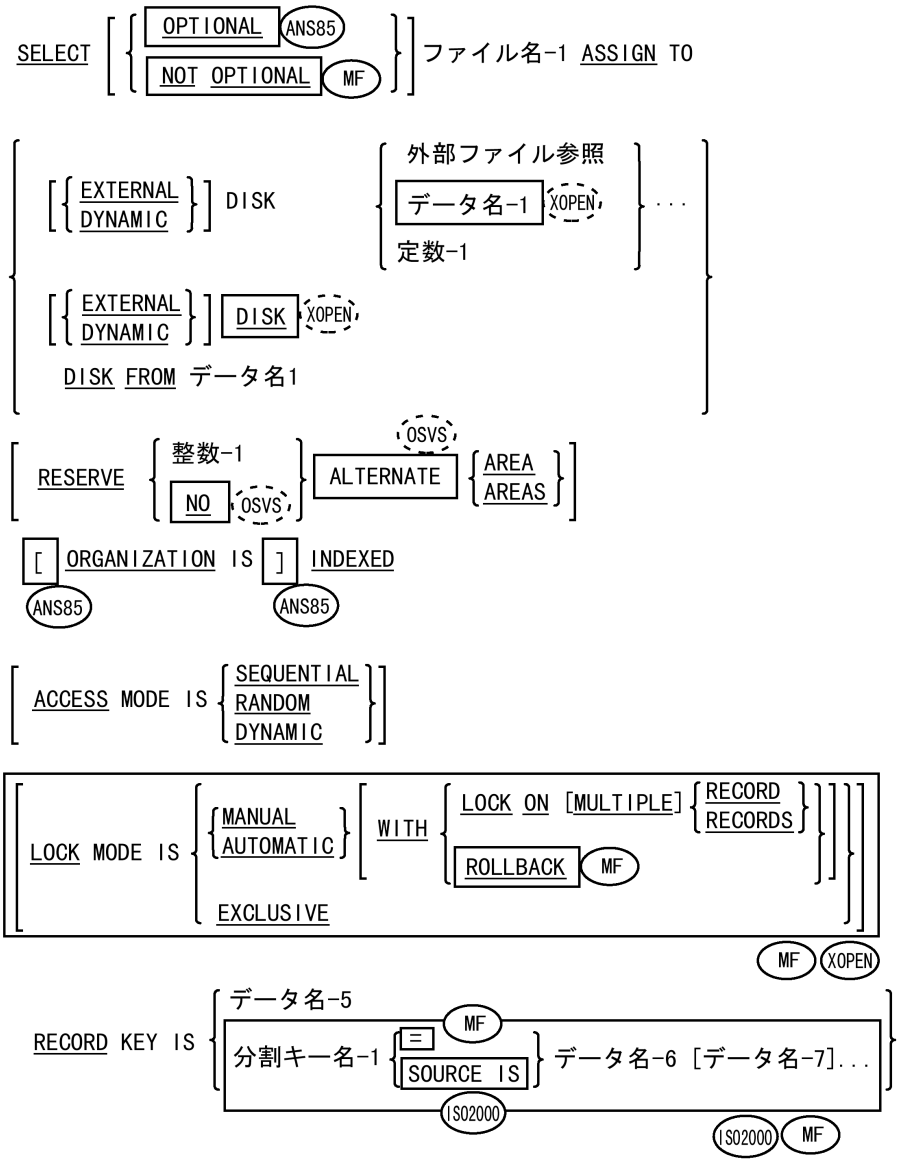
書き方 2（行順ファイル）

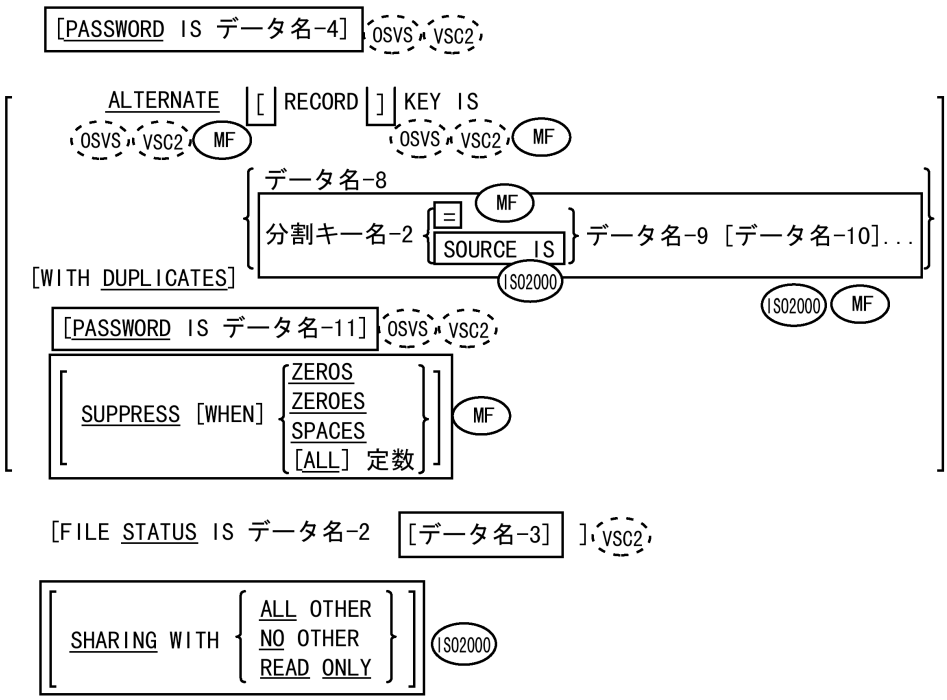

MF
XOPEN
ISO2000
SELECT
OPTIONAL
NOT OPTIONAL
ファイル名-1 ASSIGN TO
EXTERNAL
DYNAMIC
DISK FILE
外部ファイル参照
データ名-1
定数-1
DISK
KEYBOARD
DISPLAY
PRINTER
PRINTER-1
DISK FROM データ名-1
ORGANIZATION IS
LINE
SEQUENTIAL
ACCESS MODE IS SEQUENTIAL
LOCK MODE IS
MANUAL
AUTOMATIC
EXCLUSIVE
WITH
LOCK ON
MULTIPLE
RECORD
RECORDS
ROLLBACK
FILE STATUS IS データ名-2
PASSWORD IS データ名-4
SHARING WITH
ALL OTHER
NO OTHER
READ ONLY
ISO2000

書き方 3 (相対ファイル)

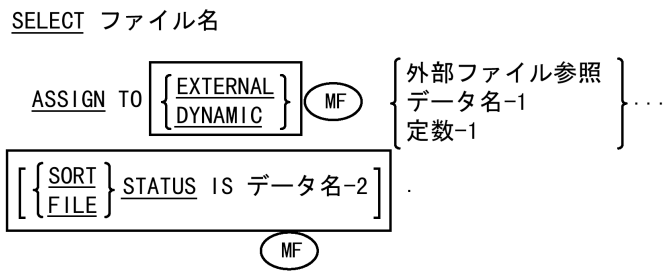

SELECT
OPTIONAL
ANS85
NOT OPTIONAL
MF
ファイル名-1 ASSIGN TO
EXTERNAL
DYNAMIC
DISK
外部ファイル参照
データ名-1
XOPEN
定数-1
DISK
XOPEN
DISK FROM データ名1
RESERVE
整数-1
NO
OSVS
ALTERNATE
OSVS
AREA
AREAS
ORGANIZATION IS
RELATIVE
ANS85
ANS85
ACCESS MODE IS
SEQUENTIAL
RELATIVE KEY IS データ名-5
RANDOM
DYNAMIC
RELATIVE KEY IS データ名-5
LOCK MODE IS
MANUAL
AUTOMATIC
EXCLUSIVE
WITH
LOCK ON
MULTIPLE
RECORD
RECORDS
ROLLBACK
MF
MF
XOPEN
FILE STATUS IS データ名-2
データ名-3
VSC2
PASSWORD IS データ名-4
OSVS
VSC2
SHARING WITH
ALL OTHER
NO OTHER
READ ONLY
ISO2000

書き方 4 (索引ファイル)


SELECT
OPTIONAL
ANS85
NOT OPTIONAL
MF
ファイル名-1 ASSIGN TO
EXTERNAL
DYNAMIC
DISK
外部ファイル参照
データ名-1
XOPEN
定数-1
DISK FROM データ名1
RESERVE
整数-1
NO
OSVS
ALTERNATE
AREA
AREAS
ORGANIZATION IS
INDEXED
ACCESS MODE IS
SEQUENTIAL
RANDOM
DYNAMIC
LOCK MODE IS
MANUAL
AUTOMATIC
EXCLUSIVE
WITH
LOCK ON [MULTIPLE]
RECORD
RECORDS
ROLLBACK
RECORD KEY IS
データ名-5
分割キー名-1
SOURCE IS
データ名-6 [データ名-7]...
ISO2000



書き方 5 (整列併合ファイル)



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - ASSIGN - 定数、データ名、 または外部ファイル参照を評価する方法を決定する。
    - SEQUENTIAL - ORGANIZATION IS SEQUENTIALとして定義されているファイルがLINE SEQUENTIALまたはRECORD SEQUENTIALのどちらであるかを判定する。

構文規則

すべての書き方(すべてのファイル)

1. ファイル管理記述項の冒頭には、 SELECT句を指定する。SELECT句の後ろに続く他の句は、どんな順序で書いてもよい。
2. データ部に記述する各ファイルには、ファイル管理段落において一度だけファイル名を付ける。ファイル管理記述項に指定した各ファイルに対応して、同じファクトリ、メソッド、オブジェクト、またはプログラムのデータ部のファイル記述項を書かなければならない。
3. ACCESS MODE句を指定しないと、暗黙的にACCESS MODE IS SEQUENTIALを指定したものとみなされる。
4. (MF)データ名-1は、データ部の中で英数字データ項目または集団項目として宣言できる。この場合、そのファイルの外部名を保持できるだけの十分な長さをとる必要がある。 データ名-1を原始要素の中で明示的に宣言しないと、COBOLシステムが暗黙的に英数字データ項目として宣言する。その長さは、ファイル名の長さとして許される最大の長さがとられる。FROM指定の中にデータ名-1を書く場合は、データ部の中で明示的に宣言しておかなければならない。
5. (MF)NOT OPTIONAL指定は、入出力両用モードでファイルを開く場合にだけ効力をもつ。
6. (MF)外部ファイル参照、データ名-1、定数-1はファイルの外部名を指定する。
   (MF)EXTERNAL指定またはDYNAMIC指定を書く場合、定数-1を指定してはならない。
   (MF)DYNAMIC指定を書く場合、外部ファイル名に指定した語は外部ファイル参照として解釈される。
7. データ名-2は、データ部の中で2文字の英数字データ項目
   (OSVS)(VSC2)(MF)または用途が表示用の2文字の数字データ項目
   として定義しておく。このデータ項目は、ファイル節
   (MF)または局所記憶節
   または通信節の中では定義できない。
8. (VSC2)データ名-3は、作業場所節または連絡節の中で6バイトの集団項目として定義しておく。
9. (OSVS)(VSC2)データ名-4は、作業場所節の中で英数字データ項目として定義しておく。
10. すべてのデータ名は、修飾できる。

書き方1 (レコード順ファイルだけ)

11. (ANS85)定数-2は、1文字の文字定数とする。
12. (ANS85)データ名-5は修飾してもよい。これはデータ部の中で、1文字の英数字データ項目として定義しておく。通信節またはファイル節または報告書節の中では定義できない。
13. (ANS85)文字列は、予約語または利用者名または定数であってはならない。

書き方1および2（レコード順ファイルおよび行順ファイル）

14. ORGANIZATION句を指定しないと、順編成であるものと想定される。
15. (MF)KEYBOARDは、操作卓入力を意味する。
16. (MF)DISPLAYは、操作卓出力を意味する。
17. (MF)PRINTERはシステムの主たる印刷装置を指す。
18. (MF)PRINTER-1はシステムの2番目の印刷装置を指す。

書き方3（相対ファイル）

19. START文の作用対象に相対ファイルを指定する場合は、そのファイルに関するRELATIVE KEYを指定しておく。
20. データ名-5は、該当するファイル名のレコード記述項の中では指定できない。
21. データ名-5は、符号なしの整数データ項目として定義する。

書き方4（索引ファイル）

22. (MF)分割キー名は、該当するファイル名のレコード内のいくつかのデータ項目をつなげたものである。これはSTART文またはREAD文の中でだけ使用できる。
23. データ名-5とデータ名-8のデータ項目
    (MF)および分割キー名-1と分割キー名-2によって参照されるデータ名
    は、該当するファイルのレコード記述項の中で定義しておく。
    (OSVS)(VSC2)(MF)これらのデータ項目は、どの項類のデータとして定義してもよい。しかし、SELECT句内で指定したファイル > に対する入出力文に関しては、キーは必ず英数字項目として扱われる。
24. データ名-5とデータ名-8のデータ項目
    (MF)および分割キー名-1と分割キー名-2によって参照されるデータ名
    に可変長項目を指定することはできない。詳細は*OCCURS句*の節を参照。
25. データ名-8の左端の文字位置がデータ名-5の左端の文字位置と合致してはならない。
26. (OSVS)(VSC2)PASSWORD句を指定する場合、対応するRECORDKEY句または ALTERNATE RECORD KEY句の直後に書く。

書き方 5（整列併合ファイル）

27. データ部に記述する各整列ファイルまたは併合ファイルには、ファイル管理段落で一度だけファイル名を付ける。ファイル管理記述項に記述した各整列ファイルまたは併合ファイルに関して、整列併合ファイル記述項を記述する。
28. ファイル名は整列ファイルまたは併合ファイルを表わすので、 ファイル管理段落中のファイル名の後ろにはASSIGN句だけを記述することができる。

一般規則

すべての書き方(すべてのファイル)

1. ASSIGN句は、ファイル名-1が表わすファイルを記憶媒体に対応付ける。最初に記述したファイル割当てだけが効力をもつ。
   (OSVS)(VSC2)(MF)1つのASSIGN句内にファイル割当てをいくつか記述した場合、最初のもの以外は注記になる。
2. RESERVE句を使用すると、利用者は必要な入出力領域の数を指定できる。
   (ANS85)使用しているオペレーティングシステムに付随するマニュアルに別途指定されていないかぎり、RESERVE句は注記になる。
3. ORGANIZATION句は、ファイルの論理構造を指定する。ファイル編成はファイルが作成されるときに確立され、後から変更することはできない。
4. FILE STATUS句を指定すると、明示的または暗黙的に該当ファイルを対象とする文が実行されるたびに、ランタイムシステムからデータ名-2のデータ項目に値が返される。この値はその文を実行した結果の状態を示している。(*入出力状態*の節を参照。)
   (MF)データ名-3は、指定しても注記になる。
5. (MF)PASSWORD句は注記になる。
6. (MF)ASSIGN句内の予約語DYNAMICは、定数-1またはデータ名-1の値が外部ファイル記憶環境におけるファイル名であることを示す。
7. (MF)ASSIGN句内の予約語EXTERNALは、指定したファイルが外部環境においては外部ファイル参照によって識別されることを示す。それを通じて、指定したファイルがさらに外部ファイル記憶環境名に対応付けられることもある。(ある操作環境での外部ファイル名の設定については、ファイル処理に関するCOBOLのマニュアルを参照。)
   (MF)外部ファイル参照中に文字"-" が含まれていると、最後の"-" の後ろに続く部分の名前だけが、外部環境においてファイルを識別するために使用される。
8. (MF)外部ファイル参照かデータ名-1か定数-1を指定せずにDISKを指定するか、FROMを指定せずにDISKを指定するかすると、そのファイルはディスク・ファイルでその名前はファイル記述中の VALUE OF FILE-ID に指定されることを表わす。このファイル記述中にVALUE OF FILE-ID句が含まれていない場合は、そのディスク・ファイルの名前は、ファイル名-1 (*内部ファイル名*) と同じであると想定される。
9. (MF)必要語DISK, KEYBOARD, DISPLAY, PRINTER, PRINTER-1のどれかの後ろに外部ファイル参照かデータ名-1か定数-1が続く場合、その必要語は無視される。
10. (MF)DISK FROMを指定すると、そのファイルはディスク・ファイルでありディスク上のファイル名はデータ名-1の値であることを表わす。しかし、そのファイルに対してOPEN文を実行すると、データ名-1はすべて空白とされて、そのディスク・ファイルの名前はファイル名-1（内部ファイル名）と同じであると想定される。
11. OPTIONAL指定は、入力モードか入出力両用モードか拡張モードで開くファイルに対してだけ適用できる。実行時要素を実際に実行するときに存在するとは限らないファイルに対して、これを指定する必要がある。

12. (ISO2000)(MF)SHARING句では、ファイルに適用する共有モードを指定する。ただし、この句はOPEN文のSHARING句によって上書きされることがある。この句ではまた、レコード・ロックが有効であるか否かを指定する。更に詳細は 共有モード中に指定する。

書き方1と3と4（レコード順ファイル、相対ファイル、索引ファイル）

13. (MF)LOCK MODE句は該当するファイルに適用するロック方式を指定するものであり、書いても書かなくてもよい。
    (MF)この句を省略した場合、ファイルを開くと、そのファイルは排他モードにおかれる。ただし、入力モードで開いた場合を除く。
    (MF)LOCK MODE IS EXCLUSIVEを指定すると、ファイルを開いた実行単位はそのファイル全体の鍵を取得する。
    (MF)LOCK MODE IS AUTOMATICまたはLOCK MODE IS MANUALを指定すると、開いたファイルは実行単位の間で共有される。ただし、出力モードで開いたファイルおよび拡張モードで開いた索引ファイルと相対ファイルは、常に排他モードにおかれる。
14. (MF)ROLLBACK句を指定すると、更新取消し機能を備えたCOBOLシステムではトランザクションのロギングが行われる。システムでの実現方法については、ファイル処理に関するCOBOLのマニュアルを参照。
    (MF)ROLLBACK句を指定すると、WITH LOCK ON MULTIPLE RECORDSが自動的に発効される。
15. (MF)WITH LOCK ON RECORD句は、ファイル中の1レコードだけをロックすることを指定する。
    (MF)WITH LOCK ON MULTIPLE RECORDS句は、ファイル中の複数のレコードをロックすることを指定する。複数のレコードをロックする必要があるときは、この句を指定しなければならない。
16. (MF)LOCK MODE IS AUTOMATIC WITH LOCK ON RECORDを指定すると、該当するファイルを対象にしたREAD文を実行したときに、該当する1レコードがロックされる。その後でそのファイルに入出力操作を行うと、そのレコードはロックを解除される。ただし、START文は例外である。
    (MF)LOCK MODE IS AUTOMATIC WITH LOCK ON MULTIPLE RECORDを指定すると、該当するファイルを対象にしたREAD文を実行したときに、該当する1レコードがロックされる。いったんロックされたレコードは、CLOSE, COMMIT, ROLLBACK, UNLOCKのどれかの文を実行するまで、または個々のロックがDELETE文で削除されるまで解除されない。
17. (MF)LOCK MODE IS MANUAL WITH LOCK ON RECORDを指定すると、該当するファイルを対象にしたWITH LOCK指定を伴うREAD文を実行したときにだけ、該当する1レコードがロックされる。その後で、そのファイルに入出力操作を行うと、そのレコードはロックを解除される。ただし、START文は例外である。
    (MF)LOCK MODE IS MANUAL WITH LOCK ON MULTIPLE RECORDSを指定すると、該当するファイルを対象にしたREAD WITH KEPT LOCK文を実行したときに、該当する1レコードがロックされる。いったんロックされたレコードは、CLOSE, COMMIT, ROLLBACK, UNLOCKのいずれかの文を実行するまで解除されない。

18. (MF)ロックモードにMANUALまたはAUTOMATICを指定しWITH LOCK ON MULTIPLE RECORDSを指定しないと、単一レコードをロックするものと想定される。
19. (MF)EXTERNALと定義したファイルに対して、 ASSIGN 指令または必要語を用いてSELECT/ASSIGN文中でデータ名-1をファイル名に割り当てるか、あるいはファイル記述項のVALUE OF句の書き方2を用いてファイル名を割り当てる場合、下記の規則に従う。
    a. そのファイルを参照するすべての原始要素プログラム中に物理ファイル名を含めるには、同じ名前の一意名を使用する。
    b. 物理ファイル名を保持する一意名の定義にも、EXTERNAL属性を含める。

    (MF)この規則に反していても、コンパイル時にはそのことは検出されない。しかし、実行単位中の実行時要素 が実行時にこの規則から外れると、結果は保証されない。
    (MF)レコード・ロックについての詳細は、 ファイル処理に関するCOBOLのマニュアルを参照。

書き方1（レコード順ファイル）

20. CLOSE REEL文またはCLOSE UNIT文を用いてファイルを閉じることができるかまたはそうする意図がある場合、MULTIPLE REEL指定またはMULTIPLE UNIT指定を書く。
21. ファイル中のレコードは、先行・後行レコード関係によって規定される順序で呼び出される。この関係はファイルを作成または拡張したときに、WRITE文の実行によって確立されたものである。
22. (MF)LINE ADVANCING FILEを指定すると、印刷装置に適したファイルが作成される。このファイルの冒頭には、復帰文字が置かれる。各レコードを書き出す際の改行指定には、省略時解釈としてAFTER ADVANCING 1 LINEが設定される。形式についての詳細は、ファイル処理に関するCOBOLのマニュアルを参照。
23. (ANS85)(MF)PADDING CHARACTER句は注記になる。
    (ANS85)関連するファイル結合子（file connector）が外部ファイル結合子である場合、そのファイル結合子に関連する実行単位中のPADDING CHARACTER句の指定内容は、すべて同じにする。データ名-5が外部のものである場合、外部データ項目を参照する。
    (ANS85)(MF)RECORD DELIMITER句は注記になる。

書き方2（行順ファイル）

24. (MF)(XOPEN)明示的にまたは暗黙的に LINE SEQUENTIAL ORGANIZATION を指定すると、ファイルは固定長のレコードから構成され、その各レコードに1行分のデータが記録されるものとされる。このレコードが格納される際には、後行の空白は削除される。1行の定義はオペレーティングシステムによって異なる。復帰および改行のどちらか一方または両方によって1行の末尾を区切るものもあれば、固定長のレコードとして余白に充てん文字を詰めるものもある。どちらにしても、使用しているCOBOLシステムは、そのオペレーティングシステムのエディタ・ソフトウェアと互換性のあるファイルを必ず作成する。
25. (MF)(XOPEN)LOCK MODE IS句は注記になる。

書き方3（相対ファイル）

26. 呼出し法が順呼出しのとき、ファイル中のレコードは現存するレコードの相対レコード番号の昇順に呼び出される。
27. 呼出し法が乱呼出しのとき、RELATIVE KEYデータ項目の値が呼び出されるレコードを示す。
28. 相対ファイル中に格納されているすべてのレコードは、相対レコード番号によって一意に識別される。この相対レコード番号は、ファイル中のレコードの論理的な順序も示す。つまり、最初の論理レコードの相対レコード番号は1であり、以降2、3、4、...というように続く。
29. データ名-5のデータ項目は、利用者とオペレーティングシステムとの間で相対レコード番号を情報を取り交わすために使用される。

書き方4（索引ファイル）

30. 呼出し法が順呼出しのとき、ファイル中のレコードは、指定されたレコードキーの値の昇順に呼び出される。
31. 呼出し法が乱呼出しのとき、レコードキー・データ項目の値が呼び出されるレコードを示す。
32. RECORD KEY句は、該当するファイルの主レコードキーを指定する。ファイル中のレコードの間で、主レコードキーの値は一意にする。主レコードキーは、索引ファイルからレコードを呼び出す経路を形成する。
33. ファイルが1つ以上のレコード記述項をもつ場合、データ名-1は、これらのレコード記述項の1つにだけ記述されている必要がある。このどれかのレコード記述項にあるデータ名-1によって参照される同一の文字位置は、そのファイルにある他のすべてのレコード記述項のキーとして、暗黙的に参照される。
34. (ANS85)関連するファイル結合子が拡張ファイル結合子である場合、そのファイル結合子に対応する実行単位中のすべてのファイル記述項は、対応するレコード中で、同じ相対的な位置にあるデータ名-1に対して、同じデータ記述項を記述する。
35. ALTERNATE RECORD KEY句は、該当するファイルの副レコードキーを指定する。副レコードキーは、索引ファイルからレコードを呼び出す代替経路を形成する。
36. データ名-5
    (MF)または分割キー名-1
    およびデータ名-8
    (MF)または分割キー名-2
    のデータ記述ならびにレコード内でのそれらの相対位置は、ファイルが作成されたときと同じにする。また、副レコードキーの数も、ファイルが作成されたときと同じにする。このチェックは環境によっては構成可能である。("ファイル処理のためのプログラマガイド"の"呼出し可能ファイルハンドラ"に記述されている、呼出し可能ファイルハンドラ構成ファイル中のKEYCHECK属性を参照。)
37. DUPLICATESを指定すると、ファイル中の該当するレコードキーの値がレコード間で重複してもよいことを意味する。DUPLICATESを指定しない場合、該当するレコードキーの値は、ファイル中のすべてのレコードにわたって一意にする。

38. 重複する値をもつレコードを探し出すには、同じ値をもつ最初のレコードから順呼出しで読み込んでいく。
39. (MF)SUPPRESS句では、副指標（alternate index）にスパースキー（sparse key）を使用することを要求する。スパースキーは呼出し可能ファイル・ハンドラ（Callable File Handler）を通じてのみ利用可能であり、すべてのシステムで利用できるとはかぎらない（ファイル・ハンドラの詳細については、COBOLのシステム・ドキュメンテーションを参照）。

書き方5（整列併合ファイル）

40. ファイル管理記述項において、整列ファイルまたは併合ファイルに名前を付け、それと記憶媒体との関係を指定する。
41. (MF)SORT STATUS句を指定すると、各整列処理の実行が終わった時点で、データ名-2によって指定される2文字のデータ項目に値が設定される。この値は整列処理の終了結果を示す。

    (MF)整列処理の結果は、状態キー1と状態キー2の有効な組合わせによって示される。状態キー1と状態キー2の説明および状態の定義については、前述の*入出力状態*の節を参照。

    (MF)状態キー1と状態キー2の有効な組合わせには、下記のものがある。状態キー1=0で状態キー2=0であれば、正常終了を表わす。状態キー1=3で状態キー2=0であれば、パラメータ誤りを表わす。状態キー1=9であれば、状態キー2にオペレーティングシステムのエラーメッセージ番号が入れられている。

    (MF)SORT STATUS句の代わりに FILE STATUS句を使用できる。しかし、整列ファイルまたは併合ファイルに指定したFILE STATUS句は、SORT STATUSの同義語として扱われる。
42. ASSIGN句は注記となる。

6.1.3.2 入出力管理段落

機能

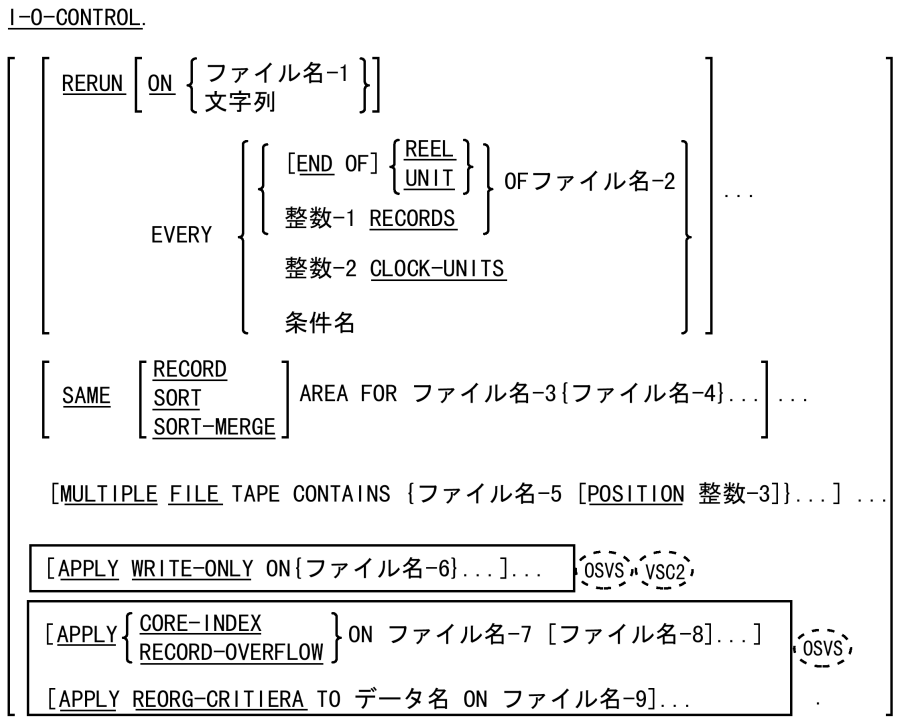
入出力管理段落では、再開点、ファイル間で共有する記憶領域、複数ファイル・リール中の順編成ファイルの位置を指定する。

(ANS85)入出力管理段落のRERUN句およびMULTIPLE FILE TAPE句は、ANSI '85標準では廃要素に分類されており、ANSI標準の次回の全面改訂の際に、削除される予定である。

(MF)この構文は、このCOBOLシステムに組み込まれているすべての方言で全面的に使用できる。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。

(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、RERUN句およびMULTIPLE FILE TAPE句は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこれらの句を使用するべきではない。

一般形式



構文規則

1. 入出力管理段落は、書いても書かなくてもよい。
2. 整数-1 RECORDS句または整数-2 CLOCK-UNITS句を指定するときは、 RERUN句に文字列を指定する。
3. SAME AREA句においては、SORTとSORT-MERGEは同義語である。
4. SAME SORT AREA句またはSAME SORT-MERGE AREA句を指定した場合、その中に最低1つは整列ファイルまたは併合ファイルであるファイル名を入れなければならない。整列ファイルでも併合ファイルでもないファイルを入れても構わない。
5. SAME句には2つの形（SAME AREA, SAME RECORD AREA）がある。これらは下記のように使い分ける。

   1つの原始要素中に複数のSAME句を指定してもよい。しかし、下記の制約がある。

   a. 1つのファイル名を複数の SAME AREA（ファイル領域の共用）句内に指定することはできない。
   b. 1つのファイル名を複数の SAME RECORD AREA（レコード領域の共用）句内に指定することはできない。

c. SAME AREA句に指定したファイル名をSAME RECORD AREA句内にも指定する場合、SAME AREA句内に指定したすべてのファイル名を、SAME RECORD AREA句内にも指定する。SAME AREA句内に指定しなかったファイル名を、SAME RECORD AREA句内に追加指定してもよい。SAME RECORD AREA句内に指定したすべてのファイルを、任意の時点で開いておくことができる。しかし、1つの SAME RECORD AREA句内に指定したファイルは、ある一時点では1つしか開くことはできない。

d. 1つの整列ファイル名または併合ファイル名を、1つのSAME AREA句内に指定してはならない。

e. 1つの整列ファイル名または併合ファイル名を、複数のSAME SORT AREA句またはSAME SORT-MERGE AREA句内に指定してはならない。

f. SAME AREA句に指定した整列ファイル名でも併合ファイル名でもないファイル名を、SAME SORT AREA句またはSAME SORT-MERGE AREA句内にも指定する場合、SAME AREA句内に指定したすべてのファイル名をSAME SORT AREA句またはSAME SORT-MERGE AREA句内にも指定する。

6. SAME AREA句，SAME SORT AREA句，SAME SORT-MERGE AREA句，SAME RECORD AREA句内に指定するファイルは、編成または呼出し法がすべて同じである必要はない。
7. (MF)文字列は、予約語または定数または利用者名であってはならない。
8. END OF REEL/UNIT句は、ファイル名-2が順編成ファイルである場合にだけ指定できる。
9. 1つのファイル名-2に、複数のRERUN句を指定してもよい。ただし、その場合は、下記の制約がある。
   a. 整数-1 RECORD句を複数指定する場合、それらに同じファイル名-2は指定できない。
   b. END OF REEL句またはEND OF UNIT句を複数指定する場合、それらに同じファイル名-2は指定できない。
10. CLOCK-UNITSを指定したRERUN句は、1つだけ指定できる。
11. (OSVS)(VSC2)入出力管理段落中の各句は、後ろに終止符（.）を置いてもよい。

一般規則

1. (ANS85)RERUN句は注記になる。
2. SAME AREA句は、2つ以上のファイルの処理用に同じ記憶領域を使用することを指定する。ただし、整列ファイルおよび併合ファイルは対象外である。共有される領域には、対象となるファイルに割り当てられるすべての記憶領域が含まれる。したがって、SAME AREA句内に指定したファイルを同時に2つ以上開くことは有効ではない。（構文規則5cを参照）
3. SAME RECORD AREA句は、現行の論理レコードを処理する間に2つ以上のファイルが同じ記憶領域を使用することを指定する。この場合、すべてのファイルを同時に開いておくことができる。SAME RECORD AREA中の論理レコードは、SAME RECORD AREA句の中に指定したファイルの内の、出力モードで開いたものの各論理レコード、および最も新しく読み込んだ入力ファイルの論理レコードを表わす。これは、共有領域を暗黙的に再定義していることに相当する。したがって、レコードは左詰めで配置される。

4. (MF)APPLY句は注記になる。
5. (MF)MULTIPLE FILE句は注記になる。
6. SAME SORT AREA句またはSAME SORT-MERGE句を指定した場合、その中に少なくとも1つは整列ファイルまたは併合ファイルを入れなければならない。その中に整列ファイルでも併合ファイルでもないファイルを入れても構わない。具体的な記憶領域の共有のされ方は下記のとおり。
    a. SAME SORT AREA句またはSAME SORT-MERGE句によって、指定した整列ファイルまたは併合ファイルを整列または併合する際に使用される記憶領域が共有される。したがって、ある整列ファイルまたは併合ファイルを整列または併合するために割り当てた任意の記憶領域を、他の整列ファイルまたは併合ファイルを整列または併合するために再利用できる。
    b. 更に、整列ファイルでも併合ファイルでもないファイルに割り当てた記憶領域を、必要に応じて、SAME SORT AREA句またはSAME SORT-MERGE句の中に指定した整列ファイルまたは併合ファイルを整列または併合するために割り当てることもできる。
    c. 整列ファイルでも併合ファイルでもないファイル同士は、互いに同じ記憶領域を共有するようにはされない。それらのファイル間で記憶領域を共有させたければ、該当原始要素中でそれらのファイルを対象にしてSAME AREA句またはSAME RECORD AREA句を指定する。
    d. この句の中に指定した整列ファイルまたは併合ファイルを使用する SORT文または MERGE文を実行する間は、この句の中に指定した整列ファイルでも併合ファイルでもないファイルを開くことはできない。

第7章：データ部

7.1 データ部

(MF)データ部の見出しは書いても書かなくてもよい。

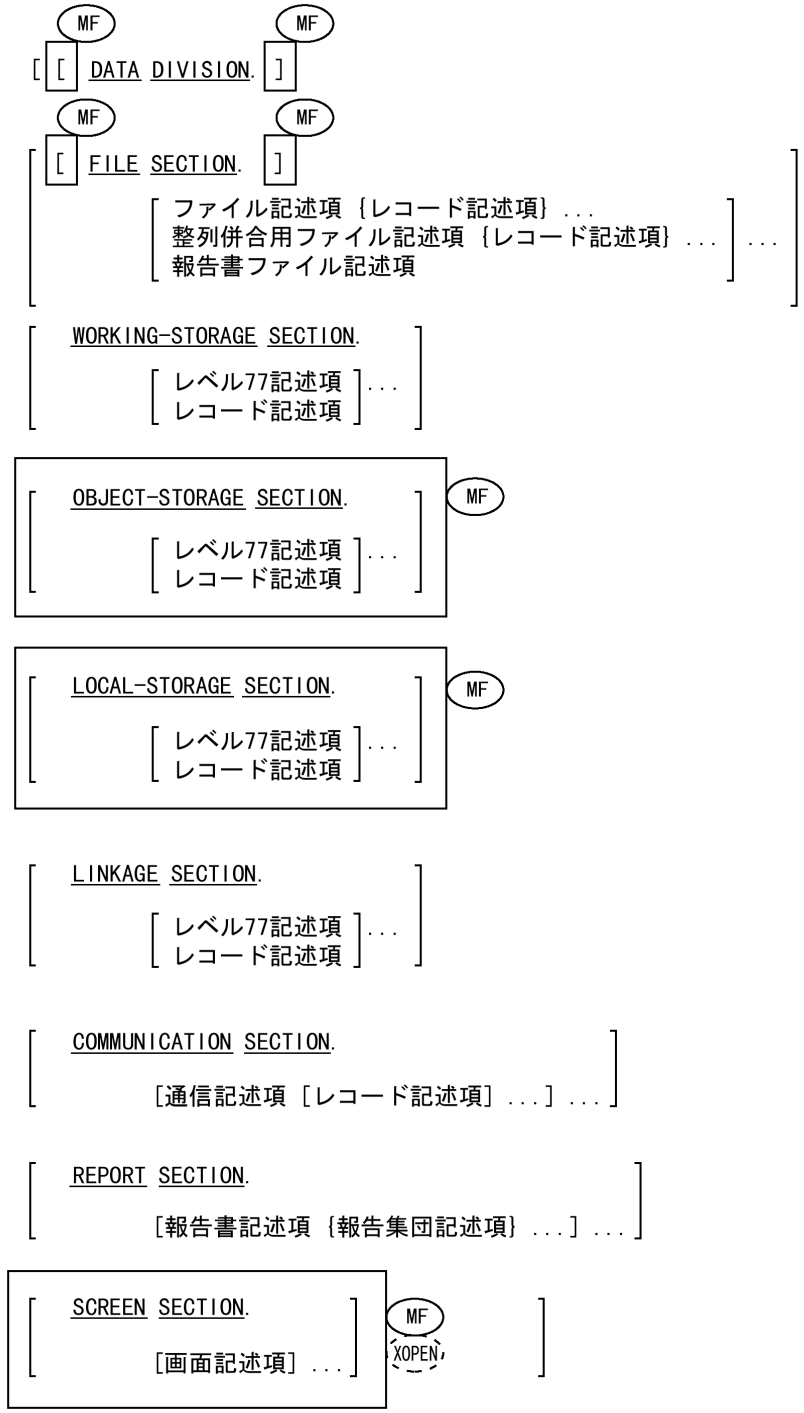
7.1.1 一般記述

データ部には、実行機能単位 が入力用として受けるデータ、または出力用として計算、作成、生成するデータを記述する。

7.1.1.1 構造

以下にデータ部内の節の一般形式を示す。 これは呼ばれるプログラム 内で示すべき順序を規定する。

一般形式


MF
MF
[DATA DIVISION.]
MF
MF
[FILE SECTION.]
ファイル記述項 {レコード記述項} ...
整列併合用ファイル記述項 {レコード記述項} ...
報告書ファイル記述項
WORKING-STORAGE SECTION.
レベル77記述項
レコード記述項
OBJECT-STORAGE SECTION.
MF
LOCAL-STORAGE SECTION.
MF
LINKAGE SECTION.
COMMUNICATION SECTION.
[通信記述項 [レコード記述項] ...] ...
REPORT SECTION.
[報告書記述項 {報告集団記述項} ...] ...
SCREEN SECTION.
[画面記述項] ...
MF
XOPEN

7.1.2 ファイル節

ファイル節の中では、ファイル記述項（FD）が構造上一番高い位置を占める。 FILE SECTION見出しの後ろにファイル記述項が続く。ファイル記述項は、レベル指示語（FD）とファイル名と一連の独立した句から構成される。 FD句には、論理レコードおよび物理レコードの大きさ、ファイルの外部名、ファイルを構成するデータレコードの名前を指定する。項目は終止符を付けて終了する。
SDファイル記述は、整列ファイルまたは併合ファイルの大きさとデータレコードの名前に関する情報を指定するためのものである。整列ファイルおよび併合ファイルに関しては、利用者が制御できるラベル手続きはない。また、ブロック化および内部記憶のされ方は、SORT文およびMERGE文に固有である。

7.1.2.1 整列併合との関係

整列併合機能単位は、データ部内の表の要素を整列させる機能を備えている。 SORT文およびMERGE文のUSING指定およびGIVING指定の対象とするファイルは、ファイル管理段落中に明示的にまたは暗黙的に、順編成で順呼出し法のファイルとして記述しておく。整列併合ファイル記述中に指定するファイルに対しては、入出力文を実行してはならない。
(ANS85)SORT文およびMERGE文の USING指定および GIVING指定の対象とするファイルは、 ファイル管理段落中に明示的に、索引編成または相対編成をとるものと記述できる。その呼出し法は、順呼出しにも動的呼出しにもできる。

7.1.2.2 レコード記述の構造

レコード記述は、個々のレコードの特性を記述するデータ記述項の組から構成される。各データ記述項は、レベル番号と、必要ならばデータ名と、必要に応じて一連の独立した句から構成される。 レコード記述は階層構造をとる。したがって、後ろに下位の記述が続くか否かによって、記述する句の形態は大幅に異なることがある。レコード記述の構造は、*COBOLの概念* の章の*レベルの概念* の節に定義してある。また、レコード記述に含まれる要素については、この後の*データ記述項の全体的な骨組み* の節に示してある。

7.1.2.3 ファイル記述項の全体的な骨組み

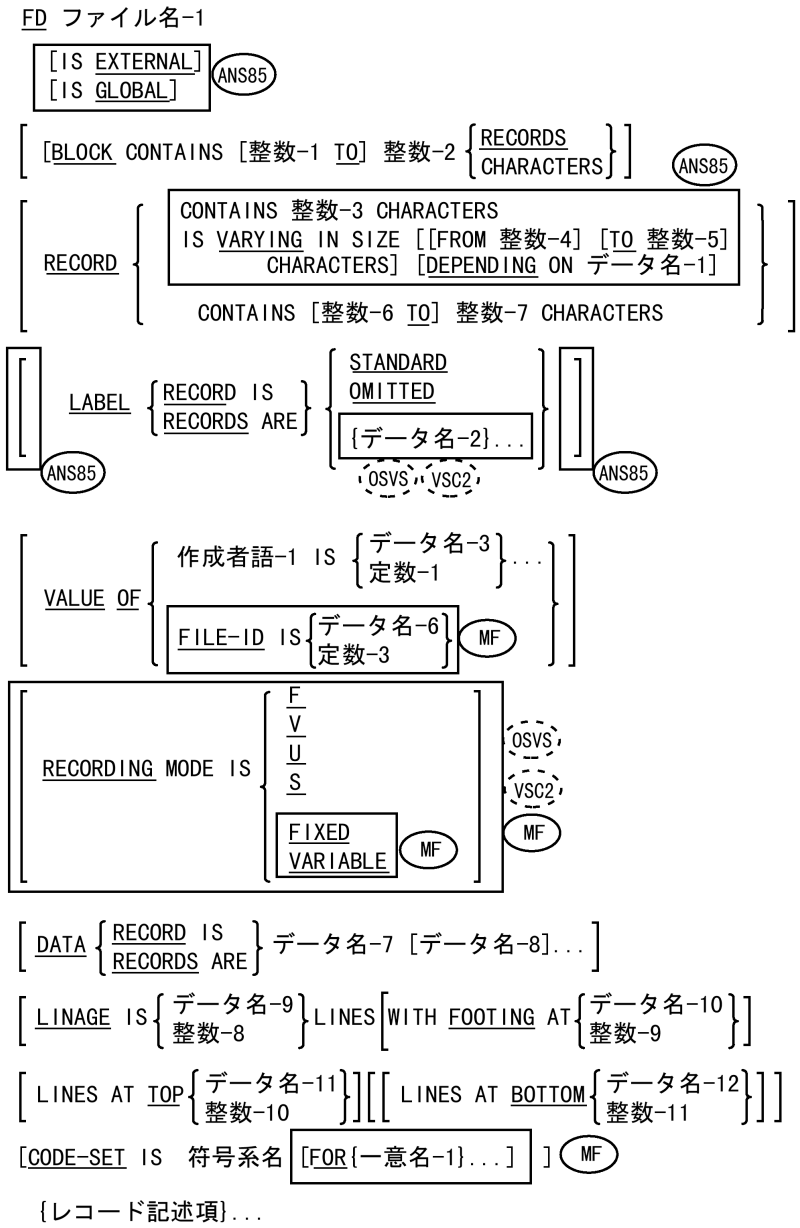
機能

ファイル記述は、ファイルの物理構造や識別やレコード名に関する情報から構成される。
(ANS85)ファイル記述では、ファイル結合子や関連するデータレコードや関連するデータ項目の内部属性や外部属性を指定する。また、ファイル記述項では、ファイル名が局所的であるのか大域的であるのかも指定する

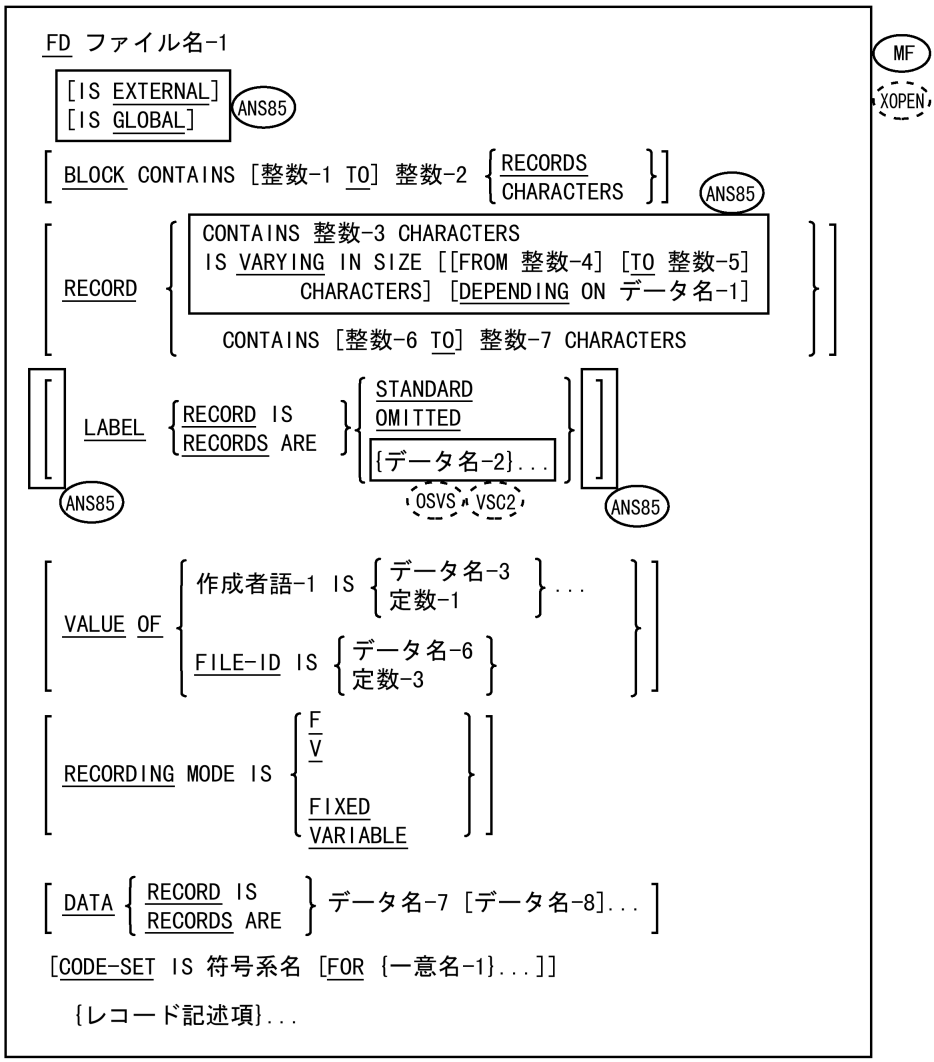
7.1.3 一般記述

一般形式

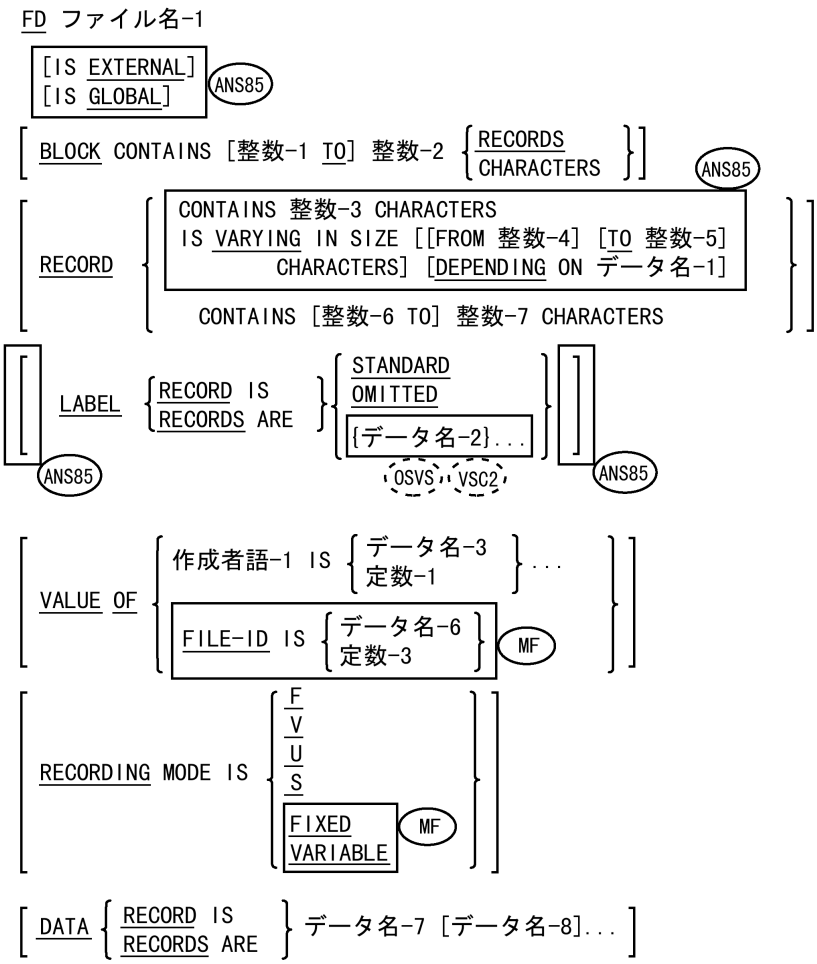
書き方1（レコード順ファイル）



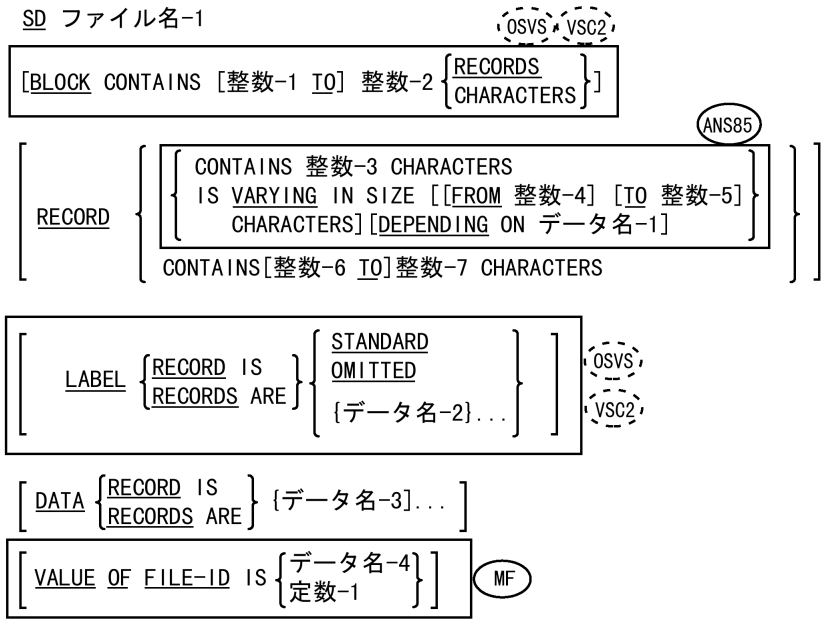
書き方2（行順ファイル）


FD ファイル名-1
[IS EXTERNAL]
[IS GLOBAL]
ANS85
MF
XOPEN
[BLOCK CONTAINS [整数-1 TO] 整数-2 { RECORDS / CHARACTERS }]
ANS85
[RECORD { CONTAINS 整数-3 CHARACTERS
IS VARYING IN SIZE [[FROM 整数-4] [TO 整数-5]
CHARACTERS] [DEPENDING ON データ名-1]
CONTAINS [整数-6 TO] 整数-7 CHARACTERS }]
[[LABEL { RECORD IS / RECORDS ARE } { STANDARD / OMITTED / {データ名-2}... }]]
ANS85
OSVS VSC2
ANS85
[VALUE OF { 作成者語-1 IS { データ名-3 / 定数-1 }... / FILE-ID IS { データ名-6 / 定数-3 } }]
[RECORDING MODE IS { F / V / FIXED / VARIABLE }]
[DATA { RECORD IS / RECORDS ARE } データ名-7 [データ名-8]...]
[CODE-SET IS 符号系名 [FOR {一意名-1}...]]
{レコード記述項}...

書き方3（相対ファイルおよび索引ファイル）



書き方4（整列併合ファイル）



構文規則

すべての書き方（すべてのファイル）

1. ファイル名-1に続く句は、どのような順序で記述してもよい。
2. (MF)VALUE OF FILE-ID 句を指定する場合、定数-1は文字定数とする。表意定数であってはならない。

書き方1、2、3（レコード順、(MF)行順、相対および索引ファイル）

3. レベル指示語のFDは、ファイル記述の開始を示すものであり、ファイル名の前に書く。
4. ファイル記述項の後ろには、レコード記述項を1つ以上続ける。

書き方4（整列併合ファイル）

5. レベル指示語のSDは、整列併合ファイル記述の開始を示すものであり、ファイル名の前に書く。
6. ファイル名の後ろに続く諸々の句は、指定しても指定しなくてもよく、指定する順番も重要ではない。
7. 整列併合ファイル記述項の後ろには、レコード記述項を1つ以上続ける。ただし、このファイルに対しては、入出力文を実行してはならない。

一般規則

書き方1（レコード順ファイル）

1. 順ファイル用のファイル記述項に LINAGE句およびEXTERNAL句を指定した場合、外部データ項目である LINAGE-COUNTERが使用される。順ファイル用のファイル記述項にLINAGE句およびGLOBAL句を指定した場合、特殊レジスタのLINAGE-COUNTERは大域名となる。

7.1.4 作業場所節

作業場所節（working-storage section）は、節の見出しで始まる。その後ろに、独立データ項目用のデータ記述項やレコード記述項を書く。
作業場所節の各レコード名および独立項目名は一意とする。それらは修飾できないためである。レコードに従属するデータ名は、 修飾することによって一意にできる場合、一意である必要はない。
(ANS85)データ名またはレコード名が参照されることがない場合、修飾して一意にしなくてもよい。

一般形式

```
WORKING-STORAGE SECTION.

    [ レコード記述項
      77レベルデータ記述項 ] ...
```

7.1.4.1 独立作業場所（独立データ記述項）

作業場所節に置く項目や定数のうち、相互に階層関係がなくさらに細分する必要もないものは、レコードにまとめる必要はない。この場合には、独立基本項目として、別々のデータ記述項に定義する。このレベル番号には特別の77という値が割り当てられている。
各データ記述項には、下記の要素を書く。

- レベル番号77
- データ名
- どちらか:
    - PICTURE句または
    - USAGE句(以下の指定付き:USAGE IS INDEX句、
      (VSC2)(MF)USAGE COMPUTATIONAL-1、USAGE COMPUTATIONAL-2、USAGE POINTER、または
      (MF)(COB370)USAGE PROCEDURE-POINTER)。

上記以外のデータ記述句は、必ずしも書く必要はない。必要があれば、項目の記述を完全にするために書けばよい。

7.1.4.2 作業場所レコード（レコード記述項)

作業場所節に置くデータ項目のうち、相互に階層関係があるものは、レコードにまとめておく。その際、レコード記述の書き方の規則に従う必要がある。ファイル節のレコード記述用のすべての句は、作業場所節のレコード記述にも使用できる。

7.1.4.3 レコード記述の構造

レコード記述（record description）は、レコードの特徴を記述する一連の データ記述項から構成される。各データ記述項（data description entry）の先頭にはレベル番号を書く。
(ANS85)必要があれば
その後ろにデータ名またはFILLER句を書く。さらにその後ろに、必要に応じて、独立の句をいくつか書く。レコード記述は階層構造をとる。したがって、1つの記述項（entry）に適用する句は、それに従属する記述項があるかないかによって、かなり違ってくる。レコード記述の構造とレコード記述項内に書ける要素については、*COBOLの概念* の章の *レベルの概念* の節、およびこの章の *データ記述項の全体的な骨組み* に説明してある。

7.1.4.4 初期値

作業場所節内の任意の項目の初期値は、指標データ項目、
(MF)または型定義は例外として、
データ項目のVALUE句を使用して指定する。指標データ項目およびVALUE句を指定していないデータ項目の初期値はどうなるかわからない。この章の*プログラムの初期値* に説明してある。

7.1.5 局所記憶節

(ISO2002)(MF)

機能

プログラムが呼ばれるたびに、 局所記憶節の写しが別途作られる。作られた局所記憶節の写しは、そのプログラムが呼ばれている間だけ、存在する。局所記憶節は、再帰型のプログラム呼出しのために特に用意したものである。.
局所記憶節（local-storage section）は、節の見出しで始まる。その後ろに、独立データ項目用のデータ記述項やレコード記述項を書く。局所記憶節の各レコード名および独立項目名は一意とする。それらは修飾できないためである。レコードに従属するデータ名は、修飾することによって一意にできるならば、一意である必要はない。
データ名またはレコード名が参照されることがない場合、修飾して一意にしなくてもよい。

一般形式

```
LOCAL-STORAGE SECTION.

    [ レコード記述項
      77レベルデータ記述項 ]
```

構文規則

1. EXTERNALデータは使用できない。
2. GLOBALデータは使用できない。
3. 「CALL ... USING局所データ項目」という構文は使用できる。
4. 「ENTRY ... USING局所データ項目」という構文は使用できない。
5. 「PROCEDURE DIVISION ... USING局所データ項目」という構文は使用できない。
6. 入れ子になったプログラムの中では、LOCAL-STORAGE SECTION見出しは使用できない。
7. COBOLシステムは、プログラムに局所記憶節がある場合にだけプログラムの再帰を行う。局所記憶節内にレコードが存在する必要はない。局所記憶節の見出しがないと、実行時に再帰処理が受け入れられない。

一般規則

1. DEFAULT-BYTE コンパイラ 指令は、局所記憶節内で定義した項目には適用できない。
2. 局所記憶節を含むプログラムをコンパイルする際には、MF指令セットを指定する。
3. VALUE句を書くことができるが、注記にとどまる。

7.1.6 連絡節

(OSVS)(VSC2)(MF)以下の説明において、手続き部のUSING句に触れている箇所はすべて、ENTRY USINGにも当てはまる。
(VSC2)(MF)*SET文* の節に記述されている場合は例外として、
プログラム中の連絡節（linkage section）が意味をもつのは、呼ぶプログラムからCALL文にUSING句を指定して、実行用プログラムが呼ばれた場合だけである。
(ISO2002)(MF)連絡節はメソッドにおいて常に意味を持つ。
呼ばれるプログラムの連絡節に指定した正式なパラメータと戻される項目は、制御が移された時に、その呼ばれるプログラムと呼ぶプログラムの両方から参照される。
(MF)このようなデータは、ファイル節にも作業場所節にも記述できる。 .
連絡節に記述されているデータ項目に対して、そのプログラムの中では領域は割り当てられない。これらのデータ項目を手続き部で参照すると、呼ぶプログラムの中で実行時に使用している場所を参照することになる。指標名については、そのような対応は付けられない。呼ぶプログラム中の指標名と呼ばれるプログラム中の指標名は、常に別の指標を参照する。

連絡節の中で定義されているデータ項目を、呼ばれるプログラムの手続き部の中で参照できるのは、そのデータ項目が手続き部の見出しのUSING句の作用対象に指定されるか、またはその作用対象に従属するかし、かつ実行用プログラムがUSING句を伴うCALL文の制御下にある場合だけである。
連絡節の構造は先に述べた作業場所節と同じである。つまり、節の見出しで始まり、その後ろに独立データ項目のデータ記述項やレコード記述項が続く。
連絡節内の各レコード名および独立項目名は呼ばれるプログラムの中で一意とする。それらは修飾できないためである。 (ANS85)この制限は手続き部で参照する場合に適用される。連絡節の中で定義した項目のうち、手続き部で参照できるのは、手続き部の見出しのUSING句内のデータ名-1、データ名-2などと、それらのデータ名に従属するデータ項目と、それらに関連する条件名と指標名だけである。.
連絡節に置くパラメータと戻される項目の指定は、*パラメータと戻される項目の基準*に記述された内容に従わなければならない。

7.1.6.1 独立連絡場所

連絡節に置く項目のうち、相互に階層関係のないものは、レコードにまとめる必要はない。この場合には、 独立基本項目として、別々のデータ記述項に定義する。データ記述は特別のレベル番号である77から始まる。
各データ記述項には、下記の要素を書く。

- レベル番号77
- データ名
- どちらか:
    - PICTURE句または
    - USAGE句(以下の指定付き: USAGE INDEX、
      (VSC2)(MF)USAGE COMPUTATIONAL-1、USAGE COMPUTATIONAL-2、USAGE POINTER、
      または
      (MF)(COB370)USAGE PROCEDURE-POINTER)。

上記以外のデータ記述句は、必ずしも書く必要はない。必要があれば、項目の記述を完全にするために書けばよい。

7.1.6.2 連絡レコード

連絡節に置くデータ項目のうち、相互に階層関係があるものは、レコードにまとめる。その際、レコード記述の書き方の規則に従う必要がある。入力レコードまたは出力レコードを記述するために使用する句のすべてを、連絡節でも使用できる。

7.1.6.3 初期値

連絡節では、VALUE句を指定できない。ただし、条件名記述項（88レベル）は例外である。

(OSVS)(VSC2)(MF)VALUE句を書くことができるが、注記にとどまる。

第8章：データ部 - データ記述

8.1 データ記述

8.1.1 データ記述

8.1.1.1 データ記述項の全体的な骨組み

機能

データ記述項は、個々のデータ項目の特徴を指定する。

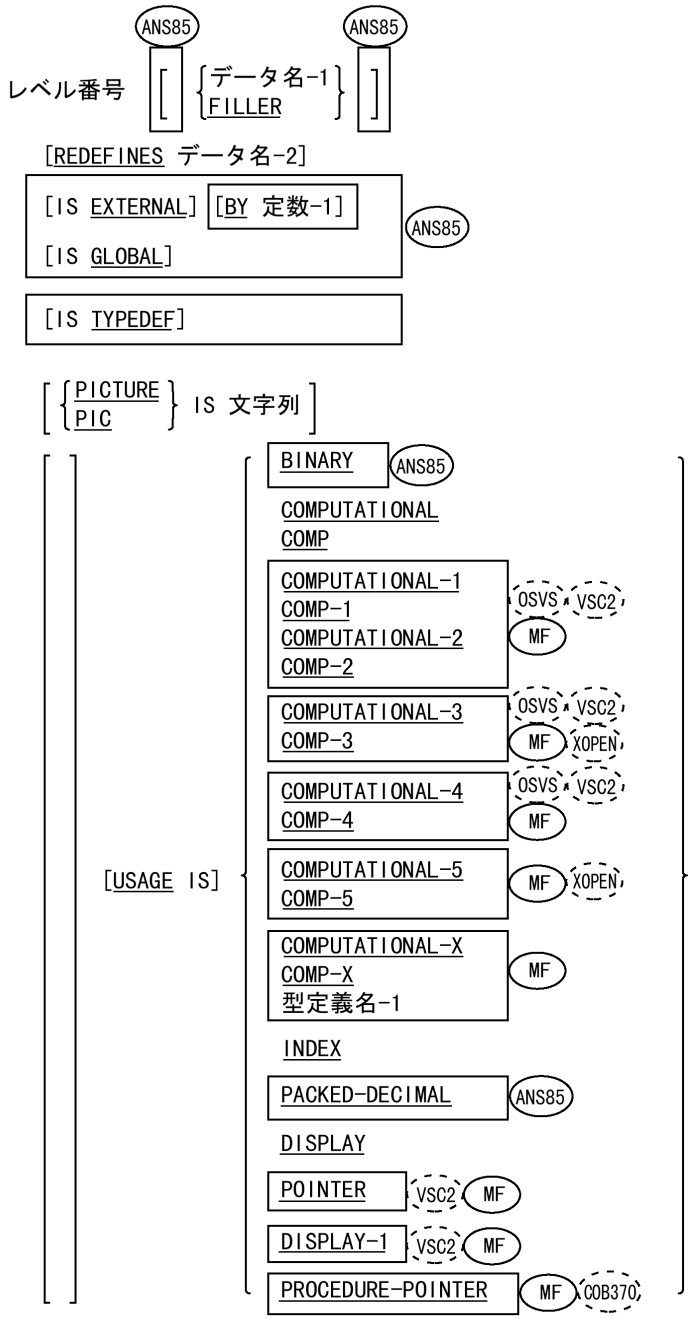
(MF)またはプログラマが定義したデータ型の特徴を記述して、複数のデータ項目の記述を指定するのに使用する。

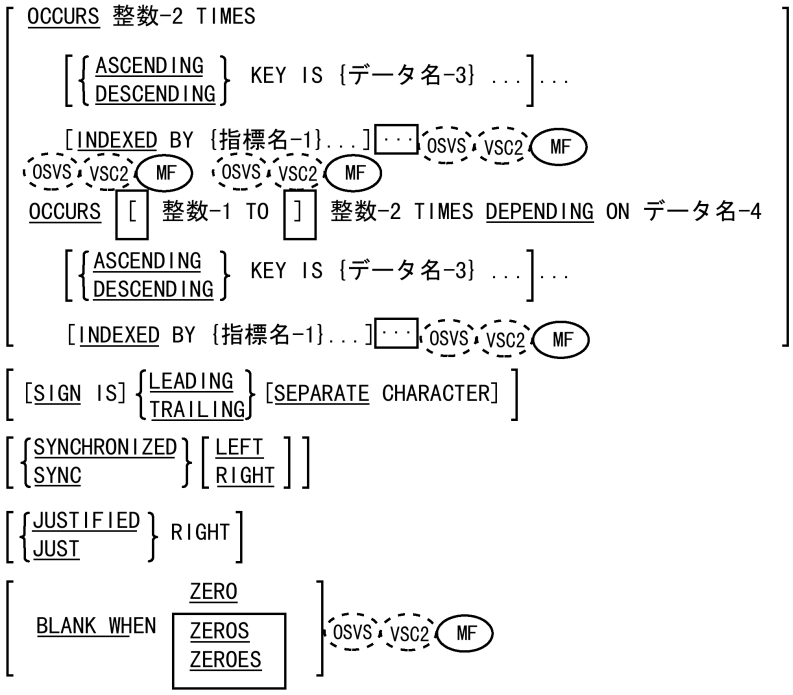
(ANS85)作業場所節またはファイル節内の01レベルのデータ記述項によって、データ・レコードおよびその下位のデータ項目に局所名（local name）または大域名（global name）が含まれるか否かが決まる。

(ANS85)作業場所節内の01レベルのデータ記述によって、データ・レコードとその下位のデータ項目の内部属性または外部属性が決まる。

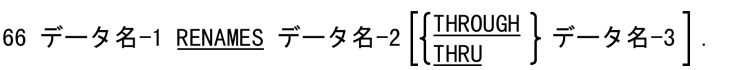
一般形式

書き方 1

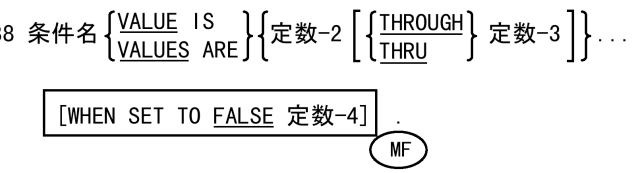

ANS85
ANS85
レベル番号
データ名-1
FILLER
[REDEFINES データ名-2]
[IS EXTERNAL] [BY 定数-1]
[IS GLOBAL]
ANS85
[IS TYPEDEF]
PICTURE
PIC
IS 文字列
[USAGE IS]
BINARY
ANS85
COMPUTATIONAL
COMP
COMPUTATIONAL-1
COMP-1
COMPUTATIONAL-2
COMP-2
OSVS
VSC2
MF
COMPUTATIONAL-3
COMP-3
OSVS
VSC2
MF
XOPEN
COMPUTATIONAL-4
COMP-4
OSVS
VSC2
MF
COMPUTATIONAL-5
COMP-5
MF
XOPEN
COMPUTATIONAL-X
COMP-X
型定義名-1
MF
INDEX
PACKED-DECIMAL
ANS85
DISPLAY
POINTER
VSC2
MF
DISPLAY-1
VSC2
MF
PROCEDURE-POINTER
MF
COB370



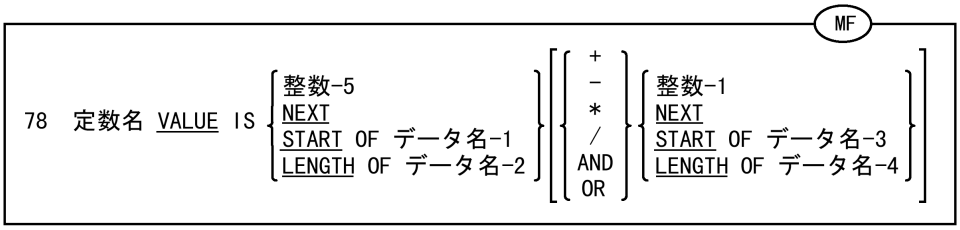
書き方 2



書き方 3



書き方 4



構文規則

1. 書き方1では、レベル番号は01から49までのどれか、または77とする。
2. 原則として、各句をどんな順序で書いてもよい。ただし、例外が2つある。データ1またはFILLER（無名項目）句
   (ANS85)を指定するとき
   は、それらはレベル番号の直後に書く。REDEFINES（再定義）句を指定するときは、データ名-1またはFILLER句
   (ANS85)が指定してあれば 直後に書く。指定がなければ、レベル番号の直後に書く。
3. 各基本項目に対して PICTURE（形式）句を指定する。 ただし、
   - 指標データ項目（index data item）、
   - (ISO2002)オブジェクト参照、
   - (MF)(COB370)手続きポインタ、
   - (VSC2)(MF)ポインタ、
   - 内部浮動小数点数データ項目 - USAGE (OSVS)(VSC2)(MF)COMPUTATIONAL-1か、(OSVS)(VSC2)(MF)COMPUTATIONAL-2か、 (ISO2002)FLOAT-SHORTか、 (ISO2002)FLOAT-LONG

   は例外とする。それらに対しては、PICTURE句を指定することは禁止されている。
4. THRUとTHROUGHは同義語であり、どちらを書いてもよい。
5. (ANS85)EXTERNAL句を指定できるのは、作業場所節内の01レベルのデータ記述項の中だけである。
6. (ANS85)EXTERNAL句とREDEFINES句は、同じデータ記述項の中には指定できない。
7. (ANS85)GLOBAL（大域）句は、01レベルのデータ記述項の中でだけ指定できる。
8. (ANS85)GLOBAL句またはEXTERNAL句を含む記述項、またはGLOBAL句またはEXTERNAL句を含むファイル記述項に関連するレコード記述に対しては、データ各-1を指定する。
9. (MF)TYPEDEF句は、レベル番号01のデータ記述項にだけ記述できる。
10. (MF)TYPEDEF句は、データ名-1が記述されている時にだけ使用できる。他の語では、明示的または暗黙的な FILLER句のデータ記述項と同じものを使用できない。
    (MF)TYPEDEF句を集団項目に対して指定した場合、従属するデータ記述は明示的または暗黙的なFILLER句で定義できるので注意が必要である。
11. SYNCHRONIZED句、PICTURE句、 JUSTIFIED句、 BLANK WHEN ZERO の各句は、基本データ項目以外には指定できない。
    (OSVS)(VSC2)SYNCHRONIZED句は、集団項目に指定できる。
12. (MF)定数-5と整数-1は、浮動点小数値あるいは負数の値であってはならず、18桁を超えてはならない。
13. 定数-5が式の一部として使用されている場合、整数でなければならない。
14. (OSVS)(VSC2)データ名-2およびデータ名-3は、暗黙的に修飾できる。
15. (ISO2002)VALUE句は、字類が指標、オブジェクト、ポインタのデータ項目には指定できない。

一般規則

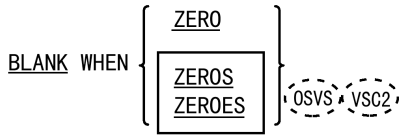
1. 書き方3は、条件名（condition-name）に適用する。 各条件名は レベル番号が88の記述項に別々に定義する必要がある。ここに、条件名とそれに対応する単独の値またはいくつかの値または値の範囲を指定する。条件変数（condition variable）に対する条件名記述項は、条件名が関連する項目の記述項の直後に書く。条件名は原則として、レベル番号を含む任意のデータ記述項に関連付けることができる。ただし、下記のものを除く。
    a. 他の条件名
    b. 66レベルの項目
    c. JUSTIFIED, SYNCHRONIZED, USAGE（USAGE IS DISPLAYを除く）を含む記述項を伴う項目を含む集団項目
    d. 指標データ項目
    (VSC2)(MF)またはポインタ・データ項目
    (*USAGE IS INDEX*句の節を参照。)
    e. (MF)定数名
2. (VSC2)(MF)条件名を、内部浮動小数点数項目と関連付けることができる。
    (MF)条件名を、外部浮動小数点数項目と関連付けることができる。
3. (MF)書き方4では、定数名を定義する。定数名（constant-name）とは、定数値を表わす記号名である。COBOLシステムは定数名を対応する値で置き換える。
4. (MF)TYPEDEF句を使用すると、基本データ記述の集団を型定義として宣言できる。そして、USAGE句内でデータ名-1を型定義名-1として使用して、そのデータ記述のインスタンスを宣言できる。型定義自体は記憶領域を割り当てられたデータ項目ではない。
5. (MF)TYPEDEF句を集団のレベルで指定した場合、その型の任意のデータ項目に関してその型定義の構成要素が暗黙的に宣言される。その構成要素を参照するには、データ項目の修飾に関するCOBOLの通常の規則を適用する。

8.1.1.2 BLANK WHEN ZERO（ゼロならば空白）句

機能

BLANK WHEN ZERO（ゼロならば空白）句は、項目の値がゼロのときにそれを空白に変える。

一般形式



構文規則

1. BLANK WHEN ZERO句を指定できるのは、PICTUREが数字（明示的または暗黙的にUSAGE IS DISPLAYを伴う）または数字編集であると指定されている基本項目だけである。(後述の*PICTURE句*を参照。)
2. ゼロ抑制記号としての星印（*）とBLANK WHEN ZEROは、同じ記述項の中には指定できない。
   (OSVS)両方を同時に指定してもよい。ただし、ゼロ抑制の方がBLANK WHEN ZERO句に優先する。

一般規則

1. BLANK WHEN ZERO句を指定すると、その項目の値がゼロのとき、その内容はまったくの空白となる。
2. PICTURE（形式）が数字を表わす項目にBLANK WHEN ZERO句を指定すると、その項目の項類は数字編集とみなされる。

8.1.1.3 BLOCK CONTAINS（ブロックの大きさ）句

機能

BLOCK CONTAINS句は物理レコードの大きさを指定する。
(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、この機能は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこの句を使用するべきではない。

一般形式

```
BLOCK CONTAINS [整数-1 TO] 整数-2 { RECORDS    }
                                  { CHARACTERS }
```

一般規則

1. (MF)BLOCK CONTAINS句は注記になる。

8.1.1.4 CODE-SET（符号系）句

機能

CODE-SET句は外部媒体上のデータを表現するために使用される符号系を指定する。
CODE-SET句を指定できるのは、レコード順編成
(MF)および行順編成の
ファイルに対してだけである。

(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、この機能は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこの句を使用するべきではない。

一般形式

CODE-SET IS 符号系名 [FOR {一意名-1}...] (MF)

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - CHARSET - 何が固有の符号系とみなされるかを決定する。

構文規則

1. あるファイルにCODE-SET句を指定した場合、そのファイル中のすべてのデータ項目の用途をDISPLAYと記述し、符号付き数字データがあればSIGN IS SEPARATE句を記述する。
   (MF)上記の制限は適用しない。
2. CODE-SET句によって参照される符号系名句には、定数を指定できない。
   (MF)上記の制限は適用しない。
3. (MF)一意名-1は修飾してもよいが、添字は付けられない。
4. (MF)各一意名-1は、該当するファイルのレコード記述中に記述したデータ項目とする。しかし、それ自体がレコード記述であってはならない。すべての一意名-1は、同じレコード記述内に含まれていなければならない。
5. (MF)FORを指定すると、その対象のデータ項目にCODE-SET句に指定した符号系が適用される。FORを指定しないと、該当するファイル全体にCODE-SET句に指定した符号系が適用される。
6. どのようなファイルまたはその中のデータ項目に関しても、固有の文字集合にはCODE-SET句は適用されないものと想定される。

一般規則

1. (MF)レコード領域中のデータは、つねにASCIIコードで書かれる。特殊名段落において符号系名がEBCDICに設定されていると、CODE-SET句の影響を受けるデータは、ファイルに書かれるときにASCIIからEBCDICにコード変換される。特殊名段落において符号系名がSTANDARD-1, STANDARD-2, NATIVE, ASCIIのどれかに設定されている場合は、コード変換は必要ない。
2. (MF)上記のコード変換に際しては、 CODE-SET句が適用されるデータ項目はどれも英数字として扱われる。そのデータ記述に字類または項類がどのように記述されているかは考慮されない。

3. (MF)一意名-1にOCCURS句が指定してある場合、CODE-SET句はその最初の要素に対してだけ適用される。一意名-1の下位に属する項目にOCCURS句が指定してある場合、COSE-SET句は一意名-1全体に適用される。

8.1.1.5 DATA RECORDS（データレコード）句

機能

DATA RECORDS句は、ファイルを構成するデータレコードの名前を書き記すだけである。
(ANS85)ファイル記述項のDATA RECORDS句は、ANSI '85標準では廃要素に分類されており、ANSI標準の次回の全面改訂の際に、削除される予定である。
(MF)この構文は、このCOBOL処理系に組み込まれているすべての方言で全面的に使用できる。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。
(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、この機能は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこの句を使用するべきではない。

一般形式

```
DATA {RECORD IS  }データ名-1 [データ名-2]...
     {RECORDS ARE}
```

構文規則

1. データ名-1およびデータ名-2は、データレコードの名前である。これと同じ名前で、レベル番号が01のレコード記述を書く。
   (OSVS)(VSC2)これらのデータ名は、プログラム中のデータ記述項と対応する必要はない。

一般規則

1. データ名を複数指定することは、ファイル中にデータレコードの型が複数あることを意味する。各型のレコードは、大きさや形式などが異なっていてよい。このデータ名を書く順序には意味はない。

   注: FD中で複数の01レベル項目の使用時に、SELECT文がキー定義を含んでいる場合、キーの大きさは、そのFDの最小レコード長の範囲内になければならない。

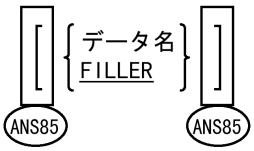
2. 概念的には、 1つのファイル内のデータレコードは、すべて同じ領域を共有する。ファイル内に複数の型のデータレコードが存在する場合でも、そのことは何ら変わらない。

8.1.1.6 データ名およびFILLER句

機能

データ名（data-name）は、記述する対象のデータの名前を表わす。語FILLERは、レコード中の明示的に参照しない基本項目を指定するために使用する。

一般形式



構文規則

1. データ名または必要語FILLER
   (ANS85)のどちらかを指定する場合、
   データ記述項のレベル番号の直後に書く。

一般規則

1. 必要語FILLERは、レコード中の基本項目または集団項目の名前として使用できる。どのような場合でも、FILLER項目は明示的に参照できない。しかし、必要語FILLERを条件変数として使用することはできる。この場合は、FILLER項目を明示的に参照するのではなく、その値を参照するだけだからである。
2. (ANS85)この句を省略すると、記述対象の項目はFILLERを指定したように扱われる。

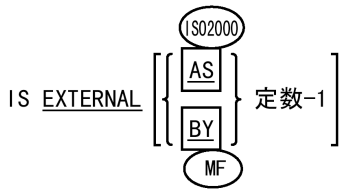
8.1.1.7 EXTERNAL（外部）句

(ANS85)

機能

EXTERNAL（外部）句は、データ項目またはファイル結合子に外部属性を指定する。外部データ・レコードを構成するデータ項目および集団項目は、そのレコードを記述している実行単位に含まれるすべての実行時要素 から利用できる。

一般形式



構文規則

1. EXTERNAL句を指定できる場所は、ファイル記述項と作業場所節内のレコード記述項だけである。
2. 同じ原始要素内で、EXTERNAL句を含む項目の外部化した名前は、EXTERNAL句を含む他の項目の外部化された名前とは異なるものにする。
3. EXTERNAL句が含まれるデータ記述項またはその下位のデータ記述項内では、VALUE句は指定できない。そのようなデータ記述項と関連する条件名記述項には、VALUE句を指定できる。
4. (MF)定数-1は文字または各国語型とし、表意定数としてはならない。
5. (ISO2000)EXTERNAL句は字類オブジェクトのデータ項目用に指定してはならない。

一般規則

1. データ名句によって名付けられたレコード中に含まれるデータは、外部用である。このデータは、それを記述している実行単位に含まれるすべての実行時要素から呼び出して処理できる。このデータを再定義してもよい。ただし、下記の一般規則に従う必要がある。
2. 実行単位に含まれる2つ以上の原始要素 において同じ外部データ・レコードを記述する場合、関連するレコード記述項の各レコード名が同じであり、かつ各レコードに含まれる標準データ形式の文字数も同じでなければならない。しかし、外部レコードを記述する原始要素 にREDEFINES（再定義）句を含むデータ記述項を記述して、外部レコード全体を再定義することができる。この再定義は、同じ実行単位の中の別々の原始要素で同じにする必要はない。 *REDEFINES句*を参照。
3. EXTERNAL句を指定することは、関連するファイル名またはデータ名が大域名であることを意味するわけではない。後述の*GLOBAL句*を参照。
4. この記述項に関連するファイル結合子は、外部ファイル結合子である。
5. (MF)EXTERNAL属性を持つファイルの詳細については、ファイル処理に関するCOBOLシステムのマニュアルを参照。
6. (MF)定数-1は外部データ項目またはファイル結合子の外部名を定義する。同じ外部データ項目またはファイル結合子を参照するすべての原始要素で、 同じ外部名を使用しなければならない。

8.1.1.8 GLOBAL（大域）句

(ANS85)

機能

GLOBAL（大域）句は、データ名やファイル名や報告書名が大域名であることを指定する。大域名は、それを定義しているプログラムに含まれるすべてのプログラムで利用できる。

一般形式

```
IS GLOBAL
```

構文規則

1. GLOBAL句を指定できる場所は、ファイル節または作業場所節内の01レベルのデータ記述項、ファイル記述項、報告書記述項だけである。
   (MF)(COB370)連絡節においても、GLOBAL句を指定できる。
2. 同じデータ部内では、同じ名前の2つのデータ項目のデータ記述項にはGLOBAL句を指定できない。
3. いくつかのファイルにSAME RECORD AREA（レコード領域の共用）句を指定した場合、それらのレコード記述項またはファイル記述項にはGLOBAL句を指定できない。

一般規則

1. GLOBAL句を使用して記述したデータ名、ファイル名、報告書名は大域名である。ある大域名の下位に属するデータ名もすべて大域名となる。大域名と関連する条件名もすべて大域名となる。
2. 大域名を定義しているプログラムの中に直接的または間接的に含まれるプログラムからは、その名前を再度定義することなく、参照できる。*COBOL言語の概念*の章の*名前の適用範囲*を参照。
3. REDEFINES句が含まれるデータ記述項の中でGLOBAL句を使用した場合、そのGLOBAL句は大域属性をもつREDEFINESだけの対象となる。

8.1.1.9 JUSTIFIED（桁寄せ）句

機能

JUSTIFIED（桁寄せ）句は、受取り側データ項目における標準的でない桁寄せを指定する。

一般形式

```
{JUSTIFIED}
{JUST     } RIGHT
```

構文規則

1. JUSTIFIED句は、基本項目にだけ指定できる。
2. JUSTは、JUSTIFIEDの省略形である。
3. 数字項目または編集を指定した項目には、JUSTIFIED句を指定できない。
4. 指標データ項目(後述の*USAGE IS INDEX*句を参照)
   (VSC2)(MF)またはポインタ・データ項目には、 JUSTIFIED句を指定できない。
5. (OSVS)(VSC2)(MF)外部浮動小数点数項目または内部浮動小数点数項目には、JUSTIFIED句を指定できない。

一般規則

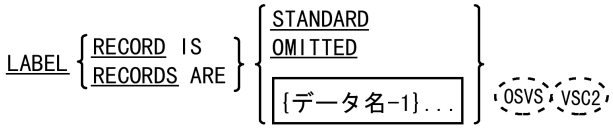
1. 受取り側のデータ項目にJUSTIFIED句が指定してあり、送出し側データ項目の方が受取り側データ項目よりも大きい場合、左端の文字が切り捨てられる。受取り側のデータ項目にJUSTIFIED句が指定してあり、受取り側データ項目の方が送出し側データ項目よりも大きい場合、データは右端を揃えられ左端には空白が詰められる。
   送出し側データ項目の内容は考慮されない（後部の空白は抑制されない）ので、注意する必要がある。
   たとえば、PIC X（4）のデータ項目があり、その値が"A　"(Aの後ろに空白が3つ)であるとする。これを、PIC X(6) JUSTIFIEDのデータ項目に転記すると、結果は"　A　"となる。同じデータ項目をPIC X（3）JUSTIFIEDのデータ項目に転記すると、左端の文字が切り落とされて、結果は"　"（空白が3つ）となる。
2. JUSTIFIED句を省略すると、基本データ項目にデータを収める際の標準桁寄せの規則が適用される。(*COBOL言語の概念*の章の*標準桁寄せ規則*の節を参照。)

8.1.1.10 LABEL RECORDS（ラベルレコード）句

機能

LABEL RECORDS句は、 ラベルの有無を指定する。
(ANS85)LABEL RECORDS句は、ANSI '85標準では廃要素に分類されており、ANSI標準の次回の全面改訂の際に、削除される予定である。
(MF)この構文は、このCOBOL処理系に組み込まれているすべての方言で全面的に使用できる。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。
(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、この機能は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこの句を使用するべきではない。

一般形式



構文規則

1. 各ファイル記述項に、LABEL RECORDS句を記述する。
   (ANS85)LABEL RECORDS句は必要ない。
2. (OSVS)(VSC2)データ名-1は、ラベルレコードの名前である。この名前で、01レベルのレコード記述を用意しておく。
3. (OSVS)(VSC2)データ名-1は、該当するファイルのDATA RECORDS句には指定できない。
4. (OSVS)(VSC2)LABEL RECORD ISとLABEL RECORDS AREはどちらも正しい構文として受け入れられる。

一般規則

1. (MF)LABEL RECORDS句は注記にとどまる。

8.1.1.11 レベル番号

機能

レベル番号（level-number）は、レコード内のデータの階層を示す。さらに、作業場所項目、連絡項目、条件名、
(MF)，定数名、
RENAMES（再命名）句の記述項にも、レベル番号を使用する。

一般形式

レベル番号

構文規則

1. レベル番号は、各データ記述項の最初の要素として必要である。
2. FD，CD，SDの各記述項の下位のデータ記述項のレベル番号は、 01から49、 66、
   (MF)78、
   88のどれかとする。(前述の*ファイル記述*の節を参照。)
3. 報告書節および画面節中のデータ記述項のレベル番号は、01から49
   (MF)または78とする。
4. 作業場所節、

(MF)局所記憶節、
連絡節中のデータ記述項のレベル番号は、 01から49、66、 77、
(MF)，78、
88のどれかとする。

5. レベル番号は、1文字または2文字の数字とする。

一般規則

1. レベル番号01は、各レコード記述の最初の記述項であることを示す。
2. レベル階層の概念が当てはまらない記述項に、特別のレベル番号が割り当てられている。下記のものがある。
    a. レベル番号77は、独立作業場所データ項目および独立連絡データ項目を示すために使用する。このレベル番号には、先に示したデータ記述項の骨組みの書き方1だけが適用できる。
    b. レベル番号66は、RENAMES（再命名）記述項を示すために使用する。このレベル番号には、先に示したデータ記述項の骨組みの書き方2だけが適用できる。
    c. レベル番号88は、 条件変数に関連する条件名を定義する記述項を示すために使用する。このレベル番号には、先に示したデータ記述項の骨組みの書き方3 だけが適用できる。
    d. (MF)レベル番号78は、定数名を定義する記述項を示すために使用する。このレベル番号には、先に示したデータ記述項の骨組みの書き方4だけが適用できる。
3. CD，FD，SDの各記述項の下位に属する複数の01レベルの記述項は、同じ領域を暗黙的に再定義する。

8.1.1.12 行数カウンタ

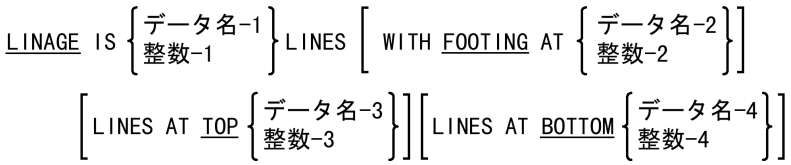
予約語のLINAGE-COUNTERは、特殊レジスタの名前である。レコード順ファイルのファイル記述項にLINAGE句を指定すると、この特殊レジスタが生成される。このレジスタには、LINAGE句中の整数-1またはデータ名-1の大きさに等しい、符号なしの整数が暗黙的にとられる。

8.1.1.13 LINAGE（行数）句

機能

LINAGE句を指定することによって、利用者は論理ページの行数を設定できる。利用者はまた、論理ページ上の上下の余白（top margin，bottom margin）およびページ本体（page body）中の脚書領域（footing area）の開始行を指定することもできる。
LINAGE句を指定することができるのは、レコード順編成のファイルに対してだけである。

一般形式



構文規則

1. データ名-1、データ名-2、データ名-3、データ名-4は、符号なしの基本整数データ項目とする。
2. 整数-1の値は、ゼロより大きくする。
3. 整数-2の値は、整数-1の値を超えてはならない。
4. 整数-3および整数-4の値は、ゼロであってもよい。

一般規則

1. LINAGE句を指定することによって、利用者は論理ページの行数を設定できる。論理ページの大きさは、FOOTING指定を除く各指定値の和である。 FOOTINGを指定しないと、脚書機能は組み込まれず、 ページあふれ条件（page overflow condition）とは別のページの終わり条件（end-of-page condition）は発生しない。
   論理ページと物理ページの大きさの間には、必ずしも関係がある必要はない。
2. 整数-1またはデータ名-1のデータ項目の値は、論理ページ上に書いたり改行したりできる行数を表わす。この値はゼロよりも大きくする。論理ページ上の書いたり改行したりできるこの部分を、ページ本体と呼ぶ。
3. 整数-2またはデータ名-2のデータ項目の値は、論理ページ上の脚書領域の開始行を表わす。この値はゼロよりも大きく、かつ整数-1またはデータ名-1のデータ項目の値を超えてはならない。
   脚書領域となるのは、整数-2またはデータ名-2のデータ項目の値以上、整数-1またはデータ名-1のデータ項目の値以下の、論理ページ中の行の部分である。
4. 整数-3またはデータ名-3のデータ項目の値は、論理ページ上の上の余白行数を表わす。この値はゼロでもよい。
   整数-4またはデータ名-4のデータ項目の値は、論理ページ上の下の余白行数を表わす。この値はゼロでもよい。
5. 整数-1と整数-3と整数-4を指定すると、OPEN文を用いて該当ファイルを出力モードで開いたときに、これらの値に基づいて論理ページの各構成部分の行数が設定される。整数-2を指定すると、その値に基づいて脚書領域が決定される。これらの値はすべて、該当実行時要素が実行される間、ファイルに書き込まれる論理ページを制御するために使用される。

6. データ名-1、データ名-3、データ名-4を指定すると、OPEN文を用いて該当ファイルを出力モードで開いたときに、これらの値に基づいて最初の論理ページの各構成部分の行数が設定される。
   データ名-2を指定すると、OPEN文を用いて該当ファイルを出力モードで開いたときに、この値に基づいて最初の論理ページの脚書領域が決定される。
7. LINAGE句を指定すると、LINAGE-COUNTERが生成される。 LINAGE-COUNTERの値は、どの時点でも、現在のページ本体の中で次に書き込まれる行番号を表わしている。LINAGE-COUNTERには下記の規則が適用される。
   a. ファイル節中のファイル記述項にLINAGE句を指定した各ファイルごとに、別々のLINAGE-COUNTERが設定される。
   b. 手続き部の文によって、LINAGE-COUNTERを参照することはできるが、変更することはできない。1つの原始要素中に複数のLINAGE-COUNTERが存在することがあるので、必要に応じて、ファイル名でLINAGE-COUNTERを修飾しなければならないことがある。
8. 各論理ページは、次の論理ページと間を空けることなく連なっている。
9. ADVANCING PAGE指定のあるWRITE文を実行したとき、またはページあふれ条件が発生したとき(後述の*WRITE*文の節を参照) 、各データ項目の値に基づいて、次の論理ページを構成する各部分の行数が決定される。
10. データ名-2を指定すると、ADVANCING PAGE指定のあるWRITE文を実行したとき、またはページあふれ条件が発生したとき、その値に基づいて、次の論理ページの脚書領域が決定される。
11. 該当するファイルに対してWRITE文を実行すると、下記の規則に従って、LINAGE-COUNTERは自動的に更新される。
    - WRITE文にADVANCING PAGE指定がある場合は、LINAGE-COUNTERの値は自動的に1に再設定される。
    - WRITE文にADVANCING 一意名-2または整数を指定している場合は、LINAGE-COUNTERの値はその整数または一意名-2の値の分だけ繰り上げられる。
    - WRITE文にADVANCING指定をしていない場合は、LINAGE-COUNTERの値は1だけ繰り上げられる。(後述の*WRITE*文の節を参照。)
    - 以降の各論理ページの書き込み可能な最初の行に装置が位置付け直されるたびに、 LINAGE-COUNTERの値は自動的に1に再設定される。(後述の*WRITE*文の節を参照。)
12. ファイルに対してOPEN文を実行すると、対応するLINAGE-COUNTERの値は自動的に1に設定される。

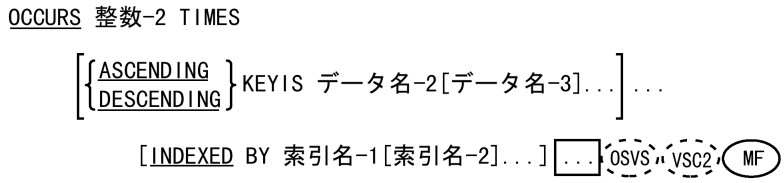
8.1.1.14 OCCURS（反復）句

機能

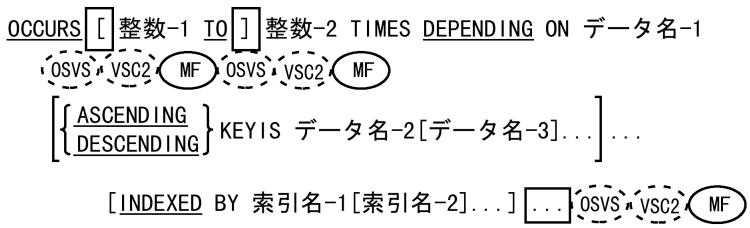
OCCURS（反復）句は、同じデータ項目が繰り返されるとき、その記述項をいちいち指定する手間を省き、添字（subscript）や指標（index）を付けるために必要な情報を与える。

一般形式

書き方1



書き方2



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - ODOOSVS - OCCURS DEPENDING ON句について、OS/VS COBOLと互換性のある処理を行うよう要求する。
    - ODOSLIDE - ネストされたOCCURS DEPENDING ON句およびその後ろに続く固定データの処理を制御する。

構文規則

1. 整数-1と整数-2を両方指定する場合、整数-1は1以上であるか
   (ANS85)(OSVS)または0に等しく、
   整数-2は整数-1よりも大きいか、
   (OSVS)(VSC2)または等しく
   する。
   (OSVS)(VSC2)(MF)書き方2において、" 整数-1 TO " を省略すると、省略時解釈として1がとられる。
2. データ名-1のデータ記述は、整数とする。
3. データ名-1、データ名-2、データ名-3は修飾できる。
4. データ名-2は、OCCURS句を含む記述項の名前であるか、またはその下位に属する記述項の名前とする。
   (OSVS)この制限は無視してよい。
5. データ名-3などは、その下位に属する記述項の名前にする。
   (OSVS)この制限は無視してよい。

6. OCCURS句の対象またはその下位に属する記述項を指標を用いて参照する場合、INDEXED BY（指標付き）句を指定する。
   (OSVS)(VSC2)ただし、他の表用に定義した指標を使用する場合は、その必要はない。(*COBOL言語の概念*の章の*指標付け*の節を参照。)
   ここに指定する指標名はデータとして扱われず、データの階層にも属さない。
7. レコード記述内で、書き方2の OCCURS句が含まれるデータ記述項の後ろに続けられるのは、その下位に属するデータ記述項だけである。
   (OSVS)(VSC2)レコード記述内で、書き方2のOCCURS句が含まれるデータ記述項の後ろに、その下位に属さないデータ記述項を続けることができる。その記述項がそのレコード内で占める位置は、 DEPENDING ON指定によって参照されるデータ項目の実行時の値によって変わる。
   (MF)ただし、NOODOSLIDEシステム指令を設定した場合は例外である。この場合は、データ名-1の値にかかわらず、そのレコードに含まれる要素は常に最大の回数だけ反復されるものとみなされる。 ODOSLIDEシステム指令を設定した場合は、
   (OSVS)(VSC2)表の後ろに続く表に属さない一意名によって参照される位置は、データ名-1の値に応じて変動する。その際、表の後ろに続くデータの内容が失われることがある。
8. OCCURS句は、下記のデータ記述項には指定できない。
   a. レベル番号が 66または 88のデータ記述項
   b. 大きさが可変な項目を記述する、データ記述項。大きさが可変な項目とは、下位の項目のデータ記述に、書き方2のOCCURS句が含まれるものである。
      (OSVS)(VSC2)書き方2のOCCURS句を指定したデータ項目の下位に属するデータ記述に、OCCURS句を指定できる。
9. OCCURS句は、 01レベルまたは 77レベルのデータ記述項には指定できない。
   (MF)この制限は無視してよい。
10. 書き方2において、データ名-1によって定義されるデータ項目は、OCCURS句を含むデータ記述項によって定義される最初の文字位置から最後の文字位置までの範囲内にあってはならない（OCCURS句の対象のデータ項目と、別の記憶域を占めるデータ項目にすること）。
    (OSVS)(VSC2)OCCURS DEPENDING ON Slide指令を設定した場合は、データ名-1は固定位置であること。
11. データ名-2がこの記述項の左辺ではないとき、下記のようにする。
    a. KEY IS（キーは）句に指定するデータ名の項目はすべて、この記述項の左辺である集団項目に含まれているものとする。
    b. KEY IS句に指定するデータ名の項目は、OCCURS句を含んではいけない。
    c. KEY IS句に指定するデータ名の項目とこの記述項の左辺との間には、OCCURS句が含まれる記述項を置いてはいけない。
12. 指標名-1、指標名-2などは、原始要素内で一意の語とする。
    (OSVS)指標名-1、指標名-2などは一意である必要はなく、この記述項の左辺のデータ名によって修飾できる。
13. (OSVS)(VSC2)(MF)外部浮動小数点数データ項目および内部浮動小数点数データ項目に対して、 OCCURS句を指定できる。

14. (ISO2000)KEY句は、字類がオブジェクトのデータ項目に対して記述してはならない。

一般規則

1. OCCURS句は、 表または同形のデータ項目の反復を定義するために使用する。OCCURS句を指定した場合、文の中でこの記述項の左辺であるデータ名を参照するときは、添字または指標を付ける。ただし、SEARCH,
   (MF), SORT
   USE FOR DEBUGGINGの各文は例外である。さらに、この記述項の左辺が集団項目の名前である場合、その集団に属するデータ項目を作用対象として使用するときには、すべて添字または指標を付ける。ただし、REDEFINES句の作用対象として用いる場合は例外である。(*COBOL言語の概念*の章の *添字付け* *指標付け* *一意名*の各節を参照。)
2. OCCURS句を指定した項目のデータ記述句はすべて、反復される各項目にも使用される。ただし、OCCURS句自体は反復しては使用されない。(*条件名以外のデータ記述項* (MF)*定数名以外のデータ記述項*の一般規則20の制限事項を参照。)
3. データ名-1は固定位置に置く。したがって、 OCCURS DEPENDING ON句を含む項目の後ろには置けない。
4. 記述項の左辺の反復回数は、下記のように定まる。
   a. 書き方1では、整数-2の値が一定の反復回数を表わす。
   b. 書き方2では、データ名-1によって参照されるデータ項目の現在の値が反復回数を表わす。
      この書き方は、この記述項の左辺の反復回数が可変であることを表わす。整数-2の値は最大反復回数を表わし、整数-1の値は最小反復回数を表わす。このことは必ずしもこの記述項の左辺の長さが可変であることを意味するのではなく、反復回数が可変であることを意味する。
      データ名-1によって参照されるデータ項目の値は、整数-1から整数-2の範囲に入らなければならない。データ名-1のデータ項目の値を小さくすると、それよりも出現番号の大きいデータ項目の内容は保証されない。
5. (ANS85)書き方2の OCCURS句で記述された項目を含む集団項目を参照すると、使用される表の領域は下記のように決定される。
   a. データ名-1によって参照されるデータ項目がその集団項目の外にあるならば、表領域のうちの参照処理が開始される時点で、そのデータ項目の値によって指定される部分が使用される。
   b. データ名-1によって参照されるデータ項目がその集団項目に含まれており、集団データ項目が送出し側としてだけ参照されているならば、表領域のうちの参照処理が開始される時点で、そのデータ項目の値によって指定される部分だけが使用される。その集団項目が受取り側になっている場合は、表の最大領域が使用される。
6. KEY IS句は、データ名-2、データ名-3などの値に従って、 昇順（ascending）または降順（descending）に反復データを並べることを指定する。昇順または降順は、作用対象の比較の規則に従って決められる。(*数字作用対象の比較*および*文字作用対象の比較*の節を参照。) これらのデータ名はキーの強さの順に左から右へ指定する。

8.1.1.15 PICTURE（形式）句

機能

PICTURE（形式）句は、基本項目の一般的性質と編集の形式を示す。

一般形式

{PICTURE / PIC} IS 文字列

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - INTLEVEL - 方言がMFのときに、数字データ項目に許される最大の長さを決定する。

構文規則

1. PICTURE句は、基本項目にだけ指定できる。
2. 文字列は、COBOL文字集合（character set）に含まれる文字を記号として用い、一定の規則に従って組み合わせたものである。この組合わせによって、基本項目の項類が決まる。
3. 文字列の長さは、最大30字である。
4. PICTURE句は、RENAMES（再命名）句の左辺に指定してはならない。
5. PICは、PICTUREの省略形である。
6. ゼロ抑制記号としての星印（*）とBLANK WHEN ZERO句は、同じ記述項に指定できない。(OSVS)ゼロ抑制記号としての星印（*）とBLANK WHEN ZERO句を同じ記述項に指定してもよい。ただし、ゼロ抑制の方がBLANK WHEN ZERO句に優先する。
7. 1文字に続けて、整数をかっこで囲んで指定すると、その文字を整数の値分の数だけ並べることを意味する。

一般規則

PICTURE句に書けるデータは、6種類の項類に分かれる。英字、数字、英数字、英数字編集、数字編集、
(OSVS)(VSC2)(MF)内部浮動小数点数
である。以下に、各項類ごとに分けて一般規則を記す。

英字（alphabetic）項目に関する規則

1. PICTURE文字列に書ける記号文字は、"A"と"B"だけである。
   (ANS85)PICTURE文字列に書ける記号文字は、"A"だけである。

2. 標準データ形式で表わした場合、その内容は1つ以上の英字とする。

数字（numeric）項目に関する規則

1. PICTURE文字列に書ける記号文字は、"9","P", "S", "V"だけである。 PICTURE文字列で表わせる桁数は、1以上18以内である。 (ISO2000)PICTURE文字列で表わせる桁数は、31までである。 (MF)PICTURE文字列で表わせる桁数は、38までである。
2. 符号を付けない場合、標準データ形式のデータはアラビア数字の"0", "1", "2", "3", "4", "5", "6", "7", "8", "9" を組み合わせたものとする。符号を付ける場合は、上記の数字の他に、"+", "-" およびその他の演算記号を含めることができる。(後述の*SIGN句*の節を参照。)

数字データは、標準データ形式以外の形式でも保持できる。(後述の*USAGE文*の節、および *COBOL言語の概念*の章の*文字の表現と基数の選定*の節を参照。)

英数字（alphanumeric）項目に関する規則

1. PICTURE文字列に書ける記号文字は、 "A", "X", "9" だけである。結果的には、記号文字をすべて"X" と書いたように扱われる。PICTURE文字列に含まれる記号文字がすべて"A"または"9"である場合、英数字項目にならない。
2. 標準データ形式で表わした場合、その内容は計算機の文字集合の任意の文字を含んでいてよい。

英数字編集（alphanumeric edited）項目に関する規則

1. PICTURE文字列に書ける記号文字は、 "A", "X", " 9", "B", "0", "/" を一定の規則に従って組み合わせたものに限られる。その規則は下記の通り。
    a. PICTURE文字列には、最低1つの "B"と最低1つの"X" 、または最低1つの "0" （ゼロ）と最低1つの"X" 、または最低1つの"/" （斜線）と最低1つの"X" を含ませる。または、
    b. PICTURE文字列には、最低1つの"0" （ゼロ）と最低1つの"A" 、または最低1つの"/" （斜線）と最低1つの"A" を含ませる。
2. 標準データ形式で表わした場合、その内容は計算機の文字集合の所定の文字を含むことができる。

数字編集（numeric edited）項目に関する規則

1. PICTURE文字列に書ける記号文字は、"B", "/", "P", "V" , "Z", "0", "9", ",", ".", "*", "+", "-", "CR", "DB", 通貨記号を、一定の規則に従って組み合わせたものに限られる。許容される記号文字の組合わせは、記号の優先順位によって決まる。その規則は下記の通り。
    a. PICTURE文字列で表わせる桁数は、1以上18以内である。 (ISO2000)PICTURE文字列で表わせる桁数は、31までである。 (MF)PICTURE文字列で表わせる桁数は、38までである。

b. PICTURE文字列には、最低1つの"0"，"B"，"/"，"Z"，"*"，"+"，","，"."，"-"，"CR"，"DB"，通貨記号を含ませる。

2. 標準データ形式で表わした場合、その内容は1つの数値とする。
3. PICTURE文字列のすべての桁位置が挿入文字によって表わされている場合、少なくとも1つの文字を小数点の左側に置く。

外部浮動小数点数

(OSVS) (VSC2) (MF)

1. PICTURE文字列は、下記の形式にする。

仮数部（mantissa）および指数部（exponent）の前に、符号文字を置く。
"+" 符号は、出力値が正であれば正号が付けられ、出力値が負であれば負号が付けられることを表わす。
"-" 符号は、出力値が正であれば符号は空白とされ、出力値が負であれば負号が付けられることを表わす。
各符号は、1バイトの記憶域を占める。
仮数部
仮数部には、下記の記号を含めることができる。
9 . V
実小数点は "."で表わし、想定小数点は "V"で表わす。仮数部には実小数点または想定小数点のどちらかがなければならない。小数点は左端にあっても、中間にあっても、右端にあってもよい。仮数部の長さは1桁以上16桁以下である。
E
指数を示す。
指数部
指数部は、PIC "99" の形式にする。

2. 外部浮動小数点数項目には、OCCURS，REDEFINES，RENAMESの各句を指定できる。
3. SIGN句は、注記としてだけ使用するものであり、符号の表現には影響しない。
4. SYNCHRONIZED句は、注記としてだけ使用する。
5. 外部浮動小数点数に関しては、下記の句は指定できない。
BLANK WHEN ZERO
JUSTIFIED
VALUE

基本項目の大きさ

基本項目の大きさとは、 標準データ形式で表わしたときの、基本項目の文字数をいう。基本項目の大きさは、 文字位置を表わすのに使用できる記号の数によって決定される。記号文字"A", ",", "X" , "9", "P", "Z", "*", "B", "/", "0", "+", "-", または通貨記号の後ろにかっこで囲んで書いた整数は、それらの記号を連続して並べる数を示す。 次の記号文字はPICTURE文字列内には１つしか書けないことに注意。"S", "V", ".",
(OSVS)(VSC2)(MF)"E",
"C R", "D B"

使用する記号

基本項目を表わすために使用する記号の働きを、以下に説明する。

A PICTURE文字列中の各 "A" は、1文字の英字または1つの空白だけを含むことのできる文字位置を表わす。

B PICTURE文字列中の各"B" は、空白文字を挿入できる文字位置を表わす。

(OSVS)(VSC2)(MF)**E** 外部浮動小数点数項目の指数部の始まりを示す。指数部は、実行時に1バイトの記憶域を占める。

P PICTURE文字列中の各"P" は、想定小数点の位置を合わせるための位取りを表わすものである。データ項目中に現れる位の中に小数点が含まれていないときに、想定小数点の位置を指定するために使用する。位取り文字である "P" は、データ項目の大きさには数えられない。数字編集項目または数字項目の最大桁数（18桁）を判定する際には、位取り文字は数に入れられる。この"P" は、PICTURE文字列の左端または右端に連続してだけ書くことができる。これは、"P" は想定小数点の位置を暗示するからである。つまり、PICTURE文字列の左端に "P" がある場合はその左に想定小数点があることになり、PICTURE文字列の右端に"P" がある場合はその右に想定小数点があることになる。"P"を含むPICTURE文字列の左端または右端に想定小数点記号の "V" を書くことは、定義の重複である。
"P" と挿入文字の"." 小数点）を、同一のPICTURE文字列に含めてはならない。データ項目の内部表現をあるものから別のものへ変換するときに、対象のデータ項目のPICTURE文字列に "P" が含まれていると、各"P" の位置に値ゼロが入っているものとみなされ、データ項目の大きさにはその桁位置も含まれる。
(ANS85)PICTURE文字列に記号 "P" が含まれるデータ項目を参照する処理で用いられるのは、そのデータ項目の実際の文字表現ではなく、そのデータ項目の代数値である。この代数値は、"P" の配置によって示される位置に小数点を想定し、各"P" をゼロで置き換えたものである。このデータ項目の大きさは、PICTURE文字列によって表わされる桁数となる。このような処理が行われる事例は下記のとおり。

a. 数字の送出し側作用対象を必要とする処理
b. 送出し側の作用対象が数字であり、そのPICTURE文字列に記号文字 "P" が含まれる場合の、基本MOVE（転記）文
c. 送出し側の作用対象が数字編集であり、そのPICTURE文字列に記号文字"P" が含まれ、受取り側の作用対象が数字または数字編集である場合の、MOVE文
d. 両方の作用対象が数字である場合の比較処理

(ANS85)上記以外のすべての処理では、記号文字 "P" は無視され、作用対象の大きさに含められない。

S PICTURE文字列中の"S" は、演算符号の存在を示す。しかし、演算符号の表現形式や位置までも示すものではない。"S"はPICTURE文字列の左端に書く。この記号文字は標準データ形式で数えた基本項目の大きさには含まれない。ただし、その記述項にSEPARATE CHARACTER句を指定した場合は、"S" は項目の大きさに含められる。(この章の*SIGN句*の節を参照。)

V PICTURE文字列中の "V" は想定小数点の位置を示す。この記号文字は1つのPICTURE文字列には1つだけ書ける。"V" は文字位置を表わすわけではないので、基本項目の大きさには含まれない。想定小数点がPICTURE文字列の右端の文字の右側にある場合は、"V" を書いても書かなくてもよい。

X PICTURE文字列中の各"X" は、1個の文字の位置を示す。そこには、計算機文字集合で許される任意の文字を含めることができる。

Z PICTURE文字列中の各 "Z" は、先行ゼロ列のゼロを空白に置き換えることを示す。"Z" は項目の大きさに含められる。

9 PICTURE文字列中の各 "9" は、数字を含む1個の桁位置を示す。"9" は項目の大きさに含められる。

0 PICTURE文字列中の各 "0" (ゼロ) は、数字のゼロを挿入すべき1個の桁位置を示す。"0" は項目の大きさに含められる。

/ PICTURE文字列中の各"/"(斜線) は、文字"/"を挿入すべき1個の桁位置を示す。 "/" は項目の大きさに含められる。

, PICTURE文字列中の各"," (コンマ) は、文字"," を挿入すべき1個の桁位置を示す。"," は項目の大きさに含められる。PICTURE文字列の最後には置けない。

(ANS85)挿入文字 "," を、PICTURE文字列の最後に置いてもよい。

. PICTURE文字列中に現れる文字"." (終止符) は、編集記号であり、桁合わせの基準とする小数点の位置を表わすとともに、文字"." を挿入する位置を表わす。"." は項目の大きさに含められる。特殊名段落にDECIMAL-POINT IS COMMAを指定すると、原始単位における終止符とコンマの機能が入れ替えられる。この場合、PICTURE句の中の終止符とコンマに関しても、終止符に関する規則がコンマに適用され、コンマに関する規則が終止符に適用される。挿入文字 "." は、PICTURE文字列の最後には置けない。

(ANS85)挿入文字 "." を、PICTURE文字列の最後に置いてもよい。

+, -, CR, DB これらの記号文字は符号編集用文字であり、符号編集用文字を置く位置を示す。1つのPICTURE文字列の中では、これらの記号文字のうちのどれか1つだけ使える。これらの文字は項目の大きさに含められる。

***** PICTURE文字列中の各"*" (星印) は、先行ゼロ列のゼロを星印に置き換えることを示す。"*" は項目の大きさに含められる。

通貨編集用文字 PICTURE文字列中の通貨編集用文字は、通貨編集文字を収める1個の文字位置を示す。通貨編集用文字は、通貨記号または特殊名段落にCURRENCY SIGN句で指定した1文字で表わされる。通貨編集用文字は、項目の大きさに含められる。 CURRENCYコンパイラ指令は、使用する通貨記号に影響を与える。

編集規則

PICTURE句の中で指定できる編集方法は、 挿入と ゼロ抑制の2通りある。挿入編集には下記の4種類がある。

1. 単純挿入 (simple insertion)
2. 特殊挿入 (special insertion)
3. 固定挿入 (fixed insertion)
4. 浮動挿入 (floating insertion)

ゼロ抑制編集には下記の2種類がある。

1. 空白によるゼロ抑制
2. 星印（*）によるゼロ抑制

項目に行える編集の種類は、その項類によって異なる。表8-1 にその関係を示す。

表 8-1: データの項類に適用できる編集の種類

項類	編集の種類
英字	"B" による単純挿入だけ[1]
数字	なし
英数字	なし
英数字編集	"0", "B" "/"による単純挿入
数字編集	すべて[2]
2バイト文字	単純挿入
外部浮動小数点数	特殊挿入

注:

1. (ANS85)ANSI '74標準では、"A" と"B" の両方が含まれるPICTURE句は英字に対する単純挿入として扱われていた。ANSI '85標準では英字に対する "B" を認めなくなり、両方が含まれるPICTURE句は、英数字編集に対する単純挿入として扱われるようになった。
2. 1つのPICTURE句の中に、浮動挿入編集とゼロ抑制編集を同時に指定することはできない。1つのPICTURE句の中には、1種類のゼロ抑制だけを指定できる。

単純挿入編集

","(コンマ), "B"(空白), "0"(ゼロ), " /"(斜線)は、挿入文字として使用される。挿入文字は項目中に文字を挿入すべき位置を表わし、項目の大きさに含められる。

特殊挿入編集

(OSVS)(VSC2)(MF)この種の編集は、数字編集項目または外部浮動小数点数項目に対して使用できる。"."(終止符)は、 は挿入文字として使用されるとともに、桁合わせの基準となる小数点を表わすためにも使用される。実小数点用に使用される挿入文字は、項目の大きさに含められる。同じPICTURE文字列の中に、記号文字"V" で表わされる想定小数点と、この挿入文字で表わされる実小数点を指定することはできない。特殊挿入編集を施すと、PICTURE文字列中に指定した位置に、指定した挿入文字が挿入される。

固定挿入編集

通貨編集用文字および符号編集用文字の"+" ,"-", "CR", "DB" は、挿入文字である。1つのPICTURE文字列の中では、1つの通貨編集用文字と、符号編集用文字のどれか1つだけを使用できる。符号編集用文字の "CR" または "DB" を使用する場合は、文字列の右端に置く。これらの符号編集用文字は項目の大きさとしては2文字分に数えられる。符号編集用文字の"+" または "-" を使用する場合は、文字列の右端または左端に置く。これらの符号編集用文字は項目の大きさとして数えられる。通貨編集用記号は、文字列の左端に置く。

表 8-2 : PICTURE 文字列における編集用記号

PICTURE句の文字列中の編集用記号	編集の結果挿入される文字	
	データ項目の値が正またはゼロ	データ項目の値が負
+	+	-
-	空白１個	-
CR	空白２個	CR
DB	空白２個	DB

浮動挿入編集

通貨編集用文字および符号編集用文字の"+"と"-" は浮動挿入文字である。これらの文字を、1つのPICTURE文字列に同時に指定することはできない。
浮動挿入編集を行うには、PICTURE文字列中に浮動挿入文字を2つ以上指定する。浮動挿入文字列の間または右隣に、単純挿入文字を書くことができる。それらの単純挿入文字は、浮動挿入文字列の一部となる。

浮動挿入文字に通貨編集用文字を使用した場合、その浮動挿入文字列の右隣に符号編集用文字の"+", "-", "CR" ,"DB" のどれか1つを書くことができる。
浮動挿入文字列の左右両端の文字はそれぞれ、データ項目中で浮動する記号の左右の限界を示す。

左端から2番目の浮動文字は、そのデータ項目の中で数字データを収めることができる左側の限界を示す。それ以降右側の浮動文字は、ゼロでない数字で置き換えられる。

PICTURE文字列の中では、浮動挿入文字を表わす方法は2通りである。1つは、小数点の左側の数字列の先頭の一部または全部を、挿入文字で書く方法である。もう1つは、PICTURE文字列の中のすべての数字位置を、挿入文字で書く方法である。

PICTURE文字列中の小数点の左側にだけ挿入文字を書いた場合、単一の浮動挿入文字が挿入される。その位置は、小数点のすぐ左、または挿入文字列が表わすデータのゼロでない最初の数字のすぐ左の、どちらか左よりの方となる。実際に挿入される挿入文字よりも左側の文字位置は、空白で置き換えられる。

PICTURE文字列の中のすべての数字位置を挿入文字で書いた場合、結果はデータの値によって異なる。値がゼロである場合は、そのデータ項目全体が空白で置き換えられる。値がゼロでない場合は、小数点の左側にだけ挿入文字を書いた場合と同じ結果となる。

(ANS85)PICTURE文字列の中のすべての数字位置を挿入文字で書いた場合、最低限1つの数字位置を、想定小数点または実小数点の左に置く。

切捨てが起こらないようにするためには、受取り側のPICTURE句の文字列の文字数を、送出し側のデータ項目の文字数に受取り側で編集するための固定挿入文字の数と浮動挿入文字のための1字とを加えたもの以上にする。

ゼロ抑制編集

数字位置の先行ゼロ列を抑制するためには、PICTURE文字列の中で英字 "Z" または "*" (星印)を使用する。1つのPICTURE文字列の中では、どちらか一方しか指定してはならない。各抑制文字は、項目の大きさに含められる。抑制文字に"Z" を指定した場合、先行ゼロ列は空白で置き換えられる。抑制文字に"*" を指定した場合、先行ゼロ列は星印で置き換えられる。

ゼロ抑制は、PICTURE文字列の中に1つ以上の抑制文字を並べて示す。この抑制文字列は、先行ゼロ列のゼロの置き換えを行おうとする先行数字位置を示す。抑制文字列の中またはそのすぐ右に書いた単純挿入文字は、その抑制文字の一部となる。

PICTURE文字列の中では、ゼロ抑制文字を表わす方法は2通りである。1つは、小数点の左側の数字列の先頭の一部または全部を、ゼロ抑制文字で書く方法である。もう1つは、PICTURE文字列の中のすべての数字位置を、ゼロ抑制文字で書く方法である。

PICTURE文字列中の小数点の左側にだけゼロ抑制文字を書いた場合、PICTURE文字列中のゼロ抑制文字に対応するデータ中の先行ゼロ列が、置換え文字によって置き換えられる。ゼロ抑制が停止される位置は、小数点またはデータ中のゼロでない最初の数字の、どちらか先に出てきた方となる。

PICTURE文字列の中のすべての数字位置をゼロ抑制文字で書いた場合、結果はデータの値によって異なる。　値がゼロでない場合は、小数点の左側にだけゼロ抑制文字を書いた場合と同じ結果となる。値がゼロである場合は、ゼロ抑制文字に"Z" を使用していれば、そのデータ項目全体は空白で置き換えられ、ゼロ抑制文字に"*" を使用していれば、そのデータ項目全体は実小数点を除いて星印で置き換えられる。

記号文字"+", "-", "*", " Z" および通貨編集用文字は、浮動置換え文字として使用するときは、1つのPICTURE文字列の中ではどれか1つだけを書く。

順序規則

PICTURE文字列の中で記号として使用するときの、文字の順序規則を表8-3に示す。"○" 印は、その列の上段に示した文字を、その行の左端に示した文字の前に書いてもよいことを示す。中かっこ{ }で囲んだ文字群は、その中の1つしか書けないことを示す。 " 通貨" は、通貨編集用文字を示す。PICTURE文字列の中には、文字 "A " , "○", "Z", " 9", "*", の少なくともどれか1つ、または記号文字"+" , "-"," 通貨" の少なくともどれか2つを書く。

表8-3の中で、固定挿入文字の"+"と"-" 、および浮動挿入文字の"Z", "*", " +","-", " 通貨", および記号文字"P" は、それぞれ2回ずつ示してある。 左の列および上の行は、小数点の左側に書く場合を表わす。右の列および下の行は、小数点の右側に書く場合を表わす。

表 8-3: PICTURE 文字列における順序規則

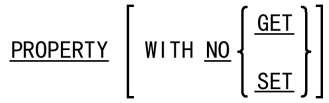
先行文字		固定挿入文字										浮動挿入文字						その他の文字						
後続文字		B	0	/	,	.	+ -	+ -	CR DB	通貨	E	Z *	Z *	+ -	+ -	通貨	通貨	9	A ○	S	V	P	P	G
固定挿入文字	B	○	○	○	○	○	○			○		○	○	○	○	○	○	○	○		○		○	○
	0	○	○	○	○	○	○			○		○	○	○	○	○	○	○	○		○		○	
	/	○	○	○	○	○	○			○		○	○	○	○	○	○	○	○		○		○	
	,	○	○	○	○	○	○			○		○	○	○	○	○	○	○			○		○	
	.	○	○	○	○		○			○		○		○		○		○						
	+ -																							
	+ -	○	○	○	○	○				○	○	○	○			○	○	○			○	○	○	
	CR DB	○	○	○	○	○				○		○	○			○	○	○			○	○	○	
	通貨						○																	
	E				○	○												○			○			
浮動挿入文字	Z *	○	○	○	○		○			○		○												
	Z *	○	○	○	○	○	○			○		○	○								○		○	
	+ -	○	○	○	○					○				○										
	+ -	○	○	○	○	○				○				○	○						○			
	通貨	○	○	○	○		○									○								
	通貨	○	○	○	○	○	○									○	○				○			
その他の文字	9	○	○	○	○	○	○			○	○	○		○		○		○	○	○	○		○	
	A ○	○	○	○														○	○					
	S																							
	V	○	○	○	○		○			○		○		○		○		○		○		○		
	P	○	○	○	○		○			○		○		○		○		○		○		○		
	P						○			○										○	○		○	
	G	○																						○

8.1.1.16 PROPERTY（属性）句

機能

(ISO2000)PROPERTY句はこのデータ項目がオブジェクトの属性であることおよび状況に応じてGETメソッドとSETメソッドのどちらか一方または両方を生成すべきことを指定する。

一般形式



構文規則

1. PROPERTY句を指定できる場所は、ファクトリ定義またはオブジェクト定義の作業場所節だけである。
2. OCCURS句の下位に属するデータ項目にはPROPERTY句を指定してはならない。
3. PROPERTY句を指定できる対象は、参照の一意性を保つために名前を修飾する必要のない、基本データ項目だけである。
4. 記述項の左辺のデータ名はスーパークラスに定義されている属性名と同じであってはならない。

注： スーパークラス内に属性名を定義できる。その方法は２通りある。ひとつは、PROPERTY指定を記述したひとつまたは一組のメソッドを定義することである。もうひとつは、PROPERTY句を指定したデータ記述項を記述することである。

一般規則

1. GET指定を書かないと、PROPERTY句は該当のデータ項目を含むファクトリまたはオブジェクトに関するメソッドを定義する働きをする。
   その記述項の左辺の字類が指標、オブジェクト、ポインタのいずれかである場合、そのメソッドは暗黙的に下記のように定義される。

```
METHOD-ID. GET PROPERTY データ名.
DATA DIVISION.
LINKAGE SECTION.
01 LSデータ名 データ記述.
PROCEDURE DIVISION RETURNING LSデータ名.
パラメータ名.
          SET LSデータ名 TO データ名
          EXIT METHOD.
END METHOD.
```

   その記述項の左辺の項類が英数字編集、国別編集、数字編集のいずれかである場合、そのメソッドは暗黙的に下記のように定義される。

```
METHOD-ID. GET PROPERTY データ名.
DATA DIVISION.
LINKAGE SECTION.
01 LSデータ名 データ記述.
PROCEDURE DIVISION RETURNING LSデータ名.
パラメータ名.
          MOVE データ名 TO LSデータ名 (1:).
          EXIT METHOD.
END METHOD.
```

注： その記述項の左辺が編集されている場合、受取り側の項目全体を部分参照したときに、そのデータに編集規則が再び適用されることはない。

そうでない場合、このメソッドの暗黙的な定義は下記のようになる。

```
METHOD-ID. GET PROPERTY データ名.
DATA DIVISION.
LINKAGE SECTION.
01 LSデータ名 データ記述.
PROCEDURE DIVISION RETURNING LSデータ名.
パラメータ名.
          MOVE データ名 TO LSデータ名
          EXIT METHOD.
END METHOD.
```

ここで、LSデータ名のデータ記述には、その記述項の左辺およびその下位のデータ項目が含まれる。ただし、下記のものは例外である。

- INDEXED BY指定
- PROPERTY句
- VALUE句
- その記述項の左辺のデータ記述内のREDEFINES句

2. SET指定を書かないと、PROPERTY句は該当のデータ項目を含むファクトリまたはオブジェクトに関するメソッドを定義する働きをする。
その記述項の左辺の字類が指標、オブジェクト、ポインタのいずれかである場合、そのメソッドは暗黙的に下記のように定義される。

```
METHOD-ID. SET PROPERTY データ名.
DATA DIVISION.
LINKAGE SECTION.
01 LSデータ名 データ記述.
PROCEDURE DIVISION USING LSデータ名.
パラメータ名.
          SET データ名 TO LSデータ名
          EXIT METHOD.
END METHOD.
```

その記述項の左辺の項類が英数字編集、国別編集、数字編集のいずれかである場合、そのメソッドは暗黙的に下記のように定義される。

```
METHOD-ID. SET PROPERTY データ名.
DATA DIVISION.
LINKAGE SECTION.
01 LSデータ名 データ記述.
PROCEDURE DIVISION USING LSデータ名.
パラメータ名.
          MOVE LSデータ名 TO データ名 (1:).
          EXIT METHOD.
END METHOD.
```

注： その記述項の左辺が編集されている場合、受取り側の項目全体を部分参照したときに、そのデータに編集規則が再び適用されることはない。

そうでない場合、このメソッドの暗黙的な定義は下記のようになる。

```
METHOD-ID. SET PROPERTY データ名.
DATA DIVISION.
LINKAGE SECTION.
01 LSデータ名 データ記述.
PROCEDURE DIVISION USING LSデータ名.
par-name.
          MOVE LSデータ名 TO データ名
          EXIT METHOD.
END METHOD.
```

ここで、LSデータ名のデータ記述には、その記述項の左辺およびその下位のデータ項目が含まれる。ただし、下記のものは例外である。

- INDEXED BY指定
- PROPERTY句
- VALUE句
- その記述項の左辺のデータ記述内のREDEFINES句

8.1.1.17 RECORD（レコードの大きさ）句

機能

(ANS85)RECORD句は、固定長レコード内の文字数または可変長レコード内の文字数の範囲を指定する。可変長レコードに関しては、最小文字数と最大文字数とを指定する。

(MF)固定形式のファイルおよび 可変形式のファイル の概念は、この節全体を通じて、行順ファイルには適用しない。使用するCOBOL での行順ファイルの使用についての詳細は、COBOLシステムのマニュアルを参照。

(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、RECORD CONTAINS句は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこの句を使用するべきではない。

一般形式

書き方1

```
RECORD CONTAINS 整数-1 CHARACTERS
```

書き方2

```
RECORD IS VARYING IN SIZE                                    (ANS85)
      [[FROM 整数-2][TO 整数-3]CHARACTERS]
      [DEPENDING ON データ名-1]
```

書き方3

```
RECORD CONTAINS 整数-4 TO 整数-5 CHARACTERS
```

構文規則

書き方1

1. 該当するファイルのレコード記述項に記述する文字数は、整数-1と同じにする。

書き方2

2. (ANS85)該当するファイルのレコード記述項に記述する文字数は、整数-2に指定する文字数より少なくても、整数-3に指定する文字数より多くてもいけない。
3. (ANS85)整数-3は、整数-2よりも大きくする。
4. (ANS85)データ名-1は、作業場所節または連絡節中で符号なしの整数の基本項目として記述しておく。

一般規則

すべての書き方

1. RECORD句を指定しないと、レコード記述項に記述する各データの大きさの和がレコードの大きさとなる。
2. (ANS85)対応するファイル結合子が外部ファイル結合子である場合、そのファイル結合子と関連する実行単位中のすべてのファイル記述項において、整数-1、整数-2、整数-3に同じ値を指定しなければならない。RECORD句を指定しない場合、そのファイル結合子に関連するレコード記述項は、すべて同じ長さにする。
3. (MF)(XOPEN)行順ファイルは、固定形式ファイルでも可変形式ファイルでもない。このため、RECORDING MODE、RECORD CONTAINS、RECORD VARYING IN SIZE句のどれかを書いても書かなくても、その性能には影響しない。

書き方1

4. 書き方1は、固定長レコードを定義するために使用する。整数-1には、ファイル中の各レコードの文字数を指定する。

書き方2

5. (ANS85)書き方2は、可変長レコードを定義するために使用する。整数-2には、ファイル中のレコードの中で最短のものの文字数を指定する。整数-3には、ファイル中のレコードの中で最長のものの文字数を指定する。
6. (ANS85)レコードの文字数は、その中に含まれる基本データ項目の文字数を合計することによって決まる。ただし、再定義および再命名分を除く。また、桁詰めによって暗黙のFILLERがとられている場合は、それも数に含める。表を指定した場合は、下記のように扱われる。
    a. レコード中に記述した表要素の最小数が、上記の合計に含められて、そのレコードの最短文字数が決定される。
    b. レコード中に記述した表要素の最大数が、上記の合計に含められて、そのレコードの最長文字数が決定される。
    c. 整数-2を指定しないと、該当するファイルのレコードの最短文字数は、そのファイルのレコード用に記述した最小の文字数に等しいものとされる。
7. (ANS85)整数-3を指定しないと、該当するファイルのレコードの最長文字数は、そのファイルのレコード用に記述した最大の文字数に等しいものとされる。
8. (ANS85)データ名-1を指定した場合、該当するファイルに対してRELEASE文、REWRITE文、WRITE文を実行する前に、該当するレコードの文字数をデータ名-1のデータ項目に設定する。
9. (ANS85)データ名-1を指定した場合、DELETE文、RELEASE文、REWRITE文、START文、WRITE文を実行しても、データ名-1のデータ項目の内容は変わらない。READ文またはRETURN文の実行が不成功に終わった場合も、データ名-1のデータ項目の内容は変わらない。
10. (ANS85)RELEASE文、REWRITE文、WRITE文を実行する間、該当するレコード中の文字数は、下記の条件によって決定される。
    a. データ名-1を指定した場合、データ名-1のデータ項目の内容によって。
    b. データ名-1を指定せず、レコードに可変反復データ項目が含まれない場合、そのレコードの文字数によって。
    c. データ名-1を指定せず、レコードに可変反復データ項目が含まれる場合、固定部分と出力文の実行時の反復数に基づく可変な表部分との和によって。

    書き出す論理レコードの文字数が、整数-2よりも少ないか整数-3よりも多いと、出力文は不成功に終わる。そして、対応する入出力状態に、その原因となった条件を示す値が設定される。ただし、RELEASE文の実行中は除く。
11. (ANS85)データ名-1を指定した場合、該当するファイルに対するREAD文またはRETURN文の実行が正常に終了すると、データ名-1のデータ項目の内容は、読み込んだばかりのレコードの文字数を示すように変更される。
12. (ANS85)READ文またはRETURN文の中にINTOを指定すると、暗黙のMOVE文の送出し側データ項目となる現在レコード中の文字数は、下記の条件によって決定される。

a. データ名-1を指定した場合は、データ名-1のデータ項目の内容によって。
b. データ名-1を指定しなかった場合は、データ名-1を指定していたならばデータ名-1のデータ項目に転記されるはずの値によって。

書き方3

13. RECORD句の書き方3を使用すると、整数-4が最短のデータレコードの文字数を表わし、整数-5が最長のデータレコードの文字数を表わす。しかし、この場合は、各データレコードの大きさは、レコード記述項に完全に定義されている。
14. 各データレコードの大きさは、論理レコードを格納するのに必要な文字数で指定する。その際、論理レコード内のデータ項目を表わすために使用される文字の型は問わない。レコードの大きさは、すべての固定長の基本項目の文字数の合計に、可変長項目が含まれればその最大の文字数の合計を加えたものによって、決定される。この合計値が実際のレコードの大きさと違っていてもよい。*COBOL言語の概念*の章の*文字の表現*と*基数の選定*の節、およびこの章の*SYNCHRONIZED句*と*USAGE句*の節を参照。

8.1.1.18 RECORDING MODE句

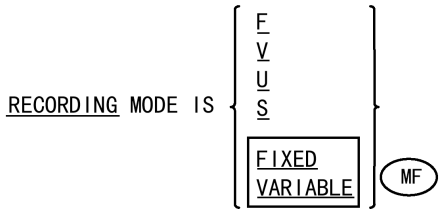
(OSVS) (VSC2) (MF)

機能

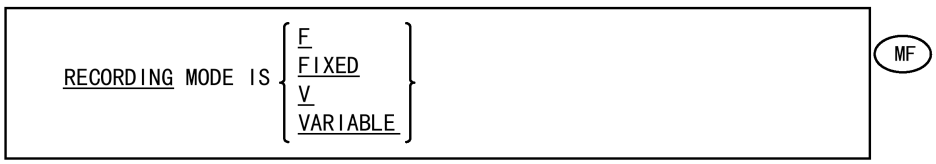
RECORDING MODE句は、ファイル中の論理レコードの形式を指定する。

一般形式

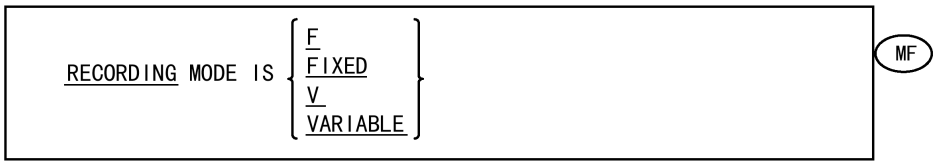
書き方1（レコード順ファイル）



書き方2（行順ファイル）



書き方3（相対ファイルおよび索引ファイル）



構文規則

1. (MF)FとFIXEDは同義語である。
2. (MF)V と VARIABLは同義語である。

一般規則

すべての書き方(すべてのファイル)

1. RECORDING MODE IS Fを指定すると、ファイル中のすべてのレコードの長さが同じとなる。

書き方1（レコード順ファイル）

2. (MF)"U" 指定は注記になる。

書き方1と3（レコード順ファイル、相対ファイル、索引ファイル）

3. RECORDING MODE IS V句を指定すると、 ファイル中のレコードは固定長でも可変長でもよくなる。各データレコードに、レコード長フィールドがとられる。このレコード長フィールドは、レコード記述の一部とは扱われない。

書き方2

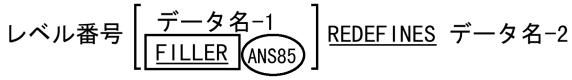
4. (MF)行順ファイルは、固定形式ファイルでも可変形式ファイルでもない。このため、RECORDING MODE、RECORD CONTAINS、RECORD VARYING IN SIZE句のどれかを書いても書かなくても、その性能には影響しない。

8.1.1.19 REDEFINES（再定義）句

機能

REDEFINES（再定義）句は、同一の計算機記憶領域を別のデータ記述項によって記述する。

一般形式



上記のレベル番号とデータ名-1

(ANS85)またはFILLER

は書き方を明確にするために示してある。レベル番号とデータ名-1

(ANS85)またはFILLER

は、REDEFINES句の一部を構成するわけではない。

構文規則

1. REDEFINES句を指定するときは、データ名-1
   (ANS85)またはFILLERの直後に書く。
   (MF)あるいは、PICTURE句またはUSAGE句の後ろにREDEFINES句を書くことができる。
2. データ名-1とデータ名-2のレベル番号は、同じにする。ただし、 66,
   (MF)78,
   88は使用できない。
3. REDEFINES句は、ファイル節内の01レベルの記述項の中では使用できない。レベル指示語のFDまたはSDの下位に属する複数の01レベルの記述項は、同じ領域を暗黙的に再定義するからである。(この章の *DATA RECORDS句*の節の一般規則2を参照。)
   (MF)この句を、ファイル節内の01レベルの記述項内で使用してもよい。
4. REDEFINES句は、通信節内の01レベルの記述項の中では使用できない。レベル指示語のCDの下位に属する複数の01レベルの記述項は、同じ領域を暗黙的に再定義するからである。
5. (OSVS)(VSC2)データ名-2のデータ記述項に、REDEFINES句が含まれていてもよい。
   データ名-2は、REDEFINES句が含まれている記述項の下位に属していてもよい。データ名-2のデータ記述項には、OCCURS句が含まれていてはならない。しかし、OCCURS句が含まれているデータ記述項の下位に、データ名-2が属していても構わない。この場合、REDEFINES句の中でデータ名-2を参照するときに、添字または指標は指定できない。元の定義および再定義のどちらも、OCCURS句で指定される可変反復データ項目を含んでいてはならない。(この章の*OCCURS句*の節を参照。)
   (MF)データ名-2のデータ記述項に、OCCURS句が含まれていてもよい。
   (MF)レベル番号が01の場合、元の定義と再定義は、OCCURS句で定義したサイズが可変の項目を含むことができる。
6. データ名-2とデータ名-1のデータ記述項の間に、データ2およびデータ1のレベル番号よりも数値的に値の小さいレベル番号の記述項は置けない。
7. (OSVS)(VSC2)(MF)外部浮動小数点数データ項目または内部浮動小数点数データ項目を、REDEFINES句の左辺または右辺に置いてもよい。
8. (ANS85)データ名-2が一意でない場合でも、データ名-2を修飾してはならない。この場合は、REDEFINES句の配置に関する規則上、データ名-2に曖昧さはないからである。
   (OSVS)データ名-2を修飾してもよいが、記述した修飾は無視される。

9. 新しい文字位置の記述を与える項目は、VALUE句を持ってはならない。ただし、条件名の項目内では例外である。
10. (ISO2000)REDEFINES句は、字類オブジェクトのデータ項目に指定してはならない。
    (MF)REDEFINES句を、字類オブジェクトのデータ項目に指定してもよい。
11. (ISO2000)データ名-2は、字類オブジェクトであってはならない。
    (MF)データ名-2は、字類オブジェクトでもよい。

一般規則

1. 再定義はデータ名-2から始まり、データ名-2のレベル番号以下のレベル番号が出てきたところで終わる。
2. データ名-1のレベル番号が01でないときは、データ名-2によって参照されるデータ項目と同じ文字数を指定する。
   (ANS85)(OSVS)ただし、データ名-1の領域が、データ名-2の領域よりも小さくてもよい場合を除く。
   (MF)ただし、データ名-1の領域がデータ名-2の領域よりも大きくてもよい場合を除く。この場合、再定義する項目と再定義される項目の大きい方を収められるだけの記憶領域がとられる。
   REDEFINES句は記憶領域を再定義するのであって、記憶されているデータ項目を再定義するのではないことを、認識することが重要である。
3. 同じ文字位置を、複数回、再定義できる。 再定義用の記述項は、再定義される領域を定義する記述項の直後に続ける。その間に、新しい領域を定義する記述項ははさめない。同じ領域を複数回再定義するときは、最初に領域を定義した記述項のデータ名を使用する。
   (OSVS)(VSC2)(MF)または、先行する再定義記述項のデータ名を使用してもよい。
4. レベル指示語（FD，CD，SD）の下位に属する複数の01レベルの記述項は、同じ領域を暗黙的に再定義する。

8.1.1.20 RENAMES（再命名）句

機能

RENAMES（再命名）句は、基本項目を組み合わせて新たな集団を作る。組合わせは互いに重ね合わせることができる。

一般形式

```
66 データ名-1 RENAMES データ名-2 [ { THROUGH } データ名-3 ].
                                  { THRU    }
```

上記のレベル番号66とデータ名-1は書き方を明確にするために示してあるだけで、レベル番号とデータ名-1はRENAMES句の一部を構成するわけではない。

構文規則

1. 1つの論理レコードに対して書けるRENAMES記述項の数に、制限はない。
2. 1つの論理レコード中のデータ項目を参照しているRENAMES記述項はすべて、そのレコード記述項の最後のデータ記述項の直後にまとめて書く。
3. データ名-2およびデータ名-3は、同一レコード中の基本項目の名前または集団項目の名前とする。しかし、同じデータ名であってはならない。66レベルの記述項は77レベル、
   (MF)78レベル、
   88レベル、01レベルまたは別の66レベルの記述項を再命名することはできない。
4. データ名-1は、修飾語として使用できない。データ名-1は対応する01レベルまたはFD，CD，SDの記述項の名前によってだけ修飾できる。データ名-2の記述項にもデータ名-3の記述項にも、OCCURS句が含まれていてはならない。また、データ名-2の記述項もデータ名-3記述項も、データ記述項にOCCURS句が含まれる項目の下位に属していてはならない。(前述の*OCCURS句*の節を参照。)
5. データ名-3によって表わされる領域の始点は、データ名-2によって表わされる領域の始点よりも左側にあってはならない。データ名-3によって表わされる領域の終点は、データ名-2によって表わされる領域の終点よりも右側にあってはならない。したがって、データ名-3はデータ名-2の下位に属していてはならない。
6. データ名-2およびデータ名-3は修飾できる。
7. THRUとTHROUGHは同義語であり、どちらを書いてもよい。
8. データ名-2からデータ名-3の範囲内にある項目には、OCCURS句によって定義される可変反復データ項目が含まれていてはならない。(前述の*OCCURS句*の節を参照。)

一般規則

1. データ名-3を指定すると、データ名-1は集団項目となる。この集団項目には、データ名-2（データ名-2が基本項目の場合）またはデータ名-2の中の最初の基本項目（データ名-2が集団項目の場合）から始まり、データ名-3（データ名-3が基本項目の場合）またはデータ名-3の中の最後の基本項目（データ名-2が集団項目の場合）で終わる、すべての基本項目が含まれる。
2. データ名-3を指定しないと、データ名-2のすべての属性がデータ名-1のデータの属性となる。
3. (OSVS)データ名-2およびデータ名-3を明示的に修飾しなかった場合、参照の曖昧性がなければ、01レベルの項目による暗黙の修飾がなされる。

8.1.1.21 SIGN（符号）句

機能

SIGN（符号）句は、演算符号の特性を明示的に記述する必要がある場合に、その演算符号の位置と表現形式を指定する。

一般形式

```
[SIGN IS] { LEADING  } [SEPARATE CHARACTER]
          { TRAILING }
```

構文規則

1. SIGN句を指定する対象となるのは、PICTURE文字列に "S" が含まれる数字データ記述項またはそのような数字データ記述項が少なくとも1つ含まれる集団項目だけである。
2. SIGN句を適用する対象の数字データ項目の用途は、明示的または暗黙的に、DISPLAYとする。
3. 数字データ記述項にSIGN句を記述する場合は、1つだけとする。
   (ANS85)この規則は廃止する。
4. ファイル記述項の中でCODE-SET（符号系）句を指定するときは、そのファイル記述項の下位に属する符号付きの数字項目に対して、SIGN IS SEPARATE（独立符号）句を記述する。
   (MF)この規則は強制しない。（前述の*CODE-SET句*を参照。）

一般規則

1. SIGN句は、それを指定した数字データ項目またはそれを指定した集団項目の下位に属する数字データ項目の、演算符号の位置と表現形式を指定する。SIGN句は、PICTURE文字列に"S" が含まれる数字データ記述項に対してだけ、適用できる。PICTURE文字列中の "S" は符号が付くことを表わすだけであり、符号の表現形式や位置を表わすわけではない。
2. PICTURE文字列に "S" が含まれるがSIGN句は指定されていない数字データ項目は、演算符号をもつが、"S" によって符号の表現形式や位置が指定されているわけではない。このような符号付き数字データ項目には、一般規則の3から5は当てはまらない。SIGN句を指定しない場合の演算符号の表現は、*COBOL言語の概念*の章の*文字の表現と基数の選定*の節を参照。
3. SEPARATE CHARACTER句を指定しないと、符号の位置と表現形式は下記のようになる。
   a. 演算符号は、LEADINGまたはTRAILINGに応じて、基本データ項目の左端または右端の桁位置に付く。具体的な符号の付き方は、*COBOL言語の概念*の章の*文字の表現と基数の選定*の節を参照。
   b. PICTURE文字列中の記号文字"S" は、標準データ形式で表わした項目の大きさには含まれない。

4. SEPARATE CHARACTER句を指定すると、符号の位置と表現形式は下記のようになる。
    a. 演算符号は、LEADINGまたはTRAILINGに応じて、基本データ項目の左端または右端の桁位置に付く。これは桁位置には入らない。
    b. PICTURE文字列中の記号文字 "S" は、標準データ形式で表わした項目の大きさに含まれる。
    c. 正号および負号は、それぞれ、標準データ形式文字の"+" と "-" である。
5. PICTURE文字列に記号文字"S" が含まれる数字データ記述項はすべて、符号付数字データ記述項となる。SIGN句付の項目に対して計算や比較のために変換が必要なときには、その変換は自動的に行われる。
6. (ANS85)SIGN句付の集団項目の下位に属する項目(基本数字項目でも集団項目でも)にSIGN句を指定すると、下位の項目に指定したSIGN句が優先する。
7. (OSVS)(VSC2)(MF)外部浮動小数点数項目に関しては、SIGN句は注記としてだけ用いられる。内部浮動小数点数項目に対しては、SIGN句は無効であり、使用すると診断メッセージが出される。

8.1.1.22 SYNCHRONIZED（桁詰め）句

機能

SYNCHRONIZED（桁詰め）句は、計算機記憶の固有の境界に従い、基本項目の配置を指定する。(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/Open COBOL言語定義では、この機能は明示的に除外されている。したがって、X/Openに準拠するCOBOL原始プログラムにおいてはこの機能を使用するべきではない。

一般形式

```
{SYNCHRONIZED}[LEFT ]
{SYNC        }[RIGHT]
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - IBMCOMP – NOIBMCOMPを指定しないと、SYNCHRONIZED句は説明のためだけの記述として扱われる。

構文規則

1. SYNCHRONIZED句は、基本項目に対してだけ指定できる。
   (OSVS)(VSC2)集団項目に対しても、SYNCHRONIZED句を指定できる。
2. SYNCは、SYNCHRONIZEDの省略形である。

一般規則

1. SYNCHRONIZED RIGHTは、基本項目を計算機の語の境界の右端の文字位置で終わらせるように割り付ける。
   この句は、 IBMCOMP指令 を設定したときだけ、効力を発揮する。
2. SYNCHRONIZED句の中でLEFT指定を指定しても、効力はない。
3. SYNCHRONIZED句の効力は、定義上、作成者がどのように実装したかに左右される。
4. SYNCHRONIZED句を指定すると、計算機の語の左右の境界の間に、対象とするもの以外はいっさい入り込まないように、データ項目が割り付けられる。
   割付対象のデータ項目を格納するために必要な文字位置の個数が、計算機の語の左右の境界の間の文字位置の個数よりも少ないとき、使用されない文字位置は他のデータ項目のためには使われない。しかし、この使用されない文字位置は、下記のようになる。
   a. その基本項目が属する集団項目の大きさには含まれる
   b. そのデータ項目がREDEFINES句の右辺であるときには、再定義される文字位置に含まれる

   SYNCHRONIZED句によって基本項目の大きさが変更されることはないが、余分の文字位置が割り当てられる。
5. SYNCHRONIZED句にRIGHTもLEFTも指定しないと、実行効率がよくなるように、基本項目が計算機の語の左右の境界の間に割り付けられる。
6. SYNCHRONIZED句を指定した項目が 参照され、その大きさに依存する操作（桁寄せ、切捨て、桁あふれなど）が行われるときには、PICTURE句に指定した元の大きさが用いられる。
7. データ項目にSYNCHRONIZED句および演算符号を指定すると、そのデータ項目の符号は通常の演算符号の位置に付けられる。この場合、SYNCHRONIZED LEFTまたはSYNCHRONIZED RIGHTのどちらを指定したかは、演算符号の位置に影響しない。
8. OCCURS句を伴うデータ項目のデータ記述項、またはその下位に属するデータ項目のデータ記述項にSYNCHRONIZED句を指定すると、下記のようになる。
   a. 反復される各データ項目が桁詰めされる。
   b. 同じ表の中の他のデータ項目のために暗黙的に生成されるFILLERは、反復される各データ項目に対して生成される。
9. (OSVS)(VSC2)集団項目にSYNCHRONIZED句を指定すると、その集団項目に属するすべての項目に適用される。
10. SYNCHRONIZED句を使用したときの効果については、*COBOL言語の概念*の章の*文字の表現と基数の選定*の節を参照。

8.1.1.23 TYPEDEF句

(MF)

機能

TYPEDEF句は、プログラマによって定義された構造体または使用法としてレコードを定義する。

一般形式

```
IS TYPEDEF
```

構文規則

1. TYPEDEF句は、レベル番号01を持つデータ記述項でのみ、指定可能である。
2. TYPEDEF句を指定すると、次の句は指定できない。
   - EXTERNAL
   - GLOBAL
   - OCCURS
   - REDEFINES
   - SYNCHRONIZED/SYNC
   - VALUE

   集団項目にTYPEDEF句が指定されると、従属項目はOCCURSまたはREDEFINES句を使用して指定できる。

   VALUE句は、TYPEDEF句を指定するデータ記述項目にも、また、TYPEDEF構造体内の条件名（88レベル項目）を除く、どの従属項目にも指定できない。
3. TYPEDEF句がデータ記述に指定された場合、そのデータ記述にデータ名を含めなければならない。つまり、これを明示または暗示のFILLER句で指定してはならない。
4. TYPEDEF句がEXTERNALプログラム （CALLプロトタイプ）に指定された場合、 TYPEDEF句はコンパイル集団の任意の後続原始コード行で参照することができる。

一般規則

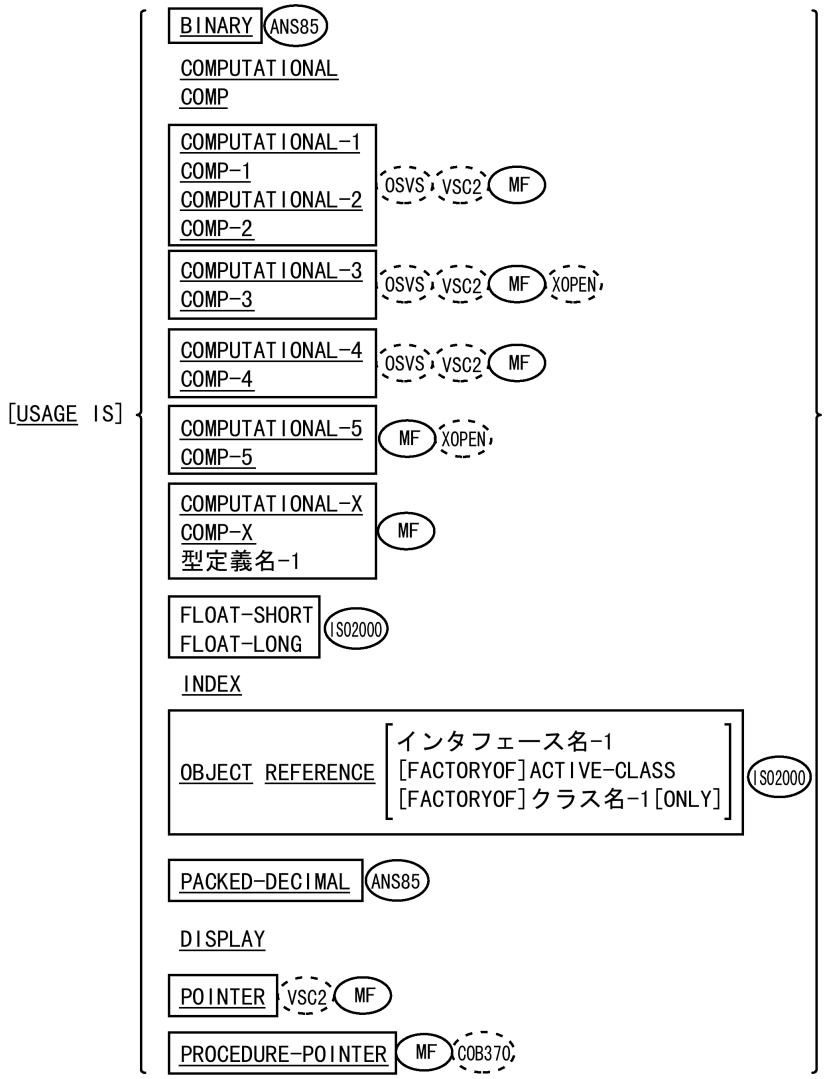
1. TYPEDEF句の使用目的は、後にUSAGE句で参照できるプログラマ定義の使用法または構造体を作成することにある。
2. TYPEDEF句を使用して宣言されたレコードは、記憶域を割り当ないが、データ名-1を型定義名-1として、後続のデータ記述項に指定できることを宣言する。

8.1.1.24 USAGE（用途）句

機能

USAGE（用途）句は、計算機記憶内の データ項目の表現形式を指定する。

一般形式



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - COBOL370 - USAGE PROCEDURE-POINTERデータ項目に割り当てる記憶領域を４バイトとするか８バイトとするかを制御する。
    - COMP-5 - USAGE COMP-5データ項目に格納するときに、演算符合の取扱いを制御する。
    - IBMCOMP - USAGE COMPおよびUSAGE COMP-5のデータ項目に割り当てる記憶領域を制御する。

構文規則

1. USAGE句はレベル66または88のデータ記述項には指定できない。
2. USAGE句を集団項目の記述項に書いた場合、それに属するすべての基本項目または集団項目の記述項にも書いて構わない。しかし、どちらの項目にも同じ用途を記述しなければならない。
3. 宣言にUSAGE句を持つ基本データ項目、または宣言にUSAGE句を持つ集団項目に従属する基本データ項目は、USAGE句がCOMPUTATIONAL
   (ANS85), BINARY, PACKED-DECIMAL
   (OSVS)(VSC2)(MF)(XOPEN), COMPUTATIONAL-3
   (OSVS)(VSC2)(MF), COMPUTATIONAL-4
   (MF)(XOPEN), COMPUTATIONAL-5
   のどれかを記述している場合、数字項目を定義するPICTURE文字列で記述しなければならない。 (たとえば、PICTURE文字列が記号 "P", "S", "V", "9" だけを含んでいること。)詳細は、前述の*PICTURE句*の節を参照。
   (MF)宣言にUSAGE句を持たない基本データ項目が、宣言にUSAGE句を持つ集団項目に従属する場合、そのUSAGE句は、英字、英数字、英数字編集、数字編集のデータ項目を定義したPICTURE文字列で記述されたリストの形式で記述してもよい。
4. (MF)(XOPEN)宣言にUSAGE句を持つ基本データ項目、または宣言にUSAGE句を持つ集団項目に従属する基本データ項目は、USAGE句が COMPUTATIONAL-5
   (MF)または COMPUTATIONAL-X
   (MF)(XOPEN)を記述している場合、数字項目を定義するPICTURE文字列で記述しなければならない。
   (MF)または、サイズが1から8バイトの数字項目を定義するPICTURE文字列で記述しなければならない。サイズが 1, 2, 3, 4, 5, 6, 7, 8 バイトの数字項目の場合、これはそれぞれ2, 4, 7, 9, 12, 14, 16, 18 の小数点位置である整数項目を記述したことと同じである。
   (MF)COMPUTATIONAL-X を指定し、PICTURE文字列が数字項目 定義している場合、その項目には符号を付けてはならない。
5. BLANK WHEN ZERO句、JUSTIFIED句、SIGN句は、用途が明示的にも暗黙的にも DISPLAY でないデータ項目には指定してはならない。

6. SYNCHRONIZED句およびVALUE句は、用途がINDEXのデータ項目に指定してはならない。
   (OSVS)(VSC2)(MF)SYNCHRONIZED句は、用途がINDEXのデータ項目に指定してもよい。
7. COMPは、COMPUTATIONALの省略形である。
   (OSVS)(VSC2)(MF)COMP-1 はCOMPUTATIONAL-1の省略形である。
   COMP-2 はCOMPUTATIONAL-2の省略形である。
   (OSVS)(VSC2)(MF)(XOPEN)COMP-3は、COMPUTATIONAL-3の省略形である。
   (OSVS)(VSC2)(MF)COMP-4は、COMPUTATIONAL-4の省略形である。
   (MF)(XOPEN)COMP-5 はCOMPUTATIONAL-5の省略形である。
   (MF)COMP-X はCOMPUTATIONAL-Xの省略形である。
8. 指標項目を明示的に参照できるのは、SEARCH（表引き）文、SET（設定）文、比較条件、手続き部の見出しのUSING（使用）句、
   (OSVS)(VSC2)(MF)ENTRY（導入）文のUSING句、
   またはCALL（呼ぶ）文のUSING句だけである。
9. (VSC2)(MF)ポインタ
   (MF)(COB370)または手続きポインタ
   (VSC2)(MF)項目を明示的に参照できるのは、SET（設定）文、比較条件、CALL（呼ぶ）文のUSING句、手続き部の見出しのUSING（使用）句、ENTRY（導入）文のUSING句、
   (MF)または CALL（呼ぶ）文のGIVING 句だけである。
10. USAGE IS INDEX,
    (ISO2002)USAGE IS OBJECT,
    (VSC2)(MF)USAGE IS POINTER,
    (MF)USAGE IS PROCEDURE-POINTER
    句のどれかで定義された基本データ項目は、条件変数であってはならない。
11. (MF)型定義名-1は、 TYPEDEF句のレコードと同じ原始ファイル内で前に定義しなければならない。
12. (MF)USAGE 型定義名-1を指定した場合、以下の句は指定できない。
    - BLANK
    - JUSTIFIED
    - PICTURE
    - SIGN
    - SYNCHRONIZED
    - VALUE
13. (MF)同じ階層の高位レベルに明示的なUSAGE 句がある場合、USAGE 型定義名-1の指定はエラーとなる。
14. (MF)USAGE 型定義名-1を指定した項目の直後に、従属する項目(78以外の高位レベル番号の項目)を記述した場合、エラーとなる。
15. (ISO2002)USAGE OBJECT REFERENCE句は、集団項目用のデータ記述項に指定してはならないが、従属する基本データ項目には指定してもよい。
16. (ISO2002)USAGE OBJECT REFERENCE句をファイル節に記述してはならない。
    (MF)USAGE OBJECT REFERENCE句をファイル節に記述してもよい。

注:オブジェクト参照が、アプリケーションロジックのアクティブなオブジェクトを参照しているかどうかによる。

17. (ISO2000)ACTIVE-CLASS指定は、ファクトリ定義、オブジェクト定義、メソッド定義にだけ記述できる。

一般規則

1. USAGE句を集団のレベルに書くと、そのUSAGE句はその集団項目に属するすべての基本項目に適用される。
   (MF)ただし、PICTURE句があり、データ項目の定義が文字である場合を除く。
2. USAGE句は、計算機記憶内でのデータ項目の表現形式を指定する。USAGE句は、データ項目の使い方には影響を与えない。ただし、手続き部の文の中には、作用対象を特定のUSAGE句を伴う基本項目に限定するものがある。USAGE句は項目の基数または文字表現の種類に影響することがある。用途の形式については、*COBOL言語の概念*の章の*文字表現と基数の選定*の節を参照。
3. USAGE IS BINARY句は、計算機記憶内で数字項目を表現するのに基数 2 を使用することを定義する。USAGE IS BINARY句はUSAGE IS COMPUTATIONAL句と同義である。
4. USAGE IS PACKED-DECIMAL句は、計算機記憶内で数字項目を表現するのに基数10 を使用することを定義する。またこの句は、各桁位置が計算機記憶内で最小限の構成を占めるように定義する。
   (OSVS)(VSC2)(MF)(XOPEN)USAGE IS PACKED-DECIMAL句は、USAGE IS COMPUTATIONAL-3と同義語である。
5. USAGE IS DISPLAY 句は、計算機記憶内でデータ項目を表現するのに(明示的または暗示的に) 標準のデータ形式を使用することを定義する。データ項目は文字境界上で整列させられる。
6. (OSVS)(VSC2)(MF)USAGE IS DISPLAY句は、下記の種類の項目に適用できる。
   - 英字
   - 英数字
   - 英数字編集
   - 数字編集
   - 外部浮動小数点数
   - 外部10進数（数字）
7. 基本項目または基本項目が属する集団項目にUSAGE句を指定しないと、用途は暗黙的にDISPLAYとされる。
8. USAGE IS INDEX句を指定した基本項目を、指標データ項目という。指標データ項目には、表の要素の出現番号に相当する値を収める。
9. MOVE（転記）文または入出力文の中で参照される集団の一部に指標データ項目、
   (VSC2)(MF)ポインタ・データ項目、
   (MF)(COB370)手続きポインタ・データ項目
   が含まれていてもよい。この場合、指標データ項目、
   (VSC2)(MF)ポインタ・データ項目、

(MF)(COB370)手続きポインタ・データ項目
の変換は行われない。

10. (OSVS)(VSC2)(MF)USAGE IS COMPUTATIONAL-4は、USAGE IS COMPUTATIONALと同義語である。
11. (VSC2)(MF)USAGE IS POINTER句を指定すると、対象データ項目の番地を記憶できる。(*SET*文の節を参照。)
12. (MF)(COB370)USAGE IS は、データ項目が 手続きの番地を記憶する手続きポインタであることを指定する。(SET文の節を参照。) 参照する対象の手続きはどんな言語で書いてあってもよい。COBOLの場合、それは入れ子になっていないポインタの手続き部を表わす。下記のどちらかによって識別する。
    - プログラム名段落のプログラム名
    - ENTRY(入口)文の入口名
13. (MF)型定義名-1を 基本項目として指定した場合、「USAGE 型定義名-1」句は、型定義名-1によって参照されるプログラマによって定義された用途と同じ属性を、基本項目に指定することになる。
14. (MF)型定義名-1を集団項目として指定した場合、「USAGE 型定義名-1」句は同一の構造をした集団項目を指定する働きをする。その効果は、型定義名-1によって識別されるデータ記述項の下位に属する複数のデータ宣言が、「USAGE 型定義名-1」句で宣言された項目の下位に、まったく同様に指定されたかのようになる。下位のデータ項目のデータ名は、型定義名-1によって参照されるプログラマによって定義された構造体の中に宣言されているものと同じになる。それらを一意に参照するには、データ名の修飾を行う。
15. (ISO2000)USAGE OBJECT REFERENCE句を用いて記述したデータ項目をオブジェクト参照と呼ぶ。オブジェクト参照は字類がオブジェクトであり項類がオブジェクト参照であるデータ項目である。オブジェクト参照の内容はオブジェクトの参照先または空である。ただし、下記の規則に従う。
    a. オブジェクト参照データ項目に割り当てられる記憶領域のサイズは4バイトである。
    b. 選択指定を何も書かないと、そのデータ項目は一般的オブジェクト参照となる。その内容は任意のオブジェクトの参照先であってよい。
    c. インタフェース名-1を指定した場合、そのデータ項目によって参照されるオブジェクトは、インタフェース名-1によって参照されるインタフェースに適合しなければならない。
    d. クラス名-1を指定した場合、そのデータ項目によって参照されるオブジェクトはクラス名-1のオブジェクトであるかまたはクラス名-1のサブクラスでなければならない。ただし、下記の規則に従う。
        1. ONLY指定を書かなかった場合:
            a. FACTORY指定を書いた場合、そのデータ項目によって参照されるオブジェクトは、指定したクラスのファクトリ・オブジェクトまたはサブクラスでなければならない。

b. FACTORY指定を書かなかった場合、そのデータ項目によって参照されるオブジェクトは、指定したクラスのオブジェクトまたはサブクラスでなければならない。

2. ONLY指定を書いた場合：
   a. FACTORY指定を書いた場合、そのデータ項目によって参照されるオブジェクトは、指定したクラスのファクトリ・オブジェクトでなければならない。
   b. FACTORY指定を書かなかった場合、そのデータ項目によって参照されるオブジェクトは、指定したクラスのオブジェクトでなければならない。

e. ACTIVE-CLASS指定を書いた場合、そのデータ項目によって参照されるオブジェクトは、そのオブジェクト参照が指定されているメソッドを呼び出すのに使用されたオブジェクトが属するクラスのオブジェクトでなければならない。
   1. FACTORY指定を書いた場合、そのデータ項目によって参照されるオブジェクトは、そのクラスのファクトリ・オブジェクトでなければならない。
   2. FACTORY指定を書かなかった場合、そのデータ項目によって参照されるオブジェクトは、そのクラスのオブジェクトでなければならない。

8.1.1.25 VALUE（値）句

機能

VALUE（値）句は、定数の値、作業場所項目の初期値、条件名に関連する値を定義する。

例:

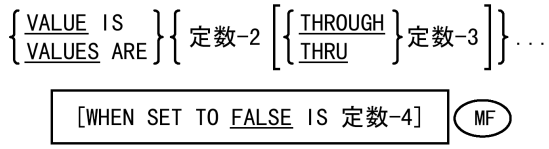
1. NEXT句の使用例は、 ***言語リファレンス - 追加トピック****の例*の章を参照。

一般形式

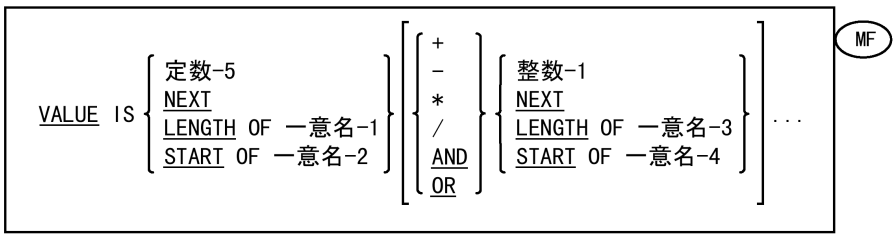
書き方1

```
VALUE IS 定数-1
```

書き方2



書き方3



構文規則

すべての書き方

1. 符号付き数字定数には、符号付き数字PICTURE文字列を指定する。
2. VALUE句の中の数字定数は、すべてPICTURE文字列で指定された値の範囲に収まるものにする。ゼロでない数字の切捨てを引き起こすものであってはならない。VALUE句の中の文字定数は、PICTURE句で示されている大きさを超えてはならない。
3. 定数が記述してある場合、代わりに書き方1 および書き方2に表意定数を記述できる。

書き方1

4. (OSVS)書き方1で、VALUES AREを使用できる。
5. (OSVS)(VSC2)(MF)外部浮動小数点数データ項目には、VALUE句を指定できない。
   (MF)外部浮動小数点数データ項目に、VALUE句を指定できる。
6. (OSVS)(VSC2)(MF)USAGE COMP-1 または USAGE COMP-2を指定した内部浮動小数点数データ項目に、VALUE句を指定できる。この場合の定数-1は、浮動小数点定数、表意定数ZERO、値ゼロを表す数字定数のどれかとする。

書き方2

7. THRUとTHROUGHは同義語であり、どちらを書いてもよい。
8. 1つのデータ項目に、書き方2の記述項をいくつでも書ける。
9. 書き方2は、条件名に関してだけ使用できる。
10. 条件名記述項では、VALUE句を必ず書く。条件名記述項に書けるのは、VALUE句と条件名自体である。
11. 定数-2は、定数-3より小さくする。

12. (OSVS)(VSC2)(MF)書き方2は、内部浮動小数点数データ項目に対応する条件変数を定義するのに使用する。外部浮動小数点数データ項目には使用できない。この場合、定数-2および定数-3は、浮動小数点定数、表意定数ZERO、値ゼロを表す数字定数のどれかとする。
    (MF)書き方2を、外部浮動小数点数データ項目に対応する条件変数を定義するのに使用できる。
13. (MF)定数-4は、定数-2から定数-3の範囲内のどの値とも等しくてはならない。つまり、定数-4は定数-2以上かつ定数-3以下であってはならない。

書き方3

14. (MF)一意名は、78レベル項目の宣言の前に、すべて定義しておく。一意名-1または一意名-3（LENGTHパラメータ）のどちらかが集団項目である場合、その集団の定義は、78レベル項目の宣言の前に、レベル番号がそれ以下の別のデータ項目が次にくることによって完結されていること。

一般規則

書き方1および2

1. VALUE句は、それを指定した対象の項目またはその項目を含む集団項目のデータ記述に書いた他の句と矛盾してはならない。下記の規則が適用される。
    a. 項目の項類が数字である場合、VALUE句内の定数はすべて数字でなければならない。作業場所項目として定義された定数は、標準桁寄せ規則に従ってデータ項目に収められる。(*COBOL言語の概念*の章の*標準桁寄せ規則*の節を参照。)
    b. 項目の項類が英字、英数字、英数字編集、数字編集である場合、VALUE句内の定数はすべて文字とする。これらの定数をデータ項目中に収めるときの扱いは、そのデータ項目が英数字として定義されている場合と同様である。(*COBOL言語の概念*の章の*標準桁寄せ規則*の節を参照。) PICTURE句内の編集文字は、データ項目の大きさに含められる。(前述の*PICTURE句*の節を参照。) しかし、そのデータ項目の初期値を定めるものではない。したがって、編集項目に対するVALUEの定数は編集済みの形式にしておく。
    c. (MF)項目の項類が数字編集である場合、VALUE句内の定数は数字定数でも文字定数でもよい。VALUE句内の定数が数字定数である場合、項目に収められる値は数字定数を数字編集項目に転記したのと同じとなる。
    d. 初期化は、BLANK WHEN ZEROまたはJUSTIFIEDの指定とは無関係に行われる。

条件名と (MF)定数名以外に関する規則 (書き方1)

2. VALUE句の使用に関する規則は、データ部のどの節で使用するかによって異なる。
    a. ファイル節内では、VALUE句は条件名記述項内でだけ使用する。

b. 作業場所節および通信節内では、条件名記述項の中にVALUE句を指定する。また、データ項目の初期値を指定するために、VALUE句を使用できる。この場合、 実行時要素が初期状態になった時に、データ項目の値はVALUE句で指定された値に設定される。データ記述項内でVALUE句を指定しないと、そのデータの初期値はどうなるかわからない。

c. 連絡節内では、VALUE句は条件名記述項内でだけ使用する。

3. (OSVS)(VSC2)(MF)ファイル節、連絡節および
(MF)局所記憶節内
(OSVS)(VSC2)(MF)のデータ記述項の中で、VALUE句を使用できる。ただし、注記にとどまる。
4. REDEFINES句が含まれるデータ記述項またはその下位に属するデータ記述項の中では、VALUE句を使用できない。この規則は、条件名記述項には当てはまらない。
5. 集団項目にVALUE句を指定するときは、定数は表意定数または文字定数にする。この場合、その集団項目に含まれる個々の基本項目または下位の集団項目を考慮することなく、その集団項目が初期化される。その集団項目の下位のレベルには、VALUE句を指定できない。
6. JUSTIFIED, SYNCHRONIZED, USAGE（USAGE IS DISPLAYを除く）が記述された項目が含まれる集団項目に、VALUE句を書いてはならない。
7. (VSC2)(MF)表意定数の NULLは、USAGE POINTERまたはUSAGE PROCEDURE-POINTERを指定したデータ項目のVALUE句内にだけ指定できる。この種のデータ項目のVALUE句内に指定できる値は、このNULLだけである。NULLを指定すると、ポインタは他のデータ項目を指さないことが保証される。
8. (ANS85)OCCURS句が含まれるデータ記述項またはその下位に属するデータ記述項の中でVALUE句を使用すると、反復される各データ項目に指定した値が割り当てられる。
9. (ANS85)可変反復データ項目に関連するデータ項目のデータ記述項の中で、VALUE句を指定すると、DEPENDING ON指定によって参照されるデータ項目の値がOCCURS句によって指定される反復回数の最大値と等しいものとして、そのデータ項目が初期化される。下記の場合に、データ項目は可変反復データ項目と関連する。

a. VALUE句を指定したデータ項目が集団項目であり、その中に可変反復データ項目が含まれる。

b. VALUE句を指定したデータ項目自体が、可変反復データ項目である。

c. VALUE句を指定したデータ項目が、可変反復データ項目の下位に属する。

DEPENDING ON指定によって参照されるデータ項目にVALUE句が関連する場合、可変反復データ項目が初期化された後で、VALUE句で指定された値がそのデータ項目に入られるものとみなされる。

10. (ISO2000)字類が オブジェクトのデータ項目は、ヌルに初期化される。初期値は、VALUE句が効力を持つ時、およびデータ項目の記憶域が割り当てられた時に効力を持つ。
11. (ISO2000)字類がポインタのデータ項目は、ヌルに初期化される。初期値は、VALUE句が効力を持つ時、およびデータ項目の記憶域が割り当てられた時に効力を持つ。

条件名に関する規則(書き方2)

10. 条件名記述項には、VALUE句を必ず指定する。条件名記述項に書けるのは、VALUE句と条件名自体だけである。条件名の性質は、 対応する条件変数の性質によって暗黙的に決まる。
11. 書き方2は、条件名に関連してだけ使用できる。THRU指定を書くときは、定数-2は定数-3よりも小さくする。
12. FALSE句は、対応する条件名がSET 条件名 TO FALSE 文を参照する場合にだけ意味を持つ。（*SET*文の節を参照。）

定数名に関する規則(書き方3)

13. (MF)書き方3は、定数名記述項にだけ使用できる。
14. (MF)定数-5を指定して演算子が続かない場合、 定数名の性質は定数-5と同様となる。そうでない場合、定数名の性質は整数と同様となる。
15. (MF)任意の数の算術演算子または 論理演算子を使用できる。式は左から右へ整数演算として評価される。式の中にかっこは使用できない。中間結果にゼロ未満のものがあると、最終結果はどうなるかわからない。整数-1の代わりに定数名を使用できる。
16. (MF)論理演算のANDとORは、2進表現のビットごとに作用する。
17. (MF)一意名-1 または一意名-3のLENGTHは、一意名-1または一意名-3に割り当てられた記憶領域の大きさである。一意名が集団項目である場合、その長さは下位のデータ項目をすべて含んだものである。
18. (MF)NEXTから返される値は、前のデータ宣言の後にある記憶域の次のバイトの相対番地である。このデータ宣言がOCCURS句によって定義された表のものであると、NEXTから返される値は、表の最初の要素の後に起こる記憶域の次のバイトの相対番地である。
19. (MF)一意名-2 または一意名-4のSTARTは、一意名-2または一意名-4が始まる相対番地である。
20. (MF)規則18と19に関して、相対番地を下記のように定義する。
    - 一意名がEXTERNALレコードまたはLINKAGEレコードの一部であるならば、相対番地は対応する01レベルの始点から数える。
    - 一意名がLOCAL-STORAGE中に定義されている場合、相対番地はLOCAL-STORAGE SECTION の始点から数える。
    - それ以外の場合、相対番地はDATA DIVISIONの始点から数える。
21. (MF)相対番地は別のCOBOL製品との間で移植性がない。相対番地は翻訳単位を超えた特定の値に依存すべきではない。

8.1.1.26 VALUE OF（ラベル項目の値）句

機能

VALUE OF句はファイルのラベル・レコード中の項目の内容を指定する。
(ANS85)ファイル記述項のVALUE OF句は、ANSI '85標準では廃要素に分類されており、ANSI標準の次回の全面改訂の際に、削除される予定である。
(MF)この構文は、このCOBOL処理系に組み込まれているすべての方言で全面的に使用することができる。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を洗い出すことができる。
(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、この機能は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこの句を使用するべきではない。

一般形式

書き方1（すべてのファイル）

```
VALUE OF { 作成者語-1 IS { データ名-1 } } ...
                         { 定数-1     }
```

書き方2（すべてのファイル）

```
VALUE OF FILE-ID IS { データ名-2 }                    MF
                    { 定数-2     }
```

構文規則

書き方1（すべてのファイル）

1. データ名-1は、必要ならば修飾してもよいが、添字や指標を付けることはできない。また、データ名-1にはUSAGE IS INDEX句を記述してあってはならない。

書き方2（すべてのファイル）

2. (MF)データ名-2は、作業場所節内に記述しておく。
3. (MF)定数-2は文字定数とする。表意定数であってはならない。
4. (MF)ファイル管理記述項のASSIGN句の中で、外部ファイル参照、データ名-1、定数-1のどれかを指定した場合、 VALUE OF FILE-ID句は使用できない。(前述の*ファイル管理記述項*の節を参照。)

一般規則

書き方1（すべてのファイル）

1. (MF)VALUE OF句は注記になる。

書き方2（すべてのファイル）

2. (MF)定数-2またはデータ名-2に指定した文字列は、外部ファイル名と解釈される。

第9章: データ部 - 画面節

9.1 画面節

9.1.1 画面節

(MF)(XOPEN)

画面節（screen section）は、 ACCEPT（受け取り）文およびDISPLAY（表示）文で画面を操作するための機能を提供するものである。

- 画面節を記述することによって、画面上の領域を構成する 静止型の定形画面（form）を表示できる。画面節の記述項を、画面記述（screen description）という。画面記述は形式的にはデータ記述に似ている。しかし、データ記述が記憶領域を定義するのに対して、画面記述は 画面項目（screen item）または画面領域（screen area）を定義する。
- (MF)画面節を記述して、静止型の定形画面を構成するデータ項目を表示できる。使用する画面領域の詳細については、関連するACCEPTおよびDISPLAYの操作の箇所で説明する。

画面節には、 画面上の各項目を記述する。画面節に記述した項目は、書き方4のACCEPT文または書き方2のDISPLAY文を用いて操作できる。このような項目を画面項目と呼ぶ。画面項目の多くは画面上の項目のレイアウトを記述するだけであり、明示的に参照されることはない。
画面節を記述することによって、操作員は下記の操作を行える。

- 項目の正確な位置を指定する
- 指定された位置から入力されたデータを受け取る
- 指定された位置に定数字句を表示する
- 画面属性を定義する
- 操作卓機能を制御する

表4-6に、 そのための画面記述句、画面オプション、データ記述句を一覧にして示す。 これらは、
(MF)ACCEPT文およびDISPLAY文に適用するために、
画面節内で使用する。また、この表には、各句またはオプションが下記のどの種類の項目に使用できるかも示す。

- 入力項目　-　 画面記述にTO指定が含まれる画面項目
- 出力項目　- 画面記述にFROM指定が含まれる画面項目
- 更新項目　- 画面記述にUSING指定が含まれる画面項目
- 定数項目　-　 画面記述にPICTURE句が含まれない基本画面項目

数字画面項目および数字編集画面項目にデータを入力する方法は、 固定方式（fixed format mode）と 自由方式（free format mode）の2通りある。 詳細については、ユーザインタフェースに関するCOBOLシステムのマニュアルを参照。
固定方式は、数字画面項目または数字編集画面項目にデータを入力する省略時解釈の方法である。この方式では、受け取った入力データを、項目のPICTURE指定に従って形式を整え画面上に表示するとともに、カーソルを移動させる。数字、"+", "-" , 小数点以外の文字は入力できない。編集項目中の挿入文字は、カーソルが前後に移動される際には飛ばされる。符号を示す記号は、通常の仕様に従って置き換えられる。浮動記号は、項目内を左右に移動する。挿入記号は、数字が挿入または削除されるにつれて現れたり消えたりする。 カーソルが項目内の最後の文字位置に達すると、どんな文字を入力しても最後の文字に置き換えられる。
自由方式は、省略時解釈の固定方式に代わる、数字画面項目および数字編集画面項目にデータを入力するもう1つの方法である。この方式では、適当な長さのPIC X項目にデータを入力させることができる。入力したデータはカーソルを項目の外に出した時点で、はじめてPICTURE仕様に合わせて編集される。入力した文字のうち、数字と符号と小数点以外はすべて無視される。その後で、数値が抽出され、格納され、編集される。このときに適用される規則は、PICTUREの同じ画面項目または作業場所項目にMOVE（転記）する通常のCOBOLの規則と同じである。通常、その後で、入力した数値は画面上に表示される。利用可能な構成オプションの詳細については、ユーザインタフェースに関するCOBOLシステムのマニュアルを参照。

表4-6: 使用可能なオプション

画面/記述句／画面オプション／データ記述句	画面節				(MF)適用句	
	入力項目	出力項目	更新項目	定数項目	ACCEPT	DISPLAY
AUTO	○		○		○	
BACKGROUND-COLOR	○	○	○	○	○	○
BELL	○	○	○	○	○	○
BLANK	○	○	○	○		○
BLANK WHEN ZERO	○	○	○			
BLINK	○	○	○	○	○	○
COLUMN	○	○	○	○		
ERASE	○	○	○	○	○	
FOREGROUND-COLOR	○	○	○	○	○	○
FULL	○		○		○	
GRID	○	○	○	○	○	○
HIGHLIGHT	○	○	○	○	○	○
JUSTIFIED	○	○	○			
LEFT-JUSTIFY					○	
LEFTLINE	○	○	○	○	○	○
LINE	○	○	○	○		
LOWLIGHT	○	○	○	○	○	○
OCCURS	○	○	○			
OVERLINE	○	○	○	○	○	○
PROMPT	○		○		○	
REQUIRED	○		○		○	
REVERSE-VIDEO	○	○	○	○	○	○
RIGHT-JUSTIFY					○	
SECURE	○		○		○	
SIGN	○	○	○			
SIZE	○	○	○	○	○	○
SPACE-FILL					○	
UNDERLINE	○	○	○	○	○	○
UPDATE					○	

○ = 句またはオプションを使用できる

以下に、句およびオプションについて、個別に説明する。

機能

画面節は、書き方4のACCEPT文および書き方2のDIAPLAY文で画面を操作するための機能を提供するものである。

一般形式

```
SCREEN SECTION.
        [画面記述項]...
```

構文規則

1. 画面節は、データ部の中の最後の節とする。
2. 画面記述項は、画面記述句で構成する。画面節の中で使用できるデータ記述句は、この節に記載してあるものに限られる。

一般規則

画面節には、画面上の各項目を記述する。画面節に記述した項目は、書き方4のACCEPT文または書き方2のDISPLAY文を用いて操作できる。このような項目を画面項目と呼ぶ。画面項目の多くは画面上の項目のレイアウトを記述するだけであり、明示的に参照されることはない。

9.1.1.1 画面記述項の全体的な骨組み

機能

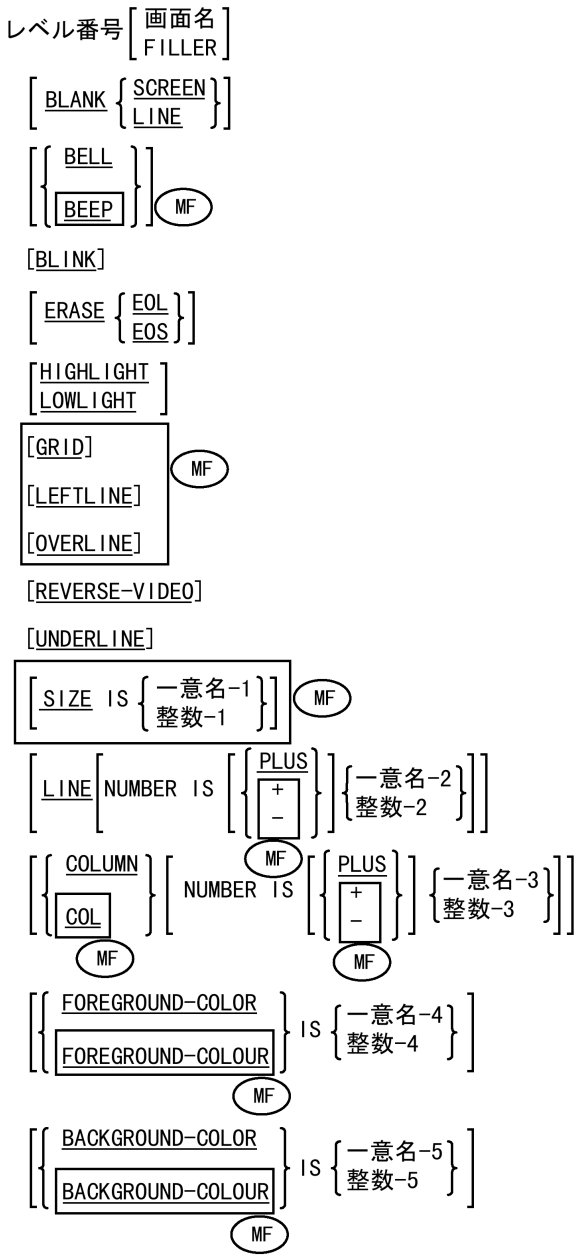
画面記述項は、実行時に受け取り用または表示用に参照される画面項目の属性、動き、大きさ、位置などを指定する。

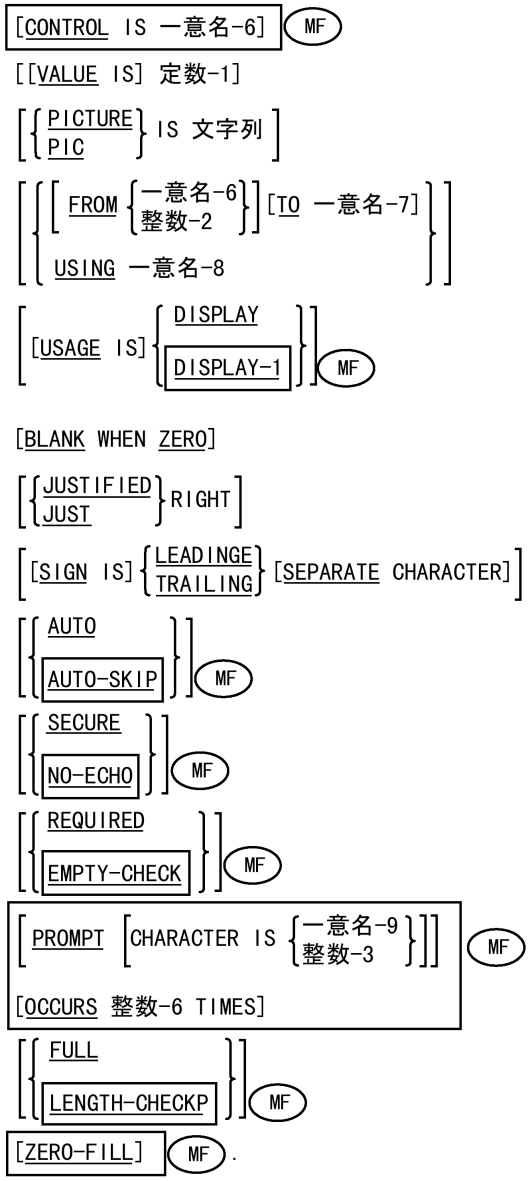
利用できる画面属性は、使用している端末ハードウェアやオペレーティングシステムやCOBOLの実行時支援機能に左右される。

(MF)下記のものは、X/Openの画面操作には**含まれない。**

ACCEPT（書き方 3 と 5）
AUTO-SKIP（AUTOの同義語）
BEEP（BELLの同義語）
COL（COLUMNの同義語）
CONTROL
DISPLAY（書き方3）
EMPTY-CHECK（REQUIREDの同義語）
GRID
LEFTLINE
LENGTH-CHECK（FULLの同義語）
NO-ECHO（SECUREの同義語）
OCCURS
OVERLINE
PROMPT
SIZE
ZERO-FILL

一般形式


レベル番号
画面名
FILLER
BLANK
SCREEN
LINE
BELL
BEEP
MF
BLINK
ERASE
EOL
EOS
HIGHLIGHT
LOWLIGHT
GRID
LEFTLINE
OVERLINE
REVERSE-VIDEO
UNDERLINE
SIZE IS
一意名-1
整数-1
LINE
NUMBER IS
PLUS
+
−
一意名-2
整数-2
COLUMN
COL
一意名-3
整数-3
FOREGROUND-COLOR
FOREGROUND-COLOUR
IS
一意名-4
整数-4
BACKGROUND-COLOR
BACKGROUND-COLOUR
一意名-5
整数-5



構文規則

1. 各画面記述項の最初にレベル番号を書く。レベル番号は01から49までとする。
2. 01レベルの項目には、画面名を指定する。
3. 画面名はレベル番号の直後に書く。画面名は、利用者語の要件を満たす必要がある。
4. 画面項目は、書き方4のACCEPT文または書き方2のDISPLAY文によってだけ参照できる。
5. 各基本画面項目には、次のものの中から少なくとも1つを指定する：BELL, BLANK LINE, BLANK SCREEN, COLUMN, LINE,PICTURE, VALUE。

6. FROM,TO,USINGの各指定の中のデータ項目は、画面項目に「関連付け」られる。USING指定は、FROM指定とTO指定を対にして同じ項目を指定したものと同義である。
7. ACCEPTを実行できるのは、FROM指定またはVALUE句を伴う画面項目、およびTO指定またはUSING指定を伴う画面項目が共に含まれる 集団画面項目に対してだけである。
8. 画面名の後ろに続ける句は、どのような順序で書いてもよい。
9. 集団画面項目の画面記述に書いた句は、その集団画面項目に属するすべての基本画面項目のうち、その句が当てはまるものに適用される。
10. 01レベル以外の画面項目には、データ名またはFILLERを指定してもよいし、あるいは何も指定しないでもよい。名前を指定しないと、FILLERを指定したものとみなされる（したがって、その項目は明示的に参照できない）。
11. 同じ画面項目に同じ句を2回以上指定すると、階層系列内で最も低いレベルに現れるものが効力を発揮する。

一般規則

1. 画面名は、画面記述中に記述された画面項目に名前を付ける働きをする。
2. 画面記述は、画面上の領域を定義する。各記述項は、レベル番号といくつかの句で構成する。画面名をはじめとする各種の句は、指定しても指定しなくてもよい。これらの句を通じて、項目の位置や操作卓機能を記述する。
3. 画面項目が表示されるとき、データは、関連するFROM指定またはUSING指定内に指名されている定数またはデータ項目から取り出される。TO指定の中にだけ指定されている項目は、FROM SPACEまたはFROM ZEROが指定されているものと扱われる。どちらに扱われるかは、その画面項目の種類による。
4. 画面項目が受け取られると、データは、画面からTO指定またはUSING指定内に指定されているデータ項目に転記される。その項目の項類に応じて、必要があれば、変換および編集が解除される。
5. 入力項目は、画面記述にTO指定が含まれる画面項目である。
6. 出力項目は、画面記述にFROM指定が含まれる画面項目である。
7. 更新項目は、画面記述にUSING指定が含まれる画面項目である。
8. 定数項目は、画面記述にPICTURE句が含まれる基本画面項目である。
9. 集団画面項目をACCEPTすることは、その集団に属する基本項目のうちの入力項目または更新項目を受け取ることである。これらの項目は、画面節内に記述されている順番に、画面記述によって示される画面位置から受け取られる。CURSOR IS句(前述の*CURSOR IS句*の節を参照。）の中で別途指定されていないかぎり、カーソルは、当初最初の項目の始点に位置付けられる。各項目のACCEPT処理が終わるにつれて、カーソルは次の項目の始点へと移動される。
10. 集団画面項目をDISPLAYすることは、その集団に属する基本項目のうちの出力項目または更新項目と定数項目を表示することである。これらの項目は、画面記述によって示される画面位置に同時に表示される。
11. ACCEPTまたはDISPLAYの対象の画面項目の長さが現在の行を超えると、次の行の頭に戻って続けられる。

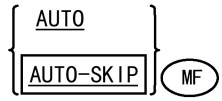
12. 画面項目が長すぎて物理画面に収まりきらない場合、切捨てが発生する。出力項目および英数字の入力項目と更新項目の場合は、画面からはみ出す最初の文字以降が切り落とされる。数字および数字編集の入力項目と更新項目の場合は、画面からはみ出す最初の項目以降が切り落とされる。

9.1.1.2 AUTO（自動）句

機能

AUTO（自動）句は、最後の文字位置に文字が入力されたときに、画面項目のACCEPT処理を自動的に終了させる。終了キーを明示的に入力する必要はない。

一般形式



構文規則

1. AUTO句は、入力項目および更新項目に対してだけ指定できる。
2. この句を集団レベルに指定すると、それに属するすべての基本項目に適用される。
3. (MF)AUTOとAUTO-SKIPは同義であり、どちらを書いてもよい。

一般規則

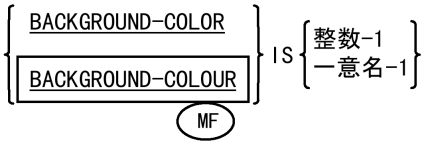
1. REQUIRED句またはFULL句が満たされるならば、カーソルは次の画面項目に位置付けられる。あるいは、ACCEPTする最後の画面項目のところでは、ACCEPT処理そのものが終了される。
2. この句は、ACCEPT文の自動飛越しおよび自動終了用の既存の構成オプションに優先する。（構成オプションの詳細については、ユーザインタフェースに関するCOBOLシステムのマニュアルを参照。）
3. 固定方式の数字編集画面項目にAUTO句を指定した場合、整数部分の桁位置がすべて入力されたときに、小数点位置が自動的に飛び越される。固定方式を選択することは、構成オプションである。（構成オプションの詳細については、ユーザインタフェースに関するCOBOLシステムのマニュアルを参照。）

9.1.1.3 BACKGROUND-COLOR（背景色）句

機能

BACKGROUND-COLOR（背景色）句は、画面項目の背景色を指定する。

一般形式



構文規則

1. (MF)BACKGROUND-COLORとBACKGROUND-COLOURは同義であり、どちらを書いてもよい。
2. BACKGROUND-COLOR句は、任意の画面項目に指定できる。
3. BACKGROUND-COLOR句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
4. 整数-1の値は、0から7までとする。

一般規則

1. BACKGROUND-COLOR句は、カラー画面に対してだけ適用できる。
2. 整数-1または一意名-1は、画面項目の背景色を指定する。使用できる色と対応する値は下記のとおり。

0	黒	8	灰色
1	青	9	明るい青
2	緑	10	明るい緑
3	藍	11	明るい藍
4	赤	12	明るい赤
5	赤紫	13	明るい赤紫
6	茶または黄	14	黄
7	白	15	高輝度の白

カラー画面では、8から15の値の場合、整数0から7を指定してBLINK句を指定することと同じである。モノクロ画面では、これは単にBLINK句を指定することと同じである。

3. BACKGROUND-COLOR句を指定しないと、背景色は省略時解釈の黒となる。
4. 画面項目にBLANK SCREEN句とBACKGROUND-COLOR句を共に指定するか、またはBLANK SCREEN句を指定した画面項目がBACKGROUND-COLOR句を指定した画面項目の下位に属する場合、DISPLAY文を用いてその画面項目を表示すると、指定した色が省略時解釈の背景色となる。この省略時解釈の背景色は、別の指定をするまで有効である。別の指定とは、上記と同じ組合わせの指定内容をもつ別の画面項目を表示するか（同じDISPLAY文の中でも別のDISPLAY文の中でもよい）、または上記の2つの句を含む書き方3のDISPLAY文を実行することである。

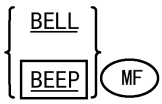
5. 一意名-1は、符号の付かない0から7までの整数とする。8以上の値を指定すると、その値を8で割った余りが指定値としてとられる。
6. 一意名-1は、OCCURS句の左辺には指定できない。
7. 整数-1または一意名-1の値に6を指定したときに、茶色と黄色のどちらが表示されるかは、端末ハードウェアによる。

9.1.1.4 BELL（警報）句

機能

BELL（警報）句は、この句が含まれる画面項目が表示されるたびに、警報音を鳴らす。

一般形式



構文規則

1. BELL句は、基本項目に対してだけ指定できる。
2. (MF)BELLとBEEPは同義であり、どちらを指定してもよい。

一般規則

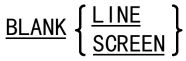
1. BELL句は、この句が指定されている画面項目が受け取られるかまたは表示されるたびに、警報音を鳴らす。
2. この句が含まれる画面項目を受け取ったときに警報音が鳴るようにするためには、書き方5のACCEPT文の中にBELL句を指定する。画面記述項の中にBELL句を指定して書き方4のACCEPT文を実行しても、警報音は鳴らない。

9.1.1.5 BLANK（空白）句

機能

BLANK（空白）句は、画面項目が表示される前に画面の行または画面全体を空白にする。

一般形式



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - OLDBLANKLINE - BLANK LINE句の効果はERASE EOLと同じであることを指定する。つまり、カーソルの右にある文字はすべて削除される。

構文規則

1. BLANK SCREEN句は、任意の画面記述項に指定できる。
2. BLANK LINE句は、基本項目に対してだけ指定できる。

一般規則

1. BLANK SCREEN句は、どこに記述されていても、画面データ項目が表示される前に実行される。この句が実行されると画面全体が空白にされ、カーソルは行1、カラム1の位置に移動される。
2. BLANK LINE句を指定すると、記述がある画面データ項目の行がカラム1から行の終端まで空白にされる。カーソルは、BLANK LINE句が実行された位置と同じところにある。
3. BLANK句もERASE句も指定しないと、画面データ項目を表示したときに、その項目の部分だけが画面上で変更される。それ以外の画面部分は、元のまま変わらない。
4. BLANK SCREEN句が実行されると、画面の前景色と背景色は省略時解釈の状態に戻される。画面項目にFOREGROUND-COLORまたはBACKGROUND-COLORを指定した場合の効果については、それぞれの節を参照。
5. ACCEPT文の中では、BLANK句は無視される。
6. 項目が表示される前に、消去が行われる。
7. BLANK SCREEN句を指定した画面には、色に関する句（FOREGROUND-COLORとBACKGROUND-COLOR）だけを使用すべきである。他の属性は受け付けられるが、無視される。

9.1.1.6 BLANK WHEN ZERO（ゼロならば空白）句-画面節中

機能

BLANK WHEN ZERO（ゼロならば空白）句は、画面項目の値がゼロのとき表示上は空白にする。

一般形式

```
BLANK WHEN ZERO
```

構文規則

1. BLANK WHEN ZERO句は項類が数字または数字編集の入力項目と出力項目と更新項目にだけ指定できる。

一般規則

1. BLANK WHEN ZERO句は、画面項目の値がゼロのとき表示上は空白にする。
2. カーソルが項目内にあるときには、この句は効力を発揮しない。つまり、現在の入力項目に関しては、たとえゼロであってもその内容は常に表示される。

9.1.1.7 BLINK（点滅）句

機能

BLINK（点滅）句は、
画面上に表示される項目を点滅させる。

一般形式

BLINK

構文規則

1. BLINK句は、どのような画面項目にも指定できる。
2. BLINK句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。

一般規則

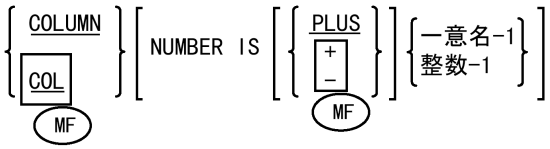
1. BLINK句は、画面上に表示される項目を点滅させる。

9.1.1.8 COLUMN（カラム）句

機能

COLUMN（カラム）句は、画面項目を画面上に表示する始点となるカラムを指定する。

一般形式



構文規則

1. 一意名-1は符号の付かない整数であり、その値は1以上255以下とする。
2. 整数-1は符号なしで、その値は1以上255以下とする。
3. (MF)PLUSと"+" は同義であり、どちらを書いてもよい。
4. NUMBERの指定を省略すると、省略時解釈として値"+1"がとられる。
5. LINE句を指定してCOLUMN句を指定しないと、カラム1が想定される。
6. COLUMN句は、任意の基本項目に指定できる。
7. (MF)COLは、COLUMNの省略形である。
8. (MF)COLUMN句をSCREEN SECTION集団項目の階層中のどこかに指定する時、以下の句のうち、最低１つを同じ集団の定義に指定しなければならない。
   BEEP, BELL, BLANK LINE, BLANK SCREEN, ERASE EOL, ERASE EOS, FROM, TO, USING, VALUE

一般規則

1. COLUMN句は、ACCEPT処理またはDISPLAY処理を行うときの、画面上に画面項目を表示するカラムを指定する。
2. COLUMN句に一意名または整数を指定し、PLUS,
   (MF)"+" , "-"
   のどれも指定しない場合、その値は絶対カラム番号を表わす。01レベルはそれぞれ、画面レコードを表わす。AT指定は、表示画面の先頭に関連付けて、画面レコードの先頭位置を指定する。ACCEPT文またはDISPLAY文にAT指定を指定しなかった場合、画面の最初のカラムがカラム1となる。
3. COLUMN句にPLUS,
   (MF)"+" , "-"
   のどれかを指定すると、一意名または整数に指定した値は、先行する画面項目の末尾からの相対カラム番号を表わす。この場合、先行する項目が実際に画面に表示されるか否かは問わない。先行する画面項目の末尾は、その項目の実行時の有効な長さによって決まる。その長さは、PICTURE句、VALUE句、SIZE句に基づいて算出される。01レベルの項目が出てくると、カラム番号は1に戻る。
4. 01レベルでない項目の画面記述にLINE句もCOLUMN句も含めないと、COLUMN+1が指定されたものとみなされる。つまり、その項目は前の項目の直後に続くことになる。
5. 画面の範囲から外れるカラム位置を指定すると、次（または前）の行の頭に繰り越してカラム位置が数えられる。

9.1.1.9 CONTROL（制御）句

(MF)

機能

CONTROL（制御）句は、画面項目に関する属性を実行時に定義できるようにする。

一般形式

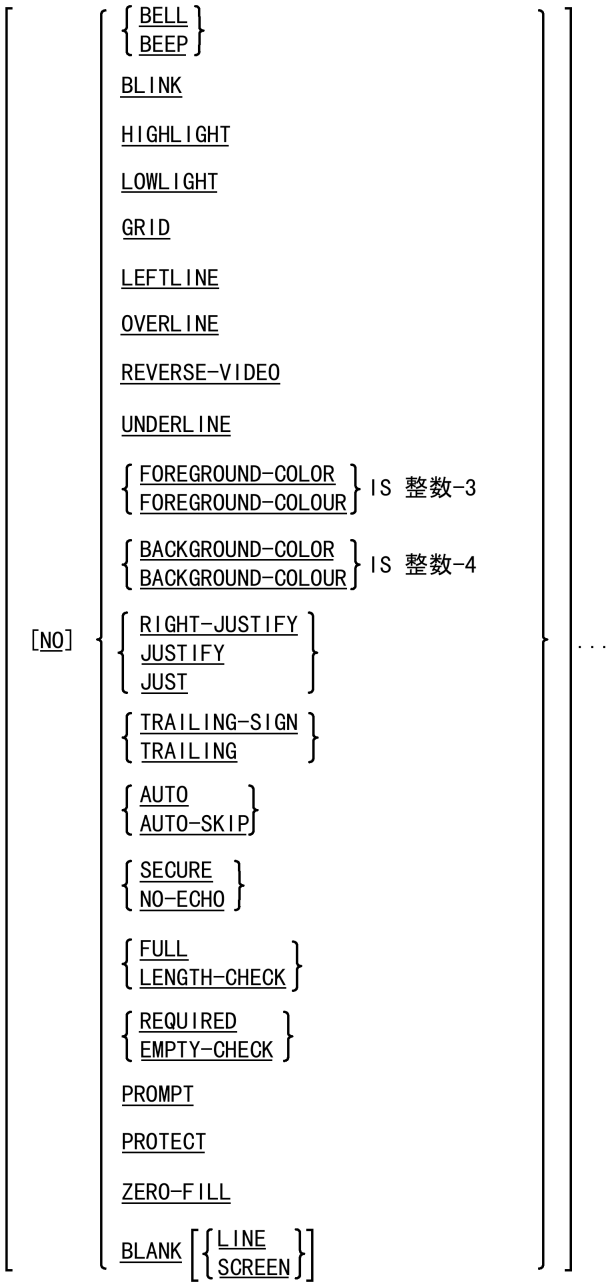
```
CONTROL IS 一意名-1
```

構文規則

1. この句は、どの画面項目にも指定できる。
2. この句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. 一意名-1は、英数字データ項目とする。この項目がとる最大の大きさは、PIC X(65535)である。
4. 一意名-1に指定した属性は、静的に定義した属性に優先する。
5. CONTROL句を入れ子にすることはできない。
6. PROTECT属性は、基本画面記述項にだけ指定できる。

一般規則

1. この項目を明示的または暗黙的に参照するACCEPT文またはDISPLAY文が実行される時、一意名-1によって参照されるデータ項目の内容は、以下に示す画面属性のサブセットを確認しなければならない。これらはどこにでも指定できる。



2. 一意名-1の内容が空白であると、項目をACCEPTまたはDISPLAYする際に、静的に定義されている属性が使用される。
3. CONTROL 一意名-1 内に指定できる各属性の意味は、この参照の以下の部分に定義する。
    a. DATA DIVISION 内の各 SCREEN SECTION CLAUSE
        - AUTO-SKIP/AUTO
        - BACKGROUND-COLOR/BACKGROUND-COLOUR

- BELL/BEEP
- BLANK LINE/SCREEN
- BLINK
- FOREGROUND-COLOR/FOREGROUND-COLOUR
- FULL/LENGTH-CHECK
- GRID
- HIGHLIGHT
- JUSTIFY/JUST
- LEFTLINE
- LOWLIGHT
- OVERLINE
- PROMPT
- REQUIRED/EMPTY-CHECK
- REVERSE-VIDEO
- SECURE/NO-ECHO
- UNDERLINE
- ZERO-FILL

b. PROCEDURE DIVISION 内のACCEPT 文一般規則

- RIGHT-JUSTIFY
- TRAILING-SIGN
- TRAILING

c. このCONTROL句内

- PROTECT

4. PROTECT属性は、ACCEPT文への入力を妨げることを指定する。

9.1.1.10 ERASE（消去）句

機能

ERASE（消去）句は、カーソルが位置している以降の行または画面の一部を空白にする。

一般形式

ERASE { EOL / EOS }

構文規則

1. ERASE句は、基本項目に対してだけ指定できる。

一般規則

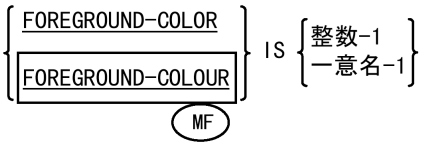
1. ERASE EOLを指定すると、その画面データ項目の座標位置（行とカラム）以降その行の末尾までが空白にされる。
2. ERASE EOSを指定すると、その画面データ項目の座標位置（行とカラム）以降その画面の末尾までが空白にされる。
3. BLANK句もERASE句も指定しないと、画面データ項目を表示する際に、その項目の部分だけが変更される。
4. ACCEPT処理の際には、この句は無視される。

9.1.1.11 FOREGROUND-COLOR（前景色）句

機能

FOREGROUND-COLOR（前景色）句は、画面項目の前景色を指定する。

一般形式



構文規則

1. FOREGROUND-COLORとFOREGROUND-COLOURは同義であり、どちらを書いてもよい。
2. FOREGROUND-COLOR句はどの画面項目にも指定できる。
3. FOREGROUND-COLOR句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
4. 整数-1の値は、0から7までとする。

一般規則

1. FOREGROUND-COLOR句はカラー画面に対してだけ適用できる。

2. 整数-1または一意名-1は画面項目の前景色を指定する。使用できる色と対応する値は下記のとおり。

0	黒	8	灰色
1	青	9	明るい青
2	緑	10	明るい緑
3	藍	11	明るい藍
4	赤	12	明るい赤
5	赤紫	13	明るい赤紫
6	茶または黄	14	黄
7	白	15	高輝度の白

カラー画面では、8から15の値の場合、整数0から7を指定してHIGHLIGHT句を指定することと同じである。モノクロ画面では、これは単にHIGHLIGHT句を指定することと同じである。

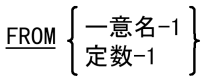
3. この句を指定しないと、前景色は省略時解釈の白となる。
4. 画面項目にBLANK SCREEN句とFOREGROUND-COLOR句を共に指定するか、またはBLANK SCREEN句を指定した画面項目がFOREGROUND-COLOR句を指定した画面項目の下位に属する場合、その画面項目をDISPLAY文を用いて表示すると、指定した色が省略時解釈の前景色となる。この省略時解釈の前景色は、別の指定をするまで有効である。別の指定とは、上記と同じ組合わせの指定内容をもつ別の画面項目を表示するか（同じDISPLAY文の中でも別のDISPLAY文の中でもよい）、または上記の2つの句を含む書き方3のDISPLAY文を実行することである。
5. HIGHLIGHT句も同時に指定すると、前景色の輝きと明るさが増大される。たとえば、ハードウェアによっては、黒が灰色に変えられ茶色が黄色に変えられることがある。しかし、これはBLANK SCREEN句には適用されない。
6. 一意名-1は符号の付かない0から7までの整数とする。8以上の値を指定すると、その値を8で割った余りが指定値としてとられる。
7. 一意名-1は、OCCURS句の左辺には指定できない。
8. 整数-1または一意名-1の値に6を指定したときに、茶色と黄色のどちらが表示されるかは、端末ハードウェアによる。

9.1.1.12 FROM（から）句

機能

FROM句は表示するデータの源を示す。

一般形式



構文規則

1. FROM句をTO句と一緒に使用することは、対応して同じ一意名を指定したUSING句を使用することに等しい。FROM句をUSING句と一緒に使用してはならない。
2. FROM句内の一意名を修飾してもよい。画面項目にOCCURS句が適用されていない場合、その一意名は添字付けまたは指標付けされていてもよい。FROM句を定義できる場所はプログラムのファイル節、作業場所節、
   (MF)局所記憶節、
   連絡節のいずれかだけである。

一般規則

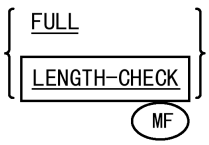
1. FROM句が記述されている画面項目に対してDISPLAY文を実行すると、対応するデータ項目から画面項目にデータが転記され、それから画面項目が画面上に表示される。

9.1.1.13 FULL（全桁）句

機能

FULL（全桁）句は、画面項目をまったく空白のままにしておくか、全桁にデータを入力するかを指定する。

一般形式



構文規則

1. FULL句は、入力項目と更新項目と集団項目に対してだけ有効である。
2. FULL句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. (MF)FULLと LENGTH-CHECKは同義であり、どちらを指定してもよい。

一般規則

1. FULL句が効力を発揮するのは、カーソルを画面項目の中に位置付けたうえでデータを受け取るACCEPT文が実行されるときである。この句の要件が満たされるまで、終了キーを押しても受け付けられない。固定方式の数字編集項目の場合、カーソルは小数点の位置に位置付け直される。
2. 画面項目の項類が英数字の場合、この句の要件が満たされるのは、次の2つの場合である。1つは、その項目全体に空白またはプロンプト文字が入れられた場合。もう1つは、その項目の最初の文字位置と最後の文字位置に、空白でもプロンプト文字でもない文字が入れられた場合。
3. 画面項目の項類が自由方式の数字または数字編集の場合、この句の要件が満たされるのは、次の2つの場合である。1つは、その項目の値が結果的にゼロになる場合。もう1つは、その項目の最初の文字位置と最後の文字位置に、空白でもプロンプト文字でもない数字が入れられた場合。
4. 画面項目の項類が固定方式の数字編集の場合、この句の要件が満たされるのは、次の2つの場合である。1つは、その項目の値がゼロである場合。もう1つは、その項目内でゼロ抑制が起こらなかった場合。
5. 固定方式の数字項目またはゼロ抑制が指定されていない数字編集項目に対しては、FULL句は効力を発揮しない。
6. 更新項目に関しては、元のデータと入力されるデータの両方がFULL句の要件を満たす要素となる。
7. ACCEPT処理を終了させるためにファンクション・キーが使用された場合は、FULL句は効力を発揮しない。(構成オプションの詳細については、ユーザインタフェースに関するCOBOLシステムのマニュアルを参照。)
8. FULL句の要件が満たされない場合、エラー・メッセージを画面上に表示させることができる。(構成オプションの詳細については、ユーザインタフェースに関するCOBOLシステムのマニュアルを参照。)

9.1.1.14 GRID（桁取り）句

(MF)

機能

GRID（桁取り）句は、画面に表示される画面項目の各文字の左側に、縦線を付ける。各縦線は、文字位置の範囲内に収まる。

一般形式

GRID

構文規則

1. GRID句は、どのような画面項目にも指定できる。
2. GRID句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. 文字の左側に縦線を引く機能のないシステム上で指定しても、この句は効力を発揮しない。

一般規則

1. GRID句は、画面に表示される画面項目の各文字の左側に、縦線を付ける。各縦線は、文字位置の範囲内に収まる。

9.1.1.15 HIGHLIGHT（強輝度）句

機能

HIGHLIGHT（強輝度）句は、画面上に表示される画面項目の輝度を強くするよう指定する。

一般形式

HIGHLIGHT

構文規則

1. HIGHLIGHT句は、入力項目、出力項目、更新項目、定数項目に対して有効である。
2. HIGHLIGHT句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。

一般規則

1. FOREGROUND-COLOR句とあわせてHIGHLIGHT句を指定すると、前景色の輝度と明るさが増大される。たとえば、ハードウエアの中には、黒が灰色に変えられ茶色が黄色に変えられるものがある。HIGHLIGHT句は、BLANK SCREEN句には適用されない。

9.1.1.16 JUSTIFIED（桁寄せ）句-画面節中

機能

JUSTIFIED（桁寄せ）句は、画面項目にデータを転記するか入力するときの、非標準的なデータの位置付けを指定する。

一般形式

```
{JUSTIFIED} RIGHT
{JUST     }
```

構文規則

1. JUSTは、JUSTIFIEDの省略形である。
2. JUSTIFIED句は、入力項目、出力項目、更新項目に対してだけ指定できる。

一般規則

1. JUSTIFIED句を指定した画面項目が出力項目または更新項目であると、送出し側項目から画面項目にデータが転記されたときに、MOVE（転記）に関する通常の規則に従って桁寄せが行われる。
2. JUSTIFIED句を指定した画面項目が入力項目または更新項目であると、画面項目へのデータの入力が済んだときに、残っているプロンプト文字の数だけ入力されたデータが右にずらされ、空いた左側に空白が埋められる。この操作は、受取り側データ項目にデータが転記される前に行われる。データが入力されなかったときには、この処理は行われない。
3. 画面項目にSECURE（機密）句が指定してある場合の処理は、SECURE句が指定しない場合の処理と同じである。ただし、その処理結果を画面上で見ることはできない。

9.1.1.17 LEFTLINE（左端位置）句

(MF)

機能

LEFTLINE（左端位置）句は、画面に表示される画面項目の左端の文字の左側に、縦線を付ける。その縦線は、文字位置の範囲内に収まる。

一般形式

```
LEFTLINE
```

構文規則

1. LEFTLINE句は、どのような画面項目にも指定できる。
2. LEFTLINE句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. 文字の左側に縦線を引く機能のないシステム上で指定しても、この句は効力を発揮しない。

一般規則

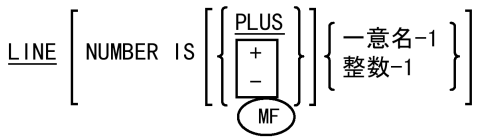
1. LEFTLINE句は、画面に表示される画面項目の左端の文字の左側に、縦線を付ける。その縦線は、文字位置の範囲内に収まる。

9.1.1.18 LINE（行）句

機能

LINE（行）句は、画面項目を画面上に表示する始点となる行を指定する。

一般形式



構文規則

1. 一意名-1は、符号の付かない整数であり、その値は1以上255以下とする。
2. 整数-1は符号なしで、その値は1以上255以下とする。
3. (MF)PLUSと"+" は同義であり、どちらを書いてもよい。
4. NUMBERの指定を省略すると、省略時解釈として値 PLUS 1がとられる。
5. LINE句の指定を省略すると、現在の行が想定される。
6. LINE句は、任意の基本項目に指定できる。
7. (MF)LINE句をSCREEN SECTION集団項目の階層中のどこかに指定する時、以下の句のうち、最低１つを同じ集団の定義に指定しなければならない。
   (MF)BEEP, BELL, BLANK LINE, BLANK SCREEN, ERASE EOL, ERASE EOS, FROM, TO, USING, VALUE

一般規則

1. LINE句は、ACCEPT処理またはDISPLAY処理を行うときの、画面上に画面項目を表示する行を指定する。

2. LINE句に一意名または整数を指定し、PLUS,
   (MF)"+" , "-",
   のどれかを指定しないと、その値は絶対行番号を表わす。01レベルはそれぞれ、画面レコードを表わす。AT指定は、表示画面の先頭に関連付けて、画面レコードの先頭位置を指定する。ACCEPT文またはDISPLAY文にAT指定を指定しなかった場合、画面の最初の行が絶対行1となる。
3. LINE句に PLUS,
   (MF)"+" , "-",
   のどれかを指定すると、一意名または整数に指定した値は先行する画面項目の末尾からの相対行番号を表わす。この場合、先行する項目が実際に画面に表示されるか否かは問わない。先行する画面項目の末尾は、その項目の実行時の有効な長さによって決まる。その長さは、PICTURE句、VALUE句、SIZE句に基づいて算出される。01レベルの項目が出てくると、行番号は1に戻る。
4. 01レベルでない項目の画面記述にLINE句もCOLUMN句も含めないと、COLUMN PLUS 1 が指定されたものとみなされる。つまり、その項目は前の項目の直後に続くことになる。
5. 画面の範囲から外れる行位置を指定すると、ACCEPTまたはDISPLAYにおいて切捨てが発生する。
6. LINE句に PLUS,
   (MF)"+" , "-",
   のどれかを指定し、画面節の項目に関連するACCEPT文またはDISPLAY文でAT LINE NUMBER句を指定した場合、この項目が示す行番号は、2つの数の和または差である。

9.1.1.19 LOWLIGHT（弱輝度）句

機能

LOWLIGHT（弱輝度）句は、画面上に表示される画面項目の輝度を弱くするよう指定する。

一般形式

LOWLIGHT

構文規則

1. LOWLIGHT句を集団項目レベルで指定すると、その下の基本項目すべてに対して適用される。

一般規則

1. LOWLIGHT句は、画面上に表示される画面項目の輝度を弱くするよう指定する。

9.1.1.20 OCCURS（反復）句-画面節中

(MF)

機能

OCCURS（反復）句は、繰り返される同形の画面項目を個々に記述する手間を省き、添字または指標を使用するのに必要な情報を提供する。

一般形式

```
OCCURS 整数-1 TIMES
```

構文規則

1. OCCURS句は、01レベルの項目には指定できない。
2. 項目にUSING句またはTO句を適用する場合、またはある項目が項目に従属する場合、送出し側項目にも同じ出現回数を指定したOCCURS句を使用するか、またはOCCURS句を使用しないようにする。これらのOCCURS句にDEPENDING句を含めてはならない。
3. 項目にFROM句 を適用する場合、またはある項目が項目に従属する場合、送出し側項目にも同じ出現回数を指定したOCCURS句を使用するか、またはOCCURS句を使用しないようにする。これらのOCCURS句にDEPENDING句を含めてはならない。

一般規則

1. OCCURS句の左辺にある画面記述では、LINE句とCOLUMN句は個々の表要素に使用される。したがって、LINE句とCOLUMN句には相対位置を指定する。これは、この2つの句に絶対位置を指定すると、すべての表要素が同じ位置に表示されてしまうためである。
2. OCCURS句を指定した画面項目が出力項目であり送出し側項目にOCCURS句が指定されていないと、DISPLAY処理において送出し側の項目が、画面項目のすべての反復要素に転記される。全面的に添字付けされた基本項目は、OCCURS句が使われていないものとみなされる。
3. OCCURS句を指定した画面項目が更新項目であるか、または送出し側項目にOCCURS句が指定されている出力項目であると、DISPLAY処理において送出し側の項目の内容が画面項目の対応する要素に転記される。
4. OCCURS句を指定した画面項目が更新項目または入力項目であると、ACCEPT処理において画面項目各要素に入力されたデータが、受取り側項目の対応する要素に転記される。

9.1.1.21 OVERLINE（上線）句

(MF)

機能

OVERLINE（上線）句は、画面に表示される画面項目のすべての文字の上側に、水平線を引く。その上線は、文字位置の範囲内に収まる。

一般形式

```
OVERLINE
```

構文規則

1. OVERLINE句は、どのような画面項目にも指定できる。
2. OVERLINE句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. 文字の上側に水平線を引く機能のないシステム上で指定しても、この句は効力を発揮しない。

一般規則

1. OVERLINE句は、画面に表示される画面項目のすべての文字の上側に、水平線を引く。その上線は、文字位置の範囲内に収まる。

9.1.1.22 PICTURE（形式）句-画面節中

機能

PICTURE（形式）句は、画面項目の長さ、一般的性質、編集要件を記述する。

一般形式

```
{PICTURE}
{PIC    } IS 文字列
```

構文規則

1. PICTURE句には、任意の標準編集文字を含めることができる。
2. 各PICTURE句には、 FROM, TO, USING のどれかの指定を含めなければならない。 USING指定は、FROM指定またはTO指定と共に使用してはならない。
3. PICは、PICTUREの省略形である。
4. PICTURE句は、基本項目だけに指定できる。

5. PICTURE句は、FROM，TO，USINGの各指定の中で参照するデータ項目のPICTURE句と必ずしも同じである必要はない。しかし、暗黙のMOVE（転記）が正しく行えるようでなければならない。

一般規則

1. 文字列は、画面項目の長さと項類を表わす。これは、データ項目用のPICTURE句の中の文字列と同じように使用する。
2. 数字画面項目は、すべて数字編集項目とするか、PICTURE句内に9だけを含めるようにすることが望ましい。対応するデータ項目と画面項目との間でデータをやり取りする際に、必要に応じて編集または編集の解除が行われる。

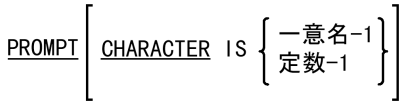
9.1.1.23 PROMPT（プロンプト）句

(MF)

機能

PROMPT（プロンプト）句は、ACCEPT処理を行う際に、画面上の画面項目の空き文字位置にプロンプトを表示するようにする。このプロンプトは、キー入力されるデータをシステムが受け取って、画面項目に入れられる状態にあることを示す。

一般形式



構文規則

1. PROMPT句は、入力項目と更新項目と集団項目にだけ指定できる。
2. PROMPT句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. 一意名-1は、1文字の英字または英数字のデータ項目とする。
4. 一意名-1は、OCCURS句の左辺には指定できない。
5. 定数-1は、1文字の英数字定数または表意定数とする。

一般規則

1. 画面節内の項目には、 :
    a. PROMPT句は常にオンである。
    b. PROMPT句が指定されていない場合、またはCHARACTER指定なしで指定されている場合、省略時解釈のプロンプト文字が使用される。
2. ACCEPT文で参照される他の項目には、

a. ACCEPT文にPROMPT句が指定されている場合にだけ、 PROMPTはオンとなる。
b. PROMPT句がCHARACTER指定なしで指定されている場合、省略時解釈のプロンプト文字が使用される。

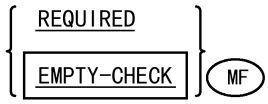
3. CHARACTER句を指定すると、該当する画面項目の空き文字位置にプロンプトが表示されるようになる。プロンプト文字は構成オプションに優先する。(構成オプションの詳細については､ユーザインタフェースに関するCOBOLシステムのマニュアルを参照｡)
4. PROMPT句を指定すると、英数字または自由方式の数字の画面項目の中の後部の空白に代わって、プロンプト文字が表示される。また、固定方式の数字編集画面項目の先行部分の抑制された文字位置に、プロンプト文字が表示される。
5. 固定方式の非編集数字画面項目、またはゼロ抑制が指定されていない数字編集画面項目に対しては、PROMPT句は効力を発揮しない。
6. あるフィールドでPROMPT句がオンの場合、プロンプト文字が示すフィールドの終端までカーソルを移動することはできない。これを行おうとすると、カーソルは次のフィールドに移動する。 .
7. あるフィールドでPROMPT句がオンでない場合、カーソルはフィールドの終わりの後にある空白に移動することができる。
8. SECURE（機密）句を指定すると、この句は効力を発揮しない。
9. 画面項目中に表示されているプロンプト文字は、ACCEPT処理が終わると空白に置き換えられる。

9.1.1.24 REQUIRED（必須）句

機能

REQUIRED（必須）句は、画面項目を必ず入力しなければならないことを指定する。

一般形式



構文規則

1. REQUIRED句は、入力項目と更新項目と集団項目にだけ指定できる。
2. REQUIRED句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
3. (MF)REQUIREDと EMPTY-CHECKは同義語であり、どちらを書いてもよい。

一般規則

1. REQUIRED句が効力を発揮するのは、ACCEPT文が実行されるときである。このときに、画面項目内にカーソルが位置付けられて、入力されたデータを受け取ることが可能となる。REQUIRED句を指定しておくと、終了キーを押しても入力を終了できず、カーソルは項目の先頭に位置付け直されるようになる。
2. この句の要件を満たすためには、英数字の画面項目の場合は、空白でもプロンプト文字でもない文字を少なくとも1つ入力しなければならない。なお、数字の画面項目の場合は、値がゼロであってはならない。
3. 更新項目に関しては、元のデータと入力されたデータの両方によって、この句の要件が満たされるか否かが決まる。
4. ACCEPT処理を終了させるためにファンクション・キーを使用した場合は、REQUIRED句は効力を発揮しない。(構成オプションの詳細については、ユーザインタフェースに関するCOBOLシステムのマニュアルを参照。)
5. REQUIRED句の要件が満たされない場合、エラー・メッセージを画面上に表示させることができる。(構成オプションの詳細については、ユーザインタフェースに関するCOBOLシステムのマニュアルを参照。)

9.1.1.25 REVERSE-VIDEO（反転画面）句

機能

REVERSE-VIDEO（反転画面）句は、画面項目を反転して表示する。

一般形式

```
REVERSE-VIDEO
```

構文規則

1. REVERSE-VIDEO句は、どのような画面項目にも指定できる。
2. REVERSE-VIDEO句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。

一般規則

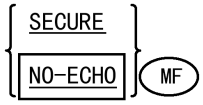
1. REVERSE-VIDEO句は、画面項目を反転して表示する。

9.1.1.26 SECURE（機密）句

機能

SECURE（機密）句は、入力されたデータを画面に表示しないようにする。

一般形式



構文規則

1. SECURE句は、入力項目と更新項目に対してだけ指定できる。
2. SECURE句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。
   (MF)SECUREと NO-ECHOは同義語であり、どちらを書いてもよい。

一般規則

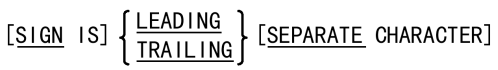
1. 入力項目にSECURE句を指定すると、画面項目内には空白とカーソルだけが現れる。更新項目にSECURE句を指定すると、元の内容は表示されるが、それを変更することはできない。

9.1.1.27 SIGN（符号）句-画面節中

機能

SIGN（符号）句は、演算符号の位置と表現形式を指定する。

一般形式



構文規則

1. SIGN句は、PICTURE文字列に記号文字"S" が含まれる入力項目と出力項目と更新項目にだけ指定できる。
2. SIGN句は基本項目に対してだけ指定できる。

一般規則

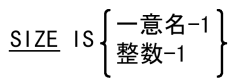
1. 画面記述中にSIGN句を指定するときは、SEPARATE句も含めることが望ましい。SEPARATE句を指定しないと、PICTURE文字列中の"S" によって表わされる符号は数字と重なって表示される。

9.1.1.28 SIZE（大きさ）句

機能

SIZE（大きさ）句は、画面項目の現在の大きさを指定する。

一般形式



構文規則

1. SIZE句は、基本画面項目に対してだけ指定できる。
2. 一意名-1は、符号の付かない整数とする。また、一意名-1はOCCURS句の左辺にあってはならない。
3. 整数-1は、符号なしとする。

一般規則

1. 指定した大きさがゼロのときは、SIZE句は効力を発揮しない。
2. 数字または数字編集の画面項目に、値がゼロでないSIZE句を指定すると、画面項目は自由方式のように扱われる。この指定は、構成オプションの設定に優先する。
3. 関連するPICTURE句またはVALUE句によって暗黙的に示されるよりも値が小さいSIZE句を指定すると、画面上では画面項目の左側の部分だけが表示される。ACCEPT文にJUSTIFIED句がある場合、画面項目の右側の部分だけが表示される。その画面項目の残りの部分は、状況に応じて、空白またはゼロとみなされる。
4. 出力項目または定数項目に、関連するPICTURE句またはVALUE句によって暗黙的に示されるよりも値が大きいSIZE句を指定すると、画面項目の右側の部分に空白が埋め込まれる。
5. 一意名-1の値を変更すると、実行時に画面項目の有効な大きさが変えられる。このため、画面節内でその項目の後ろに記述されている項目の画面上の位置が、変わることがある。(前述の*LINE句*および*COLUMN句*の節を参照。)

9.1.1.29 TO（へ）句

機能

TO句は受け取るデータの宛先を指定する。

一般形式

TO 一意名-1

構文規則

1. TO句をFROM句と一緒に使用することは、対応して同じ一意名を指定したUSING句を使用することに等しい。TO句をUSING句と一緒に使用してはならない。
2. TO句内の一意名を修飾してもよい。画面項目にOCCURS句が適用されていない場合、その一意名は添字付けまたは指標付けされていてもよい。FROM句を定義できる場所はプログラムのファイル節、作業場所節、
   (MF)局所記憶節、
   連絡節のいずれかだけである。

一般規則

1. TO句が記述されている画面項目に対してACCEPT文を実行すると、操作員が入力したデータが画面項目に受け取られ、それからそのデータが対応するデータ項目に転記される。

9.1.1.30 UNDERLINE（下線）句

機能

UNDERLINE（下線）句は、画面に表示される
画面項目に下線を引く。

一般形式

UNDERLINE

構文規則

1. UNDERLINE句は、どのような画面項目にも指定できる。
2. UNDERLINE句を集団レベルに指定すると、その集団の下位に属するすべての基本項目に適用される。

一般規則

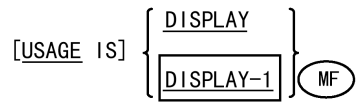
1. UNDERLINE（下線）句は、画面に表示される画面項目に下線を引く。
2. 下線を引く機能のないシステム上で指定しても、この句は効力を発揮しない。

9.1.1.31 画面節内のUSAGE（用途）句

機能

USAGE（用途）句は計算機の記憶領域内のデータ項目の形式を指定する。

一般形式



構文規則

1. 画面節のデータ項目の用途は、明示的または暗黙的に、USAGE DISPLAYまたはUSAGE DISPLAY-1と定義しなければならない。

9.1.1.32 USING（使用）句

機能

USING句は表示するデータの源を指定する。

一般形式

USING 一意名-8

構文規則

1. USING句は、対応する一意名に同じものを指定した、TO句とFROM句に等しい。UNING句をFROM句またはTO句と一緒に使用してはならない。
2. USING句内の一意名を修飾してもよい。画面項目にOCCURS句が適用されていない場合、その一意名は添字付けまたは指標付けされていてもよい。FROM句を定義できる場所はプログラムのファイル節、作業場所節、局所記憶節、連絡節のいずれかだけである。

一般規則

1. USING句が記述されている画面項目に対してDISPLAY文を実行すると、対応するデータ項目から画面項目にデータが転記され、それから画面項目が画面上に表示される。
2. USING句が記述されている画面項目に対してACCEPT文を実行すると、操作員が入力したデータが画面項目に受け取られ、それからそのデータが対応するデータ項目に転記される。

9.1.1.33 VALUE（値）句-画面節中

機能

VALUE（値）句は、画面に表示する定数の情報を指定する。

一般形式

[VALUE IS] 定数-1

構文規則

1. VALUE句に指定する定数は、文字定数とする。表意定数は指定できない。
2. VALUE句は、PICTURE句を伴わない基本項目に対してだけ指定できる。

一般規則

1. VALUE（値）句は、画面に表示する定数の情報を指定する。

9.1.1.34 ZERO-FILL（ゼロ補てん）句

(MF)

機能

ZERO-FILL（ゼロ補てん）句は、後部のプロンプト文字を空白ではなく、ゼロで置き換えることを指定する。

一般形式

ZERO-FILL

構文規則

1. ZERO-FILL句は、英字または英数字の入力項目と更新項目に対してだけ指定できる。

一般規則

1. ZERO-FILL句は、データが画面項目から受取り側項目に転記されるときに、後部のプロンプト文字を空白ではなく、ゼロで置き換えさせる。これは画面項目にデータが入力されたときにだけ行われる。
2. ZERO-FILL句を指定した画面項目の受取り側項目に、JUSTIFIED句が指定されていると、桁寄せによって空いた先行部分にゼロが埋められる。

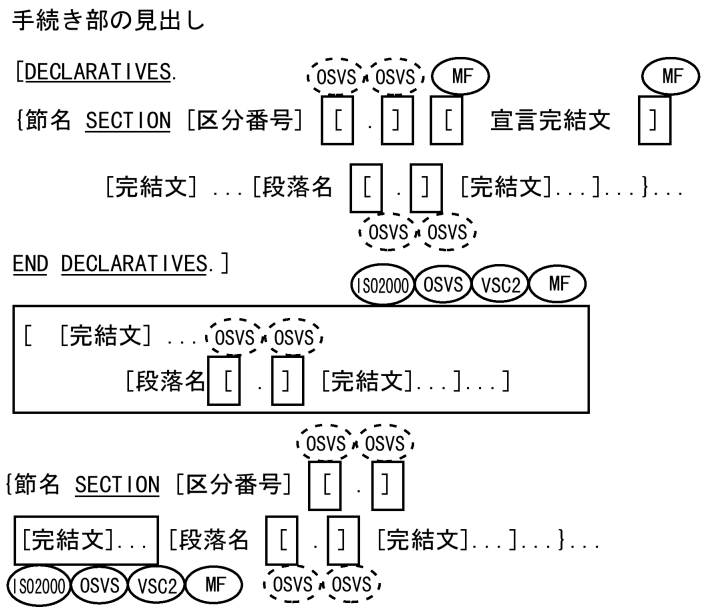
第10章：手続き部

10.1 手続き部

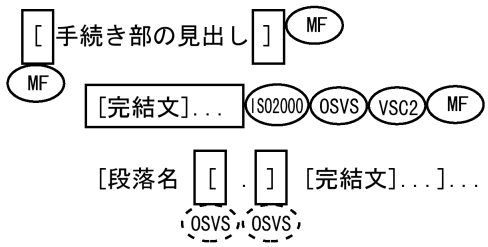
(ANS85)手続き部(procedure division)は、COBOL 原始プログラム中に任意で記述する。プログラム定義では、オブジェクトプログラム内で実行する手続きを含む。これらの手続きは、宣言しても、宣言しなくてもよい。呼出しプロトタイプでは、宣言した、または宣言しない手続きをどちらも含まない。ただし、ENTRY文は含んでも構わない。
宣言節は、手続き部の最初で集団にしなければならない。これを以下の一般形式の書き方 1に示す。

一般形式

書き方 1



書き方 2



書き方 3

```
PROCEDURE DIVISION.
[ DECLARATIVES.
{ 節名 SECTION.
                宣言完結文.
[ 完結文 ] … [ 段落名. [ 完結文 ] … ] … } …
END DECLARATIVES. ]
[ { メソッド定義 } … ]
```

「手続き部の見出し」については、次の *手続き部の見出し*の節で説明する。
上記の 宣言完結文（declarative-sentence）は、USE文である。(後述の*デバッグおよび対話式デバッグ*の節と、***言語リファレンス - 追加トピック***の*報告書作成*の節を参照。) USE文は、該当する節をいつ実行するかを指定する。
手続きは、手続き部内の段落または段落集団、あるいは節または節集団から成る。ある段落がある節内にある場合、すべてがその中になくてはならない。
OSVS VSC2 MFこの規則は強制しない。
段落は、段落名の後に続けて終止符(.)と空白を書き、後ろに完結文をいくつか書いたものである。段落の終了は、次の段落名または節名の直前、あるいは、手続き部の終わり、宣言部分のEND DECLARATIVESのところまでとなる。
節は、節見出しと、その後に続くいくつかの段落から成る。節の終了は、次の節名、あるいは手続き部の終わり、宣言部分のEND DECLARATIVESのところまでとなる。
VSC2 MFしかし、宣言中の節は宣言完結文を必要としない。これはPERFORM文によって開始できる。(PERFORM文の詳細については、後述の*PERFORM*文の節を参照。)
VSC2 MF完結文(sentence)は、上記で説明した名前付きの段落内に明示的には含まれない。名前のない暗黙的な段落内に含まれているとみなされる。段落は、節内に明示的には含まれない。名前のない暗黙的な節内に含まれているとみなされる。
プログラムの定義では、実行は、手続き部の宣言部分ではない最初の文から始まる。以降、文は記述されている順序に実行される。ただし、他の順序が示されている場合は例外である。
MF呼出しプロトタイプでは、手続き部は実行されない。
ここでは、算術式と条件式、共通指定と入出力指定、組み込み関数、COBOL動詞について説明する。

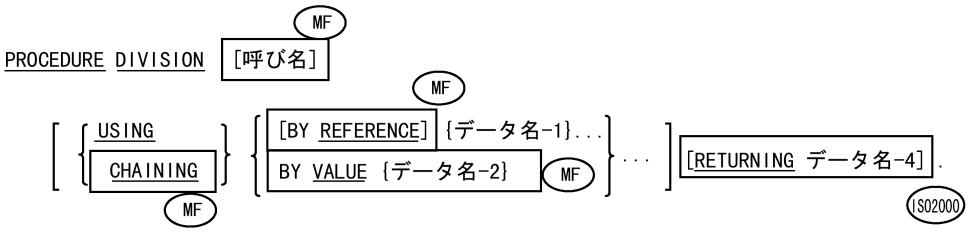
(MF)手続き部の見出しは、手続き部の書き方2では書いても書かなくてもよい。ただしこれは、、これより前に部が記述されていない場合、および手続き部がCOBOLの文で始まる場合(節見出し、段落見出し、宣言節ではなく)にだけ、任意となる。

10.1.1 手続き部の見出し

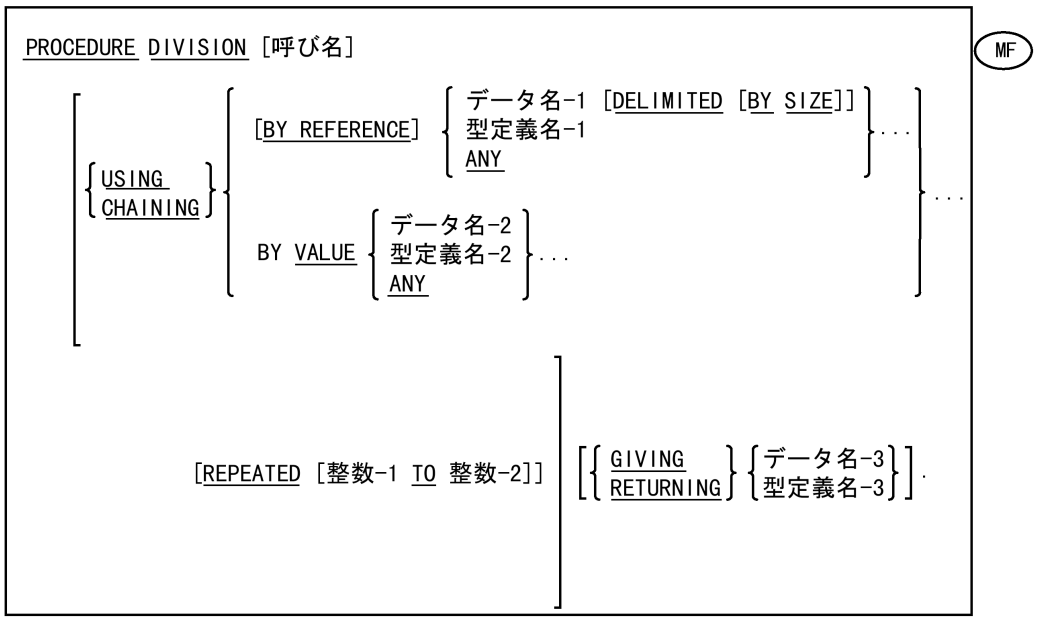
手続き部の冒頭には、下記の見出しを付けなければならない。

一般形式

書き方 1



書き方 2



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - DEFAULTCALLS - 手続き部の見出しに明示的に指定されていない場合に、省略時の呼出し方式を設定する。
    - STICKY-LINKAGE - 同じプログラム内で複数の呼出しを異なる入口点から行う間に、連絡節内のデータ項目の番地参照性を維持するか否かを制御する。

構文規則

「*」マークが付いた構文規則は、手続き部見出しとENTRY文で共通である。これらの規則で、「手続き部見出しまたはENTRY文」と記述されている箇所は、規則をどちらに適用するかによって、「手続き部見出し」または「ENTRY文」を該当させて読むこと。

書き方

1. (MF)呼び名は、目的プログラムが他のプログラムにより起動され、そのプログラムがCOBOLシステムの省略時設定として使用される以外の呼出し規則を使用している場合にだけ必要である。一般に、省略時設定のCOBOL呼出し規則は、実行時環境のCOBOL以外の言語の環境によって使用されるものと一致する。

(MF)呼び名は、特殊名段落で定義しなければならない。この方法、および実行時環境でサポートされる呼出し規則とのインタフェースをCOBOLシステムで記述する方法についての詳細は、*特殊名段落*の節を参照。

2. (MF)CHAININGおよびUSINGは同等である。
3. * USING句は、目的プログラムで、他のプログラムにより起動される呼ばれるプログラム、パラメータを渡す呼ぶプログラムの場合だけに必要である。呼ぶプログラムがCOBOLの場合、呼ばれるプログラムがUSING句を含むCALL文
   (MF)またはCHAIN文
   により起動される時に、パラメータが渡される。呼ぶプログラムがCOBOL以外の場合、パラメータが渡される方法については、その言語のマニュアルを参照のこと。
4. * 手続き部見出し
   (OSVS)(VSC2)(MF)または ENTRY文
   のUSING句は、呼ばれるプログラムにより使用される名前を、呼ぶプログラムがそれに渡すパラメータ用に認識する。呼ぶプログラムがCOBOLの場合、呼ばれるプログラムに渡されるパラメータは、呼ぶプログラムのCALLのUSING句、
   (MF)またはCHAIN文
   で認識される。名前の2つのリスト間の相応は、位置ベースで確立され（CALL文参照）、(MF)呼ばれるプログラムの手続き部見出しまたは呼ぶプログラムのCALL文で指定された呼び名によって示されるものに委ねられる。
5. (VSC2)(MF)* USING句の項目は、USAGE IS POINTERデータ項目またはADDRESS特殊レジスタの名前のどちらかであるCALL文のUSING句の項目を提供するUSAGE IS POINTERのデータ項目の名前にすることができる。
   (MF)USING句の項目は、USAGE IS PROCEDURE-POINTERデータ項目の名前であるCALL文のUSING句の項目を提供するUSAGE IS PROCEDURE-POINTERのデータ項目の名前にすることができる。

書き方 1

6. * データ名-1は、呼ばれるプログラムのリンク節か、
   (MF)ファイル節か、作業領域節
   で、レベル01またはレベル77項目として定義されなければならない。特定のユーザ定義の語は、データ名-1として 2回以上現れてはならない。
   (OSVS)(VSC2)(MF)2回以上現れてもよい。
   データ名-1の記述項目は、REDEFINES句を含んではならない。
   (OSVS)(VSC2)(MF)REDEFINES句を含んでもよい。
   しかし、データ名-1はリンク節のどこにおいても REDEFINES句の対象にできる。
7. (MF)* データ名-2は、呼ばれるプログラムのリンク節か、ファイル節か、作業領域節で、レベル01またはレベル77項目として定義されなければならない。

書き方 2

8. (MF)* 書き方2は、PROGRAM-ID段落に記述されたEXTERNAL句のあるプログラムでだけ使用することができる。 これは、呼出しプロトタイプである。

9. (MF)* データ名-1とデータ名-2は、リンク節の01レベルレコードとして定義しなければならない。
10. (MF)* 型定義名-1、型定義名-2、型定義名-3は、同じ原始ファイルにTYPEDEF句でプログラマ定義の使用としてあらかじめ定義しなければならない。

一般規則

「*」マークが付いた一般規則は、手続き部見出しとENTRY文で共通である。これらの規則で、「手続き部見出しまたはENTRY文」と記述されている箇所は、規則をどちらに適用するかによって、「手続き部見出し」または「ENTRY文」を該当させて読むこと。

書き方

1. * 手続き部見出しのUSING句およびCALL文のUSING句では、5つまでのデータ名が許される。
   (XOPEN)手続き部見出しのUSING句、CALL文のUSING句、
   (MF)ENTRY文のUSING句では、62までのデータ名が許される。
2. * BY REFERENCE句およびBY VALUE 句は、他のBY REFERENCE句またはBY VALUE句があるまで、それに続くパラメータを移行する。はじめのパラメータに先行してBY REFERENCE句またはBY VALUE句が記述されていないと、BY REFERENCE句が想定される。

書き方 1

3. (MF)呼び名は、このプログラムがそれをプログラム名あるいは入口名で起動した呼ぶプログラムにより適用されたとみなされる呼出し規則を確認する。実際に使用された呼出し規則が呼び名によって示されたものと異なる場合、COBOLシステムは混乱する。
4. (MF)* データ名-1がレベル01またはレベル77項目としてファイル節あるいは作業領域節に定義されている場合、目的プログラムは、データ項目がデータ名-1と同じデータ宣言でリンク節に定義されたかのように動作し、呼ばれるプログラムの最初の文の実行に先立って、このデータ項目の内容がデータ名-1に移動されたかのように動作する。最初のプログラムでは、これらの値はプログラムの作業領域のデータの初期化によって上書きされる。このため、呼ばれるプログラムには使用できない。
5. * 呼ぶプログラムがCOBOLの場合、以下の規則を適用する。呼ぶプログラムがCOBOLでない場合、BY REFERENCEまたはBY VALUE句を使用する必要がある時の詳細は、COBOLシステムのマニュアルを参照。
6. * 呼ばれるプログラムの手続き部見出し
   (MF)またはENTRY文
   のUSING句中の作用対象の出現列、あるいは、呼ぶプログラムがCOBOLでない場合の呼ぶプログラムのCALL文
   (MF)またはCHAIN文またはパラメータリストの相応するUSING句中の作用対象の出現列は、

呼ぶプログラムと呼ばれるプログラムにより使用されるデータ名間の相応を決定する。この相応は位置上のもので、名前が等しいことによるものではない。呼ばれるプログラムのUSING句の最初の作用対象は、呼ぶプログラムのUSING句またはパラメータリストの最初の作用対象に相応し、二番目は二番目と相応する。

(MF)作用対象のいくつかが異なるため、相応が完了しない場合、残りの相応しない作用対象がCALL文中であれば無視され、手続き部見出しまたはENTRY文中であれば呼ばれるプログラム中では参照されてはならない。位置の相応は、呼び名により示されるCOBOL以外の呼出し規則に依存する。

(MF)呼ばれるプログラムまたは呼ぶプログラムのUSING句に指定された作用対象の数が一致しないと、実行時に明らかになるだけで、FLAG指令はこの拡張をフラグ付けしない。

7. * 手続き部見出し

(MF)またはENTRY文

のUSING句中の各データ項目は、呼ぶプログラムのCALL文

(MF)またはCHAIN文

中の相応するデータ項目が内容または参照によって渡されるパラメータを宣言している場合、BY REFERENCE句が呼ばれるプログラム中に宣言または示されていなければならない。

(MF)呼ぶプログラムのCALL文中の相応するデータ項目が値によって渡されるパラメータを宣言している場合、BY VALUE句が呼ばれるプログラム中に宣言または示されていなければならない。

8. * 呼ぶプログラムのCALL文

(MF)またはCHAIN文

中の相応するデータ項目への参照が内容によって渡されるパラメータを宣言している場合、項目の値は、CALL文が実行され、CALL文の一般規則に記述されている属性を持つCOBOLシステム定義の格納項目中に置かれた時に移動される。目的プログラムは、このCOBOLシステム定義の格納項目が参照によって渡されるかのように動作し、呼ばれるプログラムの相応するデータ項目と同じ格納領域を占めるかのように動作する。CALL文のBY CONTENT句中の各パラメータに相応する各COBOLシステム定義の格納項目データ宣言は、手続き部見出し

(MF)またはENTRY文のUSING句中の相応するパラメータのデータ宣言と同じでなければならない（変換、拡張、丸めをしないで）。

9. * 呼ぶプログラムのCALL文

(MF)またはCHAIN文

中の相応するデータ項目への参照が参照によって渡されるパラメータを宣言している場合、目的プログラムは呼ばれるプログラムのデータ項目が呼ぶプログラムのデータ項目と同じ格納領域を占めるかのように動作する。呼ばれるプログラムのデータ項目の宣言は、呼ぶプログラムの一致するデータ項目によって記述されるのと同じ数の文字位置を記述しなければならない。

10. (MF)* 呼ぶプログラムのCALL文中の相応するデータ項目への参照が値によって渡されるパラメータを宣言している場合、項目の値は、CALL文が実行され、CALL文の一般規則に記述されている属性を持つシステム領域中に置かれた時に移動される。目的プログラムは、このシステム領域が参照によって渡されるかのように動作し、呼ばれるプログラムの相応するデータ項目と同じ格納領域を占めるかのように動作する。CALL文のBY VALUE句中の各パラメータに相応する各システム領域のデータ宣言は、手続き部見出しまたはENTRY文のUSING句中の相応するパラメータのデータ宣言と同じでなければならない（変換、拡張、丸めをしないで）。
11. * 呼ばれるプログラム中ではいつでも、データ名-1
(MF)およびデータ名-2
への参照は、呼ばれるプログラムのリンク節で与えられた項目の記述に従って決定される。この記述が呼ぶプログラムの相応するデータ項目よりも多くの文字位置を定義している場合、予期せぬ結果となる。この規則の適用に失敗したり、特定のランタイム環境用のシステム領域で許される最大サイズを超えると、システムはたいへんな損害を被ることになる。
12. * 呼ばれるプログラムのリンク節で定義されたデータ項目は、以下の条件をいずれか一つ満たす場合に、そのプログラムの手続き部で参照される。
    a. それらが、手続き部見出し
    (MF)またはENTRY文のUSING句の作用対象である。
    b. それらが、手続き部見出し
    (MF)またはENTRY文のUSING句の作用対象に従属する。
    c. それらがREDEFINES句またはRENAMES句で定義され、上記の条件を満たすものである。
    d. それらが、上記のいずれかの条件を満たす項目に従属する。
    e. それらが、上記のいずれかの条件を満たすデータ項目に関連した条件名または索引名である。
    f. SET ADDRESS OF文が、そのデータ項目を受取り項目として実行された。
13. (MF)* プログラムが呼ばれ、 USING句のBY REFERENCE作用対象が呼ぶプログラムのパラメータに相応する時、関連する接続が確立され、呼ぶプログラムに制御が戻るまで保たれる。プログラムが間にキャンセルせずに二度呼ばれ、同じBY REFERENCE作用対象が呼ぶプログラム中のパラメータと一致しない場合、この作用対象をSTICKY-LINKAGEコンパイラ指令を指定せずに参照してはならない。
14. USING指定を指定した場合、INITIAL句はどの通信記述にも指定してはならない。(***言語リファレンス - 追加トピック***の*通信*の章の*通信記述項 - 全体的な骨組み*の節を参照。)

書き方 2

15. (MF)* 呼出しプロトタイプ（PROGRAM-ID段落中のEXTERNAL句付きのプログラム）が 書き方 2 PROCEDURE DIVISION見出しを持ち、同じ原始ファイル中の他のプログラムがこの呼出しプロトタイプのプログラム名を参照するCALL文を持っている場合、以下の規則が提供される。

1. 呼出し規則がCALL文中に記述されている場合、これはプロトタイプに（明示的あるいは暗黙的に）記述されたものと一致しなければならない。CALL文に呼出し規則が記述されていない場合、プロトタイプに（明示的あるいは暗黙的に）記述された呼出し規則が使用される。
2. CALL文に記述されたパラメータの数は、プロトタイプに記述されたパラメータの数と同じでなければならない（REPEATED句により許されたものを除く）。
3. CALL文に記述された各パラメータには、以下の規則が適用される。
    a. BY REFERENCE句またはBY CONTENT句がパラメータと関連している場合、プロトタイプ中の相応するパラメータは、BY REFERENCEで指定されなければならない。
    b. BY VALUE句がパラメータと関連している場合、プロトタイプ中の相応するパラメータは、BY VALUEで指定されなければならない。
    c. CALL 文中のパラメータがBY REFERENCE句、BY CONTENT句、BY VALUE句を持たない場合、プロトタイプに（明示的あるいは暗黙的に）記述された句が使用される。
    d. この規則は、相応する呼出しプロトタイプが見つからない時に適用される規則を置換える。つまり、呼出しプロトタイプが使用されない時、はじめのBY句までのすべてのパラメータは、BY REFERENCEが指定され、BY句の後のすべてのパラメータ（これ自身は明示的なBY句を持たない）が最後の一つを使用するかのように扱われる。しかし、相応する呼出しプロトタイプが見つかる時は、BY句用の定義が優先する。
    e. プロトタイプ中のパラメータの字類が数字、あるいはポインタか索引データ名の場合、CALL文中のパラメータは同じデータ定義を持たなければならない。
    f. プロトタイプ中のパラメータの字類が英数字である場合、CALL文中のパラメータも英数値クラスもので、プロトタイプ中のパラメータと少なくとも同じ長さでなければならない。
    g. ANYがプロトタイプに記述されている場合、CALL文中のパラメータはどの字類でもよい。
4. CALL文が GIVING句またはRETURNING句を含んでいる場合、プロトタイプも句を含んでいなければならない（逆も同様）。CALLに記述されたデータ項目のデータ定義は、プロトタイプに記述されたデータ項目のものと同じでなければならない。

5. プロトタイプ中のDELIMITED BY SIZE句は、英数字パラメータ用にだけ使用される。この場合、CALL文中の対応するパラメータは暗黙に割当てられたデータ領域に移動され、バイナリ0（x"00"）文字がこのデータのコピーの直後に置かれる。BY SIZE句が省略されると、CALL 文中の対応するパラメータが暗黙に割当てられたデータ領域に移動され、バイナリ0（x"00"）文字がコピーデータの最後のスペース以外の文字の直後に置かれる。これは、ヌルで終わるテキスト文字列を使用するC言語のような言語とのインタフェースを取ることを目的としている。
6. REPEATED句は、少なくとも整数-1と最終パラメータの整数-2の繰返しがあることを示す。整数-1 TO整数-2が記述されないと、整数-1は0となり、整数-2は無限となる。

10.1.2 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。算術式として許される変数、数字定数、算術演算子、かっこの組合わせを表10-1に示す。

1. (OSVS)(VSC2)(MF)以下の記述において、数字データ項目というときは、浮動小数点数データ項目も含まれる。
2. (OSVS)(VSC2)(MF)以下の記述において、数字定数というときは、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。

表 10-1: 算術式の中で使用できる要素

最初の要素	2番目の要素				
	変数	* / ** + -	単項 + -	(	)
変数	-	P	-	-	P
* / ** + -	P	-	P	P	-
単項 + -	P	-	-	P	-
(	P	-	P	P	-
)	-	P	-	-	P

P　　許される要素の組合わせを示す
-　　無効な組合わせを示す
変数　　一意名または定数を示す

10.1.2.1 算術演算子

算術式に使用できる演算子（arithmetic operator）には、5つの2項演算子（binary arithmetic operator）と、2つの単項演算子（unary arithmetic operator）がある。これらの演算子は特定の記号で表わし、前後を空白で区切る。

2項演算子	意味
+	加算
-	減算
*	乗算
/	除算
**	べき乗

単項演算子	意味
+	数字定数+1を掛けることと同じ
-	数字定数-1を掛けることと同じ

10.1.2.2 書き方と評価規則

1. 算術式の中では、要素が評価される順序を指定するために、かっこを使用できる。 かっこ内の式が先に評価される。入れ子になった何組かのかっこについては、最も内側の組を最初にして、順次外側の組へと評価が進む。かっこがない場合、またはかっこ内の要素が同じ水準にある場合、評価順位は下記のようになる。

   1番目　　単項演算子の+と-
   2番目　　指数
   3番目　　乗算と除算
   4番目　　加算と減算

2. 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 " -" 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。

10.1.3 数字式

算術式は、COMPUTE文 、IF文、SEARCH文、部分参照の箇所等で使用する。
数字式は、実行時に一つの数値結果を得るために評価される。数字式の要素は、数字定数でも数字変数でもよい。

10.1.3.1 数字式の評価

この式は、一般的な評価順位で評価される。

```
    1 + 2 * 3
```

は、まず

```
     2 * 3
```

とし、次に

```
     1 + 6
```

つまり、式の結果は7となる。
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38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。38桁に格納できない桁は切り捨てられる。
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。詳細は **IBM VS COBOL for OS/VS** と**VS COBOL II Application 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(*組み込み関数*の節を参照)、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、プラットホームにより多少異なる。

10.1.4 条件式

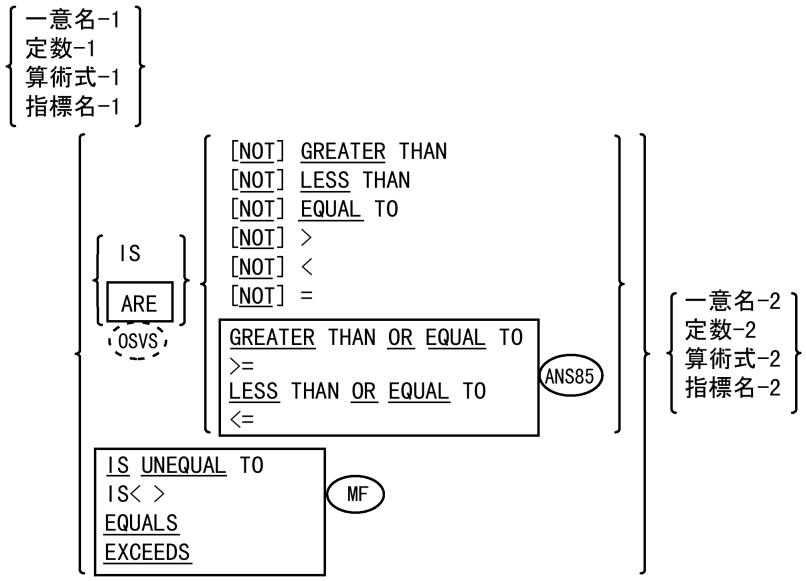
条件式（conditional expression）は、 実行用プログラムの流れを制御する条件を表わす。条件を評価した結果の真理値に応じて、プログラムの処理の経路が選択される。条件式は、
(ANS85)EVALUATE（評価）、
IF（判断）、PERFORM（実行）、SEARCH（表引き）の各文の中に指定する。条件式は単純条件と複合条件に分類される。どちらの種類の条件も、任意の数のかっこで囲むことができる。 かっこで囲んでも、条件の種類は変わらない。
(OSVS)OSVSシステム指令を設定した場合、 条件式の中で部分参照を行うことはできない。

10.1.4.1 単純条件

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

10.1.4.2 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)(VSC2)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、下線を引いていないので注意。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	S GREATER THAN OR EQUAL TO

表 10-2: 比較演算子

比較演算子	意味
[IS / ARE] (OSVS) [NOT] GREATER THAN [IS / ARE] (OSVS) [NOT] >	より大きい、または より大きくない
[IS / ARE] (OSVS) [NOT] LESS THAN [IS / ARE] (OSVS) [NOT] <	より小さい、または より小さくない
[IS / ARE] (OSVS) [NOT] EQUAL TO [IS / ARE] (OSVS) [NOT] = [IS] <> (MF)	等しい、または 等しくない
IS GREATER THAN OR EQUAL TO IS > = (ANS85)	以上
IS LESS THAN OR EQUAL TO IS < = (ANS85)	以下
EQUALS (MF)	等しい
IS UNEQUAL TO < > (MF)	等しくない
EXCEEDS (MF)	より大きい

比較文字の″<″と″>″と″=″は必要語であるが、下線を引いていない。これは、≥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。

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大小順序に基づいて行われる。（前述の*実行用計算機段落*の節を参照。）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。(*MOVE*文の節、および*使用する記号*の項のPICTURE文字"P"の説明を参照。)
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。(*MOVE*文の節、および*使用する記号*の項のPICTURE文字"P"の説明を参照。)
3. 整数ではない数字作用対象は、文字作用対象とは比較できない。

作用対象の大きさは、作用対象中の標準データ形式文字の合計数である。

(OSVS)表意定数を持つ数字編集データ項目を比較することができる。これは英数字比較として扱われる。

(MF)用途が違っていても、数字作用対象と文字作用対象を比較できる。数字作用対象は、標準データ形式で同じ大きさのUSAGE DIAPLAY項目に転記されたように扱われる。そして、この内容が文字作用対象と比較される。

比較処理は2つの場合に分けられる。

1. 作用対象の大きさが等しい場合: 両方の作用対象の大きさが等しい場合は、上位の文字位置から下位へ向けて、対応する文字が順々に比較される。この比較は、対応する文字を比較して不一致が見つかるか、または作用対象の末尾にまで達した時点で終了する。両方の作用対象が等しいと判定されるのは、最初から最後まで、対応する文字がすべて等しかったときである。

   不一致であった最初の対応する文字は、文字の大小順序を比較される。そして、大小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。

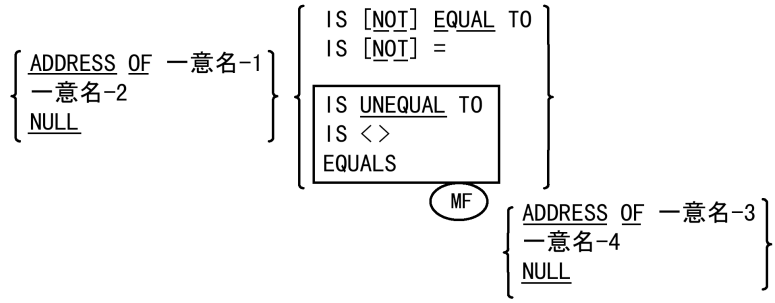
4. (OSVS)(VSC2)指標名と算術式。指標名の値に対応する出現番号が、式を計算した値と比較される。
5. 指標データ項目を 上記以外のデータ項目または定数と比較すると、結果はどうなるかわからない。

10.1.4.2.1 用途がポインタであるデータ項目に関する比較

(VSC2)(MF)

用途が明示的にまたは暗黙的に、ポインタとされている2つのデータ項目を比較できる。ポインタの比較では、等しいか等しくないかだけを検査する比較演算子だけを使用できる。

一般形式



構文規則

1. 一意名-1および一意名-3は、連絡節の中の01レベルまたは77レベルの項目を参照する。
2. (MF)一意名-1および一意名-3は、データ部の中の任意のデータ項目を参照できる。
3. 一意名-2および一意名-4は、USAGE IS POINTERと指定されている項目を参照する。

表意定数 NULLは、1つの作用対象にだけ指定できる。

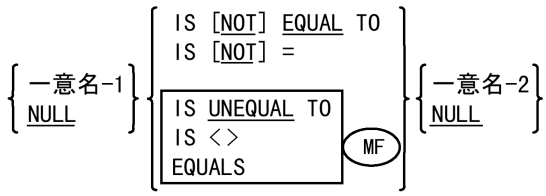
一般規則

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

10.1.4.2.2 用途が手続きポインタであるデータ項目に関する比較

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

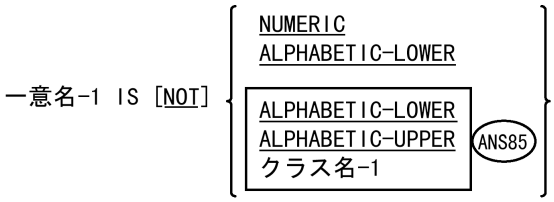
1. 2つの番地が等しいときに、作用対象は等しいとみなされる。それ以外の場合は、作用対象は等しくないとみなされる。
2. この種の比較条件は、IF，PERFORM，EVALUATE，SEARCH（書き方1）の各文の中で指定できる。 書き方2のSEARCH文（SEARCH ALL）の中では、この種の比較条件は使用できない。ポインタ・データ項目は、意味のある順序付けをすることができないからである。

10.1.4.2.3 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、用途は次のどれかとする。
   DISPLAY，
   (MF)COMPUTATIONAL，COMPUTATIONAL-X，
   (OSVS)(VSC2)(MF)(XOPEN)COMPUTATIONAL-3 ，
   (MF)(XOPEN)COMPUTATIONAL-5，
   (ANS85)PACKED-DECIMAL
3. (ANS85)一意名-1が関数一意名である場合、英数字関数を参照しなければならない。

一般規則

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*の節を参照。 .

(OSVS)(MF)NUMERIC検査を、データ記述項に演算子が記述されている基本項目から成る集団項目に対して使用できる。

3. ALPHABET検査は、データ記述に数字であると記述されているデータ項目に対しては、使用できない。検査対象のデータ項目が英字であると判定されるのは、その内容が大文字の英字"A"から"Z" と空白の任意の組合わせ、

(ANS85)および小文字の英字"a"から"z"

と空白の任意の組合わせで構成されているときだけである。

(OSVS)(VSC2)(MF)外部浮動小数点数項目（USAGE DISPLAY）および内部浮動小数点数項目（USAGE COMP-1およびUSAGE COMP-2）には、字類条件を使用できない。

4. (ANS85)ALPHABETIC-LOWER検査は、データ記述に数字であると記述されているデータ項目に対しては、使用できない。 ALPHABETIC-LOWER検査の結果は、一意名-1によって参照されるデータ項目の内容が、すべて小文字の英字"a" から"z" と空白で構成されているときに、真となる。
5. (ANS85)ALPHABETIC-UPPER検査は、データ記述に数字であると記述されているデータ項目に対しては、使用できない。ALPHABETIC-UPPER検査の結果は、一意名-1によって参照されるデータ項目の内容が、すべて大文字の英字"A" から "Z" と空白で構成されているときに、真となる。
6. (MF)字類名-1検査は、データ記述に数字であると記述されているデータ項目に対しては、使用してはならない。

10.1.4.2.4 条件名条件（条件変数）

機能

条件名条件（condition-name condition）は、条件変数（condition variable）の値が条件名に対応する値のどれかと等しいか否かを判定する。

一般形式

条件名

構文規則

1. (OSVS)(VSC2)(MF)条件名に、2バイト文字および内部浮動小数点数を使用できる。
2. (MF)条件名に、外部浮動小数点数を使用できる。

一般規則

1. 条件名にいくつかの値の範囲が関係付けられている場合、条件変数の値がその範囲内（境界値を含む）に入るか否かが検査される。
2. 条件変数と条件名の値を比較する際の規則は、比較条件の場合と同じである。
3. 条件変数の値が条件名に対応する値の中のどれかに等しければ、検査の結果は真となる。

10.1.4.2.5 スイッチ状態条件

スイッチ状態条件（switch-status condition）は、
(MF)9つある
COBOLスイッチのそれぞれが「オン」の状態にあるか「オフ」の状態にあるかを判定する。
(MF)スイッチには、順にSWITCH-0からSWITCH-8という名前が付けられている。
それらの各スイッチの値（「オン」と「オフ」）は、COBOL実行用プログラムの実行を開始するときに、操作員によって設定される。(ランタイムスイッチの詳細については、 COBOLシステムのマニュアルを参照。）条件名に対応するスイッチと「オン」または「オフ」状態は、環境部の特殊名段落の中で定義しておく。

一般形式

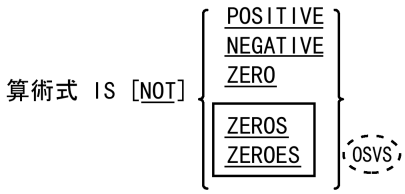
条件名

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。
COBOLスイッチがプログラムの中から設定されている場合にだけ、スイッチ状態条件を使用して検査することができる。

10.1.4.2.6 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。

一般形式



正負条件の必要語に "NOT" を付けたものは、1つの正負条件である。この条件に照らして代数検査が行われ、真理値が求められる。たとえば、 "NOT ZERO" は作用対象の値がゼロでない(正または 負) のときに真となる。作用対象は、値がゼロよりも大きいときに正であり、値がゼロよりも小さいときに負であり、値がゼロのときゼロである。算術式は、少なくとも1つの変数を参照しなければならない。
OSVS符号検査において、ZEROの代わりにZEROSまたはZEROESを使用できる。

10.1.4.3 複合条件

複合条件（complex condition）は、単純条件や複合条件を 論理連結語 10.1.4.3.1 否定単純条件
否定単純条件（negated simple condition）は、単純条件に 論理演算子の"NOT" を付けたものである。否定単純条件は、単純条件の真理値を逆転させる働きをする。したがって、単純条件の真理値が「偽」のときに否定単純条件の真理値は「真」となり、単純条件の真理値が「真」のときに否定単純条件の真理値は「偽」となる。否定単純条件は、単純条件の真理値を逆転させる働きをする。したがって、単純条件の真理値が「偽」のときに否定単純条件の真理値は「真」となり、単純条件の真理値が「真」のときに否定単純条件の真理値は「偽」となる。否定単純条件を かっこで囲んでも真理値は変わらない。

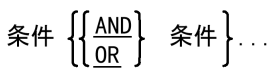
一般形式

NOT 単純条件

10.1.4.3.2 組合せ条件と否定組合せ条件

組合せ条件（combined condition）は、論理演算子の"AND" または "OR" で条件をつないだものである。

一般形式



構文規則

1. 条件には、下記のものが当てはめられる。
    - 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*の節を参照。)

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、表 10-3 に示す。左かっこと右かっこは1対1で対応し、左かっこが右かっこよりも左側にこなければならない。

表 10-3: 条件と論理演算子とかっこの組合せ

要　　素	条件式中の位置	直前にきてよい要素	直後にきてよい要素
単純条件	どこでも可	OR, NOT, AND, (	OR, AND,)
OR または AND	最初と最後は不可	単純条件,)	単純条件, NOT, (
NOT	最後は不可	OR, AND, (	単純条件, (
(	最後は不可	OR, NOT, AND, (	単純条件, NOT, (
)	最初は不可	単純条件,)	OR, AND,)

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

10.1.4.4 略記組合せ比較条件

単純比較条件または否定単純比較条件を、論理結合語で組み合わせていくつも続けて書く場合、先行する比較条件と共通の左辺または左辺と比較演算子があり、かつかっこを使用していなければ、最初のものを除く共通の比較条件を省略できる。下記の2通りの方法がある。

- 比較条件の左辺を省略する
- 比較条件の左辺と比較演算子を省略する

一般形式

比較条件 { { AND / OR } [NOT][比較演算子] オブジェクト } ...

一続きの比較条件の中で、上記の両方の形の省略方法を使用できる。略記した場合、先行する明記した最後の左辺が省略した左辺を補うように挿入され、先行する明記した最後の比較演算子が省略した比較演算子を補うように挿入される。 このように暗黙的に左辺および比較演算子が挿入された結果は、表 10-3の規則に合わなければならない。省略した左辺や比較演算子が補われるこの処理は、組合せ条件の中に略記しない単純条件が出てくると終了する。
(OSVS)条件が評価される順序を、かっこを使用することによって変更できる。（下記の例を参照）
略記組合せ比較条件の中で"NOT" を使用した場合、下記のように解釈される。

1. (ANS85)NOTの直後に GREATER, ＞, LESS, ＜, EQUAL, = のどれかが続く場合、NOTは比較演算子の一部と解釈される。ただし、GREATER THAN OR EQUAL TO, ＞=, LESS THAN OR EQUAL TO, ＜=は上記の比較演算子に含まれない。
2. 上記以外の場合、"NOT" は論理演算子として解釈される。したがって、否定比較条件の中に暗黙の左辺または比較演算子が補われる。

略記組合せ比較条件および略記否定組合せ条件と、それに相当する略さない形の例をいくつか下に示す。

略記組合せ比較条件	略さずに書いた場合
a ＞ b AND NOT ＜ c OR d	((a ＞ b) AND (a NOT ＜ c)) OR (a NOT ＜ d)
a NOT EQUAL b OR c	(a NOT EQUAL b) OR (a NOT EQUAL c)
NOT a = b OR c	(NOT (a = b)) OR (a = c)
NOT (a GREATER b OR ＜ c)	NOT ((a GREATER b) OR (a ＜ c))
NOT (a NOT ＞ b AND c AND NOT d	NOT ((((a NOT ＞ b) AND (a NOT ＞ c)) AND (NOT (a NOT ＞ d))))
x ＞ a OR y AND z	x ＞ a OR (x ＞ y AND x ＞ z)
(OSVS)x ＞ a OR (y AND z)	x ＞ a OR (x ＞ y AND x ＞ z)
(OSVS)x ＞ (a OR y) AND z	(x ＞ a OR x ＞ y) AND x ＞ z
(OSVS)x (= a OR ＞ b)	x = a OR x ＞ b
(OSVS)x = a AND (＞ b OR ＜ z)	x = a AND (x ＞ b OR x ＜ z)
a EQUAL b OR NOT GREATER OR EQUAL c OR d	(a EQUAL b) OR (NOT (a GREATER OR EQUAL c)) OR (a GREATER OR EQUAL d)
a EQUAL b OR NOT ＞=c OR d	(a EQUAL b) OR (NOT (a ＞= c)) OR (a ＞= d)

10.1.4.4.1 条件評価規則

暗黙的な評価順序を変更する必要があるときは、かっこを使用して、組合せ条件の個々の条件が評価される順序を指定できる。
かっこ内の条件が先に評価される。かっこが入れ子になっている場合は、最も内側のかっこから順次外側へと評価が進められる。かっこが使用されていないかまたはかっこ内の条件が同じレベルにあるときは、下記の暗黙的な階層順序に従って論理評価が進められ、最終的な真理値が決定される。

1. (ANS85)条件に算術式または関数が含まれている場合、その条件が評価されるときに、これらの式および関数の値が算出される。同様に、組合せ条件に否定条件が含まれる場合、この組合せ条件を評価する必要があるときに、否定条件が評価される。(前述の*算術式*の節の*書き方と評価規則*を参照。)
2. 単純条件の真理値が、下記の順で決定される。
   比較（略記比較条件があれば、通常の形に展開してから）
   字類
   条件名
   スイッチ状態
   正負
3. 否定条件の真理値が決定される。
4. 組合せ条件の真理値が決定される。論理演算子 "AND" が先に評価され、つづいて論理演算子 "OR" が評価される。
5. 否定組合せ条件の真理値が決定される。
6. 一連の条件の評価順序がすべてかっこで指定されていない場合、同じレベルの複数の条件は左から右へ評価される。

10.1.5 多くの文に共通する句と一般規則

以降に説明する各文において、次のような指定が頻繁に出てくる。ROUNDED（四捨五入）指定、
ON SIZE ERROR（桁あふれ）指定、
(ANS85)NOT ON SIZE ERROR（桁あふれなし）指定、
CORRESPONDING（対応）指定などである。以下に、これらを個々に説明する。
以降の説明の中で、 「結果の一意名」という用語を使用する。これは算術演算の結果を入れる一意名を表わす。

10.1.5.1 ROUNDED指定

小数点の位置をそろえたとき、算術演算の結果の 小数部の桁数が結果の一意名の小数部の桁数よりも大きいと、結果の一意名の大きさに応じて 切捨てが発生する。 四捨五入が要求されると、あふれる桁の最上位の値が5以上であると、結果の一意名の最下位の桁の絶対値に1が加えられる。
結果の一意名の整数部分の下位が PICTURE文字の "P" を用いて表わされているときには、 四捨五入や 切捨ては、実際に記憶場所を割り当てられている部分の右端の整数部に対して行われる。
(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。

10.1.5.2 ON SIZE ERROR指定

(ANS85)と NOT ON SIZE ERROR指定

小数点の位置をそろえたとき、算術演算の結果の絶対値が対応する結果の一意名がとる最大値を超えると、桁あふれ条件が発生する。 除数がゼロである除算が行われると、つねに桁あふれ条件が発生する。ON SIZE ERROR指定を指定しなかったときの ゼロによる除算の結果はどうなるかわからないので、注意する必要がある。一般に、 桁あふれ条件は最終結果に対してだけ適用される。ただし、MULTIPLY（乗算）文およびDIVIDE（除算）文の場合は、 中間結果に対しても桁あふれ条件が適用される。
(ANS85)指数の評価規則から外れると、つねに、算術演算は停止され、桁あふれ条件が発生する。
ROUNDED指定を書くと、四捨五入の後で桁あふれの検査が行われる。その結果桁あふれ条件が発生すると、ON SIZE ERROR指定を指定してあったか否かによって、下記のように処理が分かれる。

10.1.5.2.1 ON SIZE ERROR 指定を指定していなかった場合

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 0 ランタイムスイッチを使用するとよい。

10.1.5.2.2 ON SIZE ERROR 指定を指定してあった場合

桁あふれ条件が発生すると、桁あふれの影響を受ける結果の一意名の値は変更されない。桁あふれ条件の発生しなかった結果の一意名の値が、この演算処理が実行される間に他の結果の一意名に対して発生した桁あふれ条件の影響を受けることはない。 この演算処理が終了すると、ON SIZE ERROR指定内の無条件命令が実行される。
CORRESPONDING指定を伴うADD（加算）文およびSUBTRACT（減算）文に関しては、個々の演算処理において桁あふれ条件が発生しても、その時点ではON SIZE ERROR指定内の無条件命令は実行されない。個々の加算または減算がすべて終了した後に、ON SIZE ERROR指定内の無条件命令が実行される。
(ANS85)桁あふれ条件が発生すると、ON SIZE ERROR指定が指定されていても指定されていなくても、NOT ON SIZE ERROR指定は無視される。
(ANS85)ON SIZE ERROR指定とNOT ON SIZE ERROR指定の両方を指定したときに、実行された方の指定の中に制御を明示的に移す文が含まれていないと、必要があればその指定の実行が終わった時点で、その算術文の末尾に制御が暗黙的に移される。

10.1.5.2.3 NOT ON SIZE ERROR 指定

(ANS85)

算術演算文を実行したときに桁あふれ条件が発生しなかった場合にNOT ON SIZE ERROR指定が指定されていれば、その中に記述されている無条件命令が実行される。この場合、ON SIZE ERROR指定が指定されていても無視され、その中に記述されている無条件命令は実行されない。

10.1.5.3 CORRESPONDING指定

以下の説明で、d1とd2という記号を使用する。このd1とd2はそれぞれ別々の集団項目を指す一意名を表わす。d1とd2から1つずつ取った一組のデータ項目は、下記の条件が満たされるとき対応するという。

1. d1中のデータ項目およびd2中のデータ項目は、FILLER（無名項目）ではなくデータ名が同じであり、同じように修飾されている。ただし、d1とd2そのものは修飾語に使われていない。
2. d1中のデータ項目およびd2中のデータ項目は、少なくとも1つが基本項目である。CORRESPONDINGを伴うMOVE（転記）文に関しては、結果として各データ項目に適用される転記は転記規則に合致している。CORRESPONDINGを伴うADD文およびSUBTRACT文に関しては、両方のデータ項目とも基本数字データ項目である。
3. d1およびd2のデータ記述にレベル番号66，77，
   (MF)78，
   88，USAGE IS INDEX句，＞，
   (MF)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)，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)，USAGE IS OBJECT
   の各句が含まれるデータ項目の下位に属するデータ項目も同様に無視される。しかし、d1およびd2のデータ記述中にREDEFINES句またはOCCURS句が含まれていてもよい。あるいは、データ記述中にREDEFINES句またはOCCURS句が含まれるデータ項目の下位にd1およびd2が属していてもよい。
   (ANS85)d1もd2も、部分参照できない。
5. 上記の条件を満たす各データ項目の名前は、暗黙の修飾を使用した後では、一意でなければならない。

10.1.5.4 算術文

算術文には、ADD，COMPUTE，DIVIDE，MULTIPLY，SUBTRACTがある。これらの文には、下記の共通の特徴がある。

1. 作用対象のデータ記述は、同じである必要はない。必要に応じて、 データ形式の変換および 小数点の位置合わせが行われる。
2. 各作用対象の 最大の大きさは 18桁である。また、 作用対象の合成も 18桁を超えてはならない。ここで、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。(後述の*ADD*文，*DIVIDE*文,*MULTIPLY*文,*SUBTRACT*文の節を参照。)

10.1.5.5 作用対象の重なり

ある文の送出し側項目と受取り側項目とが記憶領域の一部を共有するとき
(ANS85)同じデータ記述項目によってまだ定義されていない場合、
その文を実行した結果はどうなるかわからない。
(MF)重なりの転記は、MOVE文が使用され、作用対象が部分参照も添え字も使用していない場合にだけ、翻訳時に検出される。コンパイラ指令 WARNING"3"を設定してある場合、前方への重なりの転記があると警告メッセージが出される。 コンパイラ指令 FLAG"方言"を設定してある場合、その他のタイプであればフラグメッセージが出される。ただし、 OSVS以外の"方言"とする。同じ記憶域を共有する項目を送受信する他の操作は、検出されない。
(MF)COBOL原始コードの移植性は、原始プログラム内でMOVE文の重なりが起きない時にだけ保証されるが、このCOBOLシステムはこのような文を認めていない。このような文の動作は、BYTE-MODE-MOVE指令の指定により変わる。

10.1.5.6 算術文における複数個の答

ADD, COMPUTE, DIVIDE, MULTIPLY, SUBTRACTの各文は複数個の答をもつことがある。これらの文は、下記のように書かれたものとして実行される。

1. 初期評価部分のすべてのデータ項目を演算処理し、その結果を一時的に記憶し、その一時結果を使用して必要なすべての演算を行い、受取り側項目に答を収める。
2. 中間結果を記憶する領域を利用して、一連の作用対象を順々に取り上げて演算を施し、最終的に1つの答を得る。この形の文は、列挙されている複数の作用対象を左から右に順々に取り上げて、別々の文で記述したものとみなされる。

   例１:
   ```
    ADD a, b, c TO c, d (c), e
   ```
   は、下記のように書いたのと等しい。
   ```
    ADD a, b, c GIVING temp
       ADD temp TO c
       ADD temp TO d (c)
       ADD temp TO e
   ```
   例２:
   ```
    MULTIPLY a(i) BY i, a(i)
   ```
   は、下記のように書いたのと等しい。
   ```
    MOVE a(i) to temp
       MULTIPLY temp by i
       MULTIPLY temp BY a(i)
   ```
   ここで、temp はCOBOLコンパイラによって使用される、 中間結果を一時的に記憶する場所である。

10.1.5.7 矛盾するデータ

字類条件（前述の*字類条件*の節を参照）の中で参照された場合は除いて、手続き部で参照されたデータ項目の内容が、そのデータ項目のPICTURE句

(ANS85)または関数定義
に指定されている字類と合わないと、そのデータ項目を参照した結果はどうなるかわからない。
(MF)数字項目を参照したところ、その項目に文字または無効なデータが含まれていた場合、結果はどうなるかわからない。このような条件は、実行時に検出されてエラーとされる。この動作は、F ランタイムスイッチの影響を受ける。
(MF)英字項目を参照したところ、その項目に英字でないデータが含まれていた場合、プログラムの実行は続けられるが、結果はどうなるかわからない。

10.1.5.8 符号付き受取り側項目

算術文またはMOVE文中の受取り側項目が、符号付きの数字または数字編集項目である場合、受取り側項目に符号が転記される。この符号の転記は、数字データの絶対値が切り捨てられたか否かに関係なく行われる。したがって、数値はゼロであるが、符号が負ということが起こり得る。

10.1.6 ファイル入出力の概要

10.1.6.1 ファイル位置指示子

ファイル位置指示子（file position indicator）は、ファイルに一連の入出力操作を施している際に、次に呼び出されるレコードを指す働きをする。ファイル位置指示子の設定に影響する文は、CLOSE, OPEN, START, READだけである。出力モードまたは拡張モードで開かれたファイルに対しては、ファイル位置指示子は影響を及ぼさない。

10.1.6.2 入出力状態

ファイル管理記述項にFILE STATUS STATUS句を指定すると、ファイル入出力操作の結果を示す2文字のデータ項目がとられる。OPEN, CLOSE, READ, WRITE, DELETE, UNLOCK, STARTの各文を実行すると、その結果を示す値がこのデータ項目に設定される。実行する条件に該当するUSE手続きがある場合、それが実行される前にこの値が設定される。
以下に記述するファイル状態キーは、ANSI標準に全面的に準拠したものであるとともに、このCOBOLシステムで使用できる拡張機能を、何も使用していないファイルに関するものである。
(MF)たとえば、拡張機能を使用してファイルをLINE SEQUENTIALと指定すると、返される状態キーに影響が出る。詳しいことは関連する動詞の箇所に説明してある。

10.1.6.2.1 状態キー1

FILE STATUSデータ項目の左端の文字を、状態キー1という。どのような編成のものでも、ファイルの入出力操作が終了すると、このデータ項目に下記の条件のどれかを示す値が設定される。

"0"　　- 正常終了（successful completion）
"1"　　- ファイル終了（at end）
"2"　　- 無効キー（invalid key）
"3"　　- 永続誤り（permanent error）
(ANS85)"4"　- 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"　(ANS85)論理誤り： 入出力文の実行が不成功に終わった。その理由は2通りある。ファイルに対する入出力操作の順序が不適切である。 または、利用者によって設定されている制限を超えた処理が行われた。
"9"　ランタイム・システム・エラーメッセージ： ランタイム・システムによって定義されているエラーメッセージ条件が発生したために、入出力文の実行が不成功に終わった。状態キー1または状態キー1と状態キー2の組合せによって予め定義されている条件に該当しない条件が発生した場合にだけ、この状態値が使用される。

10.1.6.2.2 状態キー2

FILE STATUSデータ項目の右端の文字を、状態キー2という。状態キー2は入出力操作の結果の、さらに詳細を示すために使用される。
状態キー1と状態キー2の組合せによって、入出力操作の結果が下記のように細かく定義される。

正常終了
状態キー1の値が"0" で、 入出力操作が正常に終了したことが示されている場合、 状態キー2にはその原因を示す下記のどれかの値が設定される。

"0"　すべてのファイル）それ以上詳しい情報はない。
"2"　索引ファイルだけ）下記の2通りの可能性を示す。
・READ文で読み込んだレコードの参照キーの値が、その索引における次のレコードの参照キーと等しい。
・WRITE文またはREWRITE文によって書かれたレコードの副レコードキーの値が、最低1つ既にファイルに存在している。ただし、その副レコードキーには重複が許されている。
"4"　(ANS85)（すべてのファイル）処理したレコードの長さが、そのファイルのファイル固有属性に従っていない。
"5"　(ANS85)（すべてのファイル）OPEN文が実行されたとき、参照した不定ファイルが存在しない。
"7"　(ANS85)（レコード順ファイルだけ）NO REWIND, REEL/UNIT, FOR REWINDのどれかの指定を伴うCLOSE文、またはNOREWIND指定を伴うOPEN文が実行されたが、対象となったファイルはリール/ユニット媒体ではない。

ファイル終了条件による不成功

状態キー1の値が"1" で、 ファイル終了条件（at end condition）が発生したことが示されている場合、状態キー2にはその原因を示す下記のどれかの値が設定される。
"0"　すべてのファイル）次の論理レコードが存在しないことを示す。この条件が発生する場合は下記の2通りある。
・ファイルの末尾に到達した。
・不定入力ファイル（optional input file）に対して、順呼出しREAD文が初めて実行されたが、対象とするファイルが存在しない。
"4"　(ANS85)（相対ファイルだけ）使用された相対レコード番号の桁数が、ファイル用に定義されている相対キーデータ項目の桁数よりも大きい。

無効キー条件による不成功

状態キー1の値が "2" で、無効キー条件（invalid key condition）が発生したことが示されている場合、 状態キー2にはその原因を示す下記のどれかの値が設定される。

"1"　索引ファイルを順呼び出しした場合）順序誤り（sequence error）があったことを示す。その原因は、次の2通りある。連続するキーの値が昇順になっていなかった。(後述の*WRITE*文の節を参照。)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。

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, 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)（すべてのファイル）OPEN文の対象となったファイルには、指定されたモードが適用できないことを示す。 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。
Ⓜ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

Ⓜ

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は**エラーメッセージ**を参照。

10.1.6.2.3 状態キー1と2の有効な組み合わせ

以下の表で、 "S" はレコード順ファイル、
(MF)"L " は行順ファイル、
"R" は相対ファイル、
"I" は索引ファイルを示す。
表 10-4中の状態キー1の行と状態キー2の列が交差する部分に文字が記されている場合に、そのファイル編成に関してその状態キー1と状態キー2の組合せが有効である。

表 10-4: 状態キー1と2の有効な組み合わせ

状態キー 1		状態キー 2									
		0	1	2	3	4	5	6	7	8	9
正常終了	0	SRIL		I		SRIL	SRIL		S		
ファイル終了	1	SRIL				R					
無効キー	2		I	RI	RI	RI					
永続誤り	3	SRIL				SL	SRIL		SRIL	SRIL	SRI
論理誤り	4		SRIL	SRIL	SRIL	SRIL		SRIL	SRIL	SRIL	SRIL
作成者定義	9	ランタイムシステムのエラーメッセージ (SRIL)									

10.1.6.3 ファイル終了条件

READ文を実行した結果、ファイル終了条件が発生することがある。その原因の詳細については、後述の*READ文*の節を参照。

10.1.6.4 無効キー条件

START文、READ文、WRITE文、REWRITE文、DELETE文を実行した結果、無効キー条件が発生することがある。その原因の詳細については、後述の*START文*,*READ文*, *WRITE文*, *REWRITE文*,*DELETE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 宣言部分を使用するかして、別途誤りを捕られなければならない。

10.1.6.5 マルチユーザーシステム上でのファイルの共有

ランタイムシステム（RTS）では、このCOBOLシステムのマルチユーザー機能をサポートしている。このマルチユーザー機能によって、マルチユーザー環境下で利用者がファイルを共有すること、あるプログラムがデータを更新する間は他のプログラムが該当するファイル全体またはそのレコードをアクセスできないようにすることが可能になる。
シングルユーザー環境下では、マルチユーザー用の構文は実行時に効力を発揮しない。しかし、シングルユーザーとマルチユーザーの両方の環境で使用できるように、プログラムを作成しておくことができる。
ファイルはアクティブか非アクティブのどちらかの状態をとる。アクティブ・ファイルとはいくつかの実行単位に対して開かれているものである。非アクティブ・ファイルとはどの実行単位に対しても開かれていないものである。
アクティブ・ファイルには、排他モード（exclusive mode）と共有モード（sharable mode）の2つのモードがある。

10.1.6.5.1 排他モード

MF XOPEN

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。

10.1.6.5.2 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。

10.1.6.5.3 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。

10.1.6.5.4 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。

10.1.6.5.5 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。

10.1.6.5.6 レコードのロック

ファイルへのアクセスを共有する各実行単位は、そのファイル中の単一のレコードまたは複数のレコードをロックすることができる。ただし、1つの実行単位が同じファイルに対して、単一レコードのロックと複数レコードのロックの両方を選択することはできない。

10.1.6.5.7 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、マニュアルと自動の2通りがある。

10.1.6.5.8 マニュアルのレコードロック

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。

10.1.6.5.9 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。

10.1.6.5.10 単一レコードのロック解除

ロック解除はロックした実行単位によって、下記のどれかの方法によって行われる。

- 該当するファイル中の任意のレコードを、任意のファイル操作によって呼び出す（ただし、相対ファイルおよび索引ファイルに対するSTARTを除く）
- 該当するファイルに対して、UNLOCK文を実行する
- COMMIT文を実行する
- ROLLBACK文を実行する
- 該当するファイルを閉じる

10.1.6.5.11 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、マニュアルと自動の2通りがある。

10.1.6.5.12 マニュアルのレコードロック

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。

10.1.6.5.13 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。

10.1.6.5.14 複数レコードのロック解除

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

表 10-5，10-6，10-7 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、後述の*SELECT*文の節を参照。)
(XOPEN)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（*ファイル管理記述項*の節を参照）を含めるかして、ロックの型を選択できる。

第11章: 手続き部 - 組み込み関数

(ANS85)
組み込み関数を使用すると、参照したデータ項目の値が、実行用プログラムを実行中に自動的に算出される。

組み込み関数を使用した場合、浮動小数点モジュールが必要となる。ランタイムシステム支援モジュールの詳細については、構築とリンクに関するCOBOLシステムのマニュアルを参照。

11.1 関数名

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。 (この章で後述する*関数の定義* を参照。)

11.2 関数定義と戻り値

関数の型に応じて、下記のことが示される。

- 英数字関数
  (MF)および各国語型
  の場合は、戻り値の大きさ。
- 数字関数および整数関数の場合は、戻り値の符号と整数であるか否か。r
- その他、関数によっては、戻り値。

11.3 関数一意名

関数一意名は、COBOL原始プログラムの手続き部の中で使用する、関数の名前である。 (*COBOLの概念* の章の *関数一意名*の節を参照。)

11.4 関数からの戻り値

関数からの戻り値（関数値）は、データ値とみなされる。関数を使用すると、実行時にこのデータ値が決定される。そのためには、プログラムから関数に、基となるデータ値をパラメータとして引き渡す必要がある。このパラメータを 引数という。関数によっては、引数の取る値の範囲などに制約があるものがある。関数の実行時に引数の値がその関数の制約から外れる場合、その関数からの戻り値は保証されない。

11.5 引数

引数は、関数値を求めるために使用する値である。引数は、関数一意名中に指定する。指定する引数の型には、一意名と算術式と定数がある。各関数の定義の説明で、必要な引数の数を具体的に示してある。関数によって、引数は必要ないもの、1つだけ必要とするもの、複数個必要とするものがある。また、中には、指定できる引数の数が変えられる関数もある。 関数一意名中に引数を指定する順序に基づいて、それぞれの引数が解釈され、関数値が決定される。指定する引数が、特定の字類またはその部分集合に属することが要求される場合がある。引数の型を下に示す。

1. 数　字：　算術式を指定する。演算符号を含む算術式の値が、関数値を求めるために使用される。
2. 英　字：　字類が英字である基本データ項目、または英字だけを含む文字定数を指定する。引数の大きさが、関数値の決定に使用されることもある。
3. 英数字：　字類が英字か英数字である基本データ項目、または文字定数を指定する。引数の大きさが、関数値の決定に使用されることもある。
4. (MF)各国語型。字類が各国語型であるデータ項目、または各国語型文字を指定する。引数の大きさが、関数値の決定に使用されることもある。
5. 整　数：　結果が必ず整数になる算術式を指定する。演算符号を含む算術式の値が、関数値を求めるために使用される。

意味のある関数値を求めるために、引数に指定できる値に制約が設けられることがある。関数の実行時に引数の値がその関数の制約から外れる場合、その関数からの戻り値は保証されない。

関数定義において引数を任意の回数繰り返し指定することが認められている場合、表を使用することができる。この場合、使用する表を識別するデータ名と、必要に応じて修飾語を指定し、その直後に添字を続ける。添字の1つまたはいくつかに語ALLを指定できる。

添字にALLを指定すると、その添字の位置に対応する表要素をすべて指定したのと同様の効果が得られる。この暗黙の指定は、表の要素を左から右に順に指定することを意味する。つまり、指定の対象となる最初（左端）の表要素は、添字に指定した語ALLのそれぞれを、1で置き換えた一意名によって表わされるものとなる。指定の対象となる次の表要素は、右端のALLに対応する添字の値を1繰り上げた一意名によって表わされるものとなる。

それ以降、暗黙の指定の対象となる表要素は、右端のALLに対応する添字の値を順々に1繰り上げていった一意名によって表わされるものとなる。そして、そのALLに対応する添字の値がその最大値に達したところで、その添字の繰り上げ処理は終わりとなる。ALLを指定した添字が複数ある場合は、次に、ALLを指定した右端の添字のすぐ左のALLを指定した添字が1繰り上げられ、ALLを指定した右端の添字の値は1に設定し直される。右端の添字の値を順々に1繰り上げていくことが、再度繰り返される。そして、右端の添字の値が最大値に達するたびに、その左側の添字の値が1繰り上げられる。左側の添字の値が最大値に達したところで、その添字の繰り上げ処理は終わりとなる。その左側にもALLを指定した添字がまだあれば、同様のプロセスが繰り返され、ALLを指定した左端の添字の値が最大値に達するまで、続けられる。

この処理の対象となる添字にOCCURS DEPENDING ON句が指定してある場合は、該当する添字の値の範囲は、OCCURS DEPENDING ON句の右辺の項目に基づいて決められる。ALLを指定した添字が評価された結果、少なくとも引数が1つ存在しなければならない。引数がないと、戻り値は保証されない。

例：

```
01 Test-Fields.
    10 OT-Elem                  PIC 9(02).
    10 Arr.
        15 Ind occurs 5 times        PIC 9(02).
    compute OT-Elem = function sum (IND(ALL)).
```

下記のコーディングと同じ意味になる。

```
    compute OT-Elem = function sum (IND(1), IND(2), IND(3),
                                    IND(4), IND(5)).
```

ALL を指定した添字は、以下の例のように、多くの要素を持つ表でも使用できる。

```
01 Test-Group.
    03 OT-Elem               pic 9(15) binary.
    03 table-length          pic s9(9) binary.
    03 array.
        05 Ind pic 9(2) occurs 1 to 200 times depending on
                            table length
     ...
    move 100 to table-length.
    compute OT-Elem = function sum (Ind(ALL))
```

これらの文は、表の最初の100個の要素を合計する。table-lengthに指定された数が、関数の参照時に、加える要素の数として決定される。

11.6 関数の型

データ項目関数は基本データ項目であり、英数字か
(MF)各国語型文字か

数字か整数の値を返す。 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だけ、
      (MF)またはMOVE文の送出し側としてだけ、 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だけ、
      (MF)またはMOVE文の送出し側としてだけ、使用できる。
    - 整数の作用対象が必要とされる箇所で、それに符号が付いてもよい場合、整数関数を使用することができる。
4. (MF)各国語型関数： この型の関数は、字類および項類が各国語型に属する。このデータ項目中の文字数は、関数定義中に示される。各国語型関数の用途は、暗黙的にNATIONALとされる。

11.7 日付変換関数

日付変換関数では、グレゴリオ歴を使用している。1601年1月1日を開始日とする。この日は月曜日であるので、開始日からの通算日と曜日の関係がとらえやすくなっている。

たとえば、下記の文を実行すると、指定した日の曜日を表わす整数が返される。

```
compute DoW = function rem (function integer-of-date (date-field) , 7)
```

値が0の場合は日曜日、1の場合は月曜日といった具合いになる。

11.8 三角関数

三角法は三角形の辺と角の間の関係を取り扱う。角を測る単位には度、ラジアン、グラードがある。コンピュータでは三角関数の値は通常、連続的近似によって算出される。したがって、SIN，COS，TANの各関数の引数またはASIN，ACOS，ATANの各関数から返される値として使用するときは、角度をラジアンで表す。

１ラジアンは長さが半径に等しい円弧である。半径（ｒ）と円周（ｃ）の関係は式C = 2 * π * *r*で表される。円の一周は360度である。よって、360 = 2 * π * *r*となる。このことから、下記の変換率が得られる。

１ラジアン = 180/π = 57.295780度
１度 = π/180 = 0.01745329ラジアン

その他に、角度を表すのに、グラード（直角の百分の１）という単位が用いられることがある。その変換率は下記のとおりである。

１ラジアン = 200/π = 63.661977グラード
１グラード = π/200 = 0.01570796ラジアン

11.9 関数の定義

使用できる関数を要約して、下記に示す。

The " 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

関数名	引数	関数型	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大小順序中の、位置Int1にある文字

関数名	引数	関数型	戻り値
(MF)CHAR-NATIONAL	Int1	Nat	各国語型プログラムの大小順序中の、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-YYMMDD	Int1	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	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
(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 か Anum1 か (MF)Nat1 か Num1	Int	文字番号の引数の長さ
(MF)LENGTH-AN	Alph1 か Anum1 か Int1 か Nat1 か Num1	Int	英数字文字番号の引数の長さ positions
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 か Anum1 か (MF)Nat1	引数による	引数中のすべての文字を小文字にする

関数名	引数	関数型	戻り値
MAX	Alph1 ... か Anum1 ... か Int1 ... か (MF)Nat1 ... か 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 ... か Anum1 ... か Int1 ... か (MF)Nat1 ... か 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）
(MF)NATIONAL-OF	Anum1, (MF)Nat1	Nat	引数Anum1のUSAGE NATIONAL型の表現
NUMVAL-C	Anum1 かAnum2 か (MF)Nat1, 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 か Anum1 か (MF)Nat1	Int	文字の大小順序における、引数の順序番号
ORD-MAX	Alph1 ... か Anum1 ... か (MF)Nat1 ... か Num1	Int	最大の引数の順序番号
ORD-MIN	Alph1 ... か Anum1 ... か (MF)Nat1 ... か Num1	Int	最小の引数の順序番号
(MF)PI	なし	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 ... か 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 か Anum1 か (MF)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

関数名	引数	関数型	戻り値
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 ... か 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 か Anum1 か (MF)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

11.9.1 ABS 関数

(MF)

ABS関数は引数の絶対値を返す。

この関数の型は引数の型に応じて下記のようになる。

引数の型	関数の型
整数	整数
数字	数字

一般形式

```
FUNCTION ABS(引数-1)
```

引数

1. 引数-1は字類が数字でなければならない。

戻り値

1. 戻り値は下記のようになる。
    a. 引数-1の値がゼロまたは正である場合、（引数-1）
    b. 引数-1の値が負である場合、-（引数-1）

11.9.2 ACOS 関数

ACOS関数は、引数-1に指定された角または弧（単位はラジアン）のアークコサイン（逆余弦）の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION ACOS(引数-1)
```

引数

2. 引数-1の字類は、数字とする。
3. 引数-1の値は、-1以上+1以下とする。

戻り値

1. 戻り値は、引数-1のアークコサインの近似値である。その値は0以上π以下である。

   注: 度およびグラードからラジアンへの変換係数をこの章の「三角関数」の節に示してある。

2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.3 ANNUITY 関数

ANNUITY（即時年金）関数は、引数-2に指定された数の期間後に、最初の一括掛金に対して、指定された期間にわたって各期末に支払われる、年金の比率の近似値を返す。利子率は引数-1に指定する。利息は各期末に、支払の前に算出される。この関数の型は数字である。

一般形式

```
FUNCTION ANNUITY(引数-1 引数-2)
```

引数

1. 引数-1の字類は、数字とする。
2. 引数-1の値は、0以上とする。
3. 引数-2は、正の整数とする。

戻り値

1. 引数-1の値が0である場合、この関数の戻り値は下記の式の近似値である。
   1 / 引数-2
2. 引数-1の値が0でない場合、この関数の戻り値は下記の式の近似値である。

引数-1 / (1 - (1 + 引数-1) ** (- 引数-2))

3. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.4 ASIN 関数

ASIN関数は引数-1に指定された角または弧（単位はラジアン）のアークサイン（逆正弦）の近似値を返す。この関数の型は数字である。

一般形式

FUNCTION ASIN（引数-1）

引数

1. 引数-1の字類は、数字とする。
2. 引数-1の値は、-1以上+1以下とする。

戻り値

1. 戻り値は、引数-1のアークサインの近似値である。その値は、-π/2以上+π/2以下である。

   注: 度およびグラードからラジアンへの変換係数をこの章の「三角関数」の節に示してある。

2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.5 ATAN 関数

ATAN関数は、引数-1に指定された角または弧（単位はラジアン）のアークタンジェント（逆正接）の近似値を返す。この関数の型は数字である。

一般形式

FUNCTION ATAN（引数-1）

引数

1. 引数-1の字類は、数字とする。

戻り値

1. 戻り値は、引数-1のアークタンジェントの近似値であり、-π/2より大きく+π/2より小さい。

注: 度およびグラードからラジアンへの変換係数をこの章の「三角関数」の節に示してある。

2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.6 CHAR 関数

CHAR関数は、プログラムの文字の大小順序の中から、引数-1に指定された順序位置にある文字を返す。この関数の型は英数字である。

一般形式

```
FUNCTION CHAR（引数-1）
```

引数

1. 引数-1は、整数とする。
2. 引数-1の値は、0よりも大きく、文字の大小順序の文字位置の数以下とする。

戻り値

1. プログラムの文字の大小順序中の同じ文字位置に、2つ以上の文字が割り当てられている場合、ALPHABET句中でその文字位置に指定されている最初の文字が、この関数の戻り値とされる。
2. ALPHABET句によって現在のプログラムの文字の大小順序が指定されていない場合、固有の大小順序が使用される。

11.9.7 CHAR-NATIONAL 関数

(MF)

CHAR-NATIONAL（国別文字）関数は国別の大小順に並べた一連の文字の中から、順位が引数の値に等しい文字を返す。

関数の型は国別である。

一般形式

```
FUNCTION CHAR-NATIONAL（引数-1）
```

引数

1. 引数-1は整数でなければならない。

2. 引数-1の値は、ゼロよりも大きく、国別の文字の大小順序の最大値以下でなければならない。

戻り値

1. 戻り値は、国別の大小順に並べた一連の文字の中の、順位が引数の値に等しい文字である。
2. 現行の国別の文字の大小順序に同じ順位の文字が複数ある場合には、先に出てくる方の文字が戻り値となる。

11.9.8 COS 関数

COS関数は、引数-1に指定された角または弧（単位はラジアン）のコサイン（余弦）の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION COS（引数-1）
```

引数

1. 引数-1の字類は、数字とする。

   注: 度およびグラードからラジアンへの変換係数をこの章の「三角関数」の節に示してある。

戻り値

1. 戻り値は、引数-1のコサインの近似値であり、-1以上+1以下である。
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.9 CURRENT-DATE 関数

CURRENT-DATE関数は、現在の日時に関する21文字の英数字の値を返す。その細目は、暦日、時刻、関数が実行されたシステムが稼働している地域の時刻とグリニッジ標準時刻との時差である。この関数の型は英数字である。

一般形式

```
FUNCTION CURRENT-DATE
```

戻り値

1. 戻り値の文字列の文字位置（左から右に数える）ごとの内容を下に示す。

文字位置	内　　容
1-4	グレゴリオ歴の暦年を表わす4桁の数字 calendar.
5-6	年内の月を表わす2桁の数字。値は01から12まで
7-8	月内の日を表わす2桁の数字。値は01から31まで
9-10	真夜中から経過した時間を表わす2桁の数字。値は00から23まで
11-12	時刻の分を表わす2桁の数字。値は00から59まで
13-14	時刻の秒を表わす2桁の数字。値は00から59まで
15-16	1秒の100分の1を表わす2桁の数字。値は00から99まで 1秒よりも細かく時間を測定する機能のないシステムでは、この値は00となる。
17	グリニッジ標準時間との時差の遅早を表わす。値は"-"、 " +"、"0"のいずれかである。 この値が"-" の時は、この前の文字位置に示された時間はグリニッジ標準時間よりも遅れていることを示す。この値が "+" の時は、この前の文字位置に示された時間はグリニッジ標準時間と等しいかそれよりも進んでいることを示す。この値が"0" の時は、この関数を実行したシステムにはグリニッジ標準時間との時差を測る機能がないことを示す。 グリニッジ標準時間との時差を測る機能がないシステムでは、文字位置17から21には値00000が返される。
18-19	文字位置17の値が"-" の時は、システムで使用されている時刻がグリニッジ標準時間よりも遅れている時間数が、2桁の数字でここに示される。その値は00から12までである。文字位置17の値が"+" の時は、システムで使用されている時刻がグリニッジ標準時間よりも進んでいる時間数が、2桁の数字でここに示される。その値は00から13までである。文字位置17の値が "0" の時は、ここには00が返される。
20-21	システムで使用されている時刻とグリニッジ標準時間との時差の、1時間に満たない分数を表わす。その値は00から59までである。時差の遅早は文字位置17の値が"+" であるか"-" であるかによって示される。文字位置17の値が"0" の時は、ここには00が返される。

11.9.10 DATE-OF-INTEGER 関数

DATE-OF-INTEGER関数は、グレゴリオ暦の日付を整数形式から標準日付形式（YYYYMMDD）に変換する。この関数の型は整数である。

一般形式

```
FUNCTION DATE-OF-INTEGER（引数-1）
```

引数

1. 引数-1は、グレゴリオ暦で1601年1月1日以降の通算日数を表わす正の整数とする。

戻り値

1. この関数の戻り値は、引数-1に指定した整数に相当するISO標準日付を表わす。
2. 戻り値は、YYYYMMDDの形式をしている。YYYYはグレゴリオ暦の暦年を表わし、MMとDDはそれぞれ月と日を表わす。

11.9.11 DATE-TO-YYYYMMDD 関数

DATE-TO-YYYYMMDD 関数は引数-1をYYnnnの形からYYYYnnnの形へ変換する。引数-2は実行時に年に追加された時、100年間隔の終了年を定義するか、または引数-1が起こるところへのウィンドウの移動を定義する。この関数の型は整数である。

一般形式

```
FUNCTION DATE-TO-YYYYMMDD ( 引数-1 [引数-2] )
```

引数

1. 引数-1 は 1,000,000 より小さい、正の整数とする。
2. 引数-2 は整数とする。
3. 引数-2 が省略された場合、この関数は 50 が指定されたものとして評価される。
4. 実行時の年と引数-2 の値の合計は、10,000 より小さく、1699より大きいものとする。

戻り値

1. 同等の算術式は次のとおりである。
   (FUNCTION YEAR-TO-YYYY (YY, 引数-2) * 10000 + nnnn) ここで:
   YY = FUNCTION INTEGER (引数-1/10000)
   nnnn = FUNCTION MOD (引数-1, 10000)
   INTEGER および MOD 関数の引数-1 と YEAR-TO-YYYY 関数の引数-2 は、DAY-TO-YYYYMMDD 関数自身の引数-1 と引数-2 と同じである。

注:

1. 2002年では、FUNCTION DATE-TO-YYYYMMDD (851003, 10) の戻り値は 19851003 である。1994年では、FUNCTION DATE-TO-YYYYMMDD (961002, (-10)) の戻り値は 18981002 である。
2. この関数はウィンドウ移動アルゴリズムを持っている。決まったウィンドウを指定する方法については、YEAR-TO-YYYY 関数の注意の項を参照のこと。

11.9.12 DAY-OF-INTEGER 関数

DAY-OF-INTEGER関数は、グレゴリオ暦の日付を整数形式からジュリアン日付（ユリウス暦）形式（YYYYDDD）に変換する。この関数の型は整数である。

一般形式

```
FUNCTION DAY-OF-INTEGER（引数-1）
```

引数

1. 引数-1は、グレゴリオ暦で1601年1月1日以降の通算日数を表わす正の整数とする。

戻り値

1. この関数の戻り値は、引数-1に指定した整数に相当するジュリアン日付を表わす。
2. 戻り値は、YYYYDDDの形式をしている。YYYYはグレゴリオ暦の暦年を表わし、DDDは年内の通算日を表わす。

11.9.13 DAY-TO-YYYYDDD 関数

DAY-TO-YYYYDDD 関数は、引数-1 を YYnnn の形から YYYYnnn の形へ変換する。引数-2 は実行時に年に追加された時、100年間隔の終了年を定義するか、または引数-1が起こるところへのウィンドウの移動を定義する。 この関数の型は整数である。

一般形式

```
FUNCTION DAY-TO-YYYYDDD ( 引数-1 [引数-2] )
```

引数

1. 引数-1 は 100,000 より小さい、正の整数とする。
2. 引数-2 は整数とする。
3. 引数-2 が省略された場合、この関数は 50 が指定されたものとして評価される。
4. 実行時の年と引数-2 の値の合計は、10,000 より小さく、1699より大きいものとする。

戻り値

1. 同等の算術式は次のとおりである。
   (FUNCTION YEAR-TO-YYYY (YY, 引数-2) * 1000 + nnn)
   ここで:
   YY = FUNCTION INTEGER (引数-1/1000)
   nnn = FUNCTION MOD (引数-1, 1000)
   INTEGER および MOD 関数の引数-1 と YEAR-TO-YYYY 関数の引数-2 は、DAY-TO-YYYYDDD 関数自身の引数-1 と引数-2 と同じである。

注:

1. 2002年では、FUNCTION DAY-TO-YYYYDDD (10004,20) の戻り値は20101004 である。1994年では、FUNCTION DAY-TO-YYYYDDD (95005, (-10)) の戻り値は 1995005 である。

2. この関数は、ウィンドウ移動アルゴリズムを持っている。決まったウィンドウを指定する方法については、YEAR-TO-YYYY関数の注意の項を参照のこと。

11.9.14 DISPLAY-OF 関数

(MF)

DISPLAY-OF関数は引数中の各国語型文字を英数字型の文字表現（外部表現）に変換する。

この関数の型は英数字である。

一般形式

```
FUNCTION DISPLAY-OF (引数-1 [引数-2])
```

引数

1. 引数-1の字類は各国語とする。
2. 引数-2の字類は英字または英数字とし1文字位置の長さを占める。引数-2に指定された文字は、変換時に対応する英数字表現が存在しないような各国語文字に対する置き換え文字として使われる。

戻り値

1. 引数-1中の一つ一つの各国語文字を対応する英数字表現に変換し、戻り値として返す。各国語文字と英数字との対応は稼動する環境が定義する。
2. 引数-2が指定されると、対応する英数字表現が存在しないような各国語文字に対してこの文字が戻り値に返される。
3. 引数-2を指定しない場合、対応する英数字表現が存在しないような各国語文字に対しては、各国語型の外部表現に変換した文字が戻り値に返される。その際に必要な最小限の制御機能が付加される。
4. 戻り値の長さは、変換された結果を保持するのに必要な、USAGE DISPLAYの文字位置の個数となる。この長さは引数の長さと各国語文字集合の性質によって異なる。

11.9.15 E 関数

(MF)

E関数は、自然対数の底である、ｅの近似値を返す。
この関数の型は数字である。

一般形式

```
FUNCTION E
```

戻り値

1. 戻り値は下記のとおり。
   2.718281828459045235

11.9.16 EXP 関数

(MF)

EXP関数はｅを引数の値分べき乗した値の近似値を返す。

この関数の型は数値である。

一般形式

```
FUNCTION EXP（引数-1）
```

引数

1. 引数-1は字類が数字でなければならない。

戻り値

1. 戻り値は下記のとおり。
   FUNCTION E ** 引数-1

11.9.17 EXP10 関数

(MF)

EXP10関数は10を引数の値分べき乗した値の近似値を返す。

この関数の型は数字である。

一般形式

```
FUNCTION EXP10（引数-1）
```

引数

1. 引数-1は字類が数字でなければならない。

戻り値

1. 戻り値は下記のとおり。
   10 ** 引数-1

11.9.18 FACTORIAL 関数

FACTORIAL関数は、引数-1の階乗である整数を返す。 この関数の型は整数である。

一般形式

FUNCTION FACTORIAL（引数-1）

引数

1. 引数-1は、0以上の整数とする。

戻り値

1. 引数-1の値が0の場合は、値1が返される。
2. 引数-1の値が正の場合は、その階乗値が返される。
3. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.19 FRACTION-PART 関数

(MF)

FRACTION-PART関数は引数の端数部分の数値を返す。

この関数の型は数字である。

一般形式

FUNCTION FRACTION-PART（引数-1）

引数

1. 引数-1は字類が数字でなければならない。

戻り値

1. 戻り値は下記のとおり。
   （引数-1 - FUNCTION INTEGER-PART（引数-1））
   ただし、INTEGER-PART関数の引数はFRACTION-PART関数の引数と同じとする。

注： 引数-1の値が+1.5であると、+0.5が返される。引数-1の値が-1.5であると、-0.5が返される。

11.9.20 INTEGER 関数

INTEGER関数は、引数の値以下の整数の最大値を返す。 この関数の型は整数である。

一般形式

```
FUNCTION INTEGER（引数-1）
```

引数

1. 引数-1の字類は、数字とする。

戻り値

1. この関数の戻り値は、引数-1以下の値で最大の整数である。たとえば、引数の値が-1.5であると-2が返され、引数-1の値が+1.5であると+1が返される。

11.9.21 INTEGER-OF-DATE 関数

INTEGER-OF-DATE関数は、グレゴリオ暦の日付を標準日付形式（YYYYMMDD）から整数形式に変換する。この関数の型は整数である。

一般形式

```
FUNCTION INTEGER-OF-DATE（引数-1）
```

引数

1. 引数-1は、YYYYMMDD形式の整数とする。その値は、下記の式によって算出される。
   (YYYY * 10,000) + (MM * 100) + DD
   a. YYYYはグレゴリオ暦の暦年を表わす。これは1601以上の整数とする。
   b. MMは月を表わす。これは1から12までの整数とする。
   c. DDは日を表わす。これは1から31までの整数とする。ただし、年月の組合わせに対応した有効な日数にすること。

戻り値

1. この関数の戻り値は、引数-1がグレゴリオ暦で1601年1月1日以降、通算何日目にあたるかを示す整数である。

11.9.22 INTEGER-OF-DAY 関数

INTEGER-OF-DAY関数は、グレゴリオ暦の日付をジュリアン日付（ユリウス暦）形式（YYYYDDD）から整数形式に変換する。この関数の型は整数である。

一般形式

```
FUNCTION INTEGER-OF-DAY（引数-1）
```

引数

1. 引数-1は、YYYYDDD形式の整数とする。その値は下記の式によって算出される。
   (YYYY * 1000) + DDD
   a. YYYYはグレゴリオ暦の暦年を表わす。これは1601以上の整数とする。
   b. DDDは日を表わす。これは1から366までの整数とする。ただし、対応する年に応じた有効な日数にすること。

戻り値

1. この関数の戻り値は、引数-1がグレゴリオ暦で1601年1月1日以降、通算何日目にあたるかを示す整数である。

11.9.23 INTEGER-PART 関数

INTEGER-PART関数は、引数-1の整数部分の値を返す。この関数の型は整数である。

一般形式

```
FUNCTION INTEGER-PART（引数-1）
```

引数

1. 引数-1の字類は、数字とする。

戻り値

1. 引数-1の値が0の場合は、0が返される。
2. 引数-1の値が正の場合は、引数-1の値以下の最大の整数が返される。たとえば、引数-1の値が+1.5であれば、+1が返される。
3. 引数-1の値が負の場合は、引数-1の値以上の最小の整数が返される。たとえば、引数-1の値が-1.5であれば、-1が返される。

11.9.24 LENGTH 関数

LENGTH関数は、英数字文字位置にある引数の長さと同じ整数を返す。

(MF)または、引数の字類によっては各国語型文字位置にある引数の長さ。

この関数の型は整数である。

一般形式

FUNCTION LENGTH（引数-1）

引数

1. 引数-1は、文字定数または任意の字類または項類のデータ項目であってよい。
2. 引数-1またはその下位に属するデータ項目のいずれかに、OCCURS句のDEPENDING指定がなされている場合、DEPENDING句に指定されたデータ項目によって参照されるデータ項目の内容は、この関数が実行される時点で使用される。

戻り値

1. (MF)引数-1の字類が各国語型、または各国語型定数の基本データ項目の場合、各国語型文字位置にある引数-1の長さと同じ整数が返される。
2. あるいは、引数-1が英数字定数、基本データ項目、可変反復データ項目を含まない集団項目のいずれかである場合、英数字文字位置にある引数-1の長さと同じ整数を返す。
3. 引数-1が可変反復データ項目を含む集団項目である場合、その可変反復データ項目を定義しているOCCURS句のDEPENDING指定の対象となっているデータ項目を評価した結果の文字数を表わす、整数が返される。この評価は、送出し側データ項目に関するOCCURS句の規則に従って行われる。*OCCURS 句* の節、および*USAGE 句* の節を参照。
4. 暗黙のFILLERがあれば、その文字数も戻り値に含まれる。

11.9.25 LENGTH-AN 関数

(MF)

LENGTH-AN関数は、引数を英数字の文字位置で数えた、長さを表す整数を返す。

この関数の型は整数である。

一般形式

FUNCTION LENGTH-AN（引数-1）

引数

1. 引数-1は任意の字類または項類の定数またはデータ項目であってよい。
2. 引数-1またはその下位の任意のデータ項目にOCCURS句のDEPENDING指定が記述されている場合、LENGTH-AN関数が評価されるときに、DEPENDING指定中のデータ名によって参照されるデータ項目の内容が使用される。

戻り値

1. 引数-1が基本データ項目である場合、戻り値は引数-1を英数字の文字位置で数えた長さを表す整数である。
2. 引数-1が集団データ項目である場合、
    a. 引数-1またはその下位の任意のデータ項目にOCCURS句のDEPENDING指定が記述されている場合には、戻り値は送出し側データ項目としての引数-1を英数字の文字位置で数えた長さを表す整数である。その場合のデータ項目の送出しは、OCCURS句の規則に従って、DEPENDING指定中のデータ項目を評価することによって決定される。
    b. そうでない場合には、戻り値は引数-1を英数字の文字位置で数えた長さを表す整数である。
    c. 引数-1に暗黙的な無名項目があれば、戻り値にはその分の長さも含まれる。

11.9.26 LOG 関数

LOG関数は、eを底とする引数-1の対数（自然対数）の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION LOG（引数-1）
```

引数

1. 引数-1の字類は、数字とする。
2. 引数-1の値は、0より大きくする。

戻り値

1. eを底とする引数-1の対数の近似値が返される。
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.27 LOG10 関数

LOG10関数は、10を底とする引数-1の対数の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION LOG10（引数-1）
```

引数

1. 引数-1の字類は、数字とする。

2. 引数-1の値は、0より大きくする。

戻り値

1. 10を底とする引数-1の対数の近似値が返される。
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.28 LOWER-CASE 関数

LOWER-CASE関数は、文字列中の大文字を対応する小文字に置き換えて、引数-1と同じ長さの文字列を返す。この関数の型は引数によって決まる。具体的には下記のとおり。

引数の型	関数の型
英字	英数字
英数字	英数字
(MF)各国語型	各国語型

一般形式

```
FUNCTION LOWER-CASE（引数-1）
```

引数

1. 引数-1は、英字か、英数字か、
   (MF)または各国語型文字とし、
   最低１文字の長さがなければならない。

戻り値

1. 大文字が小文字に置き換えられる点を除けば、引数-1と同じ文字列が返される。
2. 返される文字列の長さは、引数-1と同じである。
3. 使用している計算機の文字集合に小文字が含まれていない場合、文字の置換は行われない。

11.9.29 MAX 関数

MAX関数は、引数-1のうちで値が最大であるものの内容を返す。 この関数の型は引数によって決まる。具体的には下記のとおり。

引数の型	関数の型
英字	英数字
英数字	英数字
(MF)各国語型	各国語型

すべて整数 数字 (一部に整数を含んでいてもよい)	整数 数字

一般形式

```
FUNCTION MAX（{引数-1}...）
```

引数

1. 引数-1を複数指定する場合、その字類はすべて同じにする。ただし、英字と英数字の引数は混在させてもよい。

戻り値

1. 引数-1のうち、値が最大のものの内容が返される。値を比較する際は、単純条件の規則が適用される。*単純条件* の節を参照。
2. 最大値をもつ引数-1が複数存在する場合、最も左側の引数-1の内容が返される。
3. 英数字型の関数、
   (MF)または各国語型関数
   の場合、返される値の文字数は、選択された引数-1の文字数と同じである。
4. 引数-1の値が非整数の場合、結果は浮動小数点形式で戻される。

11.9.30 MEAN 関数

MEAN関数は、引数の算術平均値を返す。この関数の型は数字である。

一般形式

```
FUNCTION MEAN（{引数-1}...）
```

引数

1. 引数-1の字類は、数字とする。

戻り値

1. 一連の引数-1の算術平均値が返される。
2. この算術平均値は、一連の引数-1の和を引数-1の数で割ったものである。
3. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.31 MEDIAN 関数

MEDIAN関数は、引数のメジアン（値によって整列したときに中央に来る引数）の内容を返す。この関数の型は数字である。

一般形式

```
FUNCTION MEDIAN（{引数-1}...）
```

引数

1. 引数-1の字類は、数字とする。

戻り値

1. 一連の引数-1を値によって整列したときに、中央に来る引数の内容が返される。
2. 指定した引数-1の数が奇数の場合、引数-1の数の少なくとも半分が戻り値以上の値となり、少なくとも半分が戻り値以下の値となるような値が返される。指定した引数-1の数が偶数の場合、中央に来る2つの引数の算術平均値が返される。
3. 引数-1の値を整列する際の比較には、単純条件の規則が適用される。*単純条件*の節を参照。
4. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.32 MIDRANGE 関数

MIDRANGE関数は、引数の最小値と最大値の算術平均値を返す。この関数の型は数字である。

一般形式

```
FUNCTION MIDRANGE（{引数-1}...）
```

引数

1. 引数-1の字類は、数字とする。

戻り値

1. 引数-1の最小値と最大値の算術平均値が返される。
2. 最小値と最大値を判定する際の比較には、単純条件の規則が適用される。*単純条件* の節を参照。
3. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.33 MIN 関数

MIN関数は、引数-1のうちで値が最小であるものの内容を返す。この関数の型は引数によって決まる。具体的には下記のとおり。

引数の型	関数の型
英字	英数字
英数字	英数字
(MF)各国語型	各国語型
すべて整数	整数
数字	数字
(一部に整数を含んでいてもよい)	

一般形式

```
FUNCTION MIN（{引数-1}...）
```

引数

1. 引数-1を複数指定する場合、その字類はすべて同じにする。ただし、英字と英数字の引数は混在させてもよい。

戻り値

1. 引数-1のうち、値が最小のものの内容が返される。値を比較する際は、単純条件の規則が適用される。*単純条件*の節を参照。
2. 最小値をもつ引数-1が複数存在する場合、最も左側の引数-1の内容が返される。
3. 英数字型の関数、
   (MF)または各国語型関数
   の場合、返される値の文字数は、選択された引数-1の文字数と同じである。
4. 引数-1の値が非整数の場合、結果は浮動小数点形式で戻される。

11.9.34 MOD 関数

MOD関数は、引数-1を引数-2で割ったときのモジュロー（法、割り切れる値との差）を整数値で返す。 この関数の型は整数である。

一般形式

```
FUNCTION MOD（引数-1 引数-2）
```

引数

1. 引数-1と引数-2は、整数とする。
2. 引数-2の値は、0であってはならない。

戻り値

1. 引数-1を引数-2で割ったときのモジュローが返される。その計算式は下記のとおり。
   引数-1 - (引数-2 * FUNCTION INTEGER (引数-1 / 引数-2))
2. どのような結果が得られるかを下に例示する。

引数-1	引数-2	引数-3
11	5	1
-11	5	4
11	-5	-4
-11	-5	-1

11.9.35 NATIONAL-OF 関数

(MF)

NATIONAL-OF関数は引数中の英数字型文字を各国語型の文字表現（内部表現）に変換する。
この関数の型は各国語である。

一般形式

FUNCTION NATIONAL-OF (引数-1 [引数-2])

引数

1. 引数-1の字類は英字または英数字とする。英数字の場合、引数-1には、英数字文字、各国語文字またはその両方が外部表現の形式で含まれ得る。
2. 引数-2の字類は各国語とし1文字位置の長さを占める。引数-2に指定された文字は、変換時に対応する各国語表現が存在しないような英数字文字に対する置き換え文字として使われる。

戻り値

1. 引数-1中の一つ一つの英数字文字を対応する各国語表現に変換し、戻り値として返す。引数-1に含まれる外部表現が持つすべての制御機能は、単に文字種別を判別するためだけに用いられ、変換対象の文字とは見なされない。
2. 引数-2が指定されると、対応する内部形式の各国語表現が存在しないような英数字文字に対してこの文字が戻り値に返される。
3. 引数-2を指定しない場合、対応する内部形式の各国語表現が存在しないような英数字文字に対しては、各国語型の空白文字が戻り値に返される。

4. 戻り値の長さは、変換された結果を保持するのに必要な、USAGE NATIONALの文字位置の個数となる。この長さは引数-1の長さによって定まる。

11.9.36 NUMVAL 関数

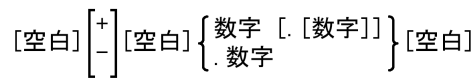
NUMVAL関数は、引数-1に指定された文字列によって表わされる数値を返す。前後の空白は無視される。この関数は数字型である。

一般形式

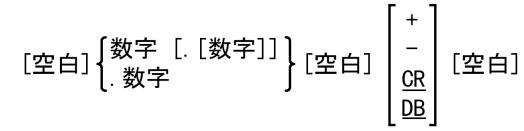
```
FUNCTION NUMVAL（引数-1）
```

引数

1. 引数-1は英数字定数
   (MF)か各国語型定数、
   または英数字データ項目
   (MF)か各国語型データ項目
   であり、下記のどちらかの形式をとる。

   または

   ここで、空白はいくつか（0個を含む）の空白文字の文字列を表わし、数字は1桁から18桁までの数字列を表わす。
2. 引数-1の合計桁数は、18を超えてはならない。
3. 特殊名段落にDECIMAL POINT IS COMMA句を指定した場合は、引数-1内でも小数点の代わりにコンマを使用する。

戻り値

1. 引数-1によって表わされる数値が返される。
2. 返される数値の桁数は18である。
3. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.37 NUMVAL-C 関数

NUMVAL-C関数は、引数-1に指定された文字列によって表わされる数値を返す。引数-2で指定された任意の通貨記号、または小数点の前にある任意のコンマは無視される。この関数は数字型である。

一般形式

FUNCTION NUMVAL-C（引数-1 [引数-2]）

引数

1. 引数-1は英数字定数か
   (MF)各国語型定数、
   または英数字データ項目か
   (MF)各国語型データ項目
   であり、下記のどちらかの形式とする。

   [空白][+/−][空白][通貨記号][空白]{数字 [,数字]... [.[数字]] / .数字}[空白]

   または

   [空白][通貨記号][空白]{数字[,数字]... [.[数字]] / .数字}[空白][+/−/CR/DB][空白]

   ここで、空白はいくつか（0個を含む）の空白文字の文字列を表わし、通貨記号は引数-2に指定した文字の何文字かの文字列を表わし、数字は何桁かの数字列を表わす。
2. 特殊名段落にDECIMAL POINT IS COMMA句を指定した場合は、引数-1中のコンマと小数点の役割が逆転される。
3. 引数-1の合計桁数は、18を超えてはならない。
4. 引数-2を指定した場合は、引数-1と同じ字類とする。
   引数-2に数字、コンマ、空白、正号（+）、負号（-）、小数点を指定してはならない。
5. 引数-2を指定しないと、引数-1の字類が英数字の時には、通貨記号はプログラムに指定したものが使用される。
   (MF); または引数-1の字類が各国語型の時には，通貨記号はプログラムで指定した各国語型表現のものが使用される。通貨記号の表現は、 FUNCTION NATIONAL-OF (cs)から戻される値である。

戻り値

1. 引数-1によって表わされる数値が返される。

2. 返される数値の桁数は18である。
3. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.38 ORD 関数

ORD関数は、プログラムの文字の大小順序中で、引数-1が位置する順序を示す整数値を返す。順序番号の最小値は1である。この関数の型は整数である。

一般形式

```
FUNCTION ORD（引数-1）
```

引数

1. 引数-1は、字類が英字か英数字か
   (MF)各国語型文字の1文字とする。

戻り値

1. 引数-1の字類が英字か英数字の場合、プログラムの英数字文字の大小順序中で、引数-1が位置する順序を示す値が返される。
2. (MF)引数-1の字類が各国語型の場合、プログラムの現行の国の大小順序中で、引数-1が位置する順序を示す値が返される。

11.9.39 ORD-MAX 関数

ORD-MAX関数は、引数-1の中で値が最大のものの順序番号を返す。この関数の型は整数である。

一般形式

```
FUNCTION ORD-MAX（{引数-1}...）
```

引数

1. 引数-1を複数指定する場合、その字類はすべて同じにする。ただし、英字と英数字の引数は混在させてもよい。

戻り値

1. 一連の引数-1のうち、値が最大のものの順序番号が返される。
2. 最大値を決めるために値を比較する際は、単純条件の規則が適用される。*単純条件* の節を参照。
3. 最大値をもつ引数-1が複数存在する場合、そのうちの最も左側のものの順序番号が返される。

11.9.40 ORD-MIN 関数

ORD-MIN関数は、引数-1の中で値が最小のものの順序番号を返す。この関数の型は整数である。

一般形式

```
FUNCTION ORD-MIN（{引数-1}...）
```

引数

1. 引数-1を複数指定する場合、その字類はすべて同じにする。ただし、英字と英数字の引数は混在させてもよい。

戻り値

1. 一連の引数-1のうち、値が最小のものの順序番号が返される。
2. 最小値を決めるために値を比較する際は、単純条件の規則が適用される。*単純条件* の節を参照。
3. 最小値をもつ引数-1が複数存在する場合、そのうちの最も左側のものの順序番号が返される。

11.9.41 PI 関数

(MF)

PI関数は円周率の近似値を返す。
この関数の型は数字である。

一般形式

```
FUNCTION PI
```

戻り値

1. 戻り値は下記のとおり。
   3.141592653589793238

11.9.42 PRESENT-VALUE 関数

PRESENT-VALUE関数は、引数-2に指定される将来の一連の期間にわたる値を、引数-1に指定される率で割り引いた、現在の値の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION PRESENT-VALUE (引数-1 {引数-2}...)
```

引数

1. 引数-1および引数-2の字類は、数字とする。
2. 引数-1の値は、-1よりも大きくする。

戻り値

1. 各期ごとに下記の式によって算出した、一連の値の和の近似値が返される。
   引数-2 / (1+引数-1) ** n
   引数-2の1つの値ごとに、1つの期が想定される。指数nは1から始まって、期が進むごとに1ずつ繰り上げられる。
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.43 RANDOM 関数

RANDOM関数は、一様分布から擬似乱数値を1つ返す。この関数の型は数字である。

一般形式

```
FUNCTION RANDOM [(引数-1)]
```

引数

1. 引数-1を指定する場合、その値は0または正の整数とする。引数-1は、一連の疑似乱数を発生させるためのシード値として使用される。
2. 後で別の引数-1を指定すると、新しい一連の擬似乱数が生成される。
3. (MF)実行単位内で最初にこの関数が実行されるときに引数-1が指定されていないと、シード値として0が使用される。
4. いずれの場合も、引数-1を指定しないでこの関数を繰り返し実行すると、現在の一連の乱数中の次の値が返される。

戻り値

1. 0以上1未満の値が返される。
2. 特定の処理系における特定のシード値に対して発生される一連の擬似乱数は、つねに同じである。
3. 引数-1の値の定義域は、別に一連の擬似乱数を発生させる。この部分集合には、0から最低32767までの値が含まれる。
4. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.44 RANGE 関数

RANGE関数は、最大の引数値と最小の引数値との差を返す。この関数の型は、引数の型によって決まる。具体的には下記のとおり。

引数の型	関数の型
すべて整数	整数
数字 (一部に整数を含んでいてもよい)	数字

一般形式

```
FUNCTION RANGE（{引数-1}...）
```

引数

1. 引数-1の字類は、数字とする。

戻り値

1. 引数-1の最大値から、引数-1の最小値を差し引いた値が返される。
2. 最大値および最小値を決めるために値を比較する際は、単純条件の規則が適用される。*単純条件*の節を参照。
3. 引数-1の値が非整数の場合、結果は浮動小数点形式で戻される。

11.9.45 REM 関数

REM関数は、引数-1を引数-2で割った剰余値を返す。この関数の型は数字である。

一般形式

```
FUNCTION REM（引数-1 引数-2）
```

引数

1. 引数-1と引数-2の字類は、数字とする。
2. 引数-2の値は、ゼロであってはならない。

戻り値

1. 引数-1を引数-2で割った剰余値が返される。その計算式は下記のとおり。
   引数-1 - (引数-2 * FUNCTION INTEGER-PART (引数-1 /引数-2))
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.46 REVERSE 関数

REVERSE関数は、引数-1の文字列の順序を逆転させた同じ長さの文字列を返す。この関数の型は、引数の型によって決まる。具体的には下記のとおり。

引数の型	関数の型
英字	英数字
英数字	英数字
(MF)各国語型	各国語型

一般形式

```
FUNCTION REVERSE（引数-1）
```

引数

1. 引数-1は、字類が英字か英数字か
   (MF)各国語型
   であり、長さは最低でも1文字とする。

戻り値

1. 引数-1の文字列の長さをn、その中の任意の文字の位置をjとすると（$1 \leqq j \leqq n$ ）、返される文字列中の文字位置jにある文字は、元の文字列のn-j+1の位置から持って来たものである。

11.9.47 SIGN 関数

(MF)
SIGN関数は引数の正負の範囲に応じて、+1，0，-1のどれかを返す。
この関数の型は整数である。

一般形式

```
FUNCTION SIGN（引数-1）
```

引数

1. 引数-1の字類は数字とする。

戻り値

1. 戻り値は下記のとおり。
   a. 引数-1が正の数である場合には、1

b. 引数-1の値がゼロである場合には、0
c. 引数-1が負の数である場合には、-1

11.9.48 SIN 関数

SIN関数は、引数-1に指定された角または弧（単位はラジアン）のサイン（正弦）の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION SIN（引数-1）
```

引数

1. 引数-1の字類は、数字とする。

注: 度およびグラードからラジアンへの変換係数をこの章の「三角関数」の節に示してある。

戻り値

1. 引数-1の、サインの近似値が返される。その値は-1以上+1以下である。
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.49 SQRT 関数

SQRT関数は、引数-1の平方根の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION SQRT（引数-1）
```

引数

1. 引数-1の字類は、数字とする。
2. 引数-1の値は0または正の整数とする。

戻り値

1. 引数-1の平方根の、近似値の絶対値が返される。
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.50 STANDARD-DEVIATION 関数

STANDARD-DEVIATION関数は、引数の標準偏差の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION STANDARD-DEVIATION（{引数-1}...）
```

引数

1. 引数-1の字類は、数字とする。

戻り値

1. 一連の引数-1の標準偏差の近似値が返される。
2. 計算手順は下記のとおり。
    a. 一連の引数-1の算術平均を算出する。各引数-1の値からその平均値を引いて、差異を算出する。各差異を2乗する。
    b. 各2乗値を加算する。 その和を引数-1の数で割る。
    c. その商の平方根を求める。 その平方根の絶対値を戻り値とする。
3. 引数-1が1つしかない場合、または一連の引数-1がすべて可変の反復データ項目から成りそのすべてについて反復数が1である場合、ゼロが返される。
4. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.51 SUM 関数

SUM関数は、引数の値の和を返す。この関数の型は、引数の型によって決まる。具体的には下記のとおり。

引数の型	関数の型
すべて整数	整数
数字 (一部に整数を含んでいてもよい)	数字

一般形式

```
FUNCTION SUM（{引数-1}...）
```

引数

1. 引数-1の字類は、数字とする。

戻り値

1. 引数の値の和が返される。
2. 一連の引数-1がすべて整数であれば、戻り値は整数である。
3. 一連の引数-1の中に整数でないものがあれば、戻り値は浮動小数形式の数字である。

11.9.52 TAN 関数

TAN関数は、引数-1に指定された角または弧（単位はラジアン）のタンジェント（正接）の近似値を返す。 この関数の型は数字である。

一般形式

```
FUNCTION TAN（引数-1）
```

引数

1. 引数-1の字類は、数字とする。

注: 度およびグラードからラジアンへの変換係数をこの章の「三角関数」の節に示してある。

戻り値

1. 引数-1のタンジェントの近似値が返される。
2. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.53 UPPER-CASE 関数

UPPER-CASE関数は、引数-1の文字列中の小文字を対応する大文字に置き換えた同じ長さの文字列を返す。この関数の型は、引数の型によって決まる。具体的には下記のとおり。

引数の型	関数の型
英字	英数字
英数字	英数字
(MF)各国語型	各国語型

一般形式

```
FUNCTION UPPER-CASE（引数-1）
```

引数

1. 引数-1は、字類が英字か英数字か

(MF)各国語型
であり、最低でも1文字の長さとする。

戻り値

1. 小文字が対応する大文字に置き換えられる点を除けば、引数-1と同じ文字列が返される。
2. 返される文字列の長さは、引数-1と同じである。

11.9.54 VARIANCE 関数

VARIANCE関数は、引数の分散の近似値を返す。この関数の型は数字である。

一般形式

```
FUNCTION VARIANCE（{引数-1}...）
```

引数

1. 引数-1の字類は、数字とする。

戻り値

1. 一連の引数-1の分散の近似値が返される。
2. 分散は標準偏差を2乗したものである。(前述の *STANDARD-DEVIATION関数*の節を参照)
3. 引数-1が1つしかない場合、または一連の引数-1がすべて可変の反復データ項目から成りそのすべてについて反復数が1である場合、ゼロが返される。
4. 数値が非整数の結果には、浮動小数点形式が使用される。

11.9.55 WHEN-COMPILED 関数

WHEN-COMPILED関数は、プログラムがコンパイルされた日付と時刻を返す。この関数の型は英数字である。

一般形式

```
FUNCTION WHEN-COMPILED
```

戻り値

1. 戻り値の、文字列の文字位置（左から右に数える）ごとの内容を下に示す。

文字位置	内　　容
1-4	グレゴリオ歴の暦年を表わす4桁の数字
5-6	年内の月を表わす2桁の数字。値は01から12まで
7-8	月内の日を表わす2桁の数字。値は01から31まで
9-10	真夜中から経過した時間を表わす2桁の数字。値は00から23まで
11-12	時刻の分を表わす2桁の数字。値は00から59まで
13-14	時刻の秒を表わす2桁の数字。値は00から59まで
15-16	1秒の100分の1を表わす2桁の数字。値は00から99まで 1秒よりも細かく時間を測定する機能のないシステムでは、この値は00となる。
17	グリニッジ標準時間との時差の遅早を表わす。値は"-"，" +"，"0" のいずれかである。この値が"-"の時は、この前の文字位置に示された時間はグリニッジ標準時間よりも遅れていることを示す。この値が"+"の時は、この前の文字位置に示された時間はグリニッジ標準時間と等しいかそれよりも進んでいることを示す。この値が"0"の時は、この関数を実行したシステムにはグリニッジ標準時間との時差を測る機能がないことを示す。 グリニッジ標準時間との時差を測る機能がないシステムでは、文字位置17から21には値0000が返される。
18-19	文字位置17の値が"-"の時は、システムで使用されている時刻がグリニッジ標準時間よりも遅れている時間数が、2桁の数字でここに示される。その値は00から12までである。文字位置17の値が"+"の時は、システムで使用されている時刻がグリニッジ標準時間よりも進んでいる時間数が、2桁の数字でここに示される。その値は00から13までである。文字位置17の値が"0"の時は、ここには00が返される。
20-21	システムで使用されている時刻とグリニッジ標準時間との時差の、1時間に満たない分数を表わす。その値は00から59までである。時差の遅早は文字位置17の値が"+"であるか"-"であるかによって示される。 文字位置17の値が"0"の時は、ここには00が返される。

2. この関数を含むプログラムがコンパイルされた日付と時刻が返される。プログラムが他のプログラムに含まれている場合には、含む方のプログラムがコンパイルされたときの日付と時刻が返される。
3. ここに返される値は、該当するプログラムの原始プログラム・リストおよび実行用プログラムに付けられているものと同じである。ただし、その表現形式と精度は異なる。

11.9.56 YEAR-TO-YYYY 関数

YEAR-TO-YYYY 関数は、引数-1 を、年の下位の 2桁から 4桁の年へ変換する。引数-2 は実行時に年に追加された時、100年間隔の終了年を定義するか、または引数-1 が起こるところへのウィンドウの移動を定義する。この関数の型は整数である。

一般形式

FUNCTION YEAR-TO-YYYY (引数-1 [引数-2])

引数

1. 引数-1 は 100 より小さい、負でない整数とする。
2. 引数-2 は整数とする。
3. 引数-2 が省略された場合、この関数は 50 が指定されたものとして評価される。
4. 実行時の年と引数-2 の値の合計は、10,000 より小さく、1699より大きいものとする。

戻り値

1. 最大年は次のように計算される。
   (FUNCTION NUMVAL (FUNCTION CURRENT-DATE (1:4)) + 引数-2)
   ここで、NUMVAL 関数の引数-2 は YEAR-TO-YYYY 関数自身の引数-2 と同じである。
1. 同等の算術式は次のとおりである。

    1. 以下の条件が真のとき:
       (FUNCTION MOD (最大年, 100) > = 引数-1)
       同等の算術式:
       引数-1 + 100 * (FUNCTION INTEGER (最大年/100))
    2. それ以外のときの、同等の算術式:
       引数-1 + 100 * (FUNCTION INTEGER (最大年/100) - 1)

注:

1. 1995年では、FUNCTION YEAR-TO-YYYY (4, 23) の戻り値は2004 である。2008年では、FUNCTION YEAR-TO-YYYY (98, (-15))の戻り値は 1898 である。
2. YEAR-TO-YYYY 関数はウィンドウ移動アルゴリズムを持っている。これをきまったウィンドウに使用するには、引数-2を次のように指定できる。ここで、固定最大年は固定した 100年間隔での最大年とする。
   (固定最大年 - FUNCTION NUMVAL (FUNCTION CURRENT-DATE (1:4)))
   決まったウィンドウが 1973 から 2072 の場合、2009 では引数-2 は値 63 を持ち、2019 では値 53 を持つ。

第12章 手続き部 - ACCEPT - DIVIDE

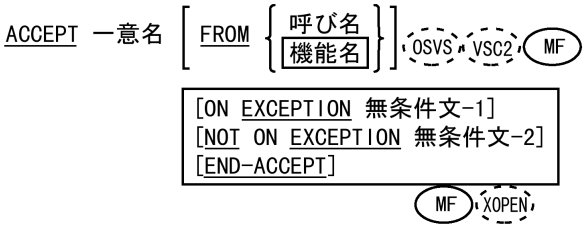
12.1 COBOLの文

12.1.1 ACCEPT（受取り）文

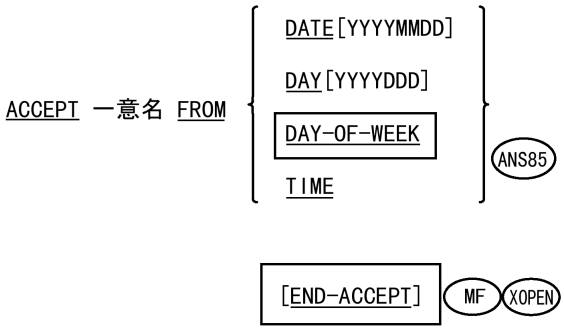
ACCEPT（受取り）文は、操作卓からキー入力されるデータ、またはオペレーティングシステムから提供されるデータを、指定されたデータ項目に入れてプログラムで利用できるようにする。

一般形式

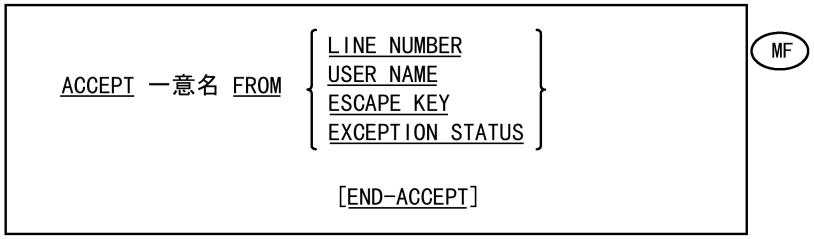
書き方 1



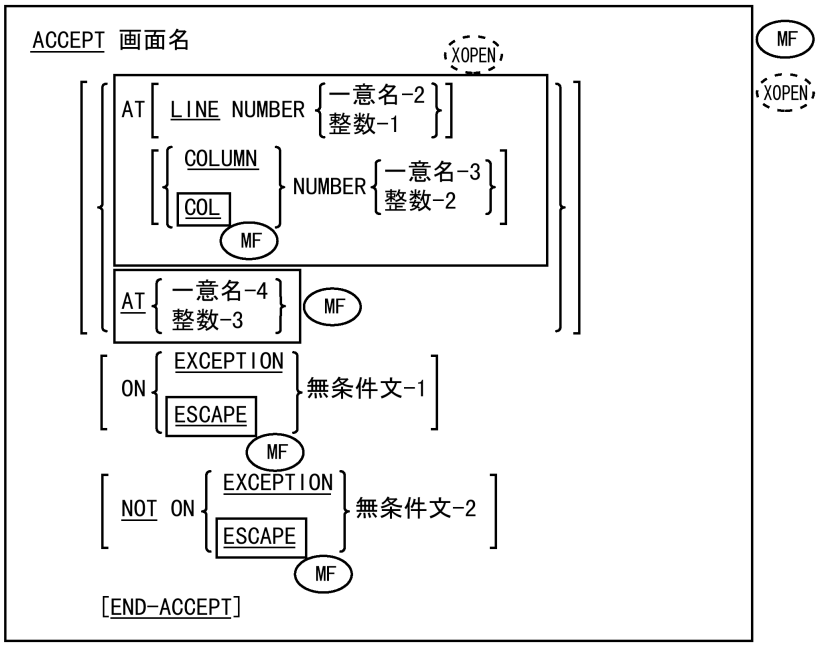
書き方 2



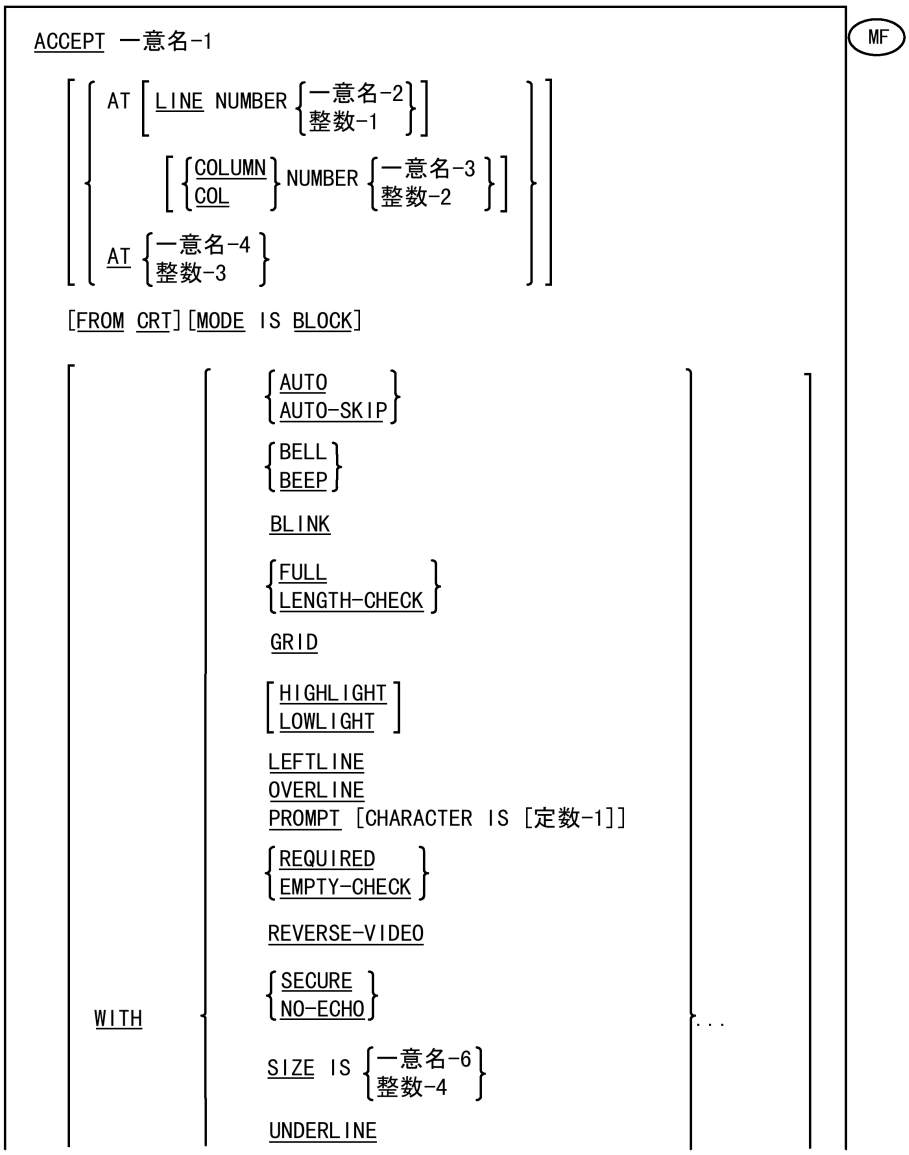
書き方 3

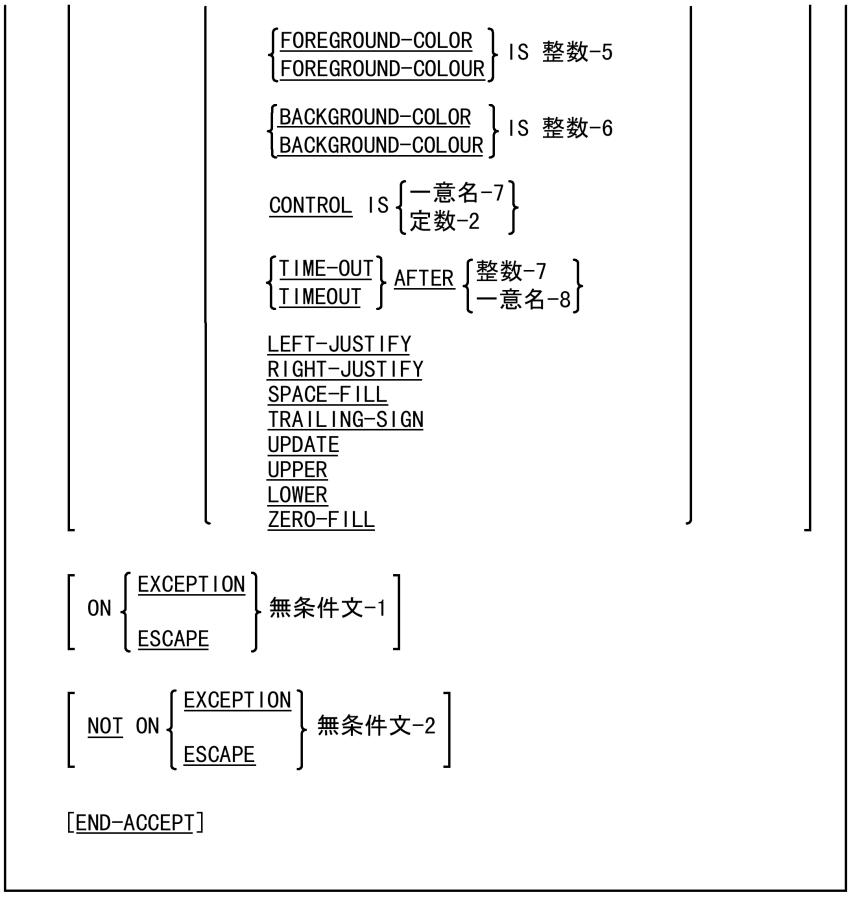

ACCEPT 一意名 FROM
LINE NUMBER
USER NAME
ESCAPE KEY
EXCEPTION STATUS
[END-ACCEPT]
MF

書き方 4


ACCEPT 画面名
XOPEN
AT LINE NUMBER 一意名-2 整数-1
COLUMN
COL
MF
NUMBER 一意名-3 整数-2
AT 一意名-4 整数-3
MF
ON EXCEPTION ESCAPE 無条件文-1
MF
NOT ON EXCEPTION ESCAPE 無条件文-2
MF
[END-ACCEPT]
MF
XOPEN

書き方 5


ACCEPT 一意名-1
MF
AT [LINE NUMBER {一意名-2 / 整数-1}]
[{COLUMN / COL} NUMBER {一意名-3 / 整数-2}]
AT {一意名-4 / 整数-3}
[FROM CRT][MODE IS BLOCK]
WITH
{AUTO / AUTO-SKIP}
{BELL / BEEP}
BLINK
{FULL / LENGTH-CHECK}
GRID
[HIGHLIGHT / LOWLIGHT]
LEFTLINE
OVERLINE
PROMPT [CHARACTER IS [定数-1]]
{REQUIRED / EMPTY-CHECK}
REVERSE-VIDEO
{SECURE / NO-ECHO}
SIZE IS {一意名-6 / 整数-4}
UNDERLINE
...



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - ACCEPTREFRESH - ACCEPT文の前に、画面節のデータに関連するデータ領域を作業場所節中の対応する項目によって更新するか否かを指定する。
    - XOPEN - ACCEPT文の前に、画面節のデータに関連するデータ領域を作業場所節中の対応する項目によって更新するか否かを指定する。

構文規則

書き方 1

1. 書き方1の呼び名は、環境部の特殊名段落中の機能名に関連させる。 有効な機能名に関しては、この章で前述した*特殊名段落*の節の中の一般規則12 を参照。
2. (OSVS)(VSC2)(MF)機能名に関連する呼び名の代わりに、機能名そのものを使用できる。
3. (MF)(XOPEN)呼び名が特殊名段落中のARGUMENT-NUMBERと関連するときは、一意名は符号の付かない整数とする。

4. (MF)(XOPEN)呼び名が特殊名段落中のARGUMENT-VALUEまたはENVIRONMENT-VALUEと関連するときは、一意名は英数字データ項目とする。
5. (VSC2)(MF)一意名は、USAGE DISPLAY-1（2バイト文字）項目でも、外部浮動小数点数データ項目でもよい。
   (MF)一意名は、内部浮動小数点数データ項目でもよい。
6. (MF)(XOPEN)EXCEPTION 指定は、FROMがENVIRONMENT-NAMEかARGUMENT-VALUEと共に指定された時、またはこれと関連する呼び名で指定された時にだけ、指定することができる。

書き方 2

7. (OSVS)(VSC2)(MF)一意名は、内部浮動小数点数データ項目でも、外部浮動小数点数データ項目でもよい。

書き方 3

8. (MF)一意名の字類、項類、用途には何の制限もない。しかし、一意名に入れられる実際の値およびその値を一意名に転記することの妥当性は、FROM指定に左右される。(詳細については、書き方3に関する一般規則を参照）。

書き方 4

9. (MF)画面名は、OCCURS句に含まれる項目であってはならない。

書き方 4 および 5

10. (MF)LINE指定およびCOLUMN指定は、どちらから先に書いてもよい。
11. (MF)EXCEPTIONとESCAPEは同義であり、どちらを書いてもよい。
12. (MF)一意名-4は、PIC 9（4）またはPIC 9（6）のデータ項目とする。

書き方 5

13. (MF)一意名-8は整数とする。符号付きとすることができる。
14. (MF)整数-7は符号付きとすることができる。
15. (MF)書き方5に該当するACCEPT文は作用対象が画面名でなく、AT指定、CRTオプションを伴うFROM指定、WITH指定、MODE IS BLOCK指定、EXCEPTION指定のどれかが指定されているもの、またはFROM指定は指定されていないが特殊名段落にCONSOLEオプション句が指定されているものである。CONSOLEオプション句を伴うFROM指定が指定されているか、FROM指定が指定されておらず特殊名段落にCONSOLE IS CRT句も指定されていないACCEPT文は、書き方1のACCEPT文として扱われる。
16. (MF)一意名の後ろに続ける指定は、どのような順序で書いてもよい。
17. (MF)SPACE-FILL, ZERO-FILL, LEFT-JUSTIFY, RIGHT-JUSTIFY, PROMPT, TRAILING-SIGNオプションを指定できるのは、作用対象が基本項目の場合だけである。
18. (MF)一意名-1に属する基本項目の用途は、USAGE DISPLAYとする。
19. (MF)一意名-1に属する非基本項目は8191バイトより大きいことがある。MODE IS BLOCK句を使用した場合、一意名-1の全体の長さは8191バイト以下にすること。

一般規則

すべての書き方

1. END-ACCEPT指定はACCEPT文の範囲を区切る。（第２章『COBOL言語の概念』の2.6.6.4「明示範囲符と暗黙範囲符」を参照。）MF(4)コンパイラ指令を設定した場合にのみ、END-ACCEPTは予約語として扱われる。

書き方 1

2. ACCEPT文は論理装置または物理装置からデータを移送する働きをする。そのデータによって、一意名が指すデータ項目の内容が置き換えられる。一意名が指すデータ項目に用途としてDISPLAYが明示的または暗黙的に指定されている場合、フォーマットの変換なしに直ちにその置換が行われる。そうでない場合には、正しいフォーマットへの変換が行われる。

   データ移送のサイズは装置とランタイム環境とによって決まる（使用可能な機能名のリストについては特殊名段落を、装置およびデータ移送サイズの制限の詳細についてはCOBOLシステム・ドキュメンテーションを、それぞれ参照）。

   装置が受取り側データ項目と同じサイズのデータを移送できる場合には、移送されたデータが受取り側データ項目に収められる。そうでない場合は、下記のように扱われる。

   a. 受取り側データ項目（または現在まだ移送されたデータによって占められていない部分）のサイズが移送されるデータのサイズを超える場合は、移送されたデータが受取り側データ項目（またはまだ空いている部分）中に左詰に収められ、さらにデータが要求される。
   b. 移送されたデータのサイズが受取り側データ項目（または現在まだ移送されたデータによって占められていない部分）のサイズを超える場合は、移送されたデータの左端から受取り側データ項目（または残りの部分）に収まる分だけが受け取られる。移送されたデータのうちの受取り側データ項目に収まらない残りの部分は無視される。

   FROMオプションを指定しなかった場合は、FROM CONSOLEと指定したのに等しい。
3. (MF)(XOPEN)機能名 COMMAND-LINE、またはそれに関連する呼び名を指定すると、受取り側データ項目はシステムのコマンド行バッファによって上書きされる。
4. (MF)(XOPEN)機能名ARGUMENT-NUMBERに関連する呼び名を使用すると、一意名はコマンド行中に含まれる引数を受け取る。（これには、すべての引数が含まれる。つまり、プログラム名の前にある引数、プログラム名自体、プログラム名の後ろに続く引数が含まれる。したがって、プログラム呼出しの種類によっては移植性を期待できない）。
5. (MF)(XOPEN)呼び名がARGUMENT-NUMBERに関連する時、一意名は現在のコマンド行の引数になる。どのコマンド行の引数を現在のものとするかの決定は、下記の通りである。
   a. はじめに、最初のコマンド行の引数を現在のものとする。
   b. ARGUMENT-NUMBERに関連する呼び名をもつDISPLAYが、現在のコマンド行の引数の数をDISPLAY文で指定された一意名または定数の値に設定する。

c. ARGUMENT-NUMBERに関連する呼び名をもつDISPLAYが最後に実行されてから、ARGUMENT-VALUEに関連する呼び名をもつACCEPTが実行された時は、現在のコマンド行の引数の数はこのACCEPT文が使用される前に増やされる。

現在の引数の数が0に設定されていると、これは、実行単位の主プログラムのプログラム名が返されることを意味する。しかし、プログラムの呼出しにいろいろな変形があるため、主プログラム名の代わりにユーティリティプログラムまたは呼ぶプログラムの名前が返されることがある。

ACCEPT文の実行時に現在の引数の数が

(XOPEN)99より大きいか、0より小さいか、

コマンド行上の実際の引数の数よりも大きい場合、無条件文-1が指定されていると、これが実行される。

6. (MF)(XOPEN)機能名ENVIRONMENTAL-VALUEに関連する呼び名を使用した場合、下記のようになる。
   a. ENVIRONMENT-NAMEに関連する呼び名をもつDISPLAYが前に実行されていると、指定された環境変数の値が一意名に入れられる。
   b. ENVIRONMENT-NAMEに関連する呼び名をもつDISPLAYが前に実行されていないか、または指定された環境変数が存在しないと、ON EXCEPTION指定に指定されている無条件文が実行される。この場合、一意名の値はどうなるかわからない。
7. (MF)(XOPEN)他のプログラムによって呼ばれたプログラムの中で、コマンド行の引数および引数の数を検索する効果は、それらを実行単位中の最初のプログラムによって検索する場合と同じである。

書き方 2

8. ACCEPT文は、要求された情報を一意名によって指定されたデータ項目に取り込む。その際、MOVE文の規則が適用される。DATE、DAY、

   (ANS85)DAY-OF-WEEK、

   およびTIMEは概念的なデータであり、COBOLプログラム中には記述されていない。
9. DATEは、年（西暦下2桁）、月、日の3つの要素から構成される。これらの要素の配列順は、高い方から低い方（左から右）、つまり年・月・日の順である。COBOLプログラムからDATEを呼び出すと、そのプログラムの中で、符号の付かない6桁の整数の基本データ項目として記述してあるように取り扱うことができる。
10. DAYは、年（西暦下2桁）と年内の通算日の2つの要素から構成される。これらの要素の配列順は、高い方から低い方（左から右）、つまり年・通算日の順である。たとえば、1968年7月1日は68183と表わされる。COBOLプログラムからDAYを呼び出すと、そのプログラムの中で、符号の付かない5桁の整数の基本データ項目として記述してあるように取り扱うことができる。
11. (ANS85)DAY-OF-WEEKは、曜日を表わす単一の要素から構成される。COBOLプログラムからDAY-OF-WEEKを呼び出すと、そのプログラムの中で、1桁の符号の付かない整数の基本データ項目として記述してあるように取り扱うことができる。DAY-OF-WEEKは1から7までの値をとり、それぞれ月曜日から日曜日を表わす。

12. TIMEは、時間、分、秒、百分の1秒の4つの要素から構成される。この時間は夜中の0時から24時間制で測られる。たとえば、午後2時41分は14410000と表わされる。COBOLプログラムからTIMEを呼び出すと、そのプログラムの中で、符号の付かない8桁の整数の基本データ項目として記述してあるように取り扱うことができる。TIMEの最小値は00000000であり、最大値は23595999である。 ハードウエアの時間測定の精度が上記よりも低い場合は、得られた値は最も近い近似値に変換される。

書き方 3

13. (MF)ACCEPT FROM LINE NUMBER句から返される値は、つねに数字である。この値は、使用しているシステムによって異なる。
14. (MF)FROM USER NAMEオプションは、UNIXシステム上のユーザー識別番号を返す。このような概念のないシステム上では、このオプションは空白を返す。
15. (MF)FROM ESCAPE KEYオプションは、終了キーによって発生させられた2文字のコードを返す。
16. (MF)EXCEPTION STATUS項目には、3桁のコードが入っている。このコードは、CALL文を実行中に発生した、例外条件の種類を識別する。
    EXCEPTION STATUSを調べる場合は、CALL文の直後で行うのがよい。 CALLとACCEPT FROM EXCEPTION STATUSとの間には、何もあってはならない。たとえば、その間にファイル入出力をはさむと、EXCEPTION STATUSが変えられてしまうので、その内容を調べる意味がなくなってしまう。

書き方 4

17. (MF)ACCEPT文のこの書き方は、画面節中に定義されている画面項目を受け取って、拡張画面操作機能によって全面的に処理できるようにする。

書き方 4 および 5

18. (MF)ACCEPT文の実行順序は、つねに以下のとおりである。
    1. AT指定
    2. BLANK指定
    3. コンパイラ指令ACCEPTREFRESHまたはXOPENのどちらかが指定されていると、USING指定で現在の画面項目の内容が表示される。
    4. BELL指定
    5. ACCEPT操作
19. (MF)AT指定は、ACCEPT処理を開始する画面上の絶対番地を指定する。
20. (MF)整数-3または整数-4の長さが4桁ならば、上2桁は行を表わし、下2桁はカラムを表わす。整数-3の長さが6桁ならば、上3桁は行を表わし、下3桁はカラムを表わす。
21. (MF)行番号とカラム番号の組合わせの中には、特殊な意味をもつものがある。それらは下記のとおり。

a. カラムの値が1行のカラム数よりも大きいと、その値から1行のカラム数が差し引かれ、行数に1が加えられる。カラムの値が1行のカラム数の範囲内におさまるまで、この処理が繰り返される。
b. 行の値が画面の範囲から外れると、画面が1行上にスクロールされる。この効果は、画面の下端の行を指定したのと同じである。
c. 与えられた行番号とカラム番号がともにゼロであると、書き方4または書き方5の前のACCEPT処理が終わった位置の直後から、ACCEPTが開始される。各行のカラム1は、前の行の最後のカラムに続くものとみなされる。
d. 指定された行番号がゼロであるがカラム番号はゼロでないと、書き方4または書き方5の前のACCEPT処理が終わった位置の次の行の指定されたカラムから、ACCEPTが開始される。
e. 指定されたカラム番号がゼロであるが行番号はゼロでないと、書き方4または書き方5の前のACCEPT処理が終わった位置の次のカラムの指定された行から、ACCEPTが開始される。

22. (MF)このACCEPT文を実行する前に、このACCEPT文に指定されたのと同じ画面名または一意名-1が指定されているDISPLAY文が実行されていなければならない。それ以降に何らかのACCEPT文またはDISPLAY文が実行されていてはならない。
23. (MF)ON EXCEPTION指定を書くと、ACCEPT処理が正常終了しなかった場合に、無条件文-1が実行される。NOT ON EXCEPTION指定を書くと、ACCEPT処理が正常終了した場合に、無条件文-2が実行される。(終了の種類については、前述の*CRT STATUS句*の項を参照。)

書き方 5

24. (MF)AT指定を指定しないと、ACCEPT処理は行1、カラム1から開始される。
25. (MF)MODE IS BLOCK指定を指定しないと、一意名が集団項目であれば、それに属する基本項目で名前がFILLERでないものが受け取られる。受け取られる項目は、データ部内でそれらが記述されている順に画面に配置され、集団内のFILLERの長さによって区切られる。この用途では、ある行の最初の位置は、その前の行の最後の位置の直後に続くものとみなされる。項目は、同じ順序で受け取られる。
CURSOR IS句（前述の*CURSOR IS 句*の項を参照）内で別途指定しないかぎり、カーソルは最初の項目の始点に位置付けられる。各項目へのACCEPT処理が終わるにつれて、カーソルは次の項目の始点に移動される。
26. (MF)一意名-1が集団項目であり、その下位に可変反復データ項目がある場合、ACCEPT文はMODE IS BLOCK句が指定されているかのように動作する。
27. (MF)MODE IS BLOCK指定は、一意名を基本項目として扱うように指定する。こうすると、一意名が集団項目であっても、1つの項目として表示される。
28. (MF)PROMPTオプションを指定しないと、空の文字位置を示すための文字は表示されない。CHARACTERオプションを抜かしてPROMPTを指定すると、構成時にプロンプト用に設定されている文字が文字位置の表示に用いられる。
29. (MF)PROMPTオプションを指定しないと、データが入力されなかった文字位置は、データ項目内では空白となる。

30. ㊎WITH指定を用いると、ある種のオプションをACCEPT処理の実行時に指定できる。（これらの句の説明については、前述の*画面節* を参照。）
    画面記述句に指定できるオプションの他に、WITH指定にはいくつかのオプションが加えられている。SPACE-FILL, ZERO-FILL, LEFT-JUSTIFY, RIGHT-JUSTIFY, TRAILING-SIGN, UPDATEである。ZERO-FILLはこのリストにも画面記述句にも現れている。これは、ZERO-FILLには2つの用法があるからである。2番目の用法については、この章で後述する。
    自由方式で、数字および数字編集の画面項目にデータを入力できるようにする構成オプションがある。COBOLでは、非編集数字データ項目は、内部形式でデータを保持することを目的としたものである。しかし、この形式を用いると、そのようなデータ項目を画面上に表示できる（詳細は、ユーザーインタフェースに関するCOBOLシステムのマニュアルを参照のこと）。自由方式がとられているときは、データは自動的に下記のように編集されて表示される。
    - 仮想小数点を終止符で表わす。
    - 符号を符号文字で表わす（負の数は"-" で、正の数は空白で）。符号は左端の数字の直前に付けられる。
    - すべての整数文字位置の先行ゼロが抑制される。ただし、最末桁を除く
    - 左に桁寄せされる。

    SPACE-FILL, ZERO-FILL, LEFT-JUSTIFY, RIGHT-JUSTIFY, TRAILING-SIGNオプションを使用すると、上記の形式を補正できる。
31. ㊎SPACE-FILLオプションは、自由方式の非編集数字データ項目の、すべての整数文字位置の先行ゼロを抑制して、画面に表示するようにする。このオプションは、自由方式の非編集数字データ項目に対してだけ働く。この機能は、データ項目中の初期データが表示されるときに効力を発揮し、そのデータ項目へのACCEPT処理が終了するときにも再び効力を発揮する。左端に付された符号があれば、右端に表示される。
32. ㊎ZERO-FILLオプションは、自由方式の非編集数字データ項目をゼロ抑制しないで、画面に表示するようにする。この機能は、データ項目中の初期データが表示されるときに効力を発揮し、そのデータ項目へのACCEPT操作が終了するときにも再び効力を発揮する。（英字または英数字のデータ項目にこのオプションを適用したときの効果については、前述の*ZERO-FILL 句*の節を参照）。
33. ㊎LEFT-JUSTIFYオプションは、注記にすぎない。
34. ㊎RIGHT-JUSTIFYオプションは、入力されたデータを項目の右に桁寄せして、画面に表示するようにする。このオプションは、自由方式の非編集数字データ項目に対してだけ働く。この機能は、データ項目中の初期データ（表示された現在の内容）が表示されるときに効力を発揮し、そのデータ項目へのACCEPT処理が終了するときにも再び効力を発揮する。
35. ㊎TRAILING-SIGNオプションは、演算符号を項目の右端の文字位置に表示するようにする。この機能は、データ項目中の初期データが表示されるときに効力を発揮し、そのデータ項目へのACCEPT処理が終了するときにも再び効力を発揮する。このオプションは、自由方式の非編集数字データ項目に対してだけ働く。

36. (MF)UPDATEオプションは、データ項目の現在の内容（初期データ）を表示してから、新しいデータの入力を促すようにする。新しいデータが何も入力されないと、初期データが入力されたかのように扱われる。UPDATEオプションを指定しないと、初期データが表示されるか否かは構成オプションによる（構成オプションの詳細については、ユーザーインタフェースに関するCOBOLシステムのマニュアルを参照のこと）。
37. (MF)WITH UPPERオプションは、入力された文字を強制的に大文字にする。
38. (MF)WITH LOWERオプションは、入力された文字を強制的に小文字にする。
39. (MF)一意名-1にREDEFINESがかかっている場合、再定義データ領域の最初の記述が使用され、以降の記述は無視される。OCCURSまたは入れ子になったOCCURSが指定されている場合、反復されるデータ項目はその反復回数だけ展開される。したがって、1つの定義が多数の項目にわたって繰り返されることになる。
40. (MF)一意名-8または整数-7の値が負の場合、時間切れによる"例外条件"を発生させないという要求を表わす。この場合、キーが打鍵されるまでの時間またはキーの打鍵間隔がどんなに長くなっても、例外条件は発生しない。
41. (MF)一意名-8または整数-7の値がゼロの場合、入力待ちの間は時間切れを起こさせないことを意味する。しかし、入力待ちでない（ACCEPTが処理された）ときには、直ちに時間切れとする。
42. (MF)ON EXCEPTION指定を書くと、TIME-OUT指定が指定してあって時間切れが発生したときに実行される。NOT ON EXCEPTION指定を書くと、TIME-OUT指定が指定してあるが時間切れ（または他の例外条件）が発生しなかったときに実行される。
43. (MF)時間切れ例外条件が発生すると、ACCEPTの結果の項目は下記のようになる。
    a. 項目の一部が既に変更されていると、その時点の内容がそのまま保持される。
    b. "FULL"や"REQUIRED" をはじめとする、通常は部分的な入力を許さない属性をもつ項目でも、時間切れ例外条件が発生したときは、その要件を満たすことを求められない。
    c. 上記の規則は、キーを打つたびに時間切れ測定用の時計が再設定されるか否かにかかわらず、適用される。
44. (MF)実行時に百分の2,147,483,647秒よりも大きな正の時間切れ間隔が検出されると、その値は百分の2,147,483,647秒（約8カ月）に設定し直される。
45. (MF)TIME-OUTの値は、ACCEPT文の実行が開始されてから時間切れ例外条件が発生するまでの時間を、秒（または十分の一秒）単位で指定する。新しいADISCF構成オプションは、キーボード上で何か操作が行われるたびに、時間切れ測定用の時計を"再設定"するか否かを制御する。ACCEPT文によって時間切れ測定用の時計を再設定したりしなかったりする応用では、設定状態の変更を必要とするACCEPT文の前または後ろで、ADIS実行時インタフェースを個別に呼ぶことができる。たとえば、時間切れ測定用の時計を再設定しないようにADISが設定されているときに、次の文を実行したとする。

```
ACCEPT INPUT-FIELD TIME-OUT AFTER +10
```

すると、ACCEPTの開始から10秒経過後に、時間切れ条件が発生する。この場合、ACCEPTの開始5秒後に何か文字を打ったとしても、時間切れ条件の発生を防ぐことにはならない。

また、時間切れ測定用の時計を再設定するようにADISが設定されているとする。すると、上記の同じ文を実行した場合、何か文字を打つたびにTIMEOUT"時計"はゼロに設定し直される。

46. (MF)ON EXCEPTION指定を指定しないと、時間切れ条件が発生したときに、CRT状態キー（指定してあれば）が更新され、処理は次の論理的な指定まで続行される。このとき、ACCEPTの受取り側項目の内容は上に説明したとおりである（このことは、NOT ON EXCEPTION指定を指定した場合にも、当てはまる）。
47. (MF)（NOT）ON EXCEPTION指定に関する説明はすべて、（NOT）ON ESCAPE指定にも当てはまる。

書き方 1および5

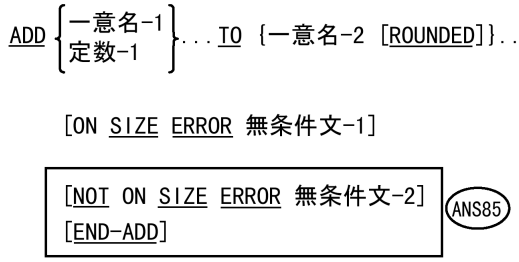
48. (MF)英数字データ項目に関しては、ACCEPT文の実行中に画面から受け取られたフィールドのサイズは、宛先のフィールドのサイズとまったく同じである。したがって、フィールドの右側の部分にデータを表示したい場合には、その位置にカーソルを合わせてデータを入力しなければならない。

12.1.2 ADD（加算）文

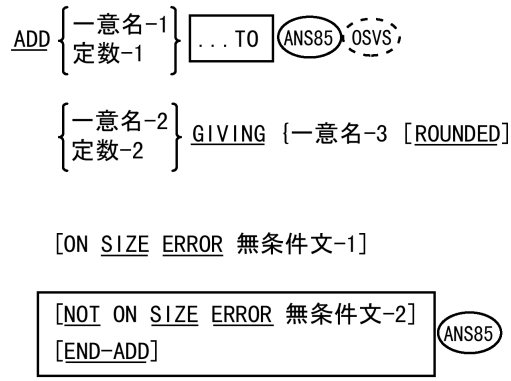
ADD（加算）文は、2つ以上の作用対象の和をとって、その結果を格納する。

一般形式

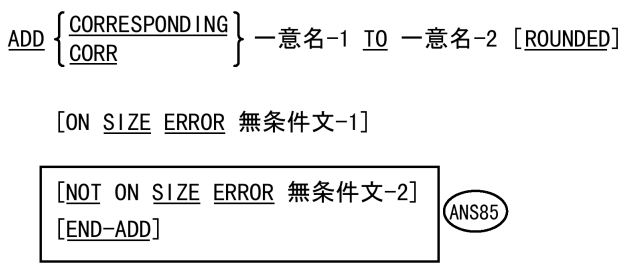
書き方 1



書き方 2



書き方 3



構文規則

すべての書き方

1. 書き方1および2において、各一意名は必ず基本数字項目を参照すること。ただし、書き方2においては、語GIVINGの後ろに続く各一意名は、基本数字項目の他に基本数字編集項目を参照してもよい。書き方3においては、各一意名は、必ず集団項目を参照すること。
2. 各定数は、数字定数とする。
3. 作用対象を合成したものは、 長さが18桁を超えてはならない。(前述の*算術文*の節を参照。)
    a. 書き方1では、作用対象の合成には、指定したすべての作用対象が用いられる。
    b. 書き方2では、作用対象の合成には、語GIVINGの後ろに続くデータ項目を除く、指定したすべての作用対象が用いられる。
    c. 書き方3では、作用対象の合成は、対応するデータ項目の組ごとに別々に行われる。
4. 書き方3では、CORRはCORRESPONDINGの省略形である。
5. (OSVS)(VSC2)(MF)数字データ項目または定数を指定できるところではどこも、浮動小数点数定数および浮動小数点数データ項目を使用できる。

一般規則

すべての書き方

1. 前述の*ROUNDED指定*、*ON SIZE ERROR指定*、*CORRESPONDING指定*、*算術文*、*作用対象の重なり*、*算術文における複数個の答え*の各節、および *COBOLの概念*の章の*明示範囲符と暗黙範囲符*、*範囲明示文*の各節を参照。
2. 書き方1では、語TOの前にある作用対象の値の和がとられ、その値が一意名-2の現在の値に加えられて、その結果が直ちに一意名-2に入れられる。語TOの後ろに続く各作用対象に対して、この処理が左から右方向に繰り返される。
3. 書き方2では、語GIVINGの前にある作用対象の値の和がとられ、その値が結果の一意名である一意名-3によって参照される各データ項目の新しい値として格納される。
4. 書き方3では、一意名-1に属するデータ項目が、一意名-2に属する対応するデータ項目に加算されて格納される。
5. (MF)計算過程で意味のある桁が失われることのないように、COBOLコンパイラによって十分な作業領域が確保される。

12.1.3 ALTER（変更）文

ALTER（変更）文は、あらかじめ定められている処理の順序を変更する。
(ANS85)ALTER文は、ANSI'85標準では廃要素に分類されており、ANSI標準の次回の全面改訂の際に、削除される予定である。
(MF)この構文は、Micro Focusに組み込まれているすべての方言で全面的に使用できる。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。
(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、この動詞は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内では、この文を使用するべきではない。

一般形式

```
ALTER 手続き名-1 TO [PROCEED TO] 手続き名-2
      [手続き名-3 TO [PROCEED TO] 手続き名-4] ...
```

構文規則

1. 各手続き名-1、手続き名-3などは、DEPENDING指定が含まれない単一のGO TO文からなる段落の名前である。
2. 各手続き名-2、手続き名-4などは、手続き部内の段落または節の名前である。

一般規則

1. ALTER文を実行すると、手続き名-1や手続き名-3などに指定された段落中のGO TO文がそれ以降に実行されたとき、それぞれ手続き名-2や手続き名-4などに制御が移されるようになる。独立の区分内の変更されたGO TO文は、状況によっては、初期状態に戻されることがある。（詳細については、***言語リファレンス - 追加トピック***中の*区分化の独立区分*の節を参照。）
2. 区分番号が50以上の節内のGO TO 文を、区分番号の異なる節内のALTER文によって参照してはならない。
   ALTER文のそれ以外の用法は、すべて有効である。たとえば、手続き名-1や手続き名-3などがオーバレイ可能な固定区分の中にあっても構わない。

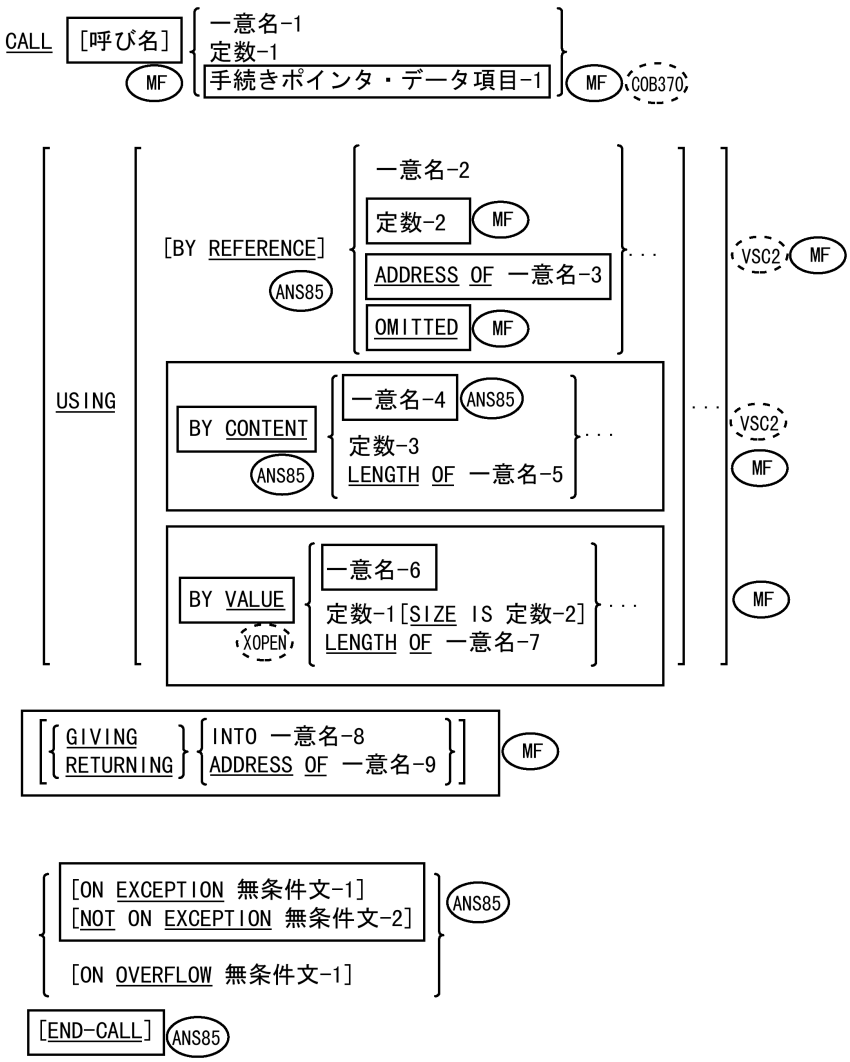
12.1.4 CALL（呼ぶ）文

CALL（呼ぶ）文は、実行単位中のある実行用プログラムから他の実行用プログラムに制御を移させる。
(MF)対象とする実行用プログラムの手続きを、番地を指定することによって、間接的に呼ぶことができる。データとデータへのポインタと手続きへのポインタを、パラメータとして呼ばれるプログラムに引き渡したり、呼ばれるプログラムから返してもらったりできる。パラメータ受渡し方式を制御することによって、COBOL以外の言語で書かれたプログラム手続きを呼ぶことができる。呼ぶばれたCOBOLプログラムは、再帰的に自分を呼ぶことができる。ただし、この場合は、局所記憶節が存在する必要がある。

例

1. CALL文の手続きポインタの使用例については、***言語リファレンス - 追加トピック***の*例*の章を参照。

一般形式



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - ALIGN - 01レベルまたは77レベルのデータ項目を割り付けるメモリーの境界を指定する。
    - CASE - 大文字と小文字を等しいとみなすかどうかを指定する。
    - DEFAULTCALLS - 省略時の呼出し方式を指定する。
    - DYNAM - CALL定数文によって呼び出されたプログラムを取り消しできるかどうかを指定する。

- FOLD-CALL-NAME - 呼ぶプログラムと呼ばれるプログラムの間でプログラム名に関して同じ大文字と小文字の使い方が同じでないときに、呼ばれた副プログラムが見つかるように保証するか否かを指定する。
- IBMCOMP - 語記憶モードを設定する。
- MAPNAME - プログラム名中の英字以外の文字の取扱いに影響する。

構文規則

星印（*）の付いている構文規則はCALL文およびCHAIN文に共通である。それらの規則を「CALL文」または「CHAIN文」に適用するさいには、「CALL文」または「CHAIN文」を読むべきである。

1. (MF)呼び名が必要なのは、呼び出す対象の実行用プログラムで用いられている呼出し方式と、COBOLシステムの省略時の呼出し方式とが異なる場合だけである。通常は、COBOLの省略時の呼出し方式は、COBOL以外の言語の主要な処理系のランタイム環境で使用されている方式と整合性がある。
   (MF)呼び名は特殊名段落中に定義しなければならない。その方法の詳細については、この章で前述した「特殊名段落」を参照。また、ランタイム環境においてサポートされている呼出し方式の詳細については、COBOLのシステム・ドキュメンテーションを参照。
2. * 一意名-1は英数字データ項目として定義しなければならない。その値がCOBOLのプログラム名であってもCOBOL以外のプログラム名であってもよいようにするためである。
3. * 定数-1、
   (MF)定数-2、
   (VSC2)(MF),および定数-3
   は文字定数でなければならず、表意定数であってはならない。
4. (MF)* 整数-1は符合付きであってよい。
5. * 一意名-2および一意名-4はファイル節、作業場所節、
   (MF)局所記憶節、
   通信節、連絡節のいずれかにデータ項目として定義されていなければならず、レベル01またはレベル77のデータ項目、
   (ANS85)または基本項目
   (MF)または任意の集団項目 でなければならない。
6. (VSC2)(MF)* 一意名-3は連絡節、
   (MF)局所記憶節または作業場所節にデータ項目として定義されていなければならず、
   (VSC2)(MF)レベル01またはレベル77のデータ項目、
   (MF)またはその他のレベルのデータ項目でなければならない。
7. (MF)* 一意名-5、一意名-6、一意名-7、一意名-8はいずれも、関数一意名であってはならない。
8. (MF)* GIVING句およびRETURNING句は、どちらを使用してもよい。
9. (MF)* 一意名-8は、4バイトのデータ項目とする。
10. (MF)一意名-9 は、連絡節中の01レベルまたは77レベルのデータ項目とする。

11. (MF)CALL文に定数-1を指定し（ただし、CALL文に一意名-1または手続きポインタ-1を指定した場合を除く）、かつ現在の翻訳単位中に呼出しプロトタイプ（プログラム名段落中にEXTERNAL句があるプログラム）が含まれていてその名前が定数-1に合致する場合、構文チェックの間に下記の項目の妥当性が検査される。
    - 必要なパラメータの数
    - パラメータの型
    - 呼出し方式

一般規則

星印（*）の付いている構文規則はCALL文およびCHAIN文に共通である。それらの規則を「CALL文」または「CHAIN文」に適用するさいには、「CALL文」または「CHAIN文」を読むべきである。

1. (MF)呼び名は、呼ばれるプログラムを呼び出すときにこのプログラムで使用する呼出し方式を識別する。呼ばれるプログラムで想定されている呼出し方式が呼び名で示されるものと異なると、COBOLシステムは障害を起こす可能性がある。
2. 定数-1、または一意名-1によって参照されるデータ項目の内容は呼ばれるプログラム(MF)または呼ばれる手続きの名前である。
   CALL文を含むプログラムは呼ぶプログラムである。呼ばれるプログラムがCOBOLプログラムである場合、定数-1、または一意名-1によって参照されるデータ項目の内容はプログラム名であって、呼ばれるプログラムのプログラム名段落に記述されているものでなければならない（この章で前述した「プログラム名段落」を参照、特に大文字と小文字の区別について）。
   (MF)あるいは、定数-1、または一意名-1によって参照されるデータ項目の内容は入口名であって、呼ばれるプログラムのENTRY文中に記述されているものでなければならない。しかも、呼ばれるプログラムはすでに呼び出されており、その後で取り消されていてはならない。別の方法として、呼ばれるプログラムの名前はそのプログラムの実行プログラム用のコードが収められているファイルを指す名前であってもよい。２つのCALL文の間でまたはCALL文とCANCEL文の間で同じファイルを指す別の名前を指定した場合にどのような結果になるかは、オペレーティング・システムによって異なる。呼ばれたプログラム、プログラム・ファイル、あるいはライブラリ・ファイル中に保持されているプログラムと指定された名前の照合方法の詳細については、インタフェースに関するCOBOLシステム・ドキュメンテーションを参照。
   (MF)呼ばれるプログラムがCOBOLプログラムではない場合のプログラム名または手続き名の構成に関する規則が、インタフェースに関するCOBOLシステム・ドキュメンテーションに記載されている。

3. (MF)手続きポインタ-1によって参照されるデータ項目の内容は呼ばれるプログラムの番地であり、そのプログラムはすでにメモリーにロードされていなければならない。呼ばれるプログラムがCOBOLプログラムである場合は、呼ばれるプログラムの番地が当初にENTRY指定を伴うSET文を用いて導出されており、その後でそのプログラムに対してCANCEL文が実行されていてはならない。呼ばれるプログラムがCOBOLプログラムではない場合は、呼ばれるプログラムの番地が当初にCOBOL以外のプログラムまたは手続きへのCALLを通じて導出されていなければならない。番地がNULLであるプログラムをCALLしようとすると、エラーとなる。
4. CALL文が実行されたときに、その対象のプログラムが実行可能な状態になると、そのプログラムに制御が移される。呼ばれたプログラムから制御が戻ると、ON OVERFLOW指定や
   (ANS85)ON EXCEPTION指定は
   指定されていても無視され、制御はCALL文の末尾に移される。
   (ANS85)あるいは、NOT ON EXCEPTION指令が指定されていると、制御は無条件文-2に移される。制御が無条件文-2に移されると、その無条件文-2の中に指定されている各文に関する規則に従って、プログラムの実行が続けられる。制御の明示移行を引き起こす手続き分岐文または条件文が実行されると、その文に関する手続きに従って制御が移される。これ以外の場合は、無条件文-2の実行が終了すると制御はCALL文の末尾に移される。
5. CALL文が実行されたときに、対象のプログラムがすでにメモリーにロードされていると、そのプログラムは実行可能な状態にされる。そのプログラムがまだメモリーにロードされていないと、そのプログラムを動的にロードする試みがなされる（インタフェースに関するCOBOLシステム・ドキュメンテーションを参照）。対象のプログラムの所在を動的に把握でき、メモリーに連続した十分な空き領域があると、そのプログラムは動的にロードされて実行可能な状態にされる。
   CALL文に指定されたプログラムがメモリーにロードされていないか、動的なロードに利用できないか、あるいは動的なロードに利用できるがメモリーに十分な空き領域がない場合には、その時点ではそのプログラムを実行可能にすることはできない。その場合、下記の２つのうちのどちらかの措置が取られる。
   a. CALL文の中にON OVERFLOW
      (ANS85)またはON EXCEPTION
      指定がある場合には、無条件文-1に制御が移される。それから、無条件文-1に指定されている各文の規則に応じて、処理が続行される。制御を明示的に移す手続き分岐または条件文が実行されると、その文の規則に応じて制御が移される。それ以外の場合には、無条件文-1の実行が完了すると、CALL文の最後に制御が移され、
      (ANS85)NOT ON EXCEPTION指定はあっても無視される。
   b. (ANS85)ON OVERFLOW指定またはON EXCEPTION指定がない場合、NOT ON EXCEPTION指定はあっても無視される。
      (MF)実行単位中の開かれているファイルに関して、明示的なCLOSE文なしにプログラムの実行が停止され、ランタイム・エラーが表示される。

6. (MF)呼ばれるプログラムを実行できるようにするさいに、その所在が把握されるまで、呼ばれるプログラムがCOBOLプログラムであるのかないのか、あるいは手続きであるのかは、実行単位には分からない。一般には、プログラム名の形式から、COBOLプログラムであるか否かを判定することはできない。COBOLプログラムとCOBOLでないプログラムの間、またはCOBOLでない２つのプログラムの間で、プログラム名が同じであるとエラーとなる。

   (ANS85)実行単位内の2つ以上のプログラムに、同じプログラム名をもたせることができる。呼ぶプログラム内でこのようなプログラム名を参照したとき、どちらのプログラムを指すかは、プログラム名の範囲の適用規則に従う。*COBOL言語の概念*の章の*プログラム名の規則*の節を参照。

   (ANS85)たとえば、CALL文の中に指定されたプログラム名と同じ名前のプログラムが、実行単位中に2つあるとする。

   a. それらの2つのプログラムのうちの1つは、そのCALL文が含まれる独立のプログラムに直接的または間接的に含まれるか、またはそのCALL文が含まれるプログラム自体が、直接的または間接的に含まれる独立のプログラムに直接的または間接的に含まれなければならない。かつ、
   b. もう1つのプログラムは、別の独立したプログラムでなければならない。

   (ANS85)この例で、参照するプログラムを見つける仕組みは、下記のとおり。

   a. CALL文中に指定された名前をもつ2つのプログラムの一方が、そのCALL文が含まれるプログラム中に直接的に含まれるならば、そのプログラムが呼ばれる。
   b. CALL文中に指定された名前をもつ2つのプログラムの一方が共通属性をもち、そのCALL文が含まれるプログラムが直接的または間接的に含まれる独立のプログラムに直接的に含まれるならば、その共通プログラムが呼ばれる。ただし、呼ぶプログラムがその共通プログラムに含まれる場合は例外である。
   c. それ以外の場合は、別のプログラムが呼ばれる。

7. 呼ばれたプログラムがCOBOLプログラムで、初期属性をもたない場合、そのプログラムおよびその中に直接的または間接的に含まれる各プログラムは、実行単位の中で最初に呼ばれたとき、および呼ばれたプログラムが取り消された後ではじめて呼ばれたときに、初期状態に置かれる。

   呼ばれたプログラムがCOBOLプログラムでない場合、実行単位の中で最初に呼ばれたときにだけ初期状態に置かれる。このようなプログラムの取り消しは効果がない。

   それ以外の場合に呼ばれたときは、呼ばれるプログラムおよびその中に直接的または間接的に含まれる各プログラムの状態は、前回に呼ばれたときのままである。COBOL以外の言語の処理系の中には、意味合いが違っているものがある。可能な場合には、インタフェースに関するCOBOLシステム・ドキュメンテーションに、その詳細が記載されている。

8. (ANS85)呼ばれたプログラムが初期属性をもつ場合、そのプログラムおよびその中に直接的または間接的に含まれる各プログラムは、実行単位の中で呼ばれるたびに初期状態に置かれる。呼ばれたプログラムが初期状態にあるか使用され終わった状態にあるかは、DYNAMコンパイラ指令の設定に左右される。

9. (ANS85)呼ばれるプログラムの内部ファイル結合子に関連するファイルは、プログラムが初期状態にあるときは、まだオープンされていない。前述の*プログラムの初期状態*の節を参照。
   (ANS85)それ以外の場合に呼ばれたときは、それらのすべてのファイルの状態と位置付けは、前回に呼ばれたときのままである。
10. (ANS85)プログラムを呼ぶ過程または呼んだプログラムから戻る過程で、外部ファイル結合子に関連するファイルの状態と位置付けが変えられることはない。
11. 呼ばれるプログラムがCOBOLプログラムである場合、その手続き部の見出し
    (OSVS)(VSC2)(MF)またはENTRY文
    の中にUSING指定がある場合にのみ、CALL文にUSING指定が含められる。その場合、各USING指定内の作用対象の数は同じでなければならない。
    (MF)作用対象の数が同じである必要はない。
12. (MF)呼ばれるプログラムがCOBOLプログラムではない場合、そのプログラムに１つ以上のパラメータが宣言されている場合にのみ、CALL文にUSING指定が含められる。その場合、USING指定内の作用対象の数は呼ばれるプログラム中のパラメータの数と同じでなければならない（COBOL以外の言語の処理系の中には、作用対象の数が同じでなくともよいものがある）。COBOL言語では、データ項目の番地の桁寄せに制約はない。他方、COBOL以外の言語では、通常は番地について前提があり、桁寄せされていないデータ項目が参照されると何らかのエラーを起こす。桁寄せは下記の措置のうちのいくつかを用いて実現される。
    - 集団項目に無名項目を追加する
    - ALIGNコンパイラ指令を使用するとともに、USING指定中の作用対象を01レベルまたは77レベルのデータ項目にする
    - SYNCHRONIZED句とIBMCOMPコンパイラ指令を併用する
13. * CALL文
    (MF)またはCHAIN文の
    USING指定および呼ばれるプログラムの手続き部の見出し
    (MF)または呼ばれるプログラムがCOBOLでない場合はENTRY文またはパラメータ・リストの中の対応するUSNIG指定に中に作用対象が置かれている順序によって、
    呼ぶプログラムと呼ばれるプログラムで使用されているデータ名の対応関係が決まる。この対応は位置で決まるのであって、名前が等しいことによるのではない。CALL文
    (MF)またはCHAIN文の
    USING指定の中の最初の作用対象が呼ばれるプログラムのUSNIG指定またはパラメータ・リストの最初の作用対象に対応し、それらの２番目同士が対応する、といったぐあいである。
    (MF)両方の作用対象の数が異なるために、完全には対応がとれないことがある。その場合、対応する相手がなくて残っている作用対象がCALL文またはCHAIN文の側にある場合、それらは無視される。残っている作用対象が手続き部の見出しまたはENTRY文またはパラメータ・リストの側にある場合、それらは呼ばれるプログラムの中で参照されてはならない。位置による対応は、呼び名によって暗黙的に示される呼出し方式によって変わることがある。

指標名の場合は、そのような対応関係は確立されない。呼ぶプログラムと呼ばれるプログラムの指標名は常に別々の指標を参照する。

14. (ANS85)* CALL文のUSING指定の中で参照されるパラメータの値は、そのCALL文が実行された時点で、呼ぶプログラムにとって利用可能になる。
15. (ANS85)* BY CONTENT、BY REFERENCE、
    (MF)BY VALUE
    (ANS85)の各指定はその後ろに続くパラメータに効力を及ぼすが、別のBY CONTENT、BY REFERENCE、
    (MF)またはBY VALUE
    (ANS85)の指定が出てきたところでその効力は停止する。
    (ANS85)最初のパラメータの前にBY CONTENT、BY REFERENCE、
    (MF)または BY VALUE
    (ANS85)のどの指定もないと、BY REFERENCE指定があるものと想定される。
16. * パラメータにBY REFERENCE 一意名-2 指定を
    (ANS85)明示的または
    暗黙的に指定すると、実行用プログラムは、呼ばれるプログラム中の対応するデータ項目が、呼ぶプログラム中のデータ項目と同じ記憶領域を占めるかのように動作する。呼ばれるプログラム中のデータ項目のデータ記述には、呼ぶプログラム中の対応するデータ項目と同じ文字数を指定する。
    (VSC2)(MF)BY REFERENCE ADDRESS OF指定を明示的または暗黙的に指定すると、プログラムの動作は次のようになる。具体的には、USAGE POINTERを用いて追加のデータ項目が指定され、「SET データ項目 TO ADDRESS OF 一意名-3」文によって取得された値がBY REFERENCEによってそのデータ項目に渡されたかのように動作する。
    (VSC2)(MF)一意名-3が連絡節にあってそのレベル番号か01または77以外であるか、または作業場所節にある場合、そのデータ項目がBY CONTENTによって渡され、一意名-3の番地を呼ばれるサブプログラムによっては変更できないことに等しい。
    (MF)「BY REFERENCE 一意名-2」を明示的または暗黙的に指定すると、オブジェクト・プログラムは定数-2を定数-3のために記述されたものとして処理する。
17. (MF)(XOPEN)* BY CONTENT指定を明示的または暗黙的に指定すると、CALL文中のUSING句内で参照されたパラメータの値は、呼ばれたプログラム側では変更できない。しかし、呼ばれたプログラムは、その手続き部の見出しの中で、対応するデータ名によって参照しているデータ項目の値を変更できる。CALL文のBY CONTENT指定中の各パラメータに関するデータ記述は、手続き部の見出しの中のUSING句の中の対応すパラメータに関するデータ記述と同じにする。つまり、変換も拡張も切捨てもあってはならない。

(MF)パラメータに関してBY CONTENT指定を明示的または暗黙的に指定すると、オブジェクト・プログラムの動作は次のようになる。具体的には、追加のデータ項目が宣言され、そのデータ項目がBY REFERENCE指定の中でパラメータとして使用されるかのように動作する。一意名-4を指定すると、追加データ項目の暗黙的なデータ記述とその内容の両方が、一意名-4のものと同じとなる。一意名-3を指定すると、追加データ項目の暗黙的なデータ記述は、サイズと内容が一意名-3と同じ英数字データ項目に等しくなる。「LENGTH OF 一意名-5」を指定すると、追加データ項目のデータ記述は、形式がPIC S9 (9) USAGE BINARYで内容が一意名-5に割り当てられた記憶領域のバイト数に設定されたものと等しくなる。

18. (MF)* BY VALUE指定を明示的または暗黙的に指定すると、CALL文のUSING指定の中で参照されているこのパラメータの値を、呼ばれるプログラムは変更できない。それにもかかわらず、呼ばれるプログラムはその手続き部の見出しの中の対応するデータ名によって参照されるデータ項目の値を変更することはできる。概念的には、システム領域（通常は「スタック」）に追加のデータ項目が宣言され、それがパラメータを渡すためにCOBOL以外の言語に利用可能であり、その記憶領域が呼ばれるプログラム中のデータ項目と同じであるかのように、オブジェクト・プログラムは動作する。一意名-6を指定すると、追加データ項目の暗黙的なデータ記述とその内容の両方が、一意名-6のものと同じとなる。整数-1を指定すると、追加データ項目の暗黙的なデータ記述は、符合付き数字項目でUSAGE COMP-5に等しくなる。その記憶領域内でのバイト数は、整数-2の指定があればその値によって示され、指定がなければ４バイトとなる。「LENGTH OF 一意名-7」を指定すると、追加データ項目のデータ記述は、形式がPIC S9 (9) USAGE BINARYで内容が一意名-7に割り当てられた記憶領域のバイト数に設定されたものと等しくなる。

(MF)CALL文のBY VALUE指定中の各パラメータには概念的な追加データ項目があり、呼ばれるプログラム中の手続き部の見出しの中のUSING指定の中に、対応するパラメータが宣言されている。呼ばれるプログラム中のそれらの各パラメータのデータ記述は、対応する概念的な追加データ項目のデータ記述と同じでなければならない。つまり、変換も拡張も切捨てもあってはならない。それに加えて、概念的なデータ項目の暗黙のサイズがシステム領域の最大サイズ（通常は４バイト）を超えてはならない。そうでないと、システムが壊滅的なエラーを起こすことがある。

(MF)呼ばれるプログラムがCOBOLプログラムである場合、CALL文のBY VALUE指定中の各パラメータに対応するパラメータが手続き部の見出しのUSING指定中に存在し、それにBY VALUE指定が明示的または暗黙的に指定されていなければならない。

(MF)呼ばれるプログラムがCOBOLプログラムではない場合に、どういうときにBY VALUE指定を使用する必要があるかの詳細については、インタフェースに関するCOBOLシステム・ドキュメンテーションを参照。

19. (MF)呼ばれるプログラムから呼ぶプログラムにデータを渡す手段には、呼ぶプログラムのUSING指定中に宣言されているパラメータを使用する方法と、戻り値を使用する方法とがある。呼ばれるプログラムはEXIT GIVING文またはGOBACK GIVING文によって値を戻す。呼ばれたプログラムから呼んだプログラムに制御が戻されると、戻り値がシステム領域に収められる。その領域は一般に、値を戻すために、COBOL以外のプログラムによって利用可能である。
    (MF)システム領域のサイズ（通常は最低で４バイト）はCOBOLシステムの外で決定され、戻り値が取りうる最大値の制約要因となる。ただし、戻り値がシステム領域の最大サイズに満たないことがある。制御が戻された後で、呼んだプログラムで戻り値を利用できるようになる。それは、暗黙的に特殊なレジスタであるRETURN-CODEに収められていることもあれば、GIVING指定に指定したデータ項目に明示的に収められていることもある。
20. (MF)GIVING指定を書かず、呼出し方式に特殊レジスタのRETURN-CODE（前出の「特殊名段落」中の「呼出し方式」を参照）を更新するように指定されている場合、オブジェクト・プログラム動作は次のようになる。具体的には、システム領域がそのサイズのままCOBOLの数字データ項目でUSAGE COPM-5が指定されているものとして定義され、そのシステム領域を送出し側とし特殊レジスタのRETURN-CODEを受取り側として、MOVE文を実行したかのように動作する。（RETURN-CODE詳細については、「COBOL言語の概念」の章の「特殊レジスタ」の項を参照。）
21. (MF)呼ぶプログラム中にGIVING INTO指定を書いた場合には、戻り値をシステム領域に収めるのに必要なのと同じ長さを一意名-8に指定しなければならない。また、一意名-8の型と用途は、呼ばれるプログラムからの値の返され方によって暗黙的に示される型と用途に等しくなければならない。呼んだプログラムに制御が戻されると、戻り値が一意名-8に収められている。
22. (MF)呼ぶプログラムにGIVING ADDRESS OF指定を書いた場合、呼ばれるプログラムが返す値は明示的または暗黙的にUSAGE POINTERと指定されたものでなければならない。オブジェクト・プログラム動作は次のようになる。具体的には、システム領域がUSAGE POINTERと指定されたCOBOLのデータ項目として宣言され、そのシステム領域を最初の作用対象とし「ADDRESS OF 一意名-9」を２番目の作用対象として、SET文を実行したかのように動作する。
23. (MF)呼ばれるプログラムが値を返さない場合にGIVING指定を書くと、エラーとなる。
24. 呼ばれたプログラムの中に、CALL文が含まれていてもよい。 しかし、
    (MF)局所記憶節が宣言されていない場合、
    呼ばれたプログラムは、呼んだプログラムを直接的または間接的に呼ぶCALL文を実行しない。宣言部分の範囲内でCALL文が実行されるとき、そのCALL文は、制御が移されたがまだ実行を終了していない呼ばれたプログラムを直接的にも間接的にも参照できない。
25. (ANS85)END-CALL指定は、CALL文の範囲を区切る。
26. (VSC2)(MF)USAGE POINTER
    (MF)またはUSAGE PROCEDURE-POINTER

(VSC2)(MF)が明示的または暗黙的に指定されており、呼ばれるプログラム中のデータ項目
(MF)または手続き
(VSC2)(MF)の番地から当初に導出された値は、その呼ばれたプログラムに対してCANCEL文が実行された後では無効になるので、使用してはならない。

27. (MF)使用されないパラメータをサブプログラムに渡すために、OMITTED指定を使用することができる。その場合、ダミーのデータ項目を宣言する必要はない。
28. (MF)OMITTEDパラメータはBY REFERENCEパラメータであり、呼ばれるプログラムに空の番地を渡すことに相当する。それはBY VALUE 0 SIZE 4と指定するのと同じであるが、BY REFERENCEパラメータとして呼び出しプロトタイプの型チェックを受ける。
29. (MF)OMITTEDの直前にBY REFERENCEを指定しないと、以降のパラメータはその前に有効であった呼出し方式に従って渡される。省いたパラメータはBY VALUE 0 SIZE 4と指定したかのように扱われる。

12.1.5 CANCEL（取消）文

CANCEL（取消）文は、参照するプログラムを次回に呼んだときに必ず初期状態にあるようにする。

一般形式

CANCEL { 一意名-1 / 定数-1 } ...

構文規則

1. 定数-1は、文字定数とする。
2. 一意名-1は、英数字データ項目を参照する。

一般規則

1. 定数-1または一意名-1によって参照されるデータ項目の内容は、取り消す対象のプログラムを指さなければならない。その名前は取消し対象のプログラムのプログラム名段落に記されているプログラム名でなければならない（この章で前述した「プログラム名段落」、特に大文字と小文字の区別に関する記述を参照）。別の方法として、その名前に取消し対象のプログラムの実行プログラム用のコードが収められているファイルを指定してもよい。CANCEL文の中で同じファイルを指す別の名前を指定した場合にどのような結果になるかは、オペレーティング・システムによって異なる。前に呼ばれたプログラム、プログラム・ファイル名、あるいは登録集ファイル中のプログラムと指定された名前の照合方法の詳細については、COBOLシステム・ドキュメンテーションを参照。
   その名前がENTRY文に含まれている入口名を指している場合、結果はどうなるか分からない。
2. 明示的または暗黙的なCANCEL文が実行された後では、そのプログラムとそれが属する実行単位との論理関係はなくなる。明示的または暗黙的なCANCEL文の対象であったプログラムが、同じ実行単位の中で後からまた呼ばれたときには、そのプログラムは初期状態にある。前述の*プログラムの初期状態* の節を参照。
3. 他のプログラム中のCANCEL文の中に指定されたプログラムは、そのプログラムから呼べるものとする。前述の*名前の適用範囲*および *CALL*文の節を参照。
4. (ANS85)明示的または暗黙的なCANCEL文が実行されると、その対象となったプログラム中に含まれるプログラムもすべて取り消される。その結果は、取り消されたプログラムの中に含まれる各プログラムに対して、それらが現れる順番と逆の順番で、有効なCANCEL文を実行したことに等しい。
5. CANCEL文中に指定したプログラムは、呼ばれたがまだEXIT PROGRAM文
   (OSVS)(VSC2)(MF)またはGOBACK文
   を実行していないプログラムを、直接的にも間接的にも指してはならない。
6. 取り消されたプログラムとの論理関係は、その後でそのプログラムを対象にしてCALL文を実行することによってだけ、成り立つ。
7. 呼ばれたプログラムは、次の場合に取り消される。1つは、CANCEL文の作用対象として指定することである。もう1つは、対象とするプログラムが属する実行単位が終了することである。
   (ANS85)または、呼ばれたプログラムが初期属性をもつ場合には、その中でEXIT PROGRAM文を実行することである。
8. 実行単位の中でまだ呼ばれていないプログラム、呼ばれたが取消し済みのプログラム、またはCOBOL以外のプログラムを指定して、明示的または暗黙的なCANCEL文を実行しても、何の措置もとられない。 制御はCANCEL文の後ろに続く、次の実行可能文に移される。
9. (ANS85)プログラム内に記述されている外部データ・レコード中のデータ項目の内容は、そのプログラムが取り消されても変わらない。

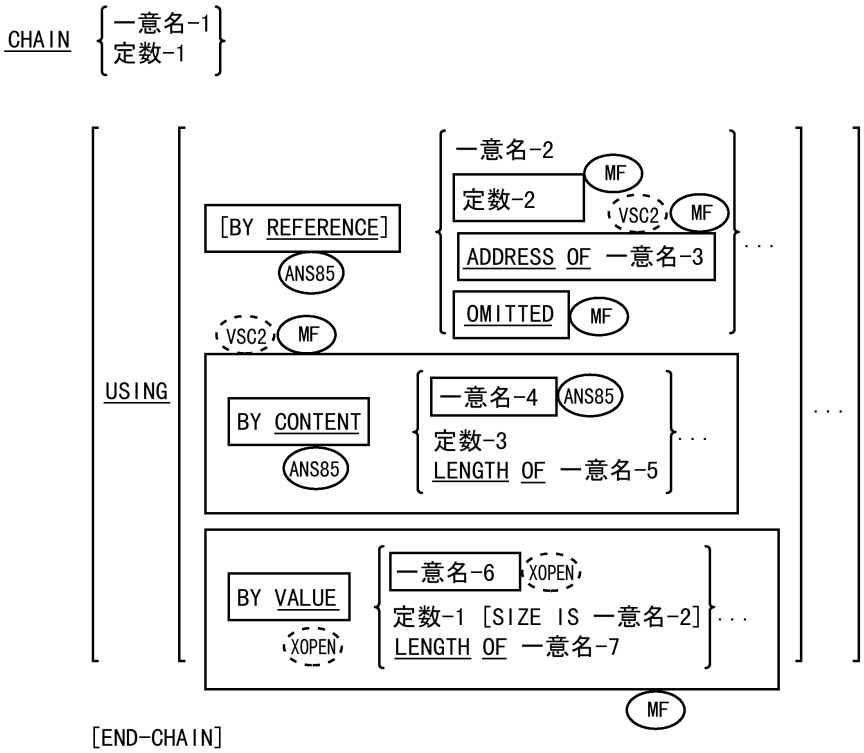
10. (ANS85)明示的または暗黙的なCANCEL文の実行中に、暗黙のCLOSE文が実行される。具体的には、CANCEL文の対象プログラム中の内部ファイル結合子に関連する、開かれているファイルがすべて閉じられる。これらのファイルに関連するUSE手続きがあっても、実行されない。
11. CANCEL文は、副プログラムを初期状態に戻すかそのままの状態で残すかを決める。CANCEL文は、必ずしも、取り消された副プログラムが記憶されていた領域を解放するとは限らない。しかし、オペレーティングシステムの中には取り消された副プログラムが記憶されていた領域を解放するものが多い。その副プログラムの外部ファイルおよび外部データ領域は、CANCEL文によって解放されることはない。

12.1.6 CHAIN（連鎖）文

(MF)

CHAIN（連鎖）文は、実行単位内のある実行用プログラムから他の実行用プログラムへ制御を移す。ただし、後から制御が戻されることを前提としない。連鎖先のプログラムは、新しい実行単位中の主プログラムであるようにみえる。

一般形式



構文規則

1. CHAIN文の構文はCALL文の構文に似ている。CALL文の項に示したように、この２つの文は共通の構文規則に従う。CHAIN文の一般形式に示されている一意名、定数、整数はすべて、CALL文の一意名、定数、整数に対応する。

一般規則

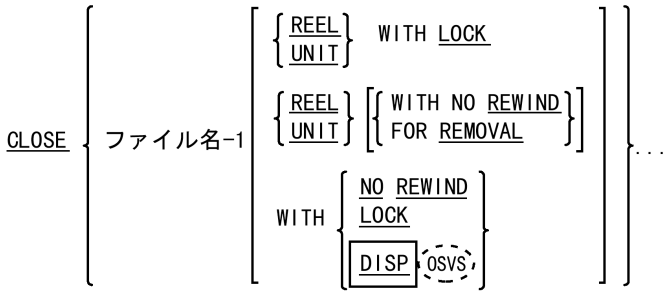
1. 定数-1、または一意名-1によって参照されるデータ項目の内容は連鎖先のプログラムの名前である。CHAIN文が記述されているプログラムは連鎖するプログラムである。連鎖先のプログラムはCOBOLプログラムでなければならず、かつ入れ子になっていてはならない。連鎖先のプログラムの名前には、そのプログラムのプログラム名段落中に記述されている名前を指定しなければならない（大文字と小文字の区別に関しては、この章で前述した「プログラム名段落」を参照）。別の方法として、連鎖先のプログラムの名前は、連鎖先のプログラムの実行プログラム用のコードが収められているファイルを指す名前であってもよい。前に連鎖されたプログラム、プログラム・ファイル名、あるいは登録集ファイル中のプログラムと指定された名前の照合方法の詳細については、インタフェースに関するCOBOLシステム・ドキュメンテーションを参照。
2. CHAIN文を実行したときに、CHAIN文に指定した連鎖先のプログラムが実行可能になれば、そのプログラムに制御が移される。連鎖するプログラムは取り消される。また、連鎖するプログラムによって呼ばれたCOBOLプログラムがあれば、それらも取り消される。連鎖先のプログラムは連鎖するプログラムの制御下にはない。したがって、連鎖されたプログラム内でEXIT PROGRAMを実行しても、効果はない。
3. CHAIN文を実行したときに、CHAIN文に指定した連鎖先のプログラムが実行可能にならなければ、致命的なランタイム・エラーとされる。
4. 連鎖されたプログラム内にCHAIN文があってもよい。連鎖されたプログラム内のCHAIN文では、そのプログラム自体以外の任意のプログラムを参照できる。
5. CHAIN文の形式はCALL文の形式に似ている。CALL文の項に示したように、この２つの文は共通の一般規則に従う。CHAIN文の一般形式に示されている一意名、定数、整数はすべて、CHAIN文の一意名、定数、整数に対応する。また、呼ぶプログラムおよび呼ばれるプログラムという表現が用いられているときは、連鎖するプログラムおよび連鎖先のプログラムと同等であるとみなす。
6. 連鎖処理によって、CHAIN文のUSING指定内で参照されている各データ項目の値が、連鎖先のプログラムの手続き部の見出しのUSING指定中に書かれている対応するデータ名によって参照される、データ項目に転記される。
7. USING指定中の作用対象として渡されたUSAGE POINTERまたはUSAGE PROCEDURE-POINTERの指定がある任意のデータ項目の値は、COBOLデータ項目から当初に導出された番地を表していてはならない。
8. END-CHAIN指定はCHAIN文の範囲を区切る。
9. コンパイラ指令のINTLEVEL"1"を使用した場合は、USING指定を用いてはならない。

12.1.7 CLOSE（閉じる）文

CLOSE（閉じる）文は リール/ユニットおよびファイルの処理を終了させる（ファイルのクローズ）。さらに、可能ならば、巻戻しやロックや取外し処理を行うこともできる。

一般形式

書き方 1（レコード順ファイル）



書き方 2（ (MF)行順、相対および索引ファイル）

```
CLOSE ファイル名-1 [WITH LOCK] [ファイル名-2 [WITH LOCK]]...
```

構文規則

すべての書き方（すべてのファイル）

1. CLOSE文で参照されるファイルは、すべてが同じ構成またはアクセスである必要はない。

書き方 1（レコード順ファイル）

2. (MF)REEL および UNITを指定できるのは、SELECT句に MULTIPLE REELまたは MULTIPLE UNIT を指定してあるファイルに対してだけである。マルチユニット・ファイルを認識しないランタイムシステムにおいては、CLOSE REEL文およびCLOSE UNIT文は" 空" 文である。つまり、これらの文は注記になる。CLOSE REEL文またはCLOSE UNIT文によって閉じた装置上には、他に開いているファイルはないことが重要である。
   (OSVS)CLOSE REEL WITH LOCK文および CLOSE UNIT WITH LOCK文は、CLOSE REEL FOR REMOVAL と同義語として扱われる。
3. (OSVS)DISP 指定は、テープ・ファイルに対してだけ適用できる。
   (MF)DISP指定は注記になる。

一般規則

すべての書き方（すべてのファイル）

1. CLOSE 文は、開かれているファイルに対してだけ実行できる。
2. CLOSE文の実行が正常に終了すると、ファイル名-1のレコード領域は使用できなくなる。CLOSE文の実行が不成功に終わった場合は、レコード領域はどうなるかわからない。
3. (ANS85)STOP RUN（実行停止）文が実行されたときに開かれているファイルがあると、そのファイルは閉じられる。呼ばれたプログラムの中でファイルが開かれ、後でそのプログラムの中でCANCEL（取消し）文が実行された場合、開かれたままのファイルがあれば閉じられる。
4. (MF)CLOSE文の実行が正常に終了すると、閉じられたファイルに関して実行単位が保持していた レコードロックまたは ファイルロックはすべて解除される。
5. CLOSE文が実行されると、ファイル名-1に関する入出力状態の値が更新される。(前述の*入出力状態*の節を参照。)
6. CLOSE文の書き方とファイルの種類ごとに、実行結果をまとめて、表12-1に示す。

表12-1: CLOSE 文の書き方とファイルの種類の関係

CLOSE文の書き方	ファイルの種類			
	非リール/ユニット	レコード順単一リール/ユニット	レコード順複数リール/ユニット	非レコード順単一/複数リール/ユニット
CLOSE	C	C, G	A, C, G	C
CLOSE WITH LOCK	C, E	C, E, G	A, C, E, G	C, E
CLOSE WITH NO REWIND レコード順だけ	(ANS85)C, H	B, C	A, B, C	X
CLOSE REEL/UNIT レコード順だけ	(ANS85)F	(ANS85)F, G	F, G	X
CLOSE REEL/UNIT FOR REMOVAL レコード順だけ	(ANS85)F	(ANS85)D, F, G	D, F , G	X
CLOSE REEL/UNIT WITH NO REWIND レコード順だけ	X	X	F, B	X

上の表の中に示した記号の意味を以降に説明する。入力ファイルか出力ファイルか入出力ファイルかによって意味が異なるときは、それぞれに分けて説明する。分けていなければ、説明は入力ファイルと出力ファイルと入出力ファイルのすべてに当てはまる。

A. **前のリール/ユニットは影響を受けない**

入力ファイルおよび入出力ファイル:

ファイル中の現在のリール/ユニットよりも前にあるリール/ユニットがすべて処理される。ただし、既に CLOSE REEL/ UNIT文が実行されているものを除く。現在のリール/ユニットがファイル中の最後のものでない場合、その後ろのリール/ユニットは処理されない。

出力ファイル:

ファイル中の現在のリール/ユニットよりも前にあるリール/ユニットがすべて処理される。ただし、既にCLOSE REEL/UNIT文が実行されているものを除く。

B. **現在のリールを巻き戻さない**

現在のリール/ユニットはそのままの位置にとどめられる。

C. **ファイルを閉じる**

入力ファイルおよび入出力ファイル(順呼出し法):

ファイルの末尾に達しており、かつファイル記述に ラベル・レコードを指定してある場合、オペレーティングシステムのラベル方式に従って、ラベルの処理が行われる。ラベル・レコードを指定しているのにラベル・レコードが存在しない場合、またはラベル・レコードを指定していないのにラベル・レコードが存在する場合、CLOSE文の動作はどうなるかわからない。ファイルの末尾に達しているが、ファイル記述にラベル・レコードが指定されていない場合、ラベルの処理は行われないが、 ランタイムシステムに依存するその他の閉じる処理は行われる。ファイルの末尾に達していない場合、ランタイムシステムに依存する閉じる処理は行われるが、終了ラベルの処理は行われない。

入力ファイルおよび入出力ファイル(乱呼出し法、動的呼出し法); 出力ファイル(乱呼出し法、動的呼出し法、順呼出し法):

ファイル記述にラベル・レコードを指定している場合、オペレーティングシステムのラベル方式に従ってラベルの処理が行われる。ラベル・レコードを指定しているのにラベル・レコードが存在しない場合、またはLABEL RECORD句を指定していないのにラベル・レコードが存在する場合、CLOSE文の動作はどうなるかわからない。ファイル記述にラベル・レコードを指定していない場合、ラベルの処理は行われないが、ランタイムシステムに依存するその他の閉じる処理は行われる。

D. **リール/ユニットを取り外す**

REELまたはUNITを指定しないでCLOSE文をファイルに対して実行し、続いてそのファイルに対してOPEN文を実行すると、そのファイル内のリールまたはユニットを適切な順番で再度呼び出すことができる。

E. **ファイルをロックする**

この実行単位の実行中は、このファイルを再び開くことはできない。

F. **リール/ユニットを閉じる**

入力ファイルおよび入出力ファイル(リール/ユニット媒体):

1. (ANS85)現在のリール/ユニットが該当するファイルの最後または唯一のリール/ユニットである場合、リール/ユニットの交換は行われない。現在のボリューム・ポインタは変更されず、ファイル位置指示子には次のリール/ユニットが存在しないと設定される。
2. 該当するファイルに他のリール/ユニットが存在する場合、リール/ユニットの交換が発生し、現在のボリューム・ポインタはそのファイル中の次のリール/ユニットを指すように更新される。その後標準開始リール/ユニット・ラベル手続きが実行されて、ファイル位置指示子が新しい現行ボリューム上に存在する最初のレコードの番号よりも1小さく設定される。現行ボリュームにデータ・レコードが存在しなければ、次のリール/ユニットの交換が発生する。

出力ファイル(リール/ユニット媒体):

下記の操作が行われる。

1. 標準終了リール/ユニット・ラベル手続きの実行
2. リール/ユニットの交換
3. 標準開始リール/ユニット・ラベル手続きの実行
4. そのファイルに対して次に WRITE WRITE文を実行すると、次の論理レコードはそのファイルの次のリール/ユニットに書き出される。

入力ファイル、入出力ファイル、出力ファイル(非リール/ユニット媒体):

(ANS85)このCLOSE文の実行は、正常に終了したものと扱われる。実際には、ファイルは開かれたままであり、ファイル位置指示子は変更されず、一般規則4に示された以外の処理は何も行われない。

G. **巻き戻す**

現在のリールまたは類似の装置は、物理的始点に位置付けられる。

H. **任意指定は無視される**

(ANS85)任意指定が何もされていないように、CLOSE文は実行される。

X. **不正**

CLOSE文の指定とファイルの種類のこの組合わせは正しくない。実行結果がどうなるかはわからない。

書き方 1 (レコード順ファイル)

以下の記述において別途断らないかぎり、 "リール" と "ユニット" という語は同義であり、CLOSE文の中でまったく同じように使うことができる。 順編成の大記憶ファイルの取扱いは、論理的にはテープまたは類似の順媒体と同じである。

7. 様々な記憶媒体に適用される種々の CLOSE の効果を示すために、すべてのファイルを下記の種類に区分する。
   a. 非リール/ユニット:巻戻しやリール/ユニットの概念が意味をもたない入出力媒体を使用したファイル。
   b. 順単一リール/ユニット:1つのリール/ユニットに完全に格納される順ファイル。

c. 順複数リール/ユニット:複数のリール/ユニットにまたがって格納される順ファイル。

8. 環境部のファイル管理段落中でファイルにOPTIONAL指定をしているが、そのファイルが存在しない場合には、そのファイルに関しては標準のファイル終了処理は行われない。
9. あるファイルに対してREELもUNITも指定せずに CLOSE 文を実行すると、以降そのファイルを対象とする入出力文（SORT文またはMERGE文でUSINGまたはGIVINGを指定したものは除く）は明示的にも暗黙的にも実行できなくなる。ただし、そのファイルに改めてOPEN文を実行すれば、再び入出力文を実行できるようになる。
10. The ファイルが存在する媒体に対して適用しなければ、WITH NO REWIND指定もFOR REMOVAL指定も実行時に効力をもたない。
11. If WITH を指定すると、そのファイルはその実行単位の現在の実行中には再度開くことはできなくなる。この点を除けば、通常のCLOSEと同じである。
    (MF)(この機能は、ファイルを共有するときのレコードまたはファイルのロックとは関係がない。)

すべての書き方（すべてのファイル)

12. あるファイルに対してCLOSE文を実行すると、以降そのファイルを対象とする文は、
    (MF)DELETE FILE文以外には、
    明示的にも暗黙的にも実行できなくなる。ただし、そのファイルに改めてOPEN文を実行すれば、再びファイルを操作する文を実行できるようになる。
13. WITH LOCKを指定すると、そのファイルはその実行単位の現在の実行中には再度開くことはできなくなる。

12.1.8 COMMIT（更新確定）文

(MF)

The COMMIT（更新確定）文は、該当する実行単位によってロックされている、ファイル中のすべてのレコードロックを解除する。SELECT文のWITH...ROLLBACK句を単なる注記としてではなく実際にサポートするCOBOLシステムでは、COMMIT文は現在のトランザクションの終わりを区切り、それまでのデータの更新内容を確定する。

一般形式

COMMIT

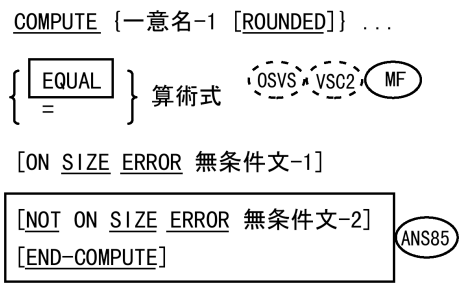
一般規則

1. COMMIT文を実行すると、実行単位によってそれまでロックされていた、ファイル中のすべてのロックが解除される。
2. 排他ファイル上のファイルロックは、COMMIT文の影響を受けない。
3. SELECT文のWITH...ROLLBACK句を単なる注記としてではなく実際にサポートするCOBOLシステムでは、COMMIT文は下記の機能を果たす。
    a. 現在のトランザクションが完了したことを示す。
    b. トランザクション処理の間になされたデータの更新内容を確定する。

12.1.9 COMPUTE（計算）文

COMPUTE（計算）文は、算術式の値をいくつかのデータ項目に代入する。

一般形式



構文規則

1. "=" の左側だけに現れる一意名は、基本数字項目か基本数字編集項目 OSVS VSC2 MF または浮動小数点数項目を指さなければならない。
2. OSVS VSC2 MF "Equal" と "=" は同義語である。

一般規則

1. 前述の*ROUNDED指定*、*ON SIZE ERROR指定*、*算術文*、*作用対象の重なり* 、*算術文における複数個の答え*の各節、および*COBOL言語の概念*の章の*明示範囲符と暗黙範囲符*、*COBOLプログラムの概念*の章の*範囲明示文*を参照。
2. 単一の一意名または定数からなる算術式を指定すると、その単一の一意名または定数の値を一意名-1の値に設定することになる。
3. 算術文のADD, SUBTRACT, MULTIPLY, DIVIDEを使用した場合は、一時的なデータ項目に中間結果を入れる必要がある。COMPUTE文を使用すると、算術処理を組み合わせながら、それらの制約を受けないで済む。式は可能なだけの桁を評価して、一意名-1中に合うように切り捨てまたは四捨五入される。

OSVS VSC2式の評価中に出された中間結果は、COBOLシステムによって決められたPICTUREのデータ項目に転記されたように切り捨てられる。この動作はARITHMETIC コンパイラ指令で選択する。

4. "=" の前に計算結果用の一意名を複数個指定すると、算術式を計算した結果の値が、個々の一意名-1の新しい値として代入される。

12.1.10 CONTINUE（継続）文

ANS85

CONTINUE（継続）文は、 処理操作を行う文ではない。実行可能な文が存在しないことを示す。

一般形式

```
CONTINUE
```

構文規則

1. 条件文または無条件文を使用できるところでは、どこでもCONTINUE文を書くことができる。

一般規則

1. CONTINUE文は、プログラムの実行には影響しない。

12.1.11 DELETE（削除）文

DELETE（削除）文は、 大記憶ファイルからレコードを論理的に削除する。DELETE文は相対編成または索引編成のファイルに対してだけ適用できる。

一般形式

```
DELETE ファイル名-1 RECORD
    [INVALID KEY 無条件文-1]

    [NOT INVALID KEY 無条件文-2]
    [END-DELETE]
```

ANS85

構文規則

1. The 順呼出し法のファイルを対象にしたDELETE文に、INVALID KEYは指定できない。

2. USE AFTER STANDARD EXCEPTION PROCEDUREを指定してない場合、順呼出し法ではないファイルを対象にしたDELETE文には、INVALID KEYを指定する。
   (OSVS)(VSC2)(MF)この規則は強制しない。

一般規則

1. DELETE文を実行する際には、対象とするファイルは入出力両用モードで開いておく。(後述する*OPEN*文の節を参照。)
2. 順呼出しモードのファイルの場合、ファイル名-1に関してDELETE文の前に最後に行われた入出力文は、正常に終了したREAD文とする。オペレーティングシステムはそのREAD文によって呼び出されたレコードを、 ファイルから論理的に削除する。
3. 乱呼出しモードまたは動的呼出しモードのファイルの場合、オペレーティングシステムは、該当するファイルの キー・データ項目の内容によって識別されるレコードをファイルから論理的に削除する。相対ファイルの場合、このキー・データ項目はRELATIVE KEYであり、索引ファイルの場合、このキー・データ項目は RECORD KEYである。キーによって指定されたレコードがファイル中に存在しないと、無効キー条件が発生する。(前述の*無効キー条件*の節を参照。)
4. DELETE文の実行が正常に終了すると、該当するレコードはファイルから論理的に削除され、呼び出すことはできなくなる。
5. DELETE文を実行することによって、ファイル名-1のレコード領域の内容が影響を受けることはない。
6. DELETE文を実行することによって、ファイル位置指示子が影響を受けることはない。
7. 該当ファイルに FILE データ項目を指定してあると、DELETE文を実行したときにその内容が更新される。(前述の*入出力状態*の節を参照。)
8. DELETE文処理が正常または不成功に終わった後、どこに制御が移されるかは、DELETE文中に INVALID KEY指定、または
   (ANS85)NOT INVALID KEY指定
   を指定したか否かに左右される。(前述の*無効キー条件*を参照。)
9. (ANS85)END-DELETE指定はDELETE文の有効範囲を区切る。(*COBOL言語の概念*の章の*明示範囲符*と*暗黙範囲符*の節を参照。)
10. (MF)DELETE文を実行する際に、対象レコードが他の実行単位によって ロックされていてはならない。
11. (MF)DELETE文の実行が正常に終了すると、その実行単位によってロックされていたレコードのロックは解除される。
12. DELETE文が実行されると、ファイル名-1に対応する入出力状態の値が更新される。(前述の*入出力状態*の節を参照。)

12.1.12 DELETE FILE（ファイル削除）文

(MF)

DELETE FILE（ファイル削除）文は、指定されたファイルをそれが格納されている物理装置から物理的に削除する。

一般形式

```
DELETE FILE {ファイル名}...
```

一般規則

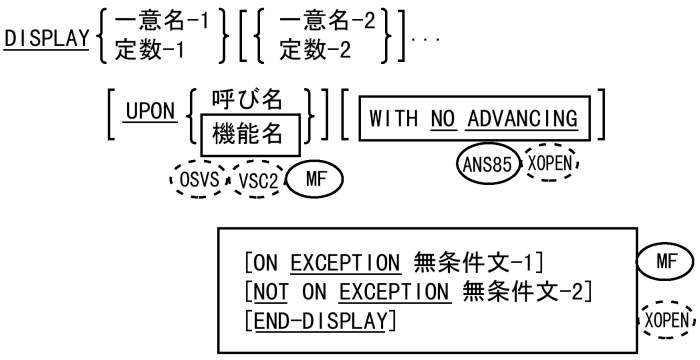
1. DELETE FILE文は、指定されたファイルをそれが格納されている物理装置から物理的に削除する。
2. DELETE FILE文を実行するとき、指定したファイルは閉じられていなければならないが、ロックされていてはならない。

12.1.13 DISPLAY（表示）文

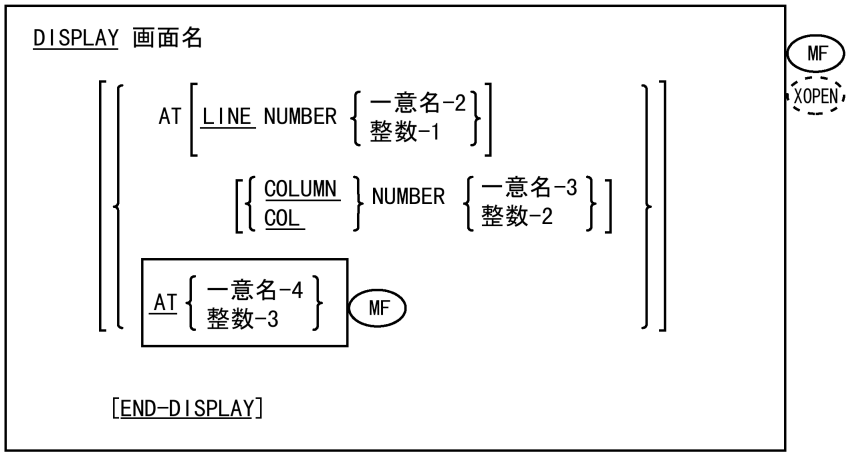
DISPLAY（表示）文は、指定されたデータ項目のデータをCRT画面のような適切なハードウエア装置に転送させる。
(MF)DISPLAY文はまた、プログラムからCRTまたはビデオ端末画面上の静止型定形画面にデータを転送して表示させる。表示されたその画面を、入力画面とすることができる。

一般形式

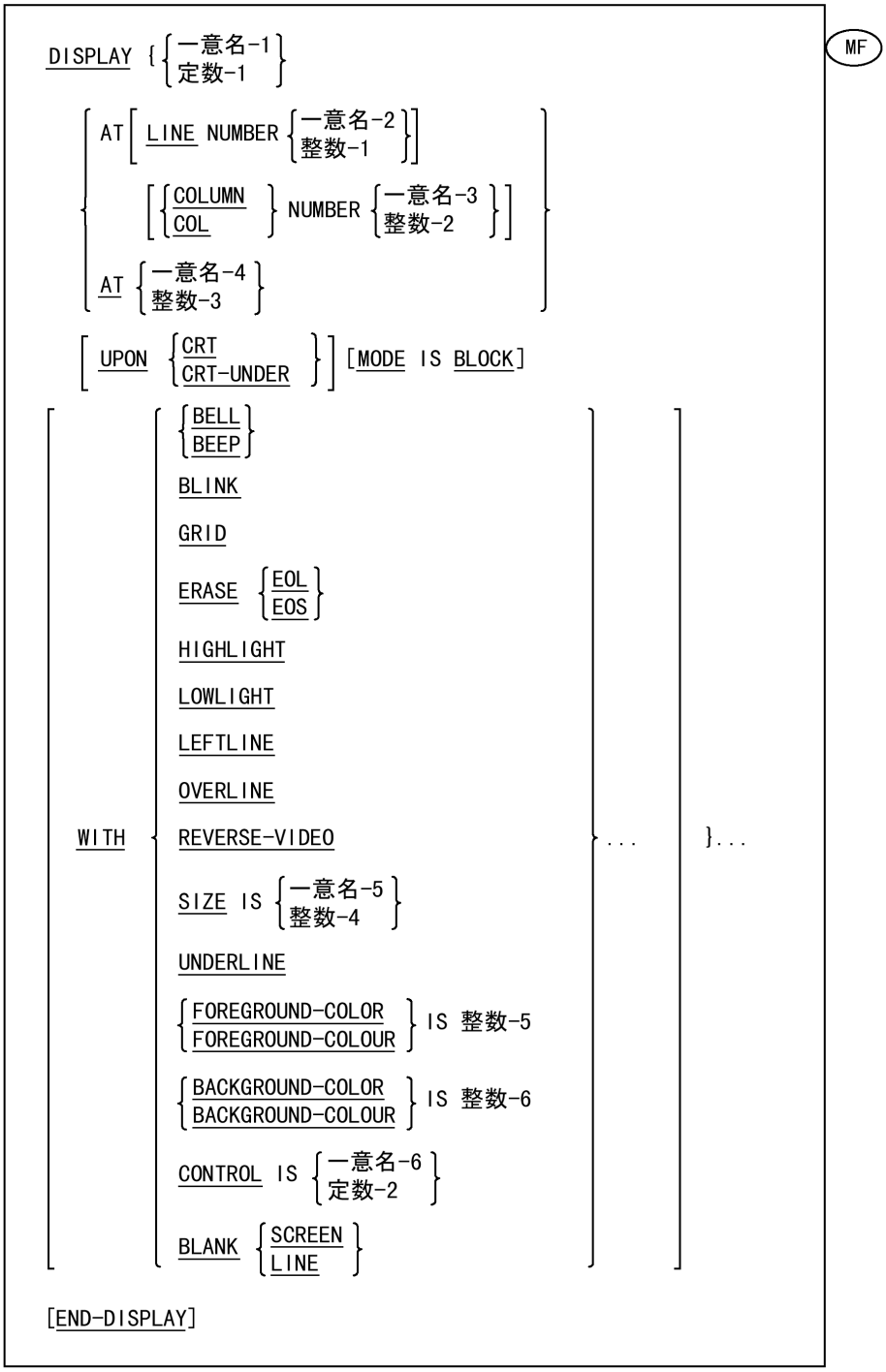
書き方 1



書き方 2


DISPLAY 画面名
AT
LINE NUMBER
一意名-2
整数-1
COLUMN
COL
NUMBER
一意名-3
整数-2
AT
一意名-4
整数-3
MF
[END-DISPLAY]
MF
XOPEN

書き方 3


DISPLAY
一意名-1
定数-1
MF
AT
LINE NUMBER
一意名-2
整数-1
COLUMN
COL
NUMBER
一意名-3
整数-2
AT
一意名-4
整数-3
UPON
CRT
CRT-UNDER
MODE IS BLOCK
WITH
BELL
BEEP
BLINK
GRID
ERASE
EOL
EOS
HIGHLIGHT
LOWLIGHT
LEFTLINE
OVERLINE
REVERSE-VIDEO
SIZE IS
一意名-5
整数-4
UNDERLINE
FOREGROUND-COLOR
FOREGROUND-COLOUR
IS 整数-5
BACKGROUND-COLOR
BACKGROUND-COLOUR
IS 整数-6
CONTROL IS
一意名-6
定数-2
BLANK
SCREEN
LINE
END-DISPLAY

構文規則

書き方 1

1. 呼び名は、環境部の特殊名段落中の機能名と関連させる。有効な機能名に関しては、前述の*特殊名段落*の節の中の「一般規則 9」を参照。
2. (OSVS)(VSC2)(MF)機能名に関連する呼び名の代わりに、機能名そのものを使用できる。
3. 各定数には、ALL以外の表意定数を使用できる。
4. 定数が数字の場合は、符号の付かない整数とする。
   (OSVS)(VSC2)(MF)この制限は無視してよい。
5. (MF)(XOPEN)呼び名がARGUMENT-NUMBERに関連しているときは、使用される一意名-1または定数-1は、それぞれ符号の付かない整数または符号の付かない整数定数として、定義されているデータ項目を指さなければならない。この場合、単一の一意名-1または定数-1だけを使用でき、WITH NO ADVANCING指定は使用できない。
6. (MF)(XOPEN)呼び名がENVIRONMENT-NAMEまたはENVIRONMENT-VALUEと関連するときは、使用される一意名-1または定数-1は、それぞれ英数字データ項目または文字定数を指さなければならない。この場合、単一の一意名-1または定数-1だけを使用でき、WITH NO ADVANCING指定は使用できない。
7. ON EXCEPTION指定およびNOT ON EXCEPTION指定は、呼び名がENVIRONMENT-NAMEまたはENVIRONMENT-VALUEと関連している場合にだけ指定できる。

書き方 2

8. (MF)画面名は、OCCURS句を伴う項目であってはならない。

書き方 2 および 3

9. (MF)LINE指定とCOLUMN指定は、どのような順番で書いてもよい。
10. (MF)整数-3および整数-4の長さは、4バイトまたは6バイトとする。

書き方 3

11. (MF)作用対象が画面名でないDISPLAY文は、その中にAT指定、CRTオプションまたはCRT-UNDERオプションを伴うUPON指定、MODE IS BLOCK指定が含まれるか、またはUPON指定は含まれないが特殊名段落中にCONSOLE IS CRT句が指定されていると、書き方3として扱われる。CONSOLEオプションを伴うUPON指定が含まれるか、またはUPON指定が含まれず特殊名段落中にCONSOLE IS CRT指定も指定されていないと、この形のDISPLAY文は書き方1として扱われる。
12. (MF)一意名に続くいろいろな指定は、どんな順番で書いてもよい。
13. (MF)一意名-1内の基本データ項目の用途は、DISPLAYとする。
14. (MF)一意名-1内の非基本データ項目は、8191バイトより大きくなることがある。MODE IS BLOCK句を使用した場合、一意名-1の全体の長さは8191バイト以下にすること。
15. (MF)DISPLAYの後に２つ以上の一意名-1が続く場合、WITH指定はその直前にある作用対象だけに適用される。

一般規則

書き方 1

1. DISPLAY文の作用対象の用途がDISPLAY以外であるかまたは符号が独立していないと、その作用対象は、用途がDISPLAYで符号が独立したものに変換される。変換後の大きさが、その作用対象の大きさとされる。
2. 作用対象の1つとして表意定数を指定すると、その表意定数が1文字だけ表示される。
3. (MF)機能名 COMMAND-LINEまたはそれに関連する呼び名を指定すると、データはシステムのコマンド行バッファに上書きされる。この後、ACCEPT FROMCOMMAND-LINE文を実行することによって、このデータを読み出すことができる。この場合は、作用対象は1つだけ指定できる。

   (MF)使用可能なそれ以外の機能名はすべて、CONSOLEに等しいものとして扱われ、各作用対象は、指定された順番に操作卓装置に転送される。表示されるデータの全体の大きさは、各作用対象の大きさの和に等しい。表示は現在カーソルが位置している所から開始され、必要があれば、以降の行に繰り越される。

   (MF)以前のリリースでは、最後の作用対象に続く空白は表示されない。
4. (ANS85)NO ADVANCINGを指定すると、カーソルは最後の文字が表示された次の位置に残される。NO ADVANCINGを指定しないと、カーソルは次の行の先頭に位置付けられる。カーソルが行頭にあるときは、いつでもスクロールできる。
5. (MF)(XOPEN)機能名ARGUMENT-NUMBERに関連する呼び名を指定すると、機能名ARGUMENT-VALUEに関連する呼び名を伴う後続のACCEPT用の位置は、指定されたコマンド行の引数を読み出すように設定される。一意名-1または定数-1に関連する値が0より小さいか、99より大きいか、コマンド行上の引数の合計数よりも大きいと、結果はどうなるかわからない。
6. (MF)(XOPEN)機能名ENVIRONMENT-NAMEに関連する呼び名を指定すると、機能名ENVIRONMENT-VALUEに関連する呼び名を伴う後続のACCEPTまたはDISPLAY中で読み取るかまたは設定すべき変数が、定数-1または一意名-1の内容として指定された変数名に設定される。ON EXCEPTION指定は、指定されていても実行されない。したがって、ENVIRONMENT-VALUEに関連する後続のACCEPTまたはDISPLAY用のENVIRONMENT-NAMEを設定しようとしている最中に問題が発生した場合、ON EXCEPTION条件の検出は、後続のACCEPTまたはDISPLAYの中で行なわなければならない。
7. (MF)(XOPEN)機能名ENVIRONMENT-VALUE に関連する呼び名を使用すると、下記のようになる。
   a. ENVIRONMENT-NAMEに関連する呼び名を伴うDISPLAYが前に実行されていると、一意名の値が指定された環境変数に入れられる。
   b. ENVIRONMENT-NAMEに関連する呼び名を伴うDISPLAYが前に実行されていないと、またはACCEPTに関しては指定された環境変数が存在しないと、またはDISPLAYに関して環境変数を格納するために割り当てられる領域が足りないと、ON EXCEPTION指定に対応する無条件文が実行される。この場合、一意名中の値はどうなるかわからない。

8. (MF)(XOPEN)機能名SYSERRに関連する呼び名を使用すると、DISPLAYはCONSOLEに対するように行われるが、出力はすべてシステムの標準エラー装置に向けられる。
9. (MF)(XOPEN)DISPLAY UPON ENVIRONMENT-VARIABLEを実行するときは、後に続く空白も値のうちに含められる。これを避けたければ、INSPECT REPLACING ALL SPACES BY LOW-VALUESを使用する。
10. (OSVS)(VSC2)(MF)一意名-1においては、内部浮動小数点数は表示用の外部浮動小数点数に変換される。その結果、下記のようになる。
    a. COMP-1項目は外部浮動小数点数用のPICTURE句-.9(8)E-99が指定されているように表示される。
    b. COMP-2項目は外部浮動小数点数用のPICTURE句-.9(18)E-99 が指定されているように表示される。（このPICTURE文字列を実際に明示的に使用するとまちがいとなる。）
11. (OSVS)(VSC2)(MF)定数-1においては、浮動小数点数定数が許される。

書き方 2

12. (MF)この書き方のDISPLAY文を実行すると、プログラムの画面節内に定義されている画面項目が表示され、拡張画面操作機能によって全面的に取り扱えるようになる。

書き方 2 および 3

13. (MF)AT指定は、DISPLAY処理を開始する画面上の絶対番地を指定する。
14. (MF)DISPLAY文が実行される順序は、つねに下記のとおりである。
    1. AT指定
    2. BLANK指定
    3. BELL指定
    4. DISPLAY処理
15. (MF)整数-3または整数-4の長さが4桁ならば、上2桁は行を表わし下2桁はカラムを表わす。整数-3または整数-4の長さが6桁ならば、上3桁は行を表わし下3桁はカラムを表わす。
16. (MF)行番号とカラム番号の組合せの中には、特殊な意味をもつものがある。それらは下記のとおり。
    a. カラムの値が1行のカラム数よりも大きいと、その値から1行のカラム数が差し引かれ、行数に1が加えられる。
    b. 行の値が画面の範囲から外れると、画面が1行上にスクロールされる。この効果は、画面の下端の行を指定したのと同じである。
    c. 与えられた行番号とカラム番号がともにゼロであると、書き方2または書き方3の前のDISPLAY処理が終わった位置の直後から、DISPLAYが開始される。各行のカラム1は、前の行の最後のカラムに続くものとみなされる。
    d. 指定された行番号がゼロであるがカラム番号はゼロでないと、書き方2または書き方3の前のDISPLAY処理が終わった位置の次の行の指定されたカラムから、DISPLAYが開始される。

e. 指定されたカラム番号がゼロであるが行番号はゼロでないと、書き方2または書き方3の前のDISPLAY処理が終わった位置の次のカラムの指定された行から、DISPLAYが開始される。

書き方 3

17. (MF)この文の一部を繰り返し指定して、いくつものデータ項目を表示するようにできる。最初の一意名にAT指定を指定しないと、行1、カラム1から表示が開始される。それ以降に続く各データ項目は、前のデータ項目を表示し終わった後のカーソルの位置から表示が開始される。
18. (MF)MODE IS BLOCK指定を指定しないと、一意名が集団項目であれば、それに属する基本項目で名前がFILLERでないものが表示される。これらの項目はデータ部内で記述されている順に画面上に表示され、集団内のFILLERの長さによって区切られる。この目的では、ある行の最初の位置は、その前の行の最後の位置の直後に続くものとみなされる。
19. (MF)MODE IS BLOCK指定は、一意名を基本項目として扱うように指定する。こうすると、一意名が集団項目であっても、1つの項目として表示される。
20. (MF)WITH指定を用いると、各種のオプションをDISPLAY処理の実行時に指定できる。(これらの指定の説明については、前述の*画面節* を参照。)
    (MF)画面記述句に指定できるオプションの他に、WITH指定にはいくつかのオプションが加えられている。SPACE-FILL, ZERO-FILL, LEFT-JUSTIFY, RIGHT-JUSTIFY, TRAILING-SIGN, UPDATEである。ZERO-FILLには2つの用法があるため、このリストにも画面記述句にも現れている。2番目の用法については、この章で後述する。
    (MF)自由方式で、数字および数字編集の画面項目に、データを入力できるようにする構成オプションがある。COBOLでは、非編集数字データ項目は、内部形式でデータを保持することを目的としたものである。自由方式を用いると、このようなデータ項目を画面上に表示できる。 詳細は、ユーザーインタフェースに関するCOBOL システムのマニュアルを参照。自由方式がとられているときは、データは自動的に下記のように編集されて表示される。
    - 仮想小数点を、終止符で表わす。
    - 符号を、符号文字で表わす（負の数は "-" で、正の数は空白で）。符号は、左端の数字の直前に付けられる。
    - すべての整数文字位置の先行ゼロが抑制される。ただし、最末桁を除く。
    - 左に桁寄せされる。

    (MF)SPACE-FILL, ZERO-FILL, LEFT-JUSTIFY, RIGHT-JUSTIFY, TRAILING-SIGNオプションを使用すると、上記の形式を補正できる。

21. (MF)一意名-1に表意定数を指定すると、次のような特別な効果がある。SPACEを指定すると、指定したカーソル位置から画面の末尾までがクリアされる。LOW-VALUEを指定すると、指定した位置にカーソルが移動される。ALL X"01" を指定すると、指定したカーソル位置からその行の末尾までがクリアされる。ALL X"02" を指定すると、画面全体がクリアされる。ALL X"07" を指定すると、警報が鳴らされる。一意名に上記以外の表意定数を指定し、SIZEオプションを指定しないと、その表意定数の値が1つ表示される。
22. (MF)特別の効果をもたない表意定数にSIZEオプションを指定すると、その表意定数の値がSIZE分の長さだけ表示される。ただし、行末から次の行頭へ表示が繰り越されるときは、表意定数の最初から表示し直される。
23. (MF)FOREGROUND-COLORオプションを指定すると、画面全体がこの表示によってクリアされた場合にだけ、指定した色が省略時解釈の前景色となる。 .
画面全体がクリアされるのは、BLANK SCREEN オプションを指定した時、または一意名-1 が SPACES で表示が行1、カラム 1から始まる時である。
24. (MF)BACKGROUND-COLORオプションを指定すると、画面全体がこの表示によってクリアされた場合にだけ、指定した色が省略時解釈の背景色となる。
25. (MF)一意名-1にREDEFINESがかかっている場合、再定義データ領域の最初の記述が使用され、以降の記述は無視される。OCCURSまたは入れ子になったOCCURSが指定されている場合、反復されるデータ項目はその反復回数分展開される。したがって、1つの定義が多数の項目にわたって繰り返されることになる。

12.1.14 DIVIDE（除算）文

DIVIDE（除算）文は、いくつかの数字データ項目を1つの数字データ項目で割って、 商と 剰余をデータ項目の値として設定する。

一般形式

書き方 1

```
DIVIDE { 一意名-1 } INTO {一意名-2 [ROUNDED]} ...
       { 定数-1   }
 [ON SIZE ERROR 無条件文-1]
 [NOT ON SIZE ERROR 無条件文-2]                (ANS85)
 [END-DIVIDE]
```

書き方 2

```
DIVIDE {一意名-1 | 定数-1} INTO {一意名-2 | 定数-2}
        GIVING {一意名-3 [ROUNDED]} ...
 [ON SIZE ERROR 無条件文-1]
 [NOT ON SIZE ERROR 無条件文-2]                    ANS85
 [END-DIVIDE]
```

書き方 3

```
DIVIDE {一意名-1 | 定数-1} BY {一意名-2 | 定数-2}
        GIVING {一意名-3 [ROUNDED]} ...
 [ON SIZE ERROR 無条件文-1]                        ANS85
 [NOT ON SIZE ERROR 無条件文-2]
 [END-DIVIDE]
```

書き方 4

```
DIVIDE {一意名-1 | 定数-1} INTO {一意名-2 | 定数-2}
        GIVING 一意名-3 [ROUNDED]
 REMAINDER 一意名-4
 [ON SIZE ERROR 無条件文-1]
 [NOT ON SIZE ERROR 無条件文-2]                    ANS85
 [END-DIVIDE]
```

書き方 5

```
DIVIDE { 一意名-1 } BY { 一意名-2 }
       { 定数-1   }    { 定数-2   }
        GIVING 一意名-3 [ROUNDED]
  REMAINDER 一意名-4
  [ON SIZE ERROR 無条件文-1]                    (ANS85)
  [NOT ON SIZE ERROR 無条件文-2]
  [END-DIVIDE]
```

構文規則

すべての書き方

1. 各一意名は、基本数字項目とする。ただし、 GIVING指定または REMAINDER指定に指定する一意名は、基本数字編集項目であってもよい。
2. 各定数は、数字定数とする。
3. 作用対象を合成したものは、 18桁を超えてはならない。ここで、作用対象を合成したものとは、すべての受取り側データ項目を小数点の位置を揃えて重ね合わせた結果としてできる、仮想のデータ項目である。ただし、REMAINDERデータ項目は、受取り側データ項目に含めない。
4. (OSVS)(VSC2)(MF)書き方1、2、3 のいずれかの場合、浮動小数点データ項目または定数は、数字データ項目または定数が指定できる箇所であればどこでも使用できる。書き方4または5 の場合、浮動小数点データ項目または定数は指定できない。

一般規則

すべての書き方

1. 前述の*ROUNDED指定*、*ON SIZE ERROR指定*、*算術文*、*作用対象の重なり*、*算術文における複数個の答え*の各節、*COBOL言語の概念*の章の*明示範囲符と暗黙範囲符*、*COBOLプログラムの概念*の章の*範囲明示文*を参照。さらに、書き方4、5に関連するROUNDED指定とON SIZE ERROR指定については、下記の一般規則5と7を参照。

書き方 1

2. 書き方1では、定数-1または一意名-1によって参照されるデータ項目の値が一時的データ項目に記憶される。次いで、一意名-2の値がこの一時的なデータ項目の値で割られる。そして、 被除数（一意名-2）の値が 商で置き換えられる。同様に、一意名-2の後ろに続く各被除数が、指定された順番に左から右に順次この一時的なデータ項目で割られる。

書き方 2

3. 書き方2では、一意名-2または定数-2が、定数-1または一意名-1のデータ項目の値で割られ、結果が一意名-3によって参照される各データ項目に入れられる。

書き方 3

4. 書き方3では、一意名-1または定数-1が、定数-2または一意名-2のデータ項目の値で割られ、結果が一意名-3によって参照される各データ項目に入れられる。

書き方 4 および 5

5. 書き方4と5は、一意名-4に 剰余を入れたいときに使用する。COBOLでは、剰余は、被除数から 除数と商（一意名-3）の積を差し引いた結果の値であると定義される。一意名-3が数字編集項目として定義されていると、剰余を計算するために用いられる商には、編集されていない値を保持している中間項目が使用される。ROUNDEDを指定すると、剰余を計算するために用いられる商には、四捨五入ではなく切り捨てられた値を保持している中間項目が使用される。
6. 書き方4と5では、REMAINDERデータ項目（一意名-4）の精度は上記の計算によって決まる。一意名-4によって参照されるデータ項目の内容は、必要に応じて、小数点の位置合わせと切捨て（四捨五入ではない）が行われる。
7. 書き方4と5においてON SIZE ERROR指定を書くと、下記の規則が適用される。
    a. 桁あふれが 商に発生した場合、 剰余を計算しても意味がない。したがって、一意名-3および一意名-4によって参照される値は変更されない。
    b. 剰余に桁あふれが発生した場合、一意名-4によって参照される値は変更されない。しかし、算術演算で複数の結果を求める他の場合と同様、利用者は自分で分析を行って、実際に何が発生したのかを突き止める必要がある。

第13章: 手続き部 - ENTER - INVOKE

13.1 COBOLの文

13.1.1 ENTER（導入）文

ENTER（導入）文は、 同じプログラム内で2つ以上の言語を使用できるようにする。
(ANS85)ENTER文は、ANSI'85標準では廃要素に分類されており、ANSI標準の次回の全面改訂の際に、削除される予定である。
(MF)このCOBOLに組み込まれているすべての方言では、この構文は注記としての役割を果たすだけである。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。
(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、この動詞は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内では、この文を使用するべきではない。

一般形式

```
ENTER 言語名 [手順名].
```

構文規則

1. 言語名および手順名には、任意の利用者語または英数字定数を書くことができる。

一般規則

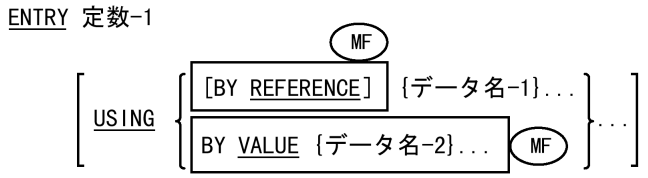
1. この文は、注記としての役割を果たすだけである。他の言語を実際に呼び出すには、CALL文を使用する。

13.1.2 ENTRY（入口）文

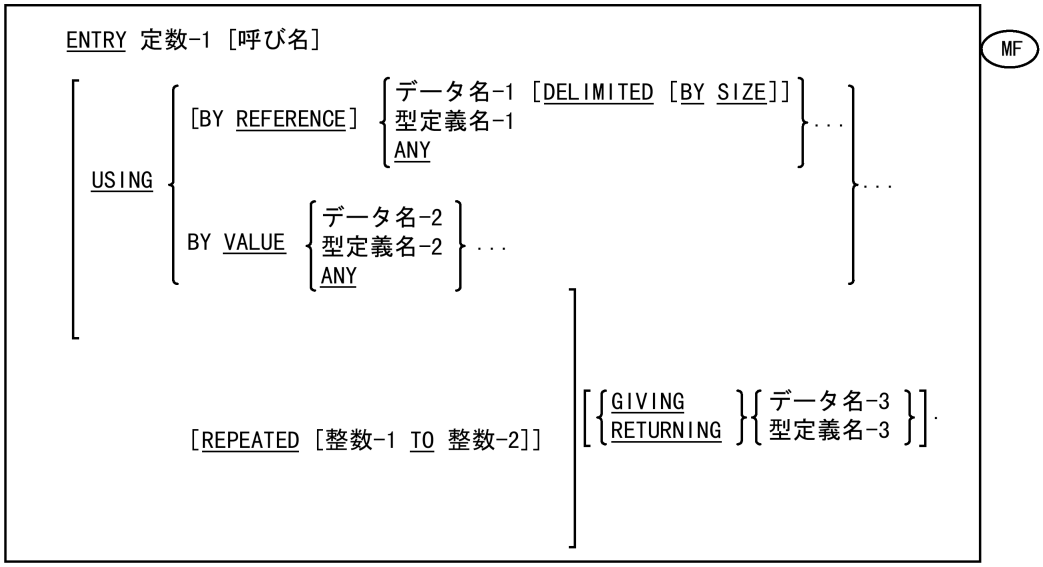
(OSVS)(VSC2)(MF)
ENTRY（入口）文は、呼ばれたCOBOLプログラムへの代替の入口点を設定する。

一般形式

書き方 1



書き方 2



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - STICKY-LINKAGE - 同じプログラム内で異なる入口点を呼び出す間に、連絡節項目の番地を指定できる状態を維持するかどうかを制御する。
    - LITLINK - **.obj**ファイルを生成するときに、入口名の有効文字数を制御する。

構文規則

すべての書き方

1. 他のプログラム内に入れ子になっているプログラム内では、ENTRY文を使用できない。
2. 定数-1は数字であってはならず、表意定数であってもならない。環境によっては、先頭の８文字だけが有効なものがある。

書き方 1

3. 書き方１はプログラム定義中にのみ指定できる。

4. データ名-1は連絡節、 (MF)ファイル節、 (MF)作業場所節 のいずれかに、レベルが01または77の項目として定義されていなければならない。
5. (MF)データ名-2は連絡節、ファイル節、作業場所節のいずれかに、レベルが01または77の項目として定義されていなければならない。データ名-2は字類は何であってもよいが、長さが８バイトを超えてはならない。

書き方 2

6. (MF)書き方２を使用できるのは、プログラム名段落にEXTERNAL句が指定されているプログラム、つまり呼び出しプロトタイプ、の中だけである。
7. (MF)データ名-1とデータ名-2は連絡節内の01レベルのレコードとして定義されていなければならない。
8. (MF)型定義名-1、型定義名-2、型定義名-3は、同じソース・ファイル中にTYPEDEF句を用いて、プログラマによって用途が定義されたものとして、予め定義されていなければならない。

一般規則

すべての書き方

1. 定数-1の内容は入口名を表す。それはプログラムへの入口点を指しており、手続き部の開始時点での省略時のCOBOL入口点の代りの入口点となる。プログラム名段落中のプログラム名は省略時のCOBOL入口点を示すと考えられる。
   実行用のCOBOLのコードが収められているプログラム・ファイルの名前は通常、基本的にプログラム名と同じであることに注意すること。呼出しが行われたときにプログラム名または入口名がまだメモリーにロードされていないと、ファイル名に基づいてプログラム・ファイルが探されるが、プログラム名に基づいて探されているように見える。そのような状況において、プログラム名を参照する呼出しは正常に行われるが、入口名を参照する呼出しはエラーとなる可能性がある。
   入口名は多くの場合にプログラム名に等しく、両者は名前の形成に関する共通の規則に従う。その規則は「プログラム名段落と入れ子になった原始プログラム」の項に記載されている。定数-1はプログラム名段落の一般形式の定数-1に対応する。
2. 定数-1に入口名を指定したCALL文を使用して呼ぶプログラムから呼ばれるプログラムを呼び出すと、ENTRY文の後ろの次の実行可能な文に制御が移される。
3. 連絡節に宣言されているがENTRY文のUSING指定に宣言されていないデータ項目を参照できるのは、SET文を実行してそれらのデータ項目をあるデータ項目にリンクしてある場合、またはコンパイラ指令のSTICKY-LINKAGEが指定されている場合だけである。
4. USING指定には最大で62個のデータ名を指定できる。
5. BY REFERENCE指定とBY VALUE指定は共に、別のBY REFERENCE指定またはBY VALUE指定が出てくるまで、後続のパラメータに効力を及ぼす。最初のパラメータの前にBY REFERENCE指定もBY VALUE指定もない場合には、BY REFERENCE指定があるものとみなされる。

書き方 1

6. (MF)データ名-1がファイル節または作業場所節の中のレベルが01または77の項目として定義されている場合、オブジェクト・プログラム動作は下記のようになる。具体的には、データ名-1と同じ内容で連絡節内にデータ項目が宣言され、呼ばれるプログラム中の最初の文を実行する前にそのデータ項目の内容がデータ名-1に転記するかのように動作する。初期プログラムでは、それらの値はそのプログラムの作業場所節のデータを初期化するさいに上書きされるので、呼ばれたプログラムからは利用できない。
7. 稼働しているランタイム要素がCOBOLである場合、下記の規則が適用される。稼働しているランタイム要素がCOBOLでない場合、どういうときにBY REFERENCE指定またはBY VALUE指定を使用する必要があるかの詳細については、インタフェースに関するCOBOLシステム・ドキュメンテーションを参照。
8. USING指定はプログラムで使用される仮パラメータまたはプログラムに渡される任意の引数を指す。プログラムに渡された引数は、稼働しているソース要素によって、CALL文のUSING指定を用いて識別される。２つの名前リストの間の対応関係は、位置に基づいて確立される。
9. 引数が内容によって渡される場合、呼ばれるプログラムの動作は下記のようになる。具体的には、呼出しを開始するプロセスの間に呼ぶランタイム要素によって連絡節内のレコードが割り当てられ、そのレコードは呼び出すランタイム要素中の引数と同じ記憶領域を占めることはないかのように動作する。この割り当てられたレコードの長さは引数とちょうど同じ文字数である。その引数がその割り当てられたレコードに何も変換せずに転記される。その後で、そのレコードは呼ばれたプログラムによって、それは引数であり参照によって渡されたかのように処理される。
10. 引数が参照によって渡された場合、仮パラメータが引数と同じ記憶領域を占めるかのように、呼ばれるプログラムは動作する。
11. (MF)引数が値によって渡される場合には、呼ばれるパラメータの動作は下記のようになる。具体的には、呼出しを開始するプロセスの間に呼ぶランタイム要素によって連絡節内のレコードが割り当てられ、そのレコードは呼び出すランタイム要素中の引数と同じ記憶領域を占めることはないかのように動作する。この割り当てられたレコードの長さは引数とちょうど同じ文字数である。その引数がその割り当てられたレコードに何も変換せずに転記される。その後で、そのレコードは呼ばれたプログラムによって、それは引数であり参照によって渡されたかのように処理される。
12. 稼働している要素においては常に、データ名-1、(MF)データ名-2および(MF)データ名-3への参照は連絡節中の記述に基づいて解明される。その連絡節中のデータ項目に定義されている文字数が、稼働している要素中の対応するデータ項目よりも多いと、結果はどのようになるか分からない。この規則に外れるか、具体的なランタイム環境において許されるシステム領域の最大許容サイズを超えるかすると、システムは壊滅的な障害を起こす可能性がある。

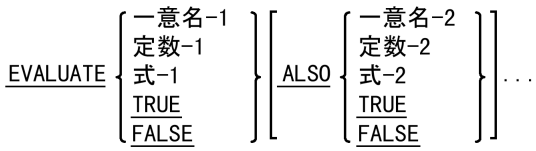
13. (MF)プログラムが呼ばれ、USING指定中のBY REFERENCEの作用対象が呼ぶプログラム中のパラメータに対応する場合、参照のリンクが確立される。そのリンクは制御が呼んだプログラムに戻されるまで持続する。その後、そのプログラムを取り消すことなく、再びそのプログラムを呼んだ場合、同じBY REFERENCEの作用対象が呼んだプログラムのパラメータに対応しなかったならば、その作用対象を参照してはならない。ただし、コンパイラ指令のSTICKY-LINKAGEを指定した場合は別である。
14. USING指定を書いた場合、いかなるCD項目中にもINITIAL句があってはならない。(『言語リファレンス - 追加トピック』中の「通信」の章の「通信記述 - 項目の完全な枠組み」を参照。)

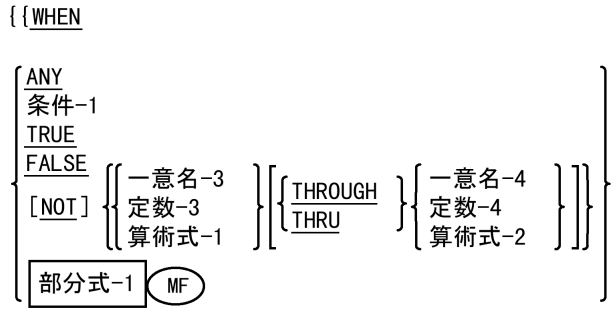
13.1.3 EVALUATE（評価）文

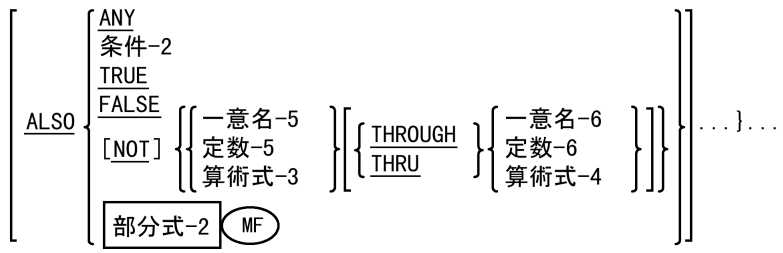
(ANS85)

EVALUATE（評価）文は、 多くの分岐点や 合流点がある構造を記述しておいて、 複数の条件を評価できるようにする。 その評価の結果によって、実行時要素のその後の動きが決まってくる。

一般形式







無条件文-1}...

[WHEN OTHER 無条件文-2]

[END-EVALUATE]

構文規則

1. EVALUATE文の最初の WHEN指定の前に現れる作用対象または語TRUEとFALSEを、個別には 選択の左辺といい、総称して選択の左辺の組という。
2. EVALUATE文の最初のWHEN指定の中に現れる作用対象または語TRUEとFALSEとANYを、個別には 選択の右辺といい、総称して選択の右辺の組という。

3. THROUGHとTHRUは同義である。
4. THROUGH指定によってつないだ2つの作用対象は、同じ字類のものにする。このようにTHROUGH指定によってつないだ2つの作用対象は、単一の選択の右辺を構成する。
5. 選択の右辺の組に含まれる選択の右辺の数は、選択の左辺の数と等しくする。
6. 選択の右辺の組に含まれる各選択の右辺は、選択の左辺の組に含まれる同じ順番の位置にある選択の左辺と対応させる。これには、下記の規則が適用される。
    a. 選択の右辺の中に現れる一意名、定数、算術式は、選択の左辺中の対応する作用対象と比較する作用対象として、有効なものにする。(前述の*比較条件*の節を参照。)
    b. 選択の右辺として現れる条件-1、条件-2あるいは語TRUEまたはFALSEは選択の左辺の組の中の条件式または語TRUEまたはFLASEと対応させる。
    c. 語ANYは、任意の種類の選択の左辺と対応できる。
    d. (VSC2)(MF)一意名を使えるところでは、浮動小数点数データ項目を参照できる。
    e. (VSC2)(MF)数字定数を使えるところでは、浮動小数点数定数を参照できる。
    f. (VSC2)(MF)一意名を使えるところでは、ポインタ・データ項目を参照できる。
7. (MF)選択の右辺として部分式-1または部分式-2を指定する場合、これらは一意名か定数か算術式である選択の左辺と対応させる。部分式-1および部分式-2は一連のCOBOLの語であり、対応する選択の左辺の後ろに続けたときに、有効な条件式となるものでなければならない。

一般規則

1. EVALUATE文を実行すると、まず、選択の左辺と選択の右辺が評価されて、1つの数値、1つの文字値、数値の範囲、文字値の範囲、真理値のどれかが割り当てられる。これらの値は下記のように決定される。
    a. 一意名-1および一意名-2によって指定される選択の左辺、および一意名-3および一意名-5によって指定される右辺のうちで、NOT指定またはTHROUGH指定を伴わないものは、その一意名によって参照されるデータ項目の値と字類を割り当てられる。
    b. 定数-1および定数-2によって指定される選択の左辺、および定数-3および定数-5によって指定される右辺のうちでNOT指定もTHROUGH指定も伴わないものは、その定数の値と字類を割り当てられる。定数-3と定数-5が表意定数のZEROであると、それらには対応する選択の左辺の字類が割り当てられる。
    c. 算術式として式-1および式-2を指定された選択の左辺、および算術式-1または算術式-3が含まれるNOT指定もTHROUGH指定も伴わない選択の右辺は、算術式の評価規則に従って数値を割り当てられる。(前述の*算術式*の節を参照。)
    d. 条件式として式-1および式-2を指定された選択の左辺、および条件-1と条件-2を指定された選択の右辺は、条件式の評価規則に従って真理値を割り当てられる。(前述の*条件式*の節を参照。)
    e. 語TRUEまたはFALSEを指定された選択の左辺または選択の右辺は、真理値を割り当てられる。語"true" が指定された項目は、真理値の「真」を割り当てられ、語"false" が指定された項目は、真理値の「偽」を割り当てられる。

f. 語ANYを指定された選択の右辺は、それ以上評価されない。
g. 選択の右辺にNOT指定を付けずにTHROUGH指定を書くと、選択の左辺と比較したときの値の範囲は、比較の規則に基づいて、最初の作用対象以上で2番目の作用対象以下となる。(前述の*比較条件*の節を参照。) 最初の作用対象の方が2番目の作用対象よりも大きいと、その範囲には値は何も含まれない。
h. 選択の対象にNOT指定を書くと、割り当てられる値は、その値と等しくないすべての値、またはNOT指定を指定しなかった場合に割り当てられる値以外のものとなる。

2. EVALUATE文の実行は、選択の左辺と選択の右辺の比較へと進む。そして、選択の左辺の組の中にWHEN指定を満たすものがあるかどうかが判定される。この比較は下記のように行われる。
   a. 最初のWHEN指定の選択の右辺の組の中の各選択の右辺が、選択の左辺の組の中の順序位置が同じ選択の左辺と比較される。
      1. 比較対象の項目に割り当てられている値が数値、文字値、数値の範囲、文字値の範囲である場合は、選択の右辺に割り当てられている値または値の範囲内のどれかの値が、比較規則に照らして選択の左辺に割り当てられている値と等しければ、その比較条件は満足される。(前述の*比較条件*の節を参照。)
      2. 比較対象の項目に割り当てられている値が真理値である場合は、選択の左辺と選択の右辺に同じ真理値が割り当てられているときに、その比較条件は満足される。
      3. 比較対象の項目に割り当てられている値が語ANYである場合は、その比較条件はつねに満足される。この場合は、選択の左辺の値は何であってもよい。
   b. 比較対象の選択の右辺の組に含まれる各選択の右辺に関して、上記の比較が満足されると、その選択の右辺の組が含まれるWHEN指定が選択の左辺の組を満たすものの1つとして選択される。
   c. 比較対象の選択の右辺の組に含まれる選択の右辺の中に、上記の比較が満足しないものがいくつかあると、その選択の右辺の組は選択の左辺の組を満たさない。
   d. 後続の選択の右辺の組に対して、上記の比較処理が、原始要素の記述されている順序で繰り返し実行される。そして、選択の左辺の組を満たすWHEN指定が選択されるか、またはすべての選択の右辺の組を比較し尽くした時点で、この処理は終わる。
3. 比較処理が終わると、EVALUATE文の実行は下記のように進む。
   a. WHEN指定が選択された場合、そのWHEN指定の後ろに続く最初の無条件文-1に処理は進む。
   b. WHEN指定が選択されなかった場合、WHEN OTHER指定が指定されていれば、無条件文-2に処理は進む。

c. EVALUATE文の実行は、選択されたWHEN指定またはWHEN OTHER指定の最後まで到達した時、あるいは選択されたWHEN指定がなくWHEN OTHER指定が指定されていない時に終了する。(*COBOL言語の概念*の章の*明示範囲符と暗黙範囲符*の節を参照。)

4. (MF)選択の右辺が部分式-1または部分式-2によって指定されている場合、対応する選択の左辺は語TRUEであるものとみなされる。選択の右辺はそれぞれ条件-1または条件-2とみなされる。ここで、条件-1または条件-2はそれぞれ、元の対応する選択の左辺-1または選択の左辺-2に続く部分式-1または部分式-2から導出される条件式である。

13.1.4 EXAMINE（検査）文

(OSVS)

EXAMINE（検査）文は、データ項目中に現れる指定された文字を置き換えたり、数を数えたりする。

一般形式

書き方 1

```
EXAMINE 一意名 TALLYING
                        { UNTIL FIRST }
                        { ALL         } 定数-1 [REPLACING BY 定数-2]
                        { LEADING     }
```

書き方 2

```
EXAMINE 一意名 REPLACING { ALL       }
                         { LEADING   }
                         { FIRST     } 定数-1 BY 定数-2
                         { UNTIL FIRST }
```

構文規則

すべての書き方

1. 一意名は、明示的または暗黙的に、USAGE IS DISPLAYと記述しておく。
2. 各定数は、1文字で構成させる。一意名が数字項目である場合、定数は数字定数、値が単一の数字である文字定数、表意定数のZEROのどれかとする。一意名の字類がそれ以外の場合は、定数は数字でも文字でも語ALL以外の任意の表意定数であってもよい。

一般規則

すべての書き方

1. 検査は、下記のように行われる。
    a. 文字データ項目に関しては、検査は左端の文字から開始され、右方向に進められる。一意名に指定したデータ項目中の各文字が、順々に検査される。

   b. EXAMINE文の対象とするデータ項目が数字の場合は、演算符号が付いてもよい。検査は符号を除く左端の文字から開始され、右方向に進められる。符号を除く各文字が順々に検査される。符号はどこに付いていても、EXAMINE文によって完全に無視される。

2. The T TALLYINGオプションを指定すると、数が数えられて、 TALLYと呼ばれる特殊レジスタの値が置き換えられる。 この数は、下記の事柄を表わす。
   a. UNTIL FIRSTオプションを指定した場合は、最初に定数-1が見つかるまでに出て来る定数-1以外の文字の数。
   b. ALLオプションを指定した場合は、定数-1が出て来る数。
   c. LEADINGオプションを指定した場合は、最初に定数-1以外の文字が見つかるまでに出て来る定数-1の数。
3. REPLACINGオプションを指定した場合は、下記の規則に基づいて置き換えが行われる。
   a. ALLオプションを指定すると、出て来る定数-1はすべて定数-2に置き換えられる。
   b. LEADINGオプションを指定すると、定数-1以外の文字が出て来るか、データ項目の右端に達したときに、定数-2による置き換えが終了される。
   c. FIRSTオプションを指定すると、最初に出て来る定数-1が定数-2に置き換えられる。
   d. UNTIL FIRSTオプションを指定すると、定数-1が出て来るか、データ項目の右端に達したときに、定数-2による置き換えが終了される。

13.1.5 EXEC(UTE)（実行）文

(MF)

The EXEC（UTE）（実行）文は、COBOL以外のサブシステムに制御を渡すための連絡機構として設けられている。

一般形式

```
EXEC[UTE]原文名 原文データ END-EXEC
```

構文規則

1. テキスト・データは、文字列END-EXECを含まなければ、どんなテキスト的データであってもよい。

一般規則

1. この文は、「CALL " テキスト名" USINGテキスト・データ・バッファ」文に翻訳される。（前述の*CALL*文の節を参照。）ここで、テキスト・データ・バッファにはEXEC文のすべてのテキスト・データが入れられ（空白は圧縮される）、これが呼ばれるプログラムである 「テキスト名」によってさらに解析される。

このことは、呼ばれる副プログラム（テキスト名）に、テキスト文字列が渡されることを意味する。このテキスト文字列は、動詞EXEC(UTE)で始まりEND-EXEC指定で終わる間のすべてのCOBOL文が含まれる。呼ばれるプログラムは、渡されるすべての情報を解析して処理しなければならない。

テキスト・データ内に一意名が含まれている場合、これは副プログラムに渡される一意名を表わしているのであって、データの値を表わしているのではない。

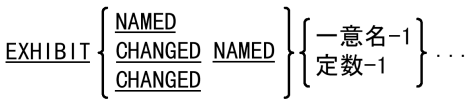
2. 特例のEXEC SQLは通常はコンパイラによって違った扱いをされる。EXEC SQL文もここに記述されているのと同様に取り扱いたい場合は、コンパイラ指令のNOSQLを必ず明示的に設定しなければならない。

13.1.6 EXHIBIT（参照）文

OSVS MF

EXHIBIT（参照）文は、指定された定数や一意名の内容を表示する。条件付きで表示させることもできる。一意名の場合は、内容の前に名前を表示させることができる。

一般形式



構文規則

1. EXHIBIT文中に指定した各一意名は、どんな字類のデータであってもよい。RETURN-CODEだけである。
2. CHANGEDとNAMEDは、両方とも省略できる。

一般規則

1. EXHIBIT文によって表示される定数および一意名は、表示行上では1つの空白によって区切られる。
2. 定数には、ALL以外の表意定数を指定できる。
3. 定数に数字を指定する場合は、符号の付かない整数とする。
4. EXHIBIT NAMED文を実行すると、指定した各一意名または定数が表示される。一意名の場合は、その名前（修飾語および添字があれば、それを含む）の後に"="（等号）を付けて、その後ろに現在の値が表示される。表示される一意名または定数は、EXHIBIT文に指定された順番に、1行に並べられる。

5. EXHIBIT CHANGED NAMED文を実行すると、指定した各一意名または定数が表示される。一意名の場合は、その名前（修飾語および添字があれば、それを含む）の後に"="（等号）を付けて、その後ろに現在の値が表示される。表示される一意名または定数は、EXHIBIT文に指定された順番に、1行に並べられる。ただし、表示される一意名（名前と値）は、前回このEXHIBIT文を実行した後で値に変更があったものだけである。値が変更されなかった一意名については、その名前も値も表示されない。値が変更された一意名が1つもなく、定数も1つも指定されていないと、何も表示されない（空白行の表示は抑制される）。
6. EXHIBIT CHANGED文を実行すると、指定した各一意名または定数の現在の値が表示される。表示される一意名または定数は、EXHIBIT文に指定された順番に、1行に並べられる。ただし、表示される値は、前回このEXHIBIT文を実行した後で値に変更があったものだけである。値が変更されなかった一意名については、その値は表示せず、代わりに空白が挿入される。値が変更された一意名が1つもなく、定数も1つも指定されていないと、空白行が表示される（空白行の表示は抑制されない）。
7. CHANGEDオプションもNAMEDオプションも指定せずにEXHIBIT文を実行すると、指定した各一意名または定数が表示される。表示される一意名または定数は、EXHIBIT文に指定された順番に、1行に並べられる。

13.1.7 EXIT（出口）文

EXIT（出口）文は、一連の手続きの共通の 終了点を設ける。
(MF)EXIT文は、内PERFORM、段落、節の出口とすることもできる。
EXIT PROGRAM文は、呼ばれたプログラムの論理的な終点を示す。
(MF)任意の言語で書かれた呼び出し元の実行時要素に、値を返すこともできる。

(ISO2000)(MF)EXIT METHOD文は、起動されたメソッドの論理的な終点を示す。

一般形式

書き方 1

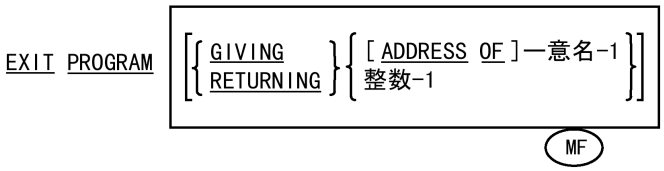
```
EXIT
```

書き方 2

```
EXIT PERFORM [CYCLE]
```
(MF)

書き方 3

書き方 4



書き方 5

構文規則

書き方 1

1. EXIT文は、それだけで完結文を構成しなければならない。
   (VSC2)(MF)この規則は強制しない。
2. EXIT文は、段落中の唯一の完結文でなければならない。
   (VSC2)(MF)この規則は強制しない。

書き方 2

3. 書き方2のEXIT文は、内PERFORMの中だけで使用できる。

書き方 4

4. 完結文内の一連の無条件文の中にEXIT PROGRAM文を指定する場合、これをその最後の文としなければならない。
   (VSC2)(MF)この規則は強制しない。
5. EXIT PROGRAM文を含める段落には、それ以外の完結文を含めてはならない。
   (ANS85)この規則は強制しない。
6. (MF)定数-1は、整数とする。
7. (MF)GIVING指定とRETURNING指定は同義である。
8. (MF)ADDRESS OF句を指定しなかった場合、一意名-1のサイズは８バイトを超えてはならない。

書き方 5

9. (ISO2000)(MF)EXIT METHOD文を指定できるのは、メソッド手続き部の中だけである。

書き方 4 および 5

10. 関連するUSE文中にGLOBAL指定が書かれている宣言手続中にEXIT文を指定してはならない。

一般規則

書き方 1

1. EXIT文は、手続き部中のある点に、利用者が手続き名を付けられるようにするだけである。 この書き方のEXIT文には、翻訳および実行に関して、それ以外の働きはない。

書き方 2

2. (MF)CYCLE指定を伴わないEXIT PERFORM文を実行すると、最も近い内PERFORM文に対応するEND-PERFORMのすぐ後ろに続く、暗黙のCONTINUE文に制御が移される。
3. (MF)CYCLE指定を伴うEXIT PERFORM文を実行すると、最も近い内PERFORM文に対応するEND-PERFORMの直前にある、暗黙のCONTINUE文に制御が移される。

書き方 3

4. (MF)PARAGRAPH指定を伴う書き方3のEXIT文を実行すると、段落の最後の文の直前にある、暗黙のCONTINUE文に制御が移される。
5. (MF)SECTION指定を伴う書き方3のEXIT文を実行すると、節の最後の段落にある最後の文の直前にある、暗黙のCONTINUE文に制御が移される。

書き方 4

6. a (ANS85)呼んだランタイム要素の制御下にないプログラム内でEXIT PORGRAM文を実行すると、次の実行可能な文に制御が移されて、処理が続行される。
7. (ANS85)初期属性を持たない呼ばれたプログラム内でEXIT PROGRAM文を実行すると、呼んだランタイム要素の中のCALL文の次の実行可能な文に制御が移されて、処理が続行される。呼んだランタイム要素の状態はCALL文を実行したときと同じである。ただし、呼んだランタイム要素と呼ばれたランタイム要素の間で共有されるデータ項目およびデータ・ファイルの内容は変更されている可能性がある。呼ばれたプログラムの状態も変更されない。ただし、呼ばれたプログラムによって実行されたすべてのPERFORM文は、その範囲の末尾に達したものとみなされる。
8. (ANS85)初期属性をもつ呼ばれたプログラム内でEXIT PROGRAMを実行することは、上記の7の処理に加えて、そのプログラムを対象とするCANCEL文を実行することに等しい。ただし、メモリーの解放は行われない。（この章で前述の*CANCEL*文を参照。）
9. (ANS85)GLOBAL句が指定されている宣言手続きが実行されている最中に、EXIT PROGRAM文を実行してはならない。ただし、その宣言手続きの実行中に呼ばれたプログラム内で、EXIT PROGRAM文を実行することは構わない。

10. (MF)呼んだランタイム要素の制御下にあるプログラム内のEXIT PROGRAM文を実行すると、システム領域に戻り値が設定される。このシステム領域は一般にCOBOL以外のプログラムが値を返すために利用できる。そのサイズは４バイトを下回ることはなく、環境によってはそれよりも大きいことがある。
(MF)GIVING指定を書かず、呼出し方式に特殊レジスタのRETURN-CODE（前出の「特殊名段落」中の「呼出し方式」を参照）を更新するように指定した場合、オブジェクト・プログラム動作は下記のようになる。具体的には、システム領域がCOBOLの数字データ項目であり、それにUSAGE COPM-5が指定されていて、そのサイズはCOBOLシステムの外の操作環境によって決定されるものとして宣言されている状態で、RETURN-CODEを送出し側としシステム領域を受取り側として、MOVE文を実行したかのように動作する。（RETURN-CODEの詳細については、「COBOL言語の概念」の章の「特殊レジスタ」の項を参照。）
(MF)GIVING ADDRESS OF指定を書いた場合、オブジェクト・プログラム動作は下記のようになる。具体的には、システム領域がUSAGE POINTERと指定されたCOBOLのデータ項目として宣言され、「ADDRESS OF 一意名-9」を最初の作用対象としシステム領域を２番目の作用対象として、SET文を実行したかのように動作する。システム領域よりも大きな値を間接的に渡すために、ADDRESS OF指定を使用できる。
(MF)「GIVING 一意名-1」指定を書いた場合、一意名-1はシステム領域に戻り値を収めるのに必要な長さがあり、その型と用途は呼んだランタイム要素によって期待されるのものでなければならない。一意名-1を送出し側項目とし、システム領域を受取り側項目として、MOVE文を実行したかのように、オブジェクト・プログラムは動作する。
(MF)「GIVING 整数-1」指定を書いた場合、整数-1の値はシステム領域に保持可能なよりも大きくてはならない。オブジェクト・プログラム動作は下記のようになる。具体的には、システム領域がCOBOLの数字データ項目であり、それにUSAGE COPM-5が指定されていて、そのサイズはCOBOLシステムの外の操作環境によって決定されるものとして宣言されている状態で、整数-1を送出し側としシステム領域を受取り側として、MOVE文を実行したかのように動作する。

書き方 5

11. (S02000)(MF)EXIT METHOD文を実行すると、実行中のメソッドが停止され、呼出し元の文に制御が戻される。呼出し元のメソッド定義にRETURNING指定があると、RETRUNING指定によって参照されたデータ項目中の値がそのメソッド呼出しの結果となる。

13.1.8 GOBACK（復帰）文

(OSVS)(VSC2)(MF)

GOBACK（復帰）文は、呼ばれたプログラムの論理的な終点を示す。
(MF)任意の言語で書かれている呼び出し元の実行時要素またはオペレーティングシステム環境へ、値を返すこともできる。

一般形式

構文規則

1. GOBACK文は、一連の無条件文の最後の文とする。
   この規則は強制されないが、その場合GOBACKの後ろに続く文は実行されない。
2. (MF)GIVING 指定とRETURNING指定は同義である。
3. (MF)ADDRESS OF句を指定しない場合、一意名-1によって参照されるデータ項目のサイズは８バイトを超えてはならない。
4. 定数-1は、整数とする。

一般規則

1. 呼び出したランタイム要素の制御下にあるプログラム内でGOBACK文を実行した場合、GOBACK文と同じ句が指定されているEXIT PROGRAM文を実行したかのように、オブジェクト・プログラムは動作する。「EXIT文」の項を参照。
2. 呼び出した ランタイム要素の制御下にないプログラム内でADDRESS OF句を指定せずにGOBACK文を実行した場合、GOBACK文と同じ句が指定されているSTOP RUN文を実行したかのように、オブジェクト・プログラムは動作する。
3. 呼び出した ランタイム要素の制御下にないプログラム内でADDRESS OF句を指定してGOBACK文を実行した場合、STOP RUN文を実行したかのように、オブジェクト・プログラムは動作する。ただし、システム領域に設定される戻り値は不定であることが異なる。GOBACK文を使用する方が、同内容のEXIT PROGRAM文やSTOP RUN文よりも、コードが簡潔になることが多い。
4. GLOBAL句が指定されている宣言手続きが実行されている最中に、GOBACK文を実行してはならない。ただし、その宣言手続きを実行中に呼ばれたプログラム内でGOBACK文を実行することはかまわない。

13.1.9 GO TO（飛越し）文

GO TO（飛越し）文は、手続き部のある部分から別の部分へ制御を移転させる。
(ANS85)書き方1のGO TO文中の手続き名-1を省略することは、ANSI'85標準では廃棄要素に分類されており、ANSI標準の次回の全面改訂の際に削除される予定である。
(MF)この構文は、Micro Focusに組み込まれているすべての方言で全面的に使用できる。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。
(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、書き方1で手続き名を省略することは明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこの文を使用するべきではない。

一般形式

書き方 1

```
GO TO [手続き名-1]
```

書き方 2

```
GO TO 手続き名-1 [手続き名-2]...DEPENDING ON 一意名
```

構文規則

すべての書き方

1. 一意名は、整数をとる数字基本項目の名前である。
2. ALTER（変更）文によって参照される段落は、段落の見出しとその後ろに書いた書き方1のGO TO文だけで構成しなければならない。

書き方 1

3. 手続き名-1を伴わない書き方1のGO TO文は、単一の文からなる段落内にだけ書ける。
   (MF)この規則は強制しない。
4. 完結文の中の一連の無条件文の1つとして、書き方1のGO TO文を使用する場合は、その最後の文とする。

書き方 2

5. 書き方2では、少なくとも2つの手続き名を書く必要がある。
   (ANS85)書き方2で、手続き名が1つだけであってもよい。

一般規則

すべての書き方

1. 書き方1で手続き名-1を指定しない場合は、このGO TO文を参照するALTER文を、このGO TO文の実行に先立って実行しておく。
   (VSC2)(MF)このGO TO文を実行する前に、このGO TO文が含まれる段階を参照するALTERが実行されていない場合には、このGO TO文はCONTINUE（継続）文と同じに扱われる。

書き方 1

2. 書き方1のGO TO文が実行されると、制御は手続き名-1に移される。ただし、ALTER文によってそのGO TO文が変更されている場合には、変更された手続き名に制御が移される。

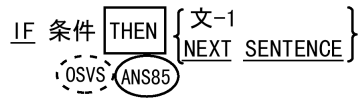
書き方 2

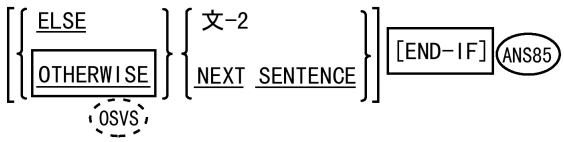
3. 書き方2のGO TO文が実行されると、一意名の値が1，2，... *n*のどれであるかに応じて、手続き名-1、手続き名-2などに制御が移される。一意名の値が正または符号の付かない1，2，... *n*でない場合は、制御の移転は行われず、通常の実行順序の次の文が実行される。

13.1.10 IF（判断）文

価する。前述の*条件式* の節を参照。）その結果が「真」であったか「偽」であったかによって、実行時要素の以降の制御の流れが変わってくる。

一般形式





構文規則

1. 文-1と文-2は、無条件文または条件文を表わす。条件文の前に無条件文を書いてもよい。
2. (ANS85)END-IF指定を書く場合、 NEXT SENTENCE指定を書いてはならない。
   (VSC2)(MF)END-IF指定と共に、NEXT SENTECE指定を書いてもよい。
   NEXT SENTENCE指定が実行されると、制御はEND-IFの次の文ではなく、後続の最も近い終止符（.）の後ろに続く文に移される。

一般規則

1. IF文の範囲は、下記のどれかによって区切られる。
   a. (ANS85)同じ入れ子レベルのEND-IF指定
   b. 分離符の終止符
   c. 入れ子になっている場合、上位レベルのIF文に対応するELSE指定
2. IF文が実行されると、下記の制御の移転が発生する。
   a. 条件が真のとき、文-1を指定してある場合は、その文が実行される。文-1に手続き分岐文または条件文が含まれると、その文の規則に従って明示的に制御が移転される。文-1に手続き分岐文または条件文が含まれないと、ELSE指定は指定してあっても無視されて、IF文の末尾に制御が移される。

b. 条件が真のとき、文-1の代わりにNEXT SENTENCE指定を指定してある場合は、ELSE指定は指定してあっても無視されて、次の実行可能文に制御が移される。
c. 条件が偽のとき、文-1またはその代わりのNEXT SENTENCE指定は無視され、文-2が指定してあればそれが実行される。文-2に手続き分岐文または条件文が含まれると、その文の規則に従って明示的に制御が移転される。文-2に手続き分岐文または条件文が含まれないと、IF文の末尾に制御が移される。ELSE文-2指定を指定してないと、文-1は無視されて、IF文の末尾に制御が移される。
d. 条件が偽のとき、ELSE NEXT SENTENCE指定を指定してある場合は、文-1は指定してあっても無視されて、次の実行可能文に制御が移される。

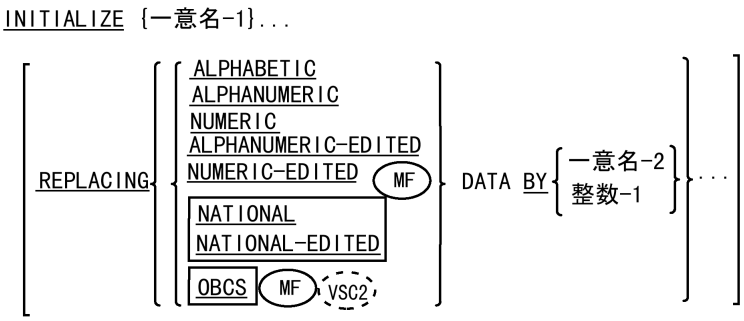
3. 文-1または文-2のどちらか一方または両方に、IF文を含めてもよい。この場合、IF文は入れ子になっている、という。
(ANS85)IF文の中に含まれるIF文は、左から右方向に、IFとELSEとEND-IFの順に組になったものとみなされる。したがって、文中に出てくるELSEは、まだ他のELSEと組み合わされていないか、または明示的にも暗示的にも終了していない、最も近い先行するIFと組み合わされる。文中に出てくるEND-IFは、明示的にも暗示的にも終了していない、最も先行するIFと組み合わされる。

13.1.11 INITIALIZE（初期化）文

(ANS85)

INITIALIZE（初期化）文は、データ項目の種類を選択して、その値を予め定義されている値に設定できるようにする。たとえば、数字データの値をゼロに設定し、 英数字データの値を空白に設定する。

一般形式



構文規則

1. 定数-1および一意名-2によって参照されるデータ項目は、送出し側項目を表わす。一意名-1によって参照されるデータ項目は、受取り側項目を表わす。

2. REPLACING指定内に記述する各項類は、対応する一意名-2によって参照されるデータ項目または定数-1を送出し側として使用した場合の、MOVE（転記）文の受取り側作用対象として許されるものにする。(前述の*MOVE*文を参照。)
3. REPLACING指定内では、同じ項類は繰り返し指定できない。
4. 一意名-1によって参照されるデータ項目またはその下位に属するデータ項目のデータ記述には、OCCURS句のDEPENDING指定が含まれていてはならない。
   (VSC2)一意名-1によって参照されるデータ項目の位置が可変であってはならない。つまり、同じレコード記述内にあってOCCURS句のDEPENDING指定があるデータ項目の後ろに、一意名-1を置くことはできない。ただし、OCCURS句のある項目の下位に一意名-1が属する場合は別である。
   (MF)一意名-1によって参照されるデータ項目は、位置が可変であってもよい。
5. 指標データ項目は、INITIALIZE文の作用対象には指定できない。
6. 一意名-1によって参照されるデータ項目のデータ記述項に、RENAMES句を含めてはならない。
7. (VSC2)(MF)数字の一意名または定数を指定できるところでは、どこでも浮動小数点数のデータ項目または定数を使用できる。
8. (MF)NATIONALおよび NATIONAL-EDITEDフィールドは国別文字の値によってのみ初期化できる。

一般規則

1. 語REPLACINGの後ろに続く必要語は、このマニュアル中に定義したデータの項類を表わす。(*COBOL言語の概念*の章の*データの字類の概念*を参照。)
2. 一意名-1によって参照されるデータ項目が基本項目であっても集団項目であっても、すべての初期化処理は、受取り側項目を基本項目とするMOVE文を1つずつ書いたように実行される。その際、下記の規則が適用される。
   REPLACING指定を書くと、下記のようになる。
   a. 一意名-1が集団項目を指すならば、その下位に属する基本項目のうちで、REPLACING指定に指定した項類に属するものだけが初期化される。
   b. 一意名-1が基本項目を指すならば、それがREPLACING指定に指定した項類に属する場合にだけ初期化される。

   この初期化処理は、一意名-2によって参照されるデータ項目または定数-1を送出し側とし、一意名-1によって参照されるデータ項目受取り側として、暗黙のMOVE文を実行する形で行われる。
   受取り側のすべての基本項目が、すべて初期化の対象となる。それには、表のすべての要素も含まれる。 ただし、下記の一般規則3と4は例外とする。
3. 指標データ項目と
   (VSC2)(MF)ポインタ・データ項目
   と基本FILLERデータ項目は、INITIALIZE文の対象とならない。

4. 受取り側項目の下位に属しかつREDEFINES句が含まれる項目、またはそのような項目の下位に属する項目は、どちらも初期化の対象から外される。ただし、受取り側の一意名自体にはREDEFINES句が含まれていてもよいし、REDEFINES句が含まれるデータ項目の下位に一意名が属していてもよい。
5. REPLACING指定を付けずにINITIALIZE文を実行すると、英字、英数字、英数字編集の項類に属するデータ項目は空白に初期化され、数字、
   (VSC2)(MF)外部浮動小数点数、
   数字編集の項類に属するデータ項目はゼロに初期化される。
   (MF)国別および国別編集の項類に属するデータ項目は国別編集の空白に初期化される。
   この場合、影響を受ける各データ項目は、移送元の定数（つまり、空白またはゼロ）が指定された基本MOVE文の受取り領域であるかのように、処理が行われる。
6. すべての場合において、一意名-1によって参照されるデータ項目の内容は、INITIALIZE文に書かれている順番（左から右）に、初期化される。その過程で集団項目が出てきた場合には、その集団の中で記述されている順に、基本項目が初期化される。
7. 一意名-1と一意名-2が同じ記憶領域を指す場合には、INITIALIZE文の実行結果はどうなるかわからない。両者が同じデータ記述項に定義されていたとしても、そのことは変わらない。(前述の*作用対象の重なり* を参照。)
8. (MF)NATIONAL項目およびNATIONAL-EDITED項目は、国別文字の値で初期化できる。

13.1.12 INSPECT（文字列検査）文

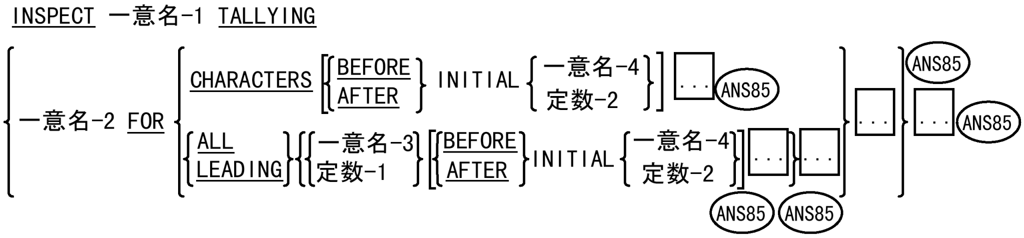
INSPECT（文字列検査）文は、データ項目内に出てくる単一の文字または一連の文字の数を数えたり（書き方1）、置き換えたり（書き方2）、数を数えて置き換えたり（書き方3）、(ANS85)変換したり（書き方4）する。

例

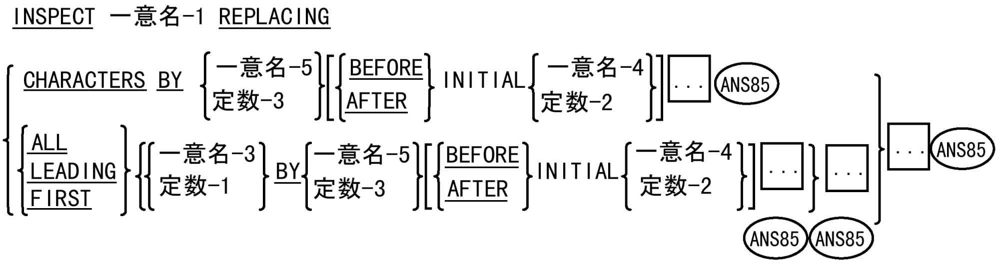
1. INSPECT TALLYING、REPLACING、CONVERTING の使用例は、***言語リファレンス - 追加トピック***の *例* を参照。

一般形式

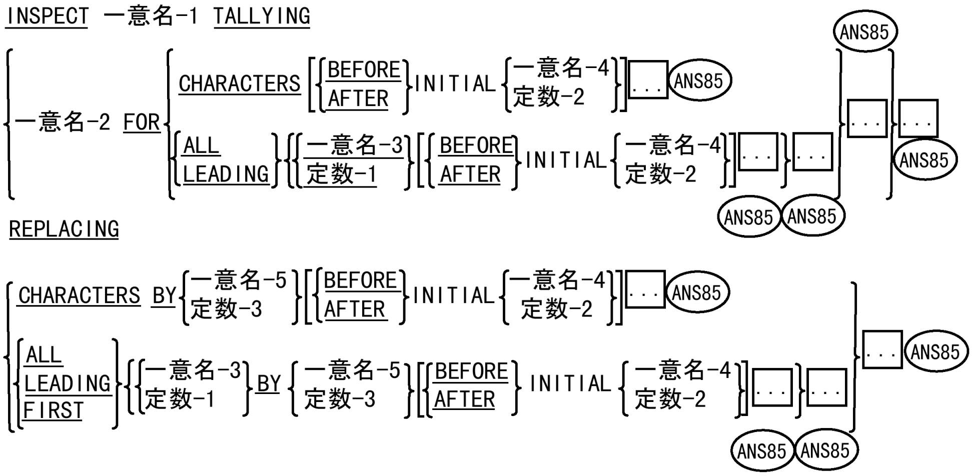
書き方 1



書き方 2



書き方 3



(ANS85)書き方 4

```
INSPECT 一意名-1 CONVERTING { 一意名-6 } TO { 一意名-7 }     (ANS85)
                            { 定数-4   }    { 定数-5   }

    [ { BEFORE } INITIAL { 一意名-4 } ]
    [ { AFTER  }         { 定数-5   } ]
```

構文規則

すべての書き方

1. 一意名-1は、集団項目、あるいは明示的または暗黙的にUSAGE IS DISPLAYと記述された基本項目とする。
2. 一意名-3から一意名-*n* までの一意名は、
   (MF)集団項目、あるいは
   明示的または暗黙的にUSAGE IS DISPLAYと記述された、英字か英数字か
   (ANS85)英数字編集か、数字編集か、
   数字の基本項目とする。
3. 各定数は、文字定数または表意定数とする。ただし、表意定数のALLは使用できない。定数-1、定数-2、定数-4が表意定数のときは、暗黙的に1文字のデータ項目を表わす。
4. 次の各指定には、 BEFORE 指定、
   (ANS85)および/
   または AFTER指定を2つ以上書くことはできない。 ALL、LEADING、 CHARACTERS、FIRST
   (ANS85)または CONVERTING
   指定。
5. 定数-1、定数-2、定数-3、定数-4、定数-5および一意名-3、一意名-4、一意名-5、一意名-6、一意名-7のデータ項目の長さは、定数またはデータ項目が取る限度内ならば、何文字でもよい。
6. (MF)一意名-2（TALLYING項目）を除くすべての一意名は、外部浮動小数点数であってもよい。英数字項目を参照するINSPECT文に関しては、外部浮動小数点項目は英数字に再定義されているように扱われる。

書き方 1 および 3

7. 一意名-2は、基本数字データ項目とする。

書き方 2 および 3

8. 定数-3または一意名-5のデータの大きさは、定数-1または一意名-3のデータの大きさと等しくする。定数-3に表意定数を指定すると、その大きさは、定数-1または一意名-3のデータ項目の大きさに等しく扱われる。
9. CHARACTERS指定を行う場合、定数-3または一意名-5のデータ項目の大きさは1文字にする。

(ANS85)定数-2および一意名-4には、この制限は適用されない。

書き方 4

10. (ANS85)定数-5または一意名-7のデータの大きさは、定数-4または一意名-6のデータの大きさと等しくする。定数-5に表意定数を指定すると、その大きさは、定数-4または一意名-6のデータ項目の大きさに等しく扱われる。
11. (ANS85)定数-4または一意名-6のデータ項目中に、同じ文字が2回以上現れてはならない。

一般規則

すべての書き方

1. (ANS85)長さを判定する目的では、一意名-1は送出し側データ項目であるかのように扱われる。(前述の*OCCURS句*を参照。)
2. 文字列検査を行うためには、比較の周期、BEFORE指定やAFTER指定の範囲の設定、計数または置換の機構などが必要である。これらの設定を含めて、文字列検査は一意名-1のデータ項目の左端の文字位置から開始され、左から右へ、右端の文字位置に達するまで進められる。この文字列検査の進め方には、字類による違いはない。具体的には、下記の一般規則11、12、13に説明する。
3. INSPECT文の中で使用するデータ項目の一意名-1、一意名-3、一意名-4、一意名-5、一意名-6、一意名-7の内容は、下記のように扱われる。
    a. 一意名-1、一意名-3、一意名-4、一意名-5、一意名-6、一意名-7のどれかが英数字と記述されているならば、これらの各一意名の内容は文字列として扱われる。
    b. 一意名-1、一意名-3、一意名-4、一意名-5、一意名-6、一意名-7のどれかが英数字編集か数字編集か符号なし数字項目と記述されているならば、これらの各一意名は英数字に再定義されているように扱われる。(一般規則3aを参照。)INSPECT文は、再定義されたデータ項目を参照するように書かれる。
    c. 一意名-1、一意名-3、一意名-4、一意名-5、一意名-6、一意名-7のどれかが符号付き数字と記述されているならば、データ項目は符号部分を除いたその符号付き数字項目と長さの等しい符号なし数字項目に転記されたように扱われる。そして、その結果に対して上記の一般規則3bが適用される。(後述する*MOVE*文の節を参照。)
4. 一般規則6から19において、定数-1、定数-2、定数-3、定数-4、定数-5に関する規則は、それぞれ、一意名-3、一意名-4、一意名-5、一意名-6、一意名-7のデータ項目の内容にも同様に当てはまる。
5. 一意名の中に添字付きのもの
   (ANS85)または関数一意名
   があると、その添字
   (ANS85)または関数一意名
   は、INSPECT文の実行の最初に1回だけ評価される。

書き方 1

6. (ANS85)一意名-1は、送出し側データ項目である。
7. 必要語のALLとLEADINGは、後続の各定数-1にかかる形容詞である。その効力の範囲は、次の形容詞が出てくるまでである。
8. 一意名-2のデータ項目の内容は、INSPECT文を実行することによって初期化されることはない。
9. 計数に関する規則は、下記のとおり。
    a. ALL指定を書くと、一意名-1の中に定数-1が出現する回数が一意名-2に加えられる。
    b. LEADING指定を書くと、一意名-1の中に定数-1が最初に出現して以降連続して出現する回数が一意名-2に加えられる。ただし、一意名-1の中に定数-1が最初に出現する場所は、定数-1が比較の対象となる最初の比較周期が始まる点とする。
    c. CHARACTERS指定を書くと、一意名-1の中にその文字が出現する回数が一意名-2に加えられる。その数え方は書き方1 および2の一般規則12eに従う。
10. 一意名-1、一意名-3、一意名-4のどれかが一意名-2と同じ記憶領域を占めると、INSPECT文の実行結果はどうなるかわからない。たとえそれらのデータ項目が同じデータ記述項に記述されたものであっても、そのことは変わらない。(前述の*作用対象の重なり*の節を参照。)

書き方 1 および 2

11. 一意名-1のデータ項目の内容を文字列検査する間、その中に定数-1が出現するたびに、計数（書き方1の場合）または定数-3による置換（書き方2の場合）が行われる。
12. 計数または置換するために定数-1の出現を調べる比較は、下記のように行われる。
    a. TALLYING指定または REPLACING指定の作用対象は、INSPECT文中に書かれた順に左から右に取り上げられる。一意名-1のデータ項目の左端の文字位置から開始して、最初の定数-1が、一意名-1中の等しい文字数と比較される。 両者を1文字ずつ比較した結果が等しく、さらに下記の条件のどれかを満たす場合、定数-1と一意名-1中の該当部分とは等しいと判定される。
        1. LEADING指定もFIRST指定も書かれていない。
        2. 定数-1に形容詞LEADINGがかかっている場合、一般規則9および15に照らして、定数-1は先頭に現れている。
        3. 定数-1に形容詞FIRSTがかかっている場合、一般規則9に照らして、定数-1は最初に現れている。
    b. 最初の定数-1が一意名-1中の比較対象部分と一致しないと、後続の定数-1が存在すれば、そのそれぞれについて比較が繰り返される。この比較は、両者が一致するか、後続の定数-1がなくなるまで続けられる。後続の定数-1がない場合には、一意名-1のデータ項目の文字位置のうちで、前回の比較周期で取り上げられた部分のすぐ右隣の部分が、次回の比較周期での比較対象部分として取り上げられる。そして、最初の定数-1との比較が繰り返される。

c. 比較結果が一致すると、一般規則9と15に説明してあるように、計数と置換のどちらか一方または両方が行われる。すると今度は、一意名-1のデータ項目中の現在比較の対象となった部分の右端の文字位置のすぐ右隣の部分が、次回の比較周期での比較対象部分として取り上げられる。そして、最初の定数-1との比較が繰り返される。

d. この比較処理は、一意名-1のデータ項目中の右端の文字位置が比較の対象として取り上げられるか、または比較部分の左端の文字位置とみなされるまで続けられる。この状態に達すると、文字列検査は終了する。

e. CHARACTERS指定を書くと、定数-1に文字を1つ暗黙に指定したものとして、上記の12aから12dに記述した比較処理が行われる。ただし、一意名-1のデータ項目の内容との比較は実際には行われない。この場合、暗黙の定数-1は現在の比較周期において、一意名-1のデータ項目の内容の比較対象部分の左端文字位置とつねに一致するものとみなされる。

13. 一般規則12に定義した比較処理は、BEFORE指定およびAFTER指定の影響を受ける。その様子は下記のとおり。

a. BEFORE指定もAFTER指定も書かないと、定数-1またはCHARACTERS指定の暗黙の作用対象が一意名-1と比較される開始点は、一意名-1の左端の文字位置からとなる。

b. BEFORE指定を書くと、定数-1またはCHARACTERS指定の暗黙の作用対象と比較される一意名-1のデータ項目の内容の範囲は、その左端の文字位置から定数-2が最初に出現するまで（定数-2を含まない）となる。
定数-2が最初に出現する位置は、一般規則12に記述した比較処理の最初の周期が始まる前に判定される。どれかの比較周期において、定数-1またはCHARACTERS指定の暗黙の作用対象が比較の対象とならない場合は、一意名-1のデータ項目の内容と一致しないものとみなされる。一意名-1のデータ項目の中に定数-2が出てこなかった場合は、定数-1またはCHARACTERS指定の暗黙の作用対象は、BEFORE指定がなかったのと同様に比較処理の対象とされる。

c. AFTER指定を書くと、定数-1またはCHARACTERS指定の暗黙の作用対象と比較される一意名-1のデータ項目の内容の範囲は、最初に出現する定数-2の右端のすぐ右の文字位置から末尾までとなる。
定数-2が最初に出現する位置は、一般規則12に記述した比較処理の最初の周期が始まる前に判定される。どれかの比較周期において、定数-1またはCHARACTERS指定の暗黙の作用対象が比較の対象とならない場合は、一意名-1のデータ項目の内容と一致しないものとみなされる。一意名-2のデータ項目の中に定数-2が出てこなかった場合は、定数-1またはCHARACTERS指定の暗黙の作用対象は、まったく比較処理の対象とならない。

書き方 2

14. 必要語のALLとLEADINGとFIRSTは、後続の各BY指定にかかる形容詞である。その効力の範囲は、次の形容詞が出てくるまでである。

15. 置換に関する規則は、下記のとおり。

a. CHARACTERS指定を書くと、一意名-1のデータ項目の中で書き方1と2の一般規則12eに照らして一致する各文字が、定数-3で置き換えられる。
b. ALL指定を書くと、一意名-1のデータ項目の中の定数-1に一致する各文字が、定数-3で置き換えられる。
c. LEADING指定を書くと、一意名-1のデータ項目の中の定数-1に一致する最初以降の連続する各文字が、定数-3で置き換えられる。ただし、一意名-1の中に定数-1が最初に出現する場所は、定数-1が比較の対象となる最初の比較周期が始まる点とする。
d. FIRST指定を書くと、一意名-1のデータ項目の中の定数-1に一致する左端の文字が、定数-3で置き換えられる。

16. 一意名-3、一意名-4、一意名-5のどれかが一意名-1と同じ記憶領域を占めると、INSPECT文の実行結果はどうなるかわからない。たとえそれらのデータ項目が同じデータ記述項に記述されたものであっても、そのことは変わらない。(前述の*作用対象の重なり*の節を参照。)

書き方 3

17. 書き方3のINSPECT文は、同じ一意名-1に対して、書き方1のTALLYING指定と書き方2のREPLACING指定を2つ続けて書いたものとして解釈され実行される。書き方1のINSPECT文の比較照合と計数に関する一般規則、および書き方2のINSPECT文の比較照合と置換に関する一般規則が、書き方3にも当てはまる。

書き方 4

18. (ANS85)書き方4のINSPECT文は、定数-4に含まれる各文字に関して1つずつ、同じ一意名-1に対してALL指定をした、書き方2のINSPECT文を続けて書いたものとして解釈され実行される。これらのALL指定をした個々の仮想のINSPECT文は、定数-1の代わりに定数-4を、定数-3の代わりに定数-5を用いたものと解釈される。定数-4の文字と定数-5の文字との対応関係は、データ項目内の順序位置によって付けられる。
19. (ANS85)一意名-4、一意名-6、一意名-7のどれかが一意名-1と同じ記憶領域を占めると、INSPECT文の実行結果はどうなるかわからない。たとえそれらのデータ項目が同じデータ記述項に記述されたものであっても、そのことは変わらない。

13.1.13 INVOKE（メソッド呼出し）文

(ISO2000)(MF)

INVOKE文はメソッドを呼び出す働きをする。

一般形式

構文規則

1. オブジェクト一意名-1はオブジェクト参照またはクラス名でなければならない。
2. (MF)一意名-1は４バイトのデータ項目でなければならない。

3. 定数-1は英数字の定数でなければならない。
4. 定数-1も定数-2も表意定数であってはならない。
5. (ISO2000)オブジェクト一意名-1が一般的オブジェクト参照を参照する場合、BY CONTENT指定もBY VALUE指定も書いてはならない。また、BY REFERENCE指定を書かなくとも、暗黙的に指定されているものとみなされる。
   (MF)BY CONTENT指定およびBY VALUE指定を書ける。
6. 一意名-2は英数字のデータ項目として定義しなければならない。
7. (ISO2000)一意名-2を指定した場合、オブジェクト一意名-1は一般的オブジェクト参照でなければならない。
   (MF)一般的オブジェクト参照であるために、オブジェクト一意名-1は必須ではない。
8. 一意名-1によって参照されるデータ項目が一般的オブジェクト参照ではない場合、「パラメータおよび返却する項目の準拠規則」に指定されている規則が当てはまる。
9. 一意名-1によって参照されるデータ項目が一般的オブジェクト参照ではなく、引数にBY CONTENT指定またはBY REFERENCE指定がある場合、手続き部の見出し中の対応する仮パラメータにBY REFERENCE指定がなければならない。
10. 一意名-3は、ファイル節、作業場所節、 (MF)オブジェクト記憶節、 局所記憶節、連絡節、通信節のいずれかに定義されているデータ項目を参照しなければならない。
11. (MF)一意名-4、一意名-5、 (MF)一意名-6、一意名-7、 (MF)一意名-8の いずれも、関数一意名であってはならない。
12. 引数にBY VALUEを指定した場合、手続き部の見出し中の対応する仮パラメータにBY VALUE指定がなければならない。
13. (MF)整数-1は符合付きでもゼロでもよい。
14. (MF)GIVINGとRETURNINGは同等である。
15. (MF)一意名-9はサイズが８バイト以内のデータ項目であり、ファイル節、作業場所節、(MF)オブジェクト記憶節、 局所記憶節、連絡節、通信節のいずれかに定義されていなければならない。
16. (MF)一意名-10は連絡節にレベル番号が01または77のデータ項目として定義されていなければならない。
17. 一意名-1によって参照されるデータ項目が一般的オブジェクト参照ではない場合、「パラメータおよび返却する項目の準拠規則」に指定されている規則が当てはまる。
18. 一意名-3は送出し側と受取り側の両方の作用対象である。
19. 一意名-1、一意名-2、一意名-4、一意名-5、一意名-6、一意名-7、一意名-8は送出し側の作用対象である。
20. 一意名-9と一意名-10は受取り側の作用対象である。

一般規則

1. INVOKE文を実行するプログラムまたはメソッドのインスタンスは起動力のあるランタイム要素である。

2. オブジェクト一意名-1または (MF)一意名-1はオブジェクト・インスタンスを指す。オブジェクト一意名-1をクラス名として指定した場合は、そのクラス名のファクトリ・オブジェクトを指す。定数-1、または一意名-2によって参照されるデータ項目の内容は、そのオブジェクト・インスタンスに対して作用するそのオブジェクトのメソッドを指す。
    a. 呼び出す対象のメソッドがCOBOLのメソッドである場合、定数-1、または一意名-2によって参照されるデータ項目の内容は、呼出し対象のメソッドのメソッド名段落に指定されている名前を指す。
    b. 呼び出す対象のメソッドがCOBOLのメソッドではない場合、メソッド名を構成する規則は関係するドメインによって異なる。
3. (MF)AS指定を書くと、COBOLのデータ項目に対してメソッドを呼び出すことができる。テンプレート-1を指定すると、クラス・テンプレートとして使用される。テンプレートを作成することは、クラス・ライブラリ中のオブジェクトによって提供される機能である。（詳細については、『オブジェクトCOBOLを使用してのｏｏプログラミング』を参照。）
4. INVOKE文を実行すると、指定したメソッドが実行可能になり、呼び出されたメソッドに制御が移される。呼び出されたメソッドがCOBOLのメソッドである場合、「手続き部の見出し」に記述されている規則に従って、メソッドが実行される。そうでない場合には、メソッドが記述されている言語の作成者によって定義されている規則に従って、メソッドが実行される。メソッドから制御が戻されると、INVOKE文の最後に制御が移される。
5. 呼び出されたメソッドを実行できるようにするさいに、その所在が把握されるまで、実行単位にはそれがCOBOLのメソッドであるか否かは分からない。メソッド名の形式からCOBOLのメソッドであるか否かを判定することはできない。
6. メソッドの呼出しまたはメソッドの終了のプロセスによって、何らかの外部ファイル結合子に関連するファイルの状態または位置づけが変更されることはない。
7. この処理系では、引数と連絡節内のデータ項目の間の同期を維持することはない。それはユーザーの責任である。

注意： COBOL言語では、データ項目の番地の桁寄せに制約はない。他方、COBOL以外の言語では、通常は番地について前提があり、桁寄せされていないデータ項目が参照されると何らかのエラーを起こす。桁寄せは下記の措置のうちのいくつかを用いて実現される。

- 集団項目に無名項目を追加する
- ALIGNコンパイラ指令を使用するとともに、USING指定中の作用対象を01レベルまたは77レベルのデータ項目にする
- SYNCHRONIZED句とIBMCOMPコンパイラ指令を併用する

INVOKE文にRETURNING指定を書いた場合、COBOL以外のメソッドは適切な形式の結果を返さなければならない。

8. INVOKE文のUSNIG指定の引数と呼び出されるメソッドの手続き部の見出しのUSING指定の中の対応する仮パラメータの順序によって、引数と仮パラメータの対応関係が決まる。この対応は位置で決まるのであって、名前が等しいことによるのではない。最初の引数が最初の仮パラメータに対応し、次にそれらの２番目同士が対応する、といったぐあいである。
   パラメータが指標名の場合は、そのような対応関係は確立されない。呼び出されたメソッドと呼び出すランタイム要素の指標名は常に別々の指標を参照する。
   起動されたランタイム要素上のUSING指定の効果については、「手続き部の見出し」の一般規則に記述されている。
9. INVOKE文のUSNIG指定の中で参照されるパラメータの値は、INVOKE文が実行されたときに、呼び出されるメソッドで利用できるようになる。
10. BY CONTENT、BY REFERENCE、BY VALUEの各指定はその後ろに続くパラメータに効力を及ぼすが、別のBY CONTENT、BY REFERENCEまたはBY VALUEの指定が出てきたところでその効力は停止する。
    最初のパラメータの前にBY CONTENT、BY REFERENCE、BY VALUEのどの指定もないと、BY REFERENCE指定があるものとみなされる。
11. BY REFERENCE ADDRESS OF指定が明示的または暗黙的に指定されている場合、USAGE POINTERを用いてデータ項目が宣言され、「SET データ項目 TO ADDRESS OF 一意名-4」文によって取得された値がBY REFERENCEによってそのデータ項目に渡されたかのように、メソッドは動作する。
12. (MF).一意名-4が連絡節にあってそのレベル番号か01または77以外であるか、または作業場所節にある場合、そのデータ項目がBY CONTENTによって渡され、一意名-4の番地を呼び出されたメソッドによっては変更できないことに等しい。「BY REFERENCE 定数-2」を明示的または暗黙的に指定すると、メソッドは定数-2を定数-3のために記述されたものとして処理する。
13. (MF)「LENGTH OF 一意名-6」を指定すると、引数のデータ記述は、形式がPIC S9 (9) USAGE BINARYで内容が一意名-6に割り当てられた記憶領域のバイト数に設定されたものと等しくなる。
14. (MF)整数-1を指定すると、引数の暗黙的なデータ記述は、符合付き数字項目でUSAGE COMP-5に等しくなる。その記憶領域内でのバイト数は、整数-2の指定があればその値によって示され、指定がなければ４バイトとなる。
15. (MF)「LENGTH OF 一意名-8」を指定すると、引数のデータ記述は、形式がPIC S9 (9) USAGE BINARYで内容が一意名-8に割り当てられた記憶領域のバイト数に設定されたものと等しくなる。
16. (MF)一意名-10を指定すると、呼び出されたメソッドは、明示的または暗黙的にUSAGE POINTERと指定された値を返す。メソッドの結果は、SET文の規則に従って、一意名-10に代入される。
17. 呼び出されるメソッド中にINVOKE文が含まれていてもよい。呼び出されるメソッドは、呼び出すメソッドを直接的または間接的に呼び出すINVOKE文を実行できる。

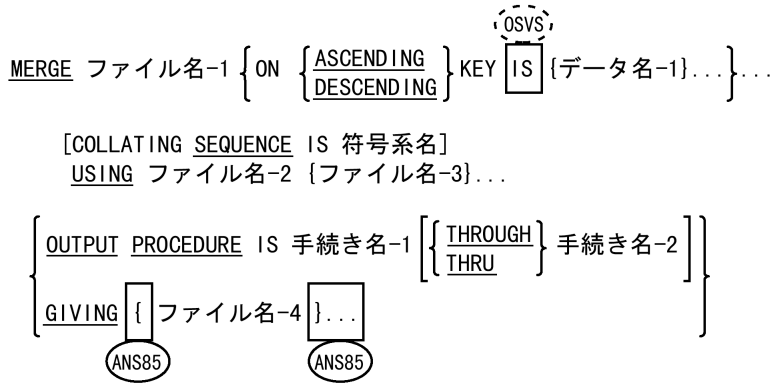
第14章: 手続き部 - MERGE - OPEN

14.1 COBOLの文

14.1.1 MERGE（併合）文

The MERGE（併合）文は、指定されたキーの組に従って同じようにそろえられているいくつかのファイルを、1つのファイルに併合する。そして、併合された順番に従って、出力手続きまたは出力ファイルへレコードを引き渡す。

一般形式



構文規則

1. ファイル名-1は、 データ部の整列併合ファイル記述項に記述しておく。
2. 手続き名-1は、出力手続きの名前を表わす。
3. 手続き名-1および手続き名-2は、節名とする。
   (ANS85)(OSVS)この制約は解除された。
4. (ANS85)ファイル名-1のファイルに含まれるレコードが可変長である場合、ファイル名-2およびファイル名-3のファイル中に含まれるレコードの大きさは、ファイル名-1のレコードの最小のもの以上かつ最大のもの以下にする。ファイル名-1のファイルに含まれるレコードが固定長である場合、ファイル名-2およびファイル名-3のファイル中に含まれるレコードの大きさは、ファイル名-1のレコードの最大のものを超えてはならない。
5. ファイル名-2、ファイル名-3、ファイル名-4は、データ部の中の整列併合ファイル記述項にではなく、ファイル記述項に記述しておく。
6. THRUとTHROUGHは同義語である。
7. データ名-1はキーのデータ名であり、下記の規則に従う。

a. キー・データ名で示すデータ項目は、ファイル名-1のレコード中に記述しておく。
b. キー・データ名は、修飾してもよい。
c. キー・データ項目は、可変長であってはならない。
d. ファイル名-1に複数のレコード記述を含める場合、キー・データ名で示すデータ項目は1つのレコード記述の中だけに指定する。1つのレコード記述の中でキー・データ名に指定した文字位置が、そのファイルの他のすべてのレコードのキーとして扱われる。
e. キー・データ名には、OCCURS句を含む記述項は記述できない。また、OCCURS句を含む記述項の下位に、キー・データ名を属させてはならない。
f. (OSVS)(VSC2)(MF)キー・データ項目は、浮動小数点項目であってもよい。
g. (OSVS)(VSC2)(MF)キー・データ項目が外部浮動小数点数である場合、キーは英数字として扱われる。レコードが併合される順序は、使用する文字の大小順序に左右される。
h. (OSVS)(VSC2)(MF)キー・データ項目が内部浮動小数点数である場合、キーの順序は数値順となる。

8. MERGE文には、複数ファイル・リール中のファイル名を複数指定してはならない。
9. MERGE文の中でファイル名を繰り返し指定してはならない。
10. MERGE文は、手続き部の宣言部分以外のどこにでも置くことができる。
11. (ANS85)ファイル名-4が索引ファイルである場合、最初のデータ名-1には ASCENDINGを指定する。また、データ名-1がレコード上で占める文字位置は、そのファイルの主レコードキーと同じにする。
12. (ANS85)GIVING を指定し、その対象であるファイル名-4のファイルに含まれるレコードが 可変長である場合、ファイル名-1のファイル中に含まれるレコードの大きさは、ファイル名-4のレコードの最小のもの以上かつ最大のもの以下にする。ファイル名-4のファイルに含まれるレコードが固定長である場合、ファイル名-1のファイル中に含まれるレコードの大きさは、ファイル名-4のレコードの最大のものを超えてはならない。
13. USING句またはGIVING句に10個以上のファイル名を指定したい場合には、コンパイラ指令のCALLSORT"EXTSM"を使用しなければならない。そうすると、255個までのファイルを指定できるようになる。

一般規則

1. MERGE文は、ファイル名-2およびファイル名-3のファイルに含まれるすべてのレコードを併合する。
2. ファイル名-1のファイルに含まれるレコードが 固定長である場合、ファイル名-2またはファイル名-3のファイル中にファイル名-1の固定長レコードよりも短いものがあると、ファイル名-2またはファイル名-3のレコードがファイル名-1のファイルに引き渡されるときに、その後部に空白が埋められる。

3. 語KEYの後ろに続くデータ名は、KEY指定での区切り方とは関係なく、強さの順に左から右に並べる。つまり、一番左のデータ名が最も強いキーであり、その右のデータ名が次に強いキーである、といった具合いである。
    a. ASCENDING を指定した場合、比較の規則に従って、キー・データ項目の内容の値の小さい方から大きい方へと、順に併合される。
    b. DESCENDINGを指定した場合、比較の規則に従って、キー・データ項目の内容の値の大きい方から小さい方へと、順に併合される。
4. 比較の規則に従って、あるデータレコードのすべてのキー・データ項目の内容が、他のいくつかのデータレコードの対応するキー・データ項目と等しい場合、引き取りの順序は下記のようになる。
    a. MERGE文に指定した入力ファイルの順。
    b. 1つの入力ファイル内にキー・データ項目の値が等しいレコードが複数ある場合は、それらがすべて他の入力ファイルからよりも先に引き取られる。
5. 文字のキー・データ項目の比較に適用される文字の大小順序は、 MERGE文の実行開始時に、下記の優先順序で決定される。
    a. 最初は、指定してあれば、MERGE文中の COLLATING SEQUENCEに指定した文字の大小順序。
    b. 次は、プログラム用に設定してある文字の大小順序。
6. ファイル名-2およびファイル名-3のファイル中のレコードが、MERGE文中に指定したASCENDING KEYまたは DESCENDING KEYの順に整列されていないと、併合処理の結果はどうなるかわからない。
7. ファイル名-2およびファイル名-3のファイル中のすべてのレコードが、ファイル名-1のファイルに引き渡される。MERGE文の実行開始時にファイル名-2およびファイル名-3のファイルが開かれていてはならない。MERGE文を実行すると、ファイル名-2およびファイル名-3の各ファイルに対して、下記の操作が行われる。
    a. ファイルの処理が開始される。これはINPUT指定をしたOPEN文を、暗黙的に実行することである。出力手続きを指定すると、この開始処理は出力手続きに制御が移される前に行われる。
    b. 論理レコードが読み込まれて、併合操作に引き渡される。これはNEXTおよびAT ENDを指定したREAD文を、暗黙的に実行することである。
       ファイル名-1のファイルのレコードが可変長であると、ファイル名-1のファイルに書き出されるレコードの大きさは、ファイル名-2またはファイル名-3から読み込んだときのレコードの大きさとなる。このことは、ファイル名-1のファイルの整列併合ファイル記述項に指定した、RECORD IS VARYING句またはOCCURS句のDEPENDING ON指定の対象となっているデータ項目の内容にかかわらず適用される。
       相対ファイルに関しては、相対キー・データ項目の内容がMERGE文の実行後にどうなっているかはわからない。
    c. ファイルの処理が終了される。これは任意指定の何もないCLOSE文を暗黙的に実行することである。出力手続きを指定した場合は、その中の最後の文に制御が移されるまで、この終了処理は実行されない。

USE AFTER EXCEPTION/ERROR手続きを指定してあれば、この暗黙のCLOSE処理において、それも対象となる。

ファイル名-2またはファイル名-3のファイル記述項中に指定した、RECORD IS VARYING句のDEPENDING ON指定の対象となっているデータ項目の内容は、MERGE MERGE文の完了時点でどのようになっているかわからない。

8. 出力手続きには、RETURN（引き取り）文を用いてファイル名-1のファイルから順に1件ずつ引き取った併合済みレコードを、選択したり変更したり複写したりする任意の必要な手続きを組み込むことができる。この出力手続きの範囲には、その中からCALL文、EXIT文、GO TO文、PERFORM文によって制御を移された結果実行される、すべての文が含まれる。また、この出力手続きの範囲内の文を実行した結果として実行される宣言手続き中のすべての文も、この出力手続きの範囲に含まれる。この出力手続きの範囲内で、MERGE文、RELEASE文、SORT文を起動するようなことがあってはならない。(*COBOL言語の概念*の章の*明示指定と暗黙指定*の節を参照。)
9. 出力手続きを指定すると、MERGE文の実行中にその手続きに制御が渡される。コンパイラは出力手続きの最後の文の末尾に復帰機構を組み込む。出力手続きの最後の文に制御が移されると、復帰機構によってMERGE文の処理が終了されて、MERGE文の後ろの次の実行可能文に制御が移される。出力手続きに制御が移される前に、併合処理は、要求されれば併合された次の順序のレコードを引き渡せる状態に達している。出力手続き中のRETURN文は、そのレコードを引き取るための要求である。
10. 出力手続きの実行中に、ファイル名-2、ファイル名-3、ファイル名-4のファイルを操作したり、そのレコード領域にアクセスしたりする文を実行することはできない。
11. GIVINGを指定すると、暗黙の出力手続きによって、併合されたすべてのレコードはファイル名-4のファイルに書き出される。MERGE文の開始時点で、ファイル名-4のファイルが開かれていてはならない。 MERGE文を実行すると、ファイル名-4の各ファイルに対して、下記の処理が行われる。
    a. ファイルの処理が開始される。これはOUTPUT指定をしたOPEN文を、暗黙的に実行することである。
    b. 論理レコードが引き取られてファイルに書き出される。これは任意指定を何も含まないWRITE文を、暗黙的に実行することである。

       ファイル名-4のファイルのレコードが可変長であると、ファイル名-4のファイルに書き出されるレコードの大きさは、ファイル名-1から読み込んだときのレコードの大きさとなる。このことは、ファイル名-4のファイルの整列併合ファイル記述項に指定した、RECORD IS VARYING句またはOCCURS句のDEPENDING ON指定の対象となっているデータ項目の内容にかかわらず適用される。

       相対ファイルに関しては、相対キー・データ項目の値は、最初のレコードが引き取られたときは"1"、2番目のレコードが引き取られたときは"2"という具合いに設定される。MERGE文の実行が終了した後で、相対キー・データ項目の内容がどうなっているかはわからない。
    c. ファイルの処理が終了される。これは任意指定の何もないCLOSE文を、暗黙的に実行することである。

USE AFTER EXCEPTION/ERROR手続きを指定してあれば、上記の暗黙の処理において、それも対象となる。しかし、そのUSE手続きの中から、ファイル名-4のファイルを操作したりそのレコード領域をアクセスしたりするような文が実行されるようなことがあってはならない。外部的に定義されているファイルの境界を超えて最初にレコードが書き出されると、そのファイルに指定してあるUSE AFTER EXCEPTION/ERROR手続きが実行される。そのUSE手続きから制御が戻されたとき、またはそのようなUSE手続きが指定されていないときは、上記の11cと同様にファイルの処理が終了される。

ファイル名-1のファイル記述項中に指定した、RECORD IS VARYING句のDEPENDING ON指定の対象となっているデータ項目の内容は、GIVINGを指定したMERGE文が完了した時点でどのようになっているかはわからない。

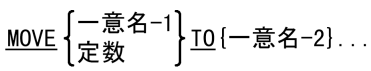
12. ファイル名-4のファイルに含まれるレコードが固定長である場合、ファイル名-1のファイル中にファイル名-4の固定長レコードよりも短いものがあると、ファイル名-4のファイルにレコードが引き渡されるときに、その後部に空白が埋められる。
13. ***言語リファレンス - 追加トピック***中の*区分化*の章に示すように、 MERGE文を含むプログラムを区分化できる。ただし、下記の制限がある。
    a. 独立区分ではない区分中の節内にMERGE文が存在する場合、そのMERGE文から呼び出される出力手続きは、下記のどちらかの状態で組み込まれていなければならない。
        1. 非独立区分中に完全に含まれる。
        2. 単一の独立区分中に完全に含まれる。
    b. 独立区分中にMERGE文が存在する場合、そのMERGE文から呼び出される出力手続きは、下記のどちらかの状態で組み込まれていなければならない。
        1. 非独立区分中に完全に含まれる。
        2. MERGE文と同じ独立区分中に完全に含まれる。
14. (OSVS)(VSC2)(MF)特殊レジスタの SORT-RETURN が MERGE文の原始要素に対して用意されている。 この特殊レジスタには、併合処理が終わったときに0（正常終了）または16（不成功）が設定される。出力手続きの中でこの特殊レジスタに値16を設定して、途中で併合処理を終わらせることができる。この場合、併合処理は次のRETURN文のところで終了される。

14.1.2 MOVE（転記）文

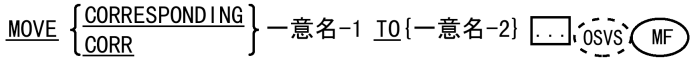
MOVE（転記）文は、編集規則に従って、データをいくつかのデータ領域に移す。

一般形式

書き方 1



書き方 2



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - BYTE-MODE-MOVE - 送出し側項目と受取り側項目がその記憶領域を部分的に共有する（つまり、重複部分のある）MOVE文の動作に影響を与える。
    - DBCS (DBSPACE) - ２バイト文字の受取り側項目に埋める空白に使用する文字に影響を与える。
    - DE-EDIT - 数字編集送出し側フィールドの値をそのPICTURE句によって決定するか否かを制御する。
    - TRUNC - 受取り側のバイナリ値のサイズよりも大きい値が送出し側項目に含まれているときに、利用可能な記憶領域またはPICTURE句に基づいて、切捨てを行うかどうかを決定する。

構文規則

すべての書き方

1. 一意名-1および定数は送出し側を表わし、一意名-2は受取り側を表わす。
2. 指標データ項目、
   VSC2 MF ポインタ・データ項目、手続きポインタ項目
   は、MOVE文の作用対象にすることはできない。(前述の*USAGE句*を参照。)

3. (VSC2)(MF)送出し側項目か受取り側項目のどちらか一方が2バイト文字（USAGE DISPLAY-1）である場合には、両方とも2バイト文字でなければならない。表意定数のSPACEは、2バイト文字の送出し側として使用できる。

書き方 1

4. (OSVS)(VSC2)(MF)書き方1では、すべての一意名は集団項目でも基本項目でもよい。送出し側のデータが、各一意名-2のデータ項目へ指定された順番に転記される。
5. (MF)一意名-1には数字の関数一意名を指定できる。

書き方 2

6. CORRは、CORRESPONDINGの省略形である。
7. CORRESPONDING指定を書くときは、すべての一意名は集団項目とする。
8. (VSC2)(MF)MOVE CORRESPONDING文中の集団項目内に、ポインタ・データ項目が含まれていてもよい。しかし、ポインタ・データの転記はいっさい行われない。

一般規則

すべての書き方

1. (MF)一意名-1が数字の関数一意名であるときは、一意名-2は数字項目か数字編集項目のどちらかでなければならない。
2. 定数または一意名-1によって指定されるデータが、一意名-2によって指定される各データ項目に指定された順番で転記される。一意名-2に関する規則は、他の受取り側にも適用される。一意名-2が添字付けまたは指標付けされていると、そのデータ項目に転記が行われる直前に添字または指標が評価される。

   (ANS85)一意名-1が部分参照されているか、添字付けされているか、あるいは関数一意名である場合、その部分参照、添字、関数一意名の評価は、受取り側の最初のデータに転記が行われる直前に、1回だけ行われる。

   (ANS85)下記の文は

   ```
   MOVE a(b) TO b, c(b)
   ```

   下記の3つの文に相当する。

   ```
   MOVE a(b) TO temp
   MOVE temp TO b
   MOVE temp TO c(b)
   ```

   ここで、temp は中間結果を保持する項目であり、COBOLコンパイラによって用意される。
3. 送出し側項目も受取り側項目も基本項目であるMOVEを、基本項目転記という。各基本項目は次の項類のどれかに属する。具体的には、数字、英字、英数字、数字編集、英数字編集、

   (OSVS)(VSC2)(MF)浮動小数点数のどれかに属する。

これらの項類はPICTURE句に説明してある。数字定数は、数字の項類に属する。文字定数は、英数字の項類に属する。表意定数のZEROは、数字項目または数字編集項目に転記される場合は数字の項類に属し、それ以外の場合には英数字の項類に属する。表意定数のSPACEは、英数字の項類に属する。その他の表意定数は、すべて英数字の項類に属する。

これらの項類の間の基本項目転記に関して、下記の規則が適用される。

a.

1. 表意定数のSPACE、英数字編集、英字のデータ項目を、数字または数字編集のデータ項目に転記してはならない。
2. 数字編集のデータ項目を、数字または数字編集のデータ項目に転記してはならない。

   (ANS85)この制限は撤廃された。

b. 数字定数、表意定数のZERO、数字データ項目、数字編集データ項目を、英数字データ項目に転記してはならない。

c. 整数でない数字定数または整数でない数字データ項目を、英数字または英数字編集のデータ項目に転記してはならない。

d. 上記以外の基本項目転記は文法的に正しく、下記の一般規則5に示す規則に従って実行される。

4. 文法的に正しい基本項目転記が実行されるときには、ある形式の内部表現から別の形式の内部表現へのデータの変換が、必要に応じて行われる。また、受取り側データ項目に指定されている編集

(ANS85)または暗黙的に示されている逆編集

もあわせて行われる。

a. 受取り側が英数字編集または英数字のデータ項目である場合、*COBOL言語の概念*の章の*標準桁寄せ規則*に定義されているように、桁寄せと必要な空白の穴埋めが行われる。送出し側の項目の大きさの方が、受取り側の項目の大きさよりも大きい場には、受取り側の項目がいっぱいになった後で、右側の過剰な文字が切り捨てられる。送出し側の項目が符号付きの数字である場合、演算符号は転記されない。演算符号が独立の文字位置を占める(前述の*SIGN句*を参照）場合、演算符号は転記されず、送出し側の項目は実際の大きさよりも1文字小さいものとして扱われる（標準データ形式として）。

b. 受取り側が数字または数字編集のデータ項目である場合、*COBOL言語の概念*の章の*標準桁寄せ規則*に定義されているように、桁寄せと必要なゼロの穴埋めが行われる。ただし、ゼロを置き換える編集が行われている場合は例外である。

   (ANS85)送出し側の項目の項類が数字編集である場合、その項目の編集されていない値を取り出すために、暗黙の 逆編集が行われる。逆編集された値には、符号が付くことがある。次いで、編集されていない値が受取り側の項目に転記される。逆編集の効果は、DE-EDIT指令の影響を受ける。

受取り側の項目が符号付きの数字である場合、送出し側の符号は受取り側項目に転記される。(前述の*SIGN句*を参照。) 必要に応じて、符号の表現形式の変換が行われる。送出し側の項目に符号が付いていない場合には、受取り側項目用に正符号が付加される。

受取り側の項目が符号なしの数字である場合、送出し側の項目の絶対値が受取り側の項目に転記され、符号は付けられない。

送出し側の項目の項類が英数字である場合、送出し側の項目が符号なしの整数として記述されているように、データが転記される。

受取り側の項目が数字または数字編集であり送出し側の項目が英数字である場合、送出し側の項目の内容が整数でないと、結果はどうなるかわからない。送出し側の英数字項目が整数でない場合は、エラーが報告されて、ゼロがその対象に転記される。(前述の*矛盾するデータ*を参照。)

(ANS85)送出し側の項目が数字である場合、転記し戻すと数字編集項目に同じ値が現れるように転記操作が行われる (ただし、切捨てが発生した場合は除く)。このデータ項目に編集文字列に適合しないデータが含まれる場合、受取り側にはゼロが転記される。

c. 受取り側の項目が英字である場合、*COBOL言語の概念*の章の*標準桁寄せ規則*の節に定義されているように、桁寄せと必要な空白の穴埋めが行われる。送出し側の項目の大きさの方が受取り側の項目の大きさよりも大きい場合には、受取り側の項目がいっぱいになった後で、右側の過剰な文字が切り捨てられる。

d. (OSVS)(VSC2)(MF)受取り側の項目が浮動小数点数である場合、送出し側の項目がまず最初に内部浮動小数点数に変換され、それから転記される。外部浮動小数点数項目へ、または外部浮動小数点数項目からデータを転記する場合は、該当データはいったん内部浮動小数点数に変換される。

e. (VSC2)(MF)受取り側の項目が2バイト文字である場合、送出し側の項目も2バイト文字でなければならない。送出し側の項目の大きさと受取り側の項目の大きさが異なる場合は、送出し側の項目の右側が切り捨てられるか、受取り側の項目の右側に2バイト文字の空白が埋められるかする。

5. 基本項目転記以外の転記は、英数字の基本項目同士の転記とまったく同じように扱われる。ただし、データの内部表現形式の変換は行われない。この形の転記においては、送出し側または受取り側の項目の下位に属する個々の基本項目または集団項目を考慮することなく、受取り側の項目にデータが詰め込まれる。ただし、OCCURS句の一般規則に述べたことは例外である。
6. 表 14-1 に、各種のMOVE文の適合性をまとめて示す。表の中に示した一般規則の参照番号は、該当事項を禁止または裏付けている規則を示す。

書き方 2

7. CORRESPONDING指定を書くと、一意名-1に属する該当する項目が、一意名-2に属する対応する項目に転記される。これに適用される規則については、前述の *CORRESPONDING指定*に説明してある。CORRESPONDING指定が含まれるMOVE文を実行することは、対応する一意名の間で個々に転記を行う MOVE文を実行することに等しい。
(OSVS)(VSC2)この処理は、各宛先集団ごとに繰り返される。

表 14-1: MOVE文のデータ・カテゴリ

送出し側のデータ項目の項類	受取り側のデータ項目の項類[1]				
	英字	英数字/英数字編集	数字（整数/非整数）	数字編集外部/内部浮動小数点数	2バイト文字
英字	Yes/4c	Yes/4a	No/3a	No/3a	No/4e
英数字	Yes/4c	Yes/4a	Yes/4b[2]	Yes/4b	No/4e
英数字編集	Yes/4c	Yes/4a	No/3a	No/3a	No/4e
数字:整数	No/3b	Yes/4a	Yes/4b	Yes/4b	No/4e
数字:非整数	No/3b	No/3c	Yes/4b	Yes/4b	No/4e
数字編集	No/3b	Yes/4a	Yes/4b	Yes/4b	No/4e
2バイト文字	No/4e	No/4e	No/4e	No/4e	Yes/4e

1. 各欄に該当する一般規則の番号を示す。
2. 非整数の英数字定数が送出し側として使用された場合、エラーとなる。

14.1.3 MULTIPLY（乗算）文

MULTIPLY（乗算）文は、数字データ項目どうしを掛け合わせて、その結果をいくつかのデータ項目に入れる。

一般形式

書き方 1

```
MULTIPLY { 一意名-1 } BY{一意名-2 [ROUNDED]}...
         { 定数-1  }

[ON SIZE ERROR 無条件文-1]

[NOT ON SIZE ERROR 無条件文-2]
[END-MULTIPLY]                    (ANS85)
```

書き方 2

```
MULTIPLY { 一意名-1 } BY { 一意名-2 }
         { 定数-1   }    { 定数-2   }

                    GIVING{一意名-3 [ROUNDED]}...

[ON SIZE ERROR 無条件文-1]

[NOT ON SIZE ERROR 無条件文-2]
[END-MULTIPLY]
```

ANS85

構文規則

すべての書き方

1. 各一意名は、数字基本項目を指さなければならない。ただし、書き方2の語GIVINGに続く各一意名は、基本数字項目の他に基本数字編集項目であってもよい。
2. 各定数は、数字定数とする。
3. 作用対象を合成したものは、 18桁を超えてはならない。ここで、作用対象の合成とは、すべての受取り側のデータ項目の小数点の位置をそろえて重ね合わせることによって得られる、仮想のデータ項目である。
4. OSVS VSC2 MF 数字データ項目または数字定数を指定できるところでは、どこでも浮動小数点数データ項目および浮動小数点数定数を使用できる。

一般規則

すべての書き方

1. 前述の*ROUNDED指定*、*ON SIZE ERROR指定*、*算術文*、*作用対象の重なり*、*算術文における複数個の答え*の各節を参照。

書き方 1

2. 書き方1では、一意名-1または定数-1の値に一意名-2の値が掛けられる。乗数（一意名-2）の値は、積によって置き換えられる。同様に、一意名-2の後続の値に対して、左から右に同じ処理が繰り返される。

書き方 2

3. 書き方2では、一意名-1または定数-1の値に一意名-2または定数-2の値が掛けられる。結果は、各一意名-3のデータ項目に入れられる。

14.1.4 NEXT SENTENCE（次の完結文）文

(MF)

NEXT SENTENCE（次の完結文）文は、制御を次のCOBOL完結文に移す。つまり、次の終止符（.）に続く完結文に制御を移す。これに対し、 動詞CONTINUEは、論理的に次のCOBOLの動詞に制御を移す。

一般形式

```
NEXT SENTENCE
```

構文規則

1. 条件文または無条件文を書けるところには、どこでもNEXT SENTENCEを使用できる。

一般規則

1. NEXT SENTENCE文は、次の終止符（.）に続く、論理的に次となるCOBOLの動詞に、 実行の流れを変えさせる。

14.1.5 NOTE（注記）文

(OSVS)

NOTE（注記）文は、完結文または段落を注記として扱う。

一般形式

```
NOTE 文字列
```

構文規則

1. 文字列には、計算機の文字集合に属する文字を自由に組み合わせて書ける。

一般規則

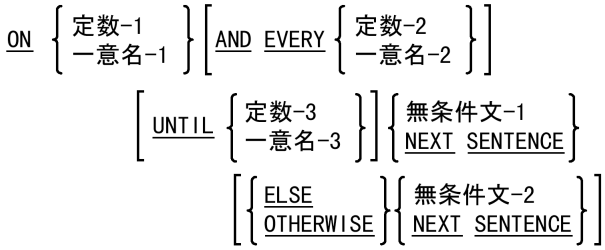
1. 段落の最初の完結文がNOTE文であると、その段落全体が注記として扱われる。
2. 段落の冒頭以外の場所にNOTE文を置くと、次の分離符である終止符（.）が出てくるまでのテキストが注記として扱われる。

14.1.6 ON（回数間隔）文

OSVS

ON（回数間隔）文は、手続き文を周期的に実行できるようにする。

一般形式



構文規則

1. 一意名-1、一意名-2、一意名-3は、符号なしの整数の基本項目とする。
2. 定数-1、定数-2、定数-3は、符号なしの数字定数とする。

一般規則

1. 各ON文が最初に実行される前に、 そのON文用に暗黙的に定義されたカウンタ（暗黙の ONカウンタ）がゼロに初期化される。
2. 指定する一意名-1、一意名-2、一意名-3は、ON文が実行されるときには、正の整数値をとらなければならない。次にON文を実行するまでの間に、これらの値を変更すると、以降のON文の実行に影響が出る。
3. 暗黙のONカウンタの内容は、 実行の流れをそのON文に移さないかぎり、変更されることはない。（呼ばれたプログラムのONカウンタを設定し直すには、そのプログラムを取り消すしかない。EXIT PROGRAM文を実行し、その後そのプログラムを再び呼び出した場合、CANCEL文をはさんでいなければONカウンタの値は元のままである。）
4. 下記の値リストが評価される。
    a. 一意名-1または定数-1の現在の値
    b. 一意名-1または定数-1の現在の値に、一意名-2または定数-2の現在の値を繰り返し加えた一連の値。この繰り返しは、一意名-1または定数-1の現在の値が、一意名-3または定数-3の値に達するまで行われる。

    次いで、暗黙のONカウンタが上記の一連の値と比較される。一連の値の中にONカウンタと一致するものがあれば、無条件文-1が実行される。そうでなければ、無条件文-2が実行される。

14.1.7 OPEN（開く）文

The OPEN（開く）文は、ファイルの処理を行うための準備（ファイルのオープン）をする。また、見出しの確認や書込み、その他の入出力操作を行う。
(ANS85)OPEN文のREVERSED指定は、ANSI '85標準では廃要素に分類されており、ANSI標準の次回の全面改訂の際に、削除される予定である。
(MF)この構文は、Micro Focus COBOLに組み込まれているすべての方言で全面的に使用できる。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。
(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、OPEN文のREVERSED指定は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこの文を使用するべきではない。

一般形式

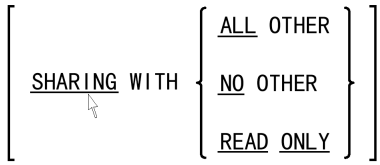
書き方 1 (レコード順ファイル)

```
       { INPUT[共有句] { ファイル名-1 [ REVERSED                  ] }...  }
       {                {             [ WITH { NO REWIND }        ] }     }
       {                {             [      { LOCK      }        ] }     }
OPEN   { OUTPUT[共有句] { ファイル名-2 [ WITH { NO REWIND } ] } ...        }...
       {                {             [      { LOCK      } ] }            }
       { I-O[共有句] {ファイル名-3 [WITH LOCK]}...                         }
       { EXTEND[共有句] {ファイル名-4 [WITH LOCK]}...                      }
```

書き方 2 (行順ファイル、相対ファイル、索引ファイル)

```
       { INPUT[共有句]{ファイル名-1 [WITH LOCK]}...  }
       { OUTPUT[共有句]{ファイル名-2 [WITH LOCK]}... }
OPEN   { I-O[共有句]{ファイル名-3 [WITH LOCK]}...    }...
       { EXTEND[共有句]{ファイル名-4 [WITH LOCK]}... }
```

SHARING句の部分:



構文規則

すべての書き方（すべてのファイル）

1. OPEN文が対象とするファイルは、編成または呼出し法が同じである必要はない。
2. (ISO2000)(MF)SHARING ALLを指定してINPUT指定を書かない場合、ファイル名-1のファイル管理記述項にLOCK MODE句を指定しなければならない。

書き方 1（レコード順ファイル）

3. (MF)NO REWINDは注記になる。
4. The I/ O 指定は、 ディスク・ファイルに対してだけ適用できる。
5. EXTEND指定は、LINAGE句を指定していないファイルに対してだけ適用できる。
6. 複数ファイル・リールに対しては、EXTENDを指定できない。

書き方 2 ((MF)行順ファイル、相対ファイル、索引ファイル)

7. (ANS85)順呼出し法のファイルに対して、EXTENDを指定できる。
8. (ISO2000)(MF)行順編成のファイルにはSHARING指定を書いてはならない。

一般規則

すべての書き方（すべてのファイル）

1. OPEN文が正常に実行され終わるまで、該当するファイルを明示的または暗黙的に対象とするどのような文も実行することはできない（USINGまたはGIVINGを指定したSORT文およびMERGE文は例外である）。
2. (ISO2000)(MF)OPEN文が正常に実行されると、ファイルの利用可能性が判定され、ファイル名によって参照されるファイル結合子がOPEN文のモードの状態に置かれる。そして、ファイル結合子を通じて、ファイルがファイル名と関係づけられる。
   必要語のINPUT, OUTPUT, I-O, およびEXTENDを用いて、ファイル結合子を通じてファイルに適用する入出力操作を指定し、OPEN文のモードを確立する。（表14-4「文とOPENモードの許される組合せ」を参照。）

ファイルが物理的に存在し、ファイル処理システムに認識されていれば、そのファイルは利用可能である。表14-2「（現在は開かれていない）利用可能なファイルと利用不可能なファイルを開く処理」に現在は開かれていない利用可能なファイルと利用不可能なファイルを開いたときの結果を示す。表14-3「現在は他のファイル結合子によって開かれている利用可能な共有ファイルを開く処理」に現在は他のファイル結合子によって開かれている利用可能なファイルと利用不可能なファイルを開いたときの結果を示す。

14-2：（現在は開かれていない）利用可能なファイルと利用不可能なファイルを開く処理

OPEN文のモード	ファイルは利用可能	ファイルは利用不可能
INPUT	正常に開ける	開けない
INPUT （不定ファイル）	正常に開ける	正常に開ける最初に読んだときに、ファイル終了条件または無効キー条件が発生する
I-O	正常に開ける	開けない
I-O （不定ファイル）	正常に開ける	開くとファイルが作成される
OUTPUT	正常に開ける；中身は空	開くとファイルが作成される
EXTEND	正常に開ける	開けない
EXTEND（不定ファイル）	正常に開ける	開くとファイルが作成される

表14-3：現在は他のファイル結合子によって開かれている利用可能な共有ファイルを開く処理

開く要求		共有の度合いとOPEN文のモード				
		他と共有せず	読出しのみ共有		他のすべてと共有	
		拡張 I-O 入力 出力	拡張 I-O 出力	入力	拡張 I-O 出力	入力
SHARING WITH NO OTHER	EXTEND I-O INPUT OUTPUT	開けない	開けない	開けない	開けない	開けない
SHARING WITH READ ONLY	EXTEND I-O	開けない	開けない	開けない	開けない	正常に開ける
	INPUT	開けない	開けない	正常に開ける	開けない	正常に開ける
	OUTPUT	開けない	開けない	開けない	開けない	開けない
SHARING WITH ALL OTHER	EXTEND I-O	開けない	開けない	開けない	正常に開ける	正常に開ける
	INPUT	開けない	正常に開ける	正常に開ける	正常に開ける	正常に開ける
	OUTPUT	開けない	開けない	開けない	開けない	開けない

3. OPEN文の実行が正常に終了すると、対象となったファイルのレコード領域が利用可能となる。
4. 表14-4において、行と列の交差する格子内の"0"は、該当の行のアクセス・モードでの入出力モードの指定を、列の上部に示されているOPEN文のモードに適用できることを示す。

表14-4: 有効な入出力文

呼出し法	OPEN文のモード				
	文	INPUT	OUTPUT	INPUT-OUTPUT	EXTEND
順呼出し	READ	0		0	
	WRITE		0		0
	REWRITE			0	
乱呼出し(非順編成ファイル)	READ	0		0	
	WRITE		0	0	
	REWRITE			0	
	START				
	DELETE			0	
動的呼出し(非順編成ファイル)	READ	0		0	
	WRITE		0	0	
	REWRITE			0	
	START	0		0	
	DELETE			0	

注: START文とDELETE文は、レコード順ファイル
(MF)または行順ファイル
には適用できない。

5. (MF)If the WITH LOCKを指定すると、OPEN文は該当するファイル全体に対して ロックする。(これは、そのファイルのSELECT文の中にLOCK MODE IS EXCLUSIVEを指定することに等しい。前述の*SELECT*文の節を参照。)
6. (ANS85)OPEN文の実行中にファイル属性の間に矛盾が検出されると、そのOPEN文の実行は不成功となる。OPEN文の実行中にどの ファイル固有属性が妥当性を検査されるかは、COBOLシステムごとに定義されている。詳細については、ファイルの取扱いに関する説明書を参照。ファイルの編成や記憶媒体によって、どのファイル固有属性が妥当性を検査されるかは変わってくる。
7. (ANS85)同じ実行単位の中で1つのファイルを開くのに、INPUT, OUTPUT, EXTEND, I-Oのどのモードでも指定できる。 ファイル結合子に関して最初にOPEN文を実行した後では、そのファイル結合子に関してCLOSE文を実行してからでないと、同じファイル結合子に関して再びOPEN文を実行することはできない。対象のファイルが不定ファイルであっても、この規則は当てはまる。
8. OPEN文を実行しても、最初のデータレコードは入手も解放もされない。
9. SELECT句中で指定したファイルの外部名は、下記のように処理される。
    a. INPUTを指定した場合、OPEN文を実行すると、入力ファイルを開くためのオペレーティングシステムの規則に従って、割り当てられた名前が検査される。
    b. OUTPUTを指定した場合、OPEN文を実行すると、出力ファイルを開くためのオペレーティングシステムの規則に従って、割り当てられた名前が書き出される。
10. ファイル名-1、ファイル名-2、ファイル名-3、ファイル名-4のファイル記述項は、そのファイルを作成したときと同じにする。

11. INPUTまたは I/O を指定して開いたファイルに関して、OPEN文は ファイル位置指示子を設定する。その値は、索引ファイルと順ファイルに関してはファイル内に現存する最初のレコードとされ、相対ファイルに関してはレコード番号1とされる。ファイル中にレコードが存在しないと、索引データまたは順ファイルが次に読まれたときにファイル終了条件が発生するように、ファイル位置指示子が設定される。ファイルそのものが存在しないと、OPEN INPUT文は誤り状態となる。
12. I-Oを指定して開いたファイルの LABEL RECORD句に、 ラベル・レコードがあると指定されていると、OPEN文の実行に下記の処理が含められる。
    a. 入出力両用ファイルのラベルの検査に関するオペレーティングシステムの規則に従って、ラベルが検査される。
    b. 入出力両用ファイルのラベルの書出しに関するオペレーティングシステムの規則に従って、ラベルが書き出される。
13. OUTPUTを指定したOPEN文の実行が正常に終了すると、ファイルが作成される。その時点では、そのファイルにはまだレコードは含まれていない。同じ名前のファイルが存在すると、そのファイルは削除される。そのファイルが書き込み保護されていると、誤りが発生する。
14. (ANS85)EXTEND を指定すると、 OPEN文はファイルへの次の書き出し位置を、そのファイルの 最後の論理レコードの直後に位置付ける。順ファイルの場合、最後の論理レコードは最後にファイルに書き出されたレコードである。相対ファイルの場合、最後の論理レコードは現存するレコードの中で相対レコード番号が最も大きいものである。索引ファイルの場合、最後の論理レコードは現存するレコードの中で主レコードキーの値が最も大きいものである。
15. OPEN文を実行すると、 FILE STATUSデータ項目の値が更新される。(前述の*入出力状態*の節を参照。)
16. (MF)該当するファイルのSELECT/ASSIGN文の中で LOCK MODE IS を指定してあると、OPEN文の実行が正常に終了したときに、ファイルはその実行単位用に排他的に ロックされる。
17. (MF)該当するファイルのSELECT/ASSIGN文の中で LOCK MODE IS AUTOMATICまたは LOCK MODE IS MANUALを指定してあると、OPEN文の実行が正常に終了したときに、ファイルは共有可能とされる。共有可能なファイルは、複数の実行単位から正常に開くことができる。
18. (MF)OUTPUT用に開いたファイルおよびEXTEND用に開いた相対および索引ファイルは、暗黙的に 排他的にロックされたものと定義される。したがって、そのファイルは共有できない。
19. (MF)I-O用に開いた共有ファイルについてだけ、 レコードロックを得ることができる。
20. OPEN文の実行が不成功に終わると、物理ファイルは影響を受けず、下記の処理が順に行われる。
    a. 該当するファイルに対応する入出力状態に、OPEN文の実行が不成功に終わった理由を示す値が設定される。
    b. 実行すべき条件に該当するUSE AFTER EXCEPTION手続きがあれば、それが実行される。(前述の*USE*文を参照。)

21. I-Oを指定してファイルを開くと、そのファイルには入力操作も出力操作も加えることができる。ファイルが存在しないときは、下記の規則が適用される。
    a. OPTIONALを指定してあれば、ファイルが作成される。
    b. NOT OPTIONALを指定してあれば、誤りが発生する。
    c. どちらの句も指定しないで、 OPTIONAL-FILE 指令を原始要素のコンパイル時に指定してあれば、ファイルが作成される。
    d. どちらの句も指定しないで、NOOPTIONAL-FILE指令を原始要素のコンパイル時に指定してあれば、誤りが発生する。
22. 入力ファイルのSELECT句に OPTIONAL指定をしていると、ファイルのオープン時に そのファイルの有無を尋ねてくる。そのファイルが存在しないと、最初にREAD文を実行したときに、ファイル終了条件が発生する。
23. EXTENDを指定すると、 OPEN文はファイルへの次の書き出し位置を、そのファイルの最後の論理レコードの直後に位置付ける。以降、そのファイルを対象にWRITE文を実行すると、OUTPUTを指定したかのように、レコードがそのファイルに追加される。ファイルが存在しない場合は、作成される。
24. (ANS85)EXTENDを指定したファイルのファイル記述項に LABEL RECORDS句が指定してあると、OPEN文の実行に下記の処理も含められる。
    a. 単一リール/ユニット・ファイルの場合だけ、開始ファイル・ラベルが処理される。
    b. 最後に存在するリール/ユニット上の開始リール/ユニット・ラベルは、そのファイルを INPUTを指定して開いたかのように処理される。
    c. 既存の終了ラベルは、そのファイルをINPUTを指定して開いたかのように処理される。その後で、終了ラベルは削除される。
    d. その後、そのファイルをOUTPUTを指定して開いたかのように処理が続行される。
25. I /Oを指定すると、 ファイルを入力と出力の両方の処理用に開ける。
    (MF)ただし、 ORGANIZATION LINE SEQUENTIALのファイルを除く。
    該当のファイルが存在しない場合には、そのファイルが作成されて、入力用には空のファイルとして使用される。
    (MF)ただし、SELECT文にNOT OPTIONALが指定された場合は別である。
    作成されたばかりのファイルから読込みを行うと、エラーとなる。
26. (ISO2000)(MF)SHARING指定は共有ファイルに対してのみ指定できる（共有 モードを参照）。
27. (ISO2000)(MF)SHARING指定は、ファイルを共有しているファイル結合子を通じて、対象のファイルにどのような処理を行えるかを示す（共有 モードを参照）。
28. (ISO2000)(MF)ファイル名に関連するファイル結合子のファイル管理記述項に記述されているSHARING指定よりも、このSHARING指定が優先する。OPEN文にSHARINGが指定されていない場合、ファイルの共有はファイル管理記述項によって全面的に指定される。OPEN文にSHARING指定がなく、ファイル管理記述項にもSHARING句が指定されていない場合、該当のファイル結合子に関する共有モードは Sharingモードに応じて設定される。

書き方 1 (レコード順ファイル)

29. ファイルの記憶媒体が巻き戻し可能なものであるときは、下記の規則が適用される。
    a. OPEN文を実行すると、ファイルの先頭に現在位置が位置付けられる。
    b. REVERSED指定があるときは、OPEN文を実行すると、ファイルの末尾に現在位置が位置付けられる。
30. REVERSED指定があるときは、そのファイルに対して以降READ文を実行すると、データは逆の順に、つまり 最後のレコードから開始して、読み込まれる。

書き方 2 (行順ファイル)

(MF)行順ファイルに対しては、I-Oモードを適用することはできない。ただし、 REWRITE-LS 指令を設定した場合は、 そのかぎりではない。

第15章：手続き部-PERFORM-ROLLBACK

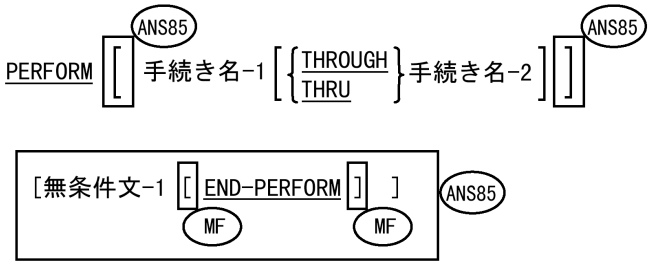
15.1 COBOLの文

15.1.1 PERFORM（実行）文

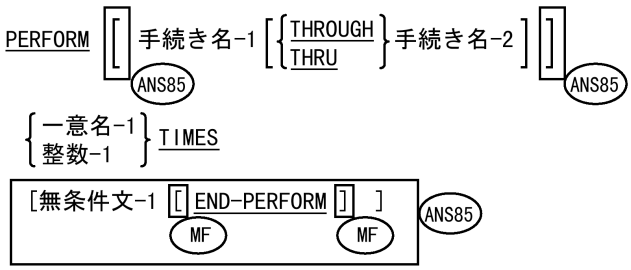
PERFORM（実行）文は、いくつかの手続きに制御を明示的に移し、その手続きの実行が終了した時点で、暗黙的に制御を戻すために使用する。
(ANS85)PERFORM文は、その有効範囲内にあるいくつかの無条件命令の実行を制御するためにも使用する。
前述の*EXIT*文の節を参照。

一般形式

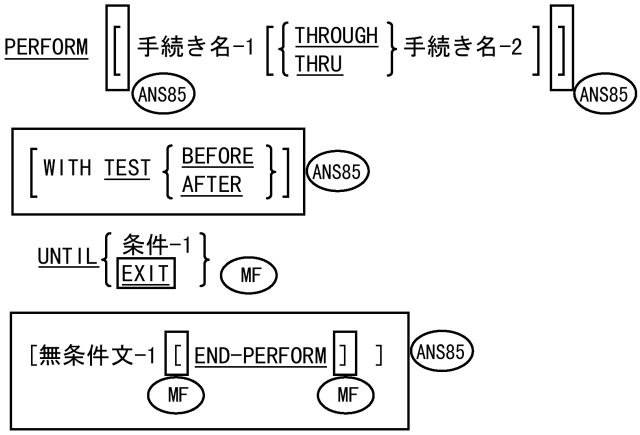
書き方 1



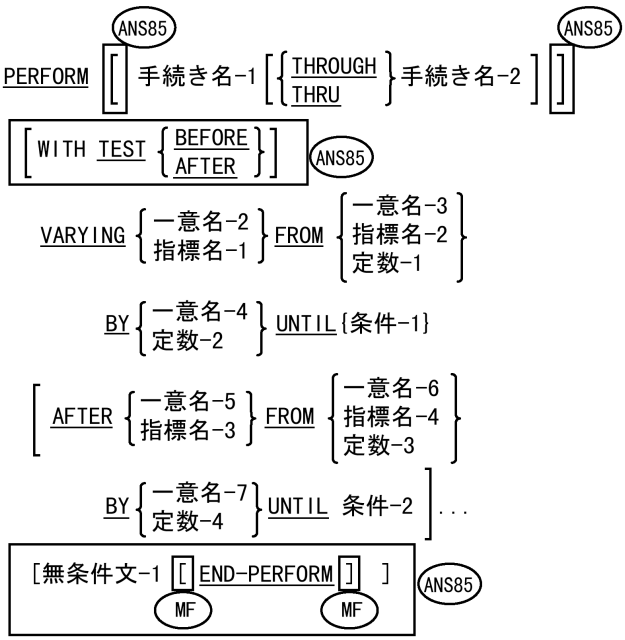
書き方 2



書き方 3



書き方 4



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - PERFORM-TYPE - 重複するPERFORMの範囲の取扱いを制御する。

構文規則

すべての書き方

1. (ANS85)手続き名-1を省略するときは、無条件文-1とEND-PERFORM指定を書く。手続き名-1を書くときは、無条件文-1とEND-PERFORM指定を書いてはならない。
   (MF)無条件文-1を書けば、END-PERFORM指定を省略してもよい。
   (COB370)手続き名-1と無条件文-1を両方とも省略してもよい。
2. 語THROUGHとTHRUは同義であり、どちらを書いてもよい。
3. 手続き名-1と手続き名-2を両方書いたときに、どちらか一方が手続き部の宣言部分に含まれる手続き名を指しているならば、両方の手続きとも同じ宣言節内になければならない。

書き方 2 および 4

4. 各一意名は、データ部に記述された数字基本項目を表わす。書き方2では、一意名-1は整数とする。
5. (OSVS)(VSC2)(MF)数字データ項目または定数を指定できるところでは、どこでも浮動小数点数のデータ項目または定数を使用できる。ただし、整数が要求される箇所には使用できない。

書き方 3

6. TEST指定を書いた場合、EXIT指定を書いてはならない。

書き方 3 および 4

7. (ANS85)TEST BEFORE指定も TEST 指定も書かないと、TEST BEFORE指定を書いたものとみなされる。
8. 条件-1、条件-2などは任意の条件式であってよい。(前述の*条件式*の節を参照。)

書き方 4

9. (ANS85)書き方4では、手続き名-1を省略するときは、AFTER指定を書いてはならない。
10. 各定数は、数字定数を表わす。
11. VARYING指定またはAFTER指定の中に指標名を指定する場合は、下記のようにする。
    a. 対応するFROM指定およびBY指定の中の一意名は、整数データ項目とする。
    b. 対応するFROM指定の中の定数は、正の整数とする。
    c. 対応するBY指定の中の定数は、ゼロでない整数とする。
12. FROM指定の中に指標名を指定する場合は、下記のようにする。
    a. 対応するVARYING指定またはAFTER指定の中の一意名は、整数データ項目とする。
    b. 対応するBY指定の中の一意名は、整数データ項目とする。
    c. 対応するBY指定の中の定数は、整数とする。
13. BY指定の中の定数は、ゼロであってはならない。

14. 書き方4では、AFTER指定を2つ書ける。
    (ANS85)書き方4では、AFTER指定を6つ書ける。
    (MF)この制限は、15にまで拡大されている。

一般規則

すべての書き方

1. 手続き名-1を指定したPERFORM文を、 外PERFORM文という。
   (ANS85)手続き名-1を省略したPERFORM文を、 内PERFORM文という。
2. 外PERFORM文の手続き名-1（手続き名-2が指定されていれば手続き名-2まで）の範囲内に含まれている文を
   (ANS85)または、内PERFORM文自体の中に含まれている文を、
   「指定された文の組」という。
3. (ANS85)END-PERFORM指定は、内PERFORM文の範囲を区切る。(*COBOL言語の概念*の章の*明示範囲符と暗黙範囲符*の節を参照。)
4. (ANS85)内PERFORM文は、基本的には、外PERFORM文用の下記の規則に従って機能する。両者の違いは、外PERFORM文では、手続き名-1（手続き名-2が指定されていれば手続き名-2まで）の範囲内に含まれている文が実行されるのに対して、内PERFORM文ではそのPERFORM文自体の中に含まれている文が実行されることである。以降の記述において、特に「内」または「外」と明示しないかぎり、外PERFORM文に当てはまる一般規則は、すべて内PERFORM文にも当てはまるものとする。
5. PERFORM文が実行されると、制御は指定された文の組の最初の文に移される（一般規則8b、8c、8dに示す場合を除く）。この制御の移行は、PERFORM文が実行されるたびに1回だけ発生する。 指定された文の組へ制御が移されると、PERFORM文の末尾に戻るための制御の暗黙移行が、下記のように用意される。
   a. 手続き名-1が段落名であり手続き名-2は指定されていない場合、手続き名-1の最後の文が実行されると、制御はPERFORM文の末尾に戻る。
   b. 手続き名-1が節名であり手続き名-2は指定されていない場合、手続き名-1の最後の段落の最後の文が実行されると、制御はPERFORM文の末尾に戻る。
   c. 手続き名-2が指定されておりそれが段落名である場合、その段落の最後の文が実行されると、制御はPERFORM文の末尾に戻る。
   d. 手続き名-2が指定されておりそれが節名である場合、その節内の最後の段落の最後の文が実行されると、制御はPERFORM文の末尾に戻る。
   e. (ANS85)内PERFORM文の実行は、 その中に含まれる最後の文が実行された後で完了する。
   f. (MF)内PERFORM文において、その中のすべての文の実行が終わる前に処理を終了させるために、 EXIT PROGRAM文を使用できる。

6. 手続き名-1と手続き名-2の間には、関連がある必要はない。ただ、手続き名-1から始まって手続き名-2で終わるように実行すればよい。手続き名-1と手続き名-2の終わりの間に、GO TO文やPERFORM文があってもよい。手続き名-1から手続き名-2の終わりに至る間にいくつかの論理経路が存在する場合は、手続き名-2を段落名とし、その中にEXIT文だけを置いて、すべての経路をそこに至らせるとよい。
7. 指定された文の組への制御の移行がPERFORM文以外の手段で行われると、その文の組の末尾から次の実行文に制御が移される。つまり、PERFORM文がその文の組を参照していないように扱われる。
8. PERFORM文は、下記のように機能する。
    a. 書き方1が基本的なPERFORM文である。この形式のPERFORMによって参照される指定された文の組が1回実行され、その後で制御はこのPERFORM文の末尾に戻る。
    b. 書き方2は PERFORM ... TIMES指定で、一定の回数だけ繰り返すPERFORM文である。指定された文の組が、整数-1または一意名-1のデータ項目の初期値によって示される回数だけ実行される。PERFORM文を実行するときに、一意名-1のデータ項目の値がゼロまたは負である場合は、制御はPERFORM文の末尾に移される。指定された文の組が指定された回数実行された後で、制御はPERFORM文の末尾に移される。
       PERFORM文の実行中は、一意名-1は指定された次の実行回数を、一意名-1のデータ項目の初期値によって示される回数から変えることはできない。
       整数-1はゼロまたは正でなければならない。符号を付けてもよい。
    c. 書き方3はPERFORM ... UNTIL指定で、条件が満たされるまで繰り返すPERFORM文である。指定された文の組が、UNTIL指定に書かれた条件が真になるまで繰り返し実行される。条件が真になると、制御はPERFORM文の末尾に移される。PERFORM文に入ったときに条件が真であり、
       (ANS85)TEST BEFOREが指定、または暗黙に指定されていると、
       制御は手続き名-1へ移行されず、PERFORM文の末尾に移される。
       (ANS85)TEST AFTERが指定されている場合も、PERFORM文はTEST BEFOREが指定されているのと同じように機能する。ただし、終了条件の判定が行われるのは、指定された文の組が実行されてからとなる。条件-1中に指定された作用対象に関連する添字付けまたは部分参照は、条件が検査されるたびに評価される。
       (MF)UNTIL EXITを指定すると、指定された文の組は、その中のどれかの文によって終了させられるまで、繰り返し実行される。外PERFORM文の場合は、この繰り返しを終了させられる文は、EXIT PROGRAMおよびSTOP RUNだけである。内PERFORM文の場合は、EXIT PERFORM文およびGO TO文によっても、この繰り返しを終了させることができる。
    d. 書き方4は PERFORM ... VARYING指定で、関連する要素の値を変化させながら条件が満たされるまで繰り返すPERFORM文である。この形のPERFORM文は、実行中にいくつかの一意名や指標名の値を、規則的に変化させるときに使用する。以下の記述において、VARYING、AFTER、FROMの各指定の作用対象としての一意名に関して述べることは指標名にも当てはまる。

指標名-1を指定する場合、PERFORM文の実行を開始する時点での一意名-3、指標名-2、定数-1の値は、指標名-1を用いている表の中の要素の出現番号と対応させる。指標名-3を指定する場合、PERFORM文の実行を開始する時点での一意名-6、指標名-4、定数-3の値は、指標名-3を用いている表の中の要素の出現番号と対応させる。

下に述べるように、指標名-1または指標名-3を増加させていく際に、指標の値は対応する表の範囲を超えてはならない。ただし、PERFORM文の実行が終了する際には、指標の値は対応する表の範囲から1増分または1減分だけ外れることがある。

一意名-2または一意名-5に添字が付けられている場合、これらの一意名のデータ項目の内容が設定または増加されるたびに、その添字が評価される。一意名-3、一意名-4、一意名-5、一意名-6に添字が付けられている場合、これらの一意名のデータ項目の内容が設定または増加の処理に使用されるたびに、その添字が評価される。条件-1または条件-2の中に指定されている作用対象に添字または部分参照が適用されていると、条件が検査されるたびにその添字または部分参照が評価される。

以降に、書き方4のPERFORM文がどのように機能するかを示す例を掲げる。

1. TEST BEFOREを明示的または暗黙的に指定した場合:

   1つの一意名のデータ項目の値を変化させる場合、PERFORM文の実行開始時点で、一意名-2のデータ項目の内容が、定数-1または一意名-3のデータ項目の現在の値に設定される。UNTIL指定の条件-1が偽であれば、指定された文の組が1回実行される。次いで、一意名-2のデータ項目の値が指定された値（定数-2または一意名-4のデータ項目の値）だけ増分または減分されて、条件-1が再び評価される。この処理が、条件-1が真になるまで繰り返される。PERFORM文の実行開始時点で条件-1が真である場合は、制御はPERFORM文の末尾に移される。

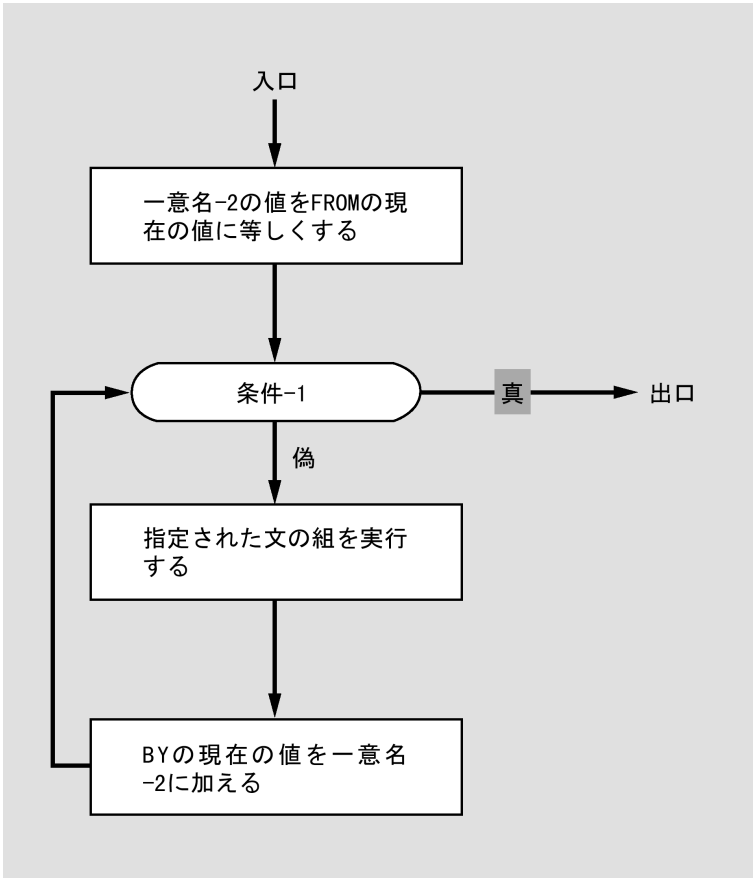


図 15-1: PERFORM文の機能の流れ図 - VARYING, TEST BEFORE, 1つの条件

2つの一意名のデータ項目の値を変化させる場合、PERFORM文の実行開始時点で、一意名-2のデータ項目の内容が、定数-1または一意名-3のデータ項目の現在の値に設定される。また、一意名-5のデータ項目の内容が、定数-3または一意名-6のデータ項目の現在の値に設定される。

これから条件-1が評価される。この結果が真である場合は、制御はPERFORM文の末尾に移される。この結果が偽である場合は、条件-2が評価される。この結果が偽であれば、指定された文の組が1回実行される。次いで、一意名-5のデータ項目の値が、定数-4または一意名-7のデータ項目の現在の値だけ増分または減分される。それから条件-2が再び評価される。

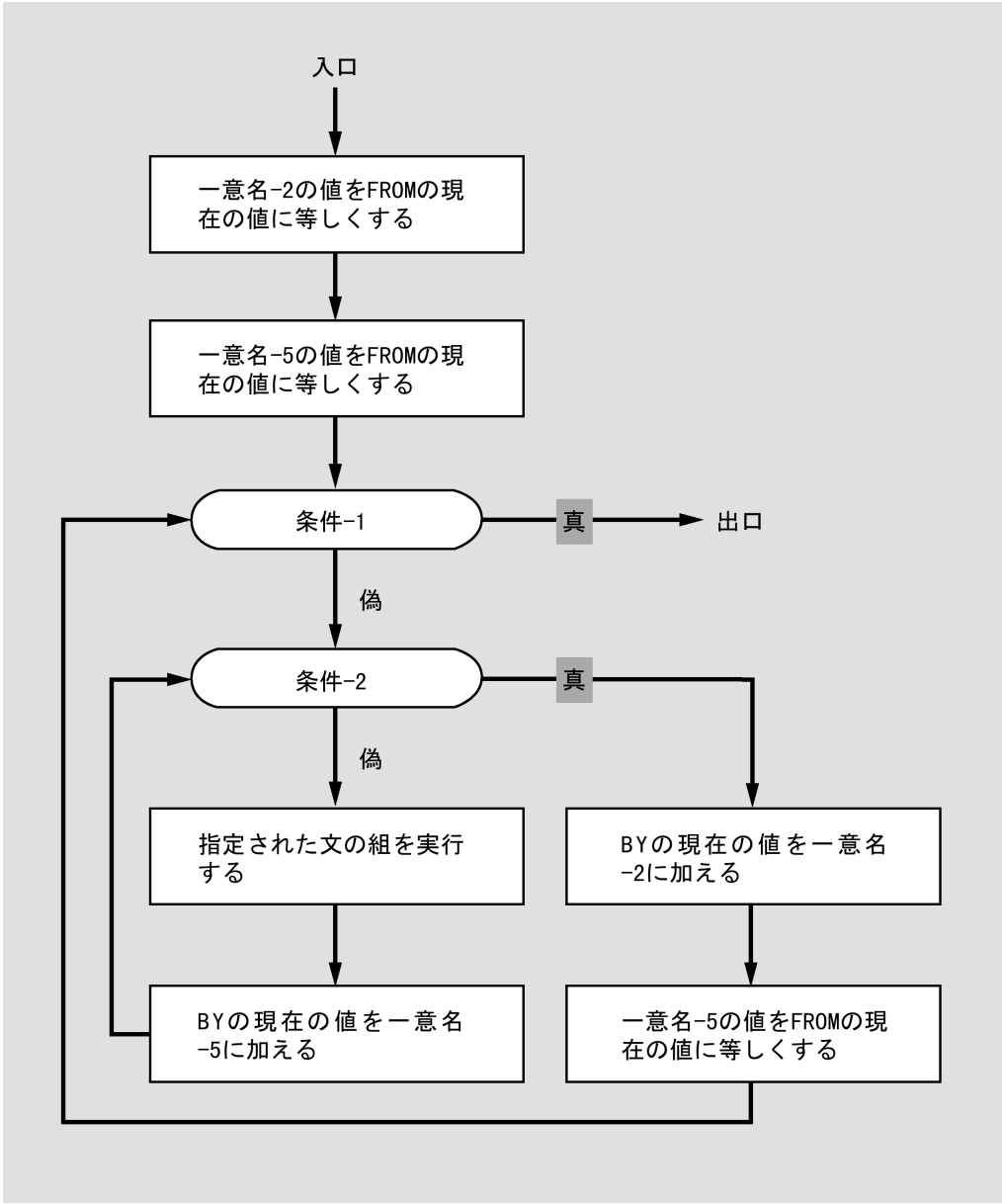


図 15-2: PERFORM文の機能の流れ図 - VARYING，TEST BEFORE，2つの条件

この処理が、条件-2が真になるまで繰り返される。条件-2が真になると、一意名-2のデータ項目の値が、定数-2または一意名-4のデータ項目の現在の値だけ増分され、一意名-5のデータ項目の内容が、定数-3または一意名-6のデータ項目の現在の値に設定され直される。この後、条件-1が再び評価される。条件-1が真である場合は、PERFORM文の実行は終了する。そうでなければ、条件-1が真になるまで、この処理が繰り返される。

PERFORM文の実行が終了した時点では、一意名-5のデータ項目には定数-3または一意名-6のデータ項目の現在の値が入っている。一意名-2のデータ項目には、増分または減分によって最後に設定された値が入っている。ただし、PERFORM文の実行が開始された時点で条件-1が真であった場合には、一意名-2のデータ項目には、定数-1または一意名-3のデータ項目の現在の値が入っている。

2. (ANS85)TEST AFTERを指定した場合:

   1つの一意名のデータ項目の値を変化させる場合、PERFORM文の実行開始時点で、一意名-2のデータ項目の内容が、定数-1または一意名-3のデータ項目の現在の値に設定される。そして、指定された文の組が1回実行される。それから、UNTIL指定の条件-1が検査される。この結果が偽であれば、一意名-2のデータ項目の値が指定された値（定数-2または一意名-4のデータ項目の値）だけ増分または減分されて、指定された文の組が再び実行される。この処理が、条件-1の検査結果が真になるまで繰り返される。条件-1の検査結果が真になると、制御はPERFORM文の末尾に移される。

   2つの一意名のデータ項目の値を変化させる場合、PERFORM文の実行開始時点で、一意名-2のデータ項目の内容が、定数-1または一意名-3のデータ項目の現在の値に設定される。また、一意名-5のデータ項目の内容が、定数-3または一意名-6のデータ項目の現在の値に設定される。そして、指定された文の組が1回実行される。それから条件-2が評価される。この結果が偽であれば、一意名-5のデータ項目の値が、定数-4または一意名-7のデータ項目の現在の値だけ増分または減分されて、指定された文の組が再び実行される。この処理が、条件-2の評価結果が真になるまで繰り返される。条件-2が真になると、今度は条件-1が評価される。この結果が偽である場合は、一意名-2のデータ項目の値が、定数-2または一意名-4のデータ項目の現在の値だけ増分または減分され、一意名-5のデータ項目の内容が、定数-3または一意名-6のデータ項目の現在の値に設定し直される。この後、指定された文の組が再び実行される。この処理が、条件-1の評価結果が真になるまで繰り返される。条件-1が真になると、制御はPERFORM文の末尾に移される。

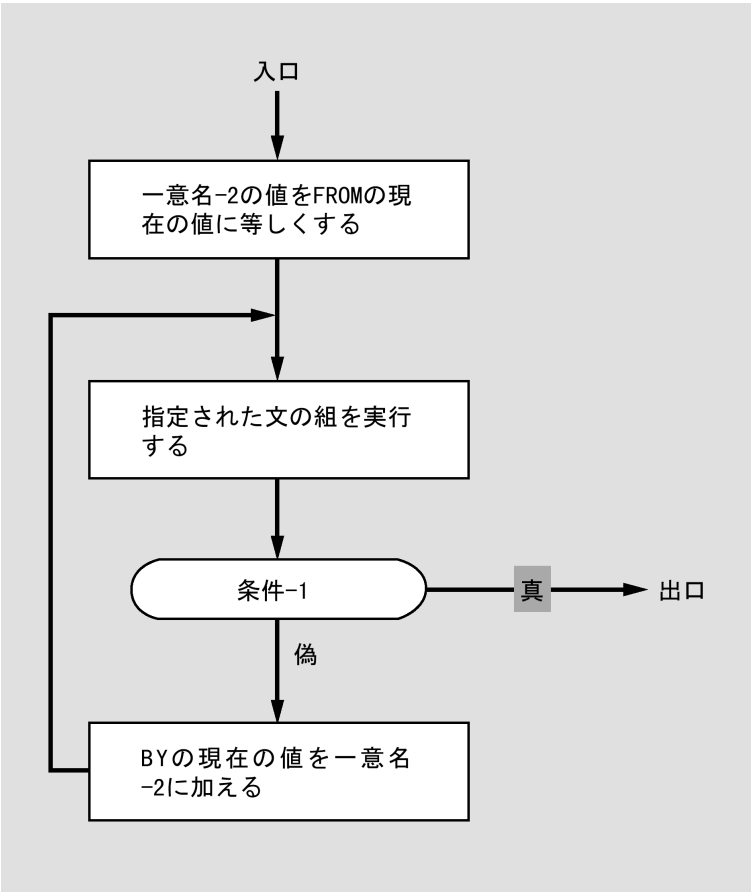


図 15-3: PERFORM文の機能の流れ図 - VARYING, TEST AFTER, 1つの条件

PERFORM文の実行が完了すると、AFTER指定またはVARYING指定によって変えられた各データ項目には、指定された文の組が最後に実行されたときの内容が入っている。図15-4を参照。

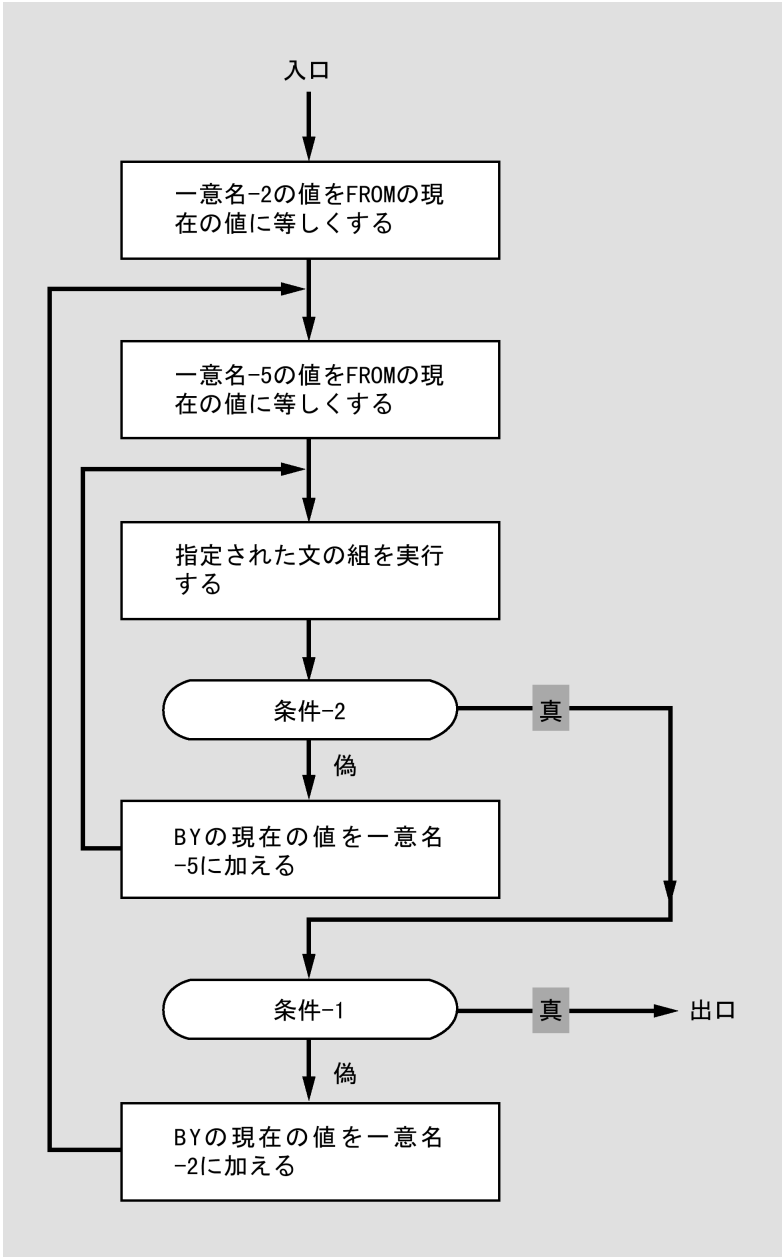


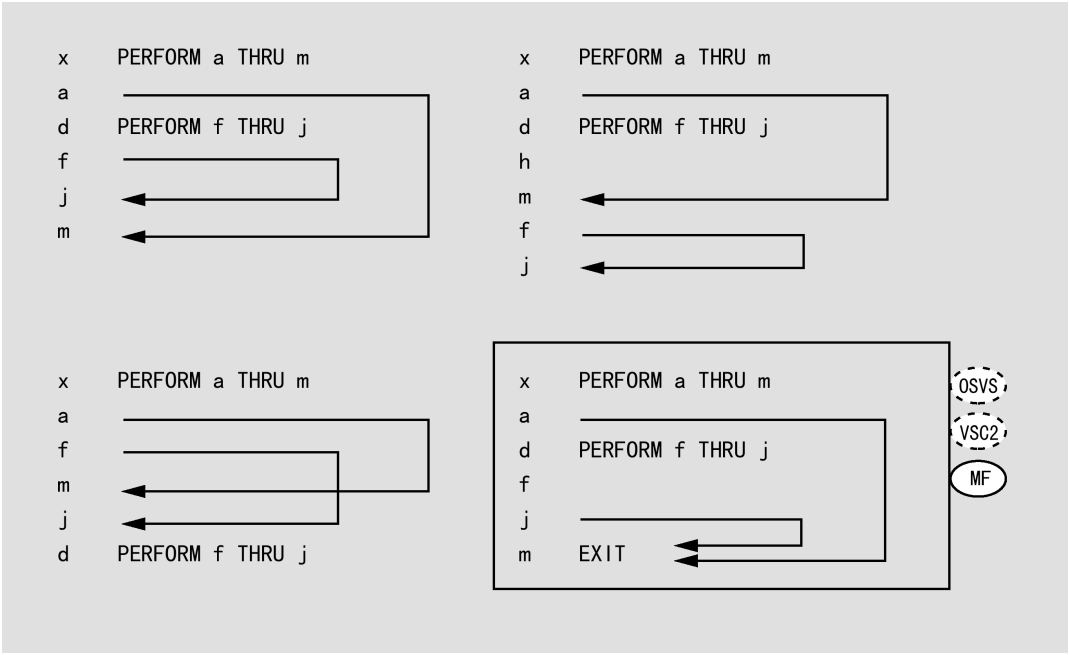
図 15-4: PERFORM文の機能の流れ図 - VARYING, TEST AFTER, 2つの条件

PERFORM文に指定された文の組が実行されている間に、VARYING変数（一意名-2のデータ項目および指標名-1）、BY変数（一意名-4のデータ項目）、AFTER変数（一意名-5のデータ項目および指標名-3）、FROM変数（一意名-3のデータ項目および指標名-2）の変化が考慮に入れられて、以降のPERFORM文の実行に影響を及ぼす。

2つの一意名のデータ項目を変化させる場合、一意名-2のデータ項目が変化するたびに、一意名-5のデータ項目は変化の全過程（FROM, BY, UNTIL）を経る。3つ以上の一意名のデータ項目の内容を変化させる場合も、その仕組みは基本的には2つの場合と同じである。ただし、AFTER指定によって変化させるデータ項目は、その前にあるAFTER指定の作用対象が増分または減分されるたびに、変化の全過程を経る。

9. PERFORM文の範囲は、論理的には、その開始によって明示的に制御を移されて実行される文に始まり、そのPERFORM文に制御が戻されるまでに実行される、すべての文に及ぶ。したがって、PERFORM文の範囲の中には、その中に含まれるCALL, GO TO, PERFORMなどの文によって制御を移されて実行される文もすべて含まれる。また、PERFORM文の範囲内の文を実行することによって実行される宣言手続き中の文もすべて含まれる。PERFORM文の範囲内に含まれる文が、原始要素内で連続して現れる必要はない。
10. 次の場合、EXIT PROGRAM文を実行したことによって制御が移されて実行される文は、PERFORM文の範囲内にあるとはみなされない。
    a. PERFORM文が指定されているのと同じプログラム中に指定されたEXIT PROGRAM文。
    b. PERFORM文の範囲内にあるEXIT PROGRAM文。
11. あるPERFORM文の範囲内に別のPERFORM文が含まれる場合、含まれる方のPERFORM文に連なる一連の手続きは、最初のPERFORM文によって参照される一連の手続きに完全に含まれるか、完全にその外にあるかの、どちらかでなければならない。したがって、あるPERFORM文の範囲内にあるPERFORM文は他のPERFORM文の出口に制御を渡してはならない。さらに、2つ以上のPERFORM文が、出口を共有してはならない。
    (MF)これらの制限は強制しない。PERFORM文を入れ子にすることも、再帰させる（あるPERFORM文の手続きの中にそのPERFORM文を含める）ことも許される。現在実行されているPERFORM文の最も内側の出口だけが認識される。これらの規則は PERFORM-TYPE コンパイラ指令 を使用することによって変更できる。

PERFORM文の正しい構成の仕方の例を、下に示す。



12. 独立区分外の節内にあるPERFORM文の範囲内には、その範囲内から呼ぶ宣言節の他に、下記のどちらか1つを含めることができる。
    a. 1つ以上の非独立区分に完全に含まれる節または段落
    b. 単一の独立区分に完全に含まれる節または段落

    (OSVS)(VSC2)これらの制限は適用しない。
13. 独立区分内にあるPERFORM文の範囲内には、その範囲内から呼ぶ宣言節の他に、下記のどれか1つを含めることができる。
    a. 1つ以上の非独立区分に完全に含まれる節または段落
    b. 該当するPERFORM文と同じ独立区分に完全に含まれる節または段落

    (OSVS)(VSC2)これらの制限は適用しない。

書き方 4

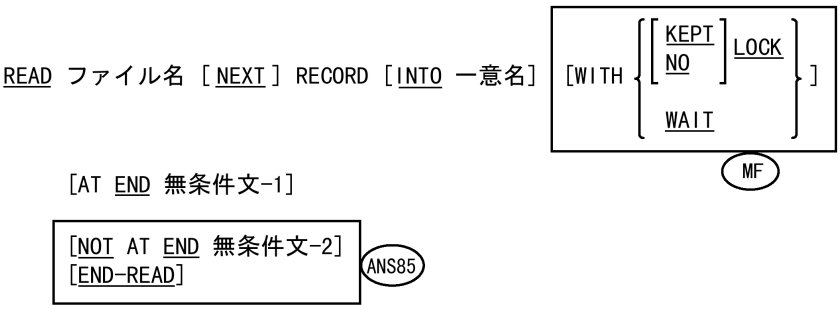
14. 一意名-4および一意名-7のデータ項目は、値がゼロであってはならない。
15. VARYING指定またはAFTER指定の中で指標名を使用し、対応するFROM指定の中で一意名を使用する場合、この一意名のデータ項目の値は、正でなければならない。

15.1.2 READ（読込み）文

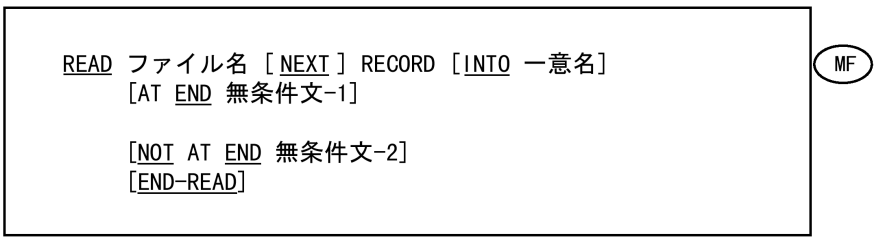
順呼出しの場合、 READ（読込み）文はファイルから 次
(MF)または 前の
論理レコードを利用できるようにする。乱呼出しの場合、READ文は大記憶ファイルから指定されたレコードを利用できるようにする。

一般形式

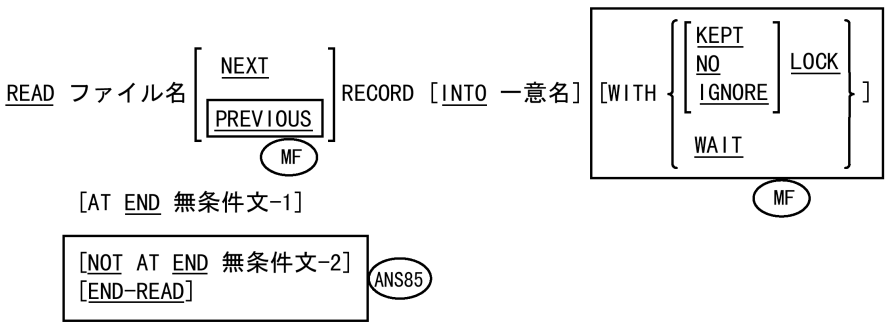
書き方 1 (レコード順ファイル)



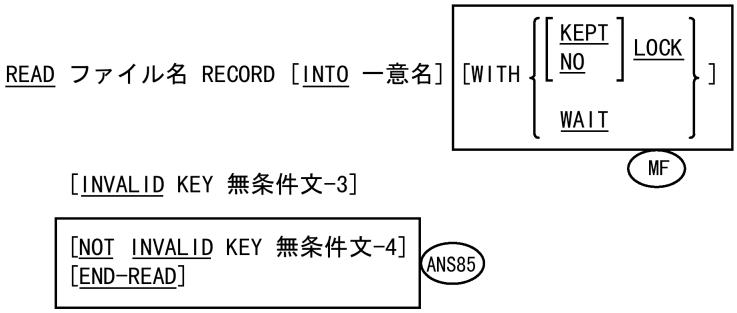
書き方 2 (行順ファイル)



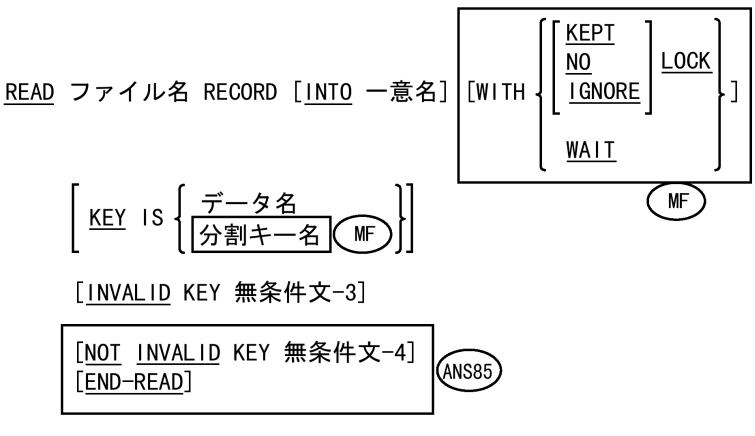
書き方 3 (相対ファイルおよび索引ファイル)



書き方 4 (相対ファイル)



書き方 5 (索引ファイル)



指令とランタイム・スイッチ

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - RETRYLOCK - ロックされているレコードの読込みを再試行させる。
2. 下記のランタイム・スイッチによって、この項に記述した意味が影響を受ける可能性がある。
    - **B, B1** - READ NEXT文を実行中にロックされたレコードが出てくると、それを飛ばすようにレコード・ポインタを更新させる。
    - **N** - 行順レコードを読むときに、制御文字の前の空文字の解釈を制御する。
    - **T** - 行順レコードを読むときに、タブ文字の解釈を制御する。

構文規則

すべての書き方 (すべてのファイル)

1. 入力ファイルの論理レコードが可変長のとき、 INTO指定を使用してはならない。
   (ANS85)この制限は廃止された。

一意名のデータの記憶領域とファイル名のレコードの記憶領域とが、同じ記憶領域を指してはならない。

2. (OSVS)(VSC2)(MF)一意名は、浮動小数点数データ項目でもよい。
3. ファイル名に適用できるUSE手続きを設定していない場合には、 INVALID KEY指定または AT END指定を書く。
   (OSVS)(VSC2)(MF)この制限は強制しない。

書き方 1、3、4、5 (レコード順ファイル、相対ファイル、索引ファイル)

4. (MF)WITH LOCKを指定できるのは、共有ファイル中で単一のレコードをマニュアルロックする場合だけである。
5. (MF)WITH NO LOCKを指定できるのは、共有ファイル中でレコードをマニュアルまたは自動的にロックする場合だけである。

書き方 3 (相対ファイルおよび索引ファイル)

6. 動的呼出し法でレコードを順検索する場合は、 NEXT
   (MF)または PREVIOUS
   を指定する。

書き方 3、4、5 (相対ファイルおよび索引ファイル)

7. (MF)WITH KEPT LOCKを指定できるのは、共有ファイル中で複数のレコードをマニュアルロックする場合だけである。

書き方 4 および 5 (相対ファイルおよび索引ファイル)

8. 乱呼出し法または 動的呼出し法でファイルのレコードを乱検索する場合は、書き方4または5を用いる。

書き方 5 (索引ファイル)

9. データ名は、ファイル名のファイルのレコードキーとして指定したデータ項目の名前とする。
   (OSVS)(MF)データ名はまた、ファイル名と対応するレコードキーとして指定した、データ項目の再定義となり、同じ長さをもつデータ名を作ることもできる。
   (OSVS)再定義は、レコードキーと異なる長さをもつことがある。
10. データ名は、修飾してもよい。
11. (MF)分割キー名は、ファイル名のファイルの レコード・キーとして指定したいくつかのデータ項目を結合したものである。

一般規則

すべての書き方 (すべてのファイル)

1. READ文を実行するときには、対象のファイルを INPUTモードまたは I-Oモードで開いておく。(前述の*OPEN*文を参照。)

2. READ文を実行すると、ファイル名に対応する FILE STATUSデータ項目が存在すれば、その値が更新される。(前述の*入出力状態*の節を参照。)
3. 1つのファイルの論理レコードが複数のレコード記述によって定義されている場合、それらのレコードは自動的に同じ記憶領域を共有する。これはレコード領域を暗黙的に再定義することに相当する。現在のレコード領域の範囲を超えてデータ項目が存在する場合、READ文の実行が終了したときにその内容はどうなっているかわからない。
4. INTOを指定すると、読み込まれたレコードは、レコード領域から一意名に指定した領域へ転記される。その際、CORRESPONDING指定のないMOVE文に関する規則が適用される。READ文の実行が不成功であったときは、この暗黙的な転記は行われない。一意名に添字付けまたは指標付けがなされている場合、レコードが読み込まれてからその一意名のデータ項目に転記される直前に評価される。
5. INTOを指定すると、読み込まれたレコードは、入力レコード領域でも一意名のデータ領域でも利用可能となる。
6. ファイル位置指示子が次の論理レコードが存在しないことを示しているか、または不定入力ファイルが存在しない場合、下記の処理が順に行われる。
    a. ファイル位置指示子の内容に応じた値が、ファイル名-1に対応する入出力状態に設定される。(前述の*入出力状態*の節を参照。)
    b. AT END指定をしてあると、制御は無条件文-1に移される。この場合、ファイル名-1のファイル結合子に対応するUSE AFTER EXCEPTION手続きが設定してあっても実行されない。その後、無条件文-1の中の各文に関する規則に従って、処理は続行される。その中で、制御の明示をもたらす分岐文または条件文が実行された場合は、その文に関する規則に従って制御が移される。そうでなければ、無条件文-1の実行が終了した時点で、制御は READ文の末尾に移される。この場合、NOT AT END指定があっても、無視される。
    c. AT END指定をしていない場合、ファイル名-1に対応するUSE AFTER STANDARD EXCEPTION手続きを設定する必要がある。この場合、そのUSE手続きが実行される。その実行が終了すると、READ文の末尾の次の実行文に制御が戻される。
    (ANS85)AT END条件が発生した場合、
    READ文の実行は不成功になる。
7. READ文の実行が不成功に終わった後では、対応するレコード領域の内容はどうなっているかわからない。また、 ファイル位置指示子は、次のレコードが存在しないことを示すように設定される。

書き方 1 (レコード順ファイル)

8. READ文によって利用可能となるレコードは、下記のように決定される。
    a. (ANS85)OPEN文の実行によってファイル位置指示子の位置が設定されている場合は、ファイル位置指示子が指すレコードが利用できるようになる。
    b. (ANS85)前のREAD文の実行によってファイル位置指示子の位置が設定されている場合は、ファイル位置指示子が次のレコードを指すように >更新され、そのレコードが利用できるようになる。

(MF)しかし、前のREAD文によってレコードがロックされている場合は、ファイル位置指示子は変更されない。そのファイル位置指示子が指すレコードが利用できるようになる。

9. READ文の実行中にリールまたはユニットの終わりが検出されたが、まだファイルの終わりには達していない場合、下記の手続きが実行される。
    a. 標準終了リール/ユニット・ラベル手続き
    b. リール/ユニットの交換
    c. 標準開始了リール/ユニット・ラベル手続き
    d. 新しいリール/ユニットの最初のデータ・レコードの読込み
10. OPTIONAL指定をしたファイルを開こうとしたときにそのファイルが存在しないと、そのファイルに対して最初にREAD文が実行されたときにAT END条件が発生し、READ文の実行は不成功に終わる。標準のファイル終了手続きは実行されない。(前述の*ファイル管理段落*、*OPEN*文、*USE*文の節を参照。) その後、 実行は、一般規則13に記してあるように進む。
11. (MF)INPUT用に開かれたファイルに関しては、READ文も READ WITH LOCK文もREAD WITH KEPT LOCK文も レコード・ロックを得ることはない。
12. (MF)レコードロック方式がAUTOMATICまたはMANUALである1つの 順ファイルを複数の実行単位からEXTENDを指定して開くと、そのファイルが共有される。ただし、そのファイルの後ろに追加されるレコードの順序は保証されない。
13. (MF)I-O用に開かれたファイルに関しては、以下のようになる。
    a. ファイルに LOCK MODE IS AUTOMATICが指定してあると、READ文にWITH NO LOCKを指定しないかぎり読み込まれた各レコードはロックされる。
    b. ファイルに LOCK MODE IS MANUALが指定してあると、READ文ではレコードロックを得ることはできない。レコードロックを得るためには、READ WITH LOCK文を使用する。READ文にWITH NO LOCKを指定しても、注記になる。
    c. レコードロックがいつ解除されるかは、単一のレコードをロックしたか複数のレコードをロックしたかによって異なる。 (前述の*ファイル管理段落*を参照。)
14. (MF)ある実行単位からはI-OまたはINPUT用に開き、別の実行単位からはEXTEND用に開いたファイルに対して READ文を実行した際に、ファイル終了状態が発生した場合、そのREAD文を実行しようとした実行単位側でそのファイルを閉じなければならない。そのファイルの状態は 終了のまま残るため、その実行単位からはファイルの後ろに追加されたレコードを呼び出すことはできない。

書き方 1、2、3 (順ファイル、相対ファイル、索引ファイル)

15. READ文の実行中に AT END条件が発生しなかった場合、AT END指定をしてあっても無視されて、以下の処理が行われる。
    a. ファイル位置指示子が設定されて、ファイル名-1に対応する入出力状態が更新される。

b. ファイル終了条件以外の例外条件が発生していると、ファイル名-1に適用できるUSE AFTER EXCEPTION手続きが指定してあればそれが実行されて、その後でUSE文の規則に従って制御が移される。(前述の*USE*文の節を参照。)

c. 例外条件が何も発生していなければ、読込み対象レコードがレコード領域中で利用できるようにされる。また、INTOを指定してあれば、暗黙の転記処理が行われる。その後で、制御はREAD文の末尾に移される。ただし、NOT AT END指定をしてあれば、制御は無条件文-2に移される。この場合、無条件文-2の中の個々の文に関する規則に従って、以降の処理が続行される。制御の明示移行をもたらす分岐文または条件文が実行された場合は、その文に関する規則に従って制御が移される。制御の明示移行をもたらす分岐文または条件文が実行されなかった場合は、無条件文-2の実行が終わると、制御はREAD文の末尾に移される。

16. NEXT指定をした READ文を実行したときに、 対象ファイル中に次の論理レコードが存在しないと、 AT END条件が発生しREAD文の実行は不成功に終わったものとみなされる。(前述の*入出力状態*の節を参照。)

17. (ANS85)順ファイルまたは
順呼出し法で呼び出すファイルに関しては、NEXT指定は書いても書かなくてもよい。READ文の実行には影響を及ぼさない。

書き方 3(相対ファイルおよび索引ファイル)

18. (MF)読み込んだレコードがロックされていると、ファイル位置指示子はそのレコードを指すように設定される。以降、READ NEXT文またはREAD PREVIOUS文は同じレコードを再び取り出す。
(MF)NOT AT END指定は、READ文の実行が正常に終了した場合にだけ実行される。

19. (MF)PREVIOUS指定をしたREAD文を実行したときに、対象ファイル中に前の論理レコードが存在しないと、AT END条件が発生しREAD文の実行は不成功に終わったものとみなされる。

20. AT END条件が検出された場合、前の論理レコードが存在しないため、
(MF)そのファイルに対して次に書き方3のREAD文を実行するならば、NEXTを指定する。
そうでなければ、
下記の処理を続ける。

a. 該当するファイルに対してCLOSE文を実行し、次いでOPEN文を実行する。いずれも正常に終了しなければならない。

b. そのファイルに対してSTART文を実行する。正常に終了しなければならない。

c. そのファイルに対して、書き方4(相対ファイル)または書き方5(索引ファイル)のREAD文を実行する。

21. 動的呼出し法を指定したファイルに関して、NEXT指定をしたREAD文を実行すると、
(MF)一般規則8に記述したように次の論理レコードが取り出される。

22. READ文によって利用可能となるレコードは、下記の規則に従って決定される。

a. (MF)OPEN文の実行によって ファイル位置指示子が設定されているところへPREVIOUSを指定したREAD文を実行すると、 AT END条件が発生する。

START文またはOPEN文の実行によってファイル位置指示子が設定されていて、ファイル位置指示子によって指されるレコードの呼出しができる場合は、そのレコードが利用可能にされる。相対ファイルのレコードを削除したり索引ファイルの 副レコードキーを変更したりすることによって、レコードが呼び出しできる状態になった場合は、参照キーによって確立されている呼出し経路中の次のレコードを指すように、ファイル位置指示子が更新される。
(MF)ただし、PREVIOUS指定をしていると、
前のレコードを指すように、ファイル位置指示子が更新される。そして、ファイル位置指示子によって指されるレコードが利用可能にされる。

b. 前のREAD文の実行によってファイル位置指示子が設定された場合、ファイル位置指示子はファイル中に存在する次のレコードを指すように更新される。
(MF)ただし、PREVIOUS指定をしていると、
前のレコードを指すように、ファイル位置指示子が更新される。
(MF)前のREAD文によってレコードがロックされた場合は、ファイル位置指示子は変更されない。そして、ファイル位置指示子によって指されるレコードが利用可能にされる。

書き方 1、3、4、5 (順ファイル、相対ファイル、索引ファイル)

23. (MF)単一のレコードを マニュアルロックするように指定したファイルを入出力両用に開くと、READ文に WITH LOCK指定をしたときにだけ、実行単位はレコードロックを取得する。単なるREAD文では、 レコードロックを得ることはできない。ロックされているレコードを読み飛ばすためには、START文を用いて ファイル位置指示子を更新する。ただし、キーの重複を認めている 副レコードキーに関しては、この方法を採ることはできない。
24. (MF)複数のレコードを マニュアルロックするように指定したファイルを入出力両用に開くと、READ文に WITH KEPT LOCK指定をしたときにだけ、実行単位はレコードロックを取得する。単なるREAD文では、 レコードロックを得ることはできない。ロックされているレコードを読み飛ばすためには、START文を用いてファイル位置指示子を更新する。ただし、キーの重複を認めている副レコードキーに関しては、この方法を採ることはできない。
25. (MF)WITH WAIT指定をすると、必要があれば待ってでも、レコードロックが必ず取得される。

書き方 3 (相対ファイル)

26. RELATIVE KEYを指定したファイルに対してREAD文を実行すると、利用可能にされたレコードの相対レコード番号が、RELATIVE KEYデータ項目に収められる。

書き方 4 (相対ファイル)

27. READ文を実行すると、ファイルのRELATIVE KEYに指定したデータ項目中に保持されている相対レコード番号がファイル位置指示子に設定され、そのレコードが利用可能にされる。ファイル中にその相対レコード番号のレコードが存在しない場合は、 無効キー条件が発生し、READ文の実行は不成功に終わる。(前述の *無効キー条件*の節を参照。)

書き方 3 および 5 (相対ファイルおよび索引ファイル)

28. (MF)IGNORE LOCK指定をすると、READ文はレコードがロックされていないかのように実行される。

書き方 3 (索引ファイル)

29. 索引ファイルを順呼び出しする場合、検索のキーとしている副レコードキーに値が同じものがあると、これらはWRITE文またはREWRITE文によって書き出された順に利用可能にされる。

書き方 5 (索引ファイル)

30. KEY指定をしないと、 主レコードキーが検索キーとして使用される。動的呼出しを指定すると、後に書き方3のREAD文を実行するときにも、この検索キーが使用される。
31. KEY指定をすると、データ名-1
    (MF)または分割キー名
    がその 参照キーとして使用される。動的呼出しを指定すると、後に書き方3の READ文を実行するときにも、この検索キーが使用される。ただし、別の参照キーを指定すると、今度はそれが使用されるようになる。
32. 書き方5のREAD文を実行すると、参照キーの値がファイル中のレコードの対応するデータ項目に格納されている値と順次比較されて、両者の値が最初に一致するものが見つけ出される。ファイル位置指示子がそのレコードを指すように設定されて、そのレコードが利用可能とされる。両者の値が一致するレコードが見つからないと、 無効キー条件が発生し、READ文の実行は不成功に終わる。(前述の*無効キー条件*の節を参照。)

15.1.3 RELEASE（引き渡し）文

RELEASE（引き渡し）文は、 整列処理の最初の段階にレコードを引き渡す。

一般形式

```
RELEASE レコード名 [ FROM 一意名 ]
```

構文規則

1. RELEASE文を使用できるのは、整列併合ファイル記述項にレコード名を記述したファイルを対象とするSORT文に関連する、入力手続きの範囲内だけである。(前述の*SORT文*の節を参照。)
2. レコード名は、関連する整列併合ファイル記述項内の論理レコードの名前とする。このレコード名は修飾してもよい。
3. (ANS85)一意名が関数一意名である場合は、英数字関数を参照しなければならない。一意名が関数一意名でない場合は、
   レコード名と一意名が同じ記憶領域を指してはならない。
4. (OSVS)(VSC2)(MF)レコード名は、浮動小数点数項目として定義してあってもよい。
5. (OSVS)(VSC2)(MF)一意名は、浮動小数点数であってもよい。

一般規則

1. RELEASE文を実行すると、レコード名が1件ずつ、整列処理の最初の段階に引き渡される。
2. FROM指定をすると、一意名のデータ領域の内容がレコード名に転記されて、それからレコード名の内容が整列ファイルに引き渡される。この転記は、CORRESPONDING指定のないMOVE文に関する規則に従って行われる。レコード領域内のデータは利用できなくなるが、一意名のデータ領域のデータは引続き利用できる。
3. RELEASE文の実行が終了した後は、レコード領域中の論理レコードは利用できなくなる。ただし、対応する整列併合ファイルをSAME RECORD AREA句に指定しておけば、利用できる。この場合、その論理レコードは そのSAME RECORD AREA句に指定した他のファイルのレコードとしても利用できる。入力手続きから制御が戻されると、整列用ファイルにはRELEASE文の実行によって引き渡されたすべてのレコードが格納されている。

15.1.4 RETURN（引き取り）文

RETURN（引き取り）文は、 整列処理の最後の段階から整列済みのレコードを引き取るか、または 併合処理中に併合済みのレコードを引き取る。

一般形式

```
RETURN ファイル名 RECORD [INTO 一意名-1]
    AT END 無条件文-1

[NOT AT END  無条件文-2]   (ANS85)
[END-RETURN]
```

構文規則

1. ファイル名は、データ部の中の整列併合ファイル記述項に記述しておく。
2. RETURN文を使用できるのは、ファイル名を対象とするSORT文またはMERGE文に関連する、出力手続きの範囲内だけである。
3. 入力ファイルのレコード記述に大きさの異なる論理レコードが含まれているときは、INTO指定をしてはならない。一意名のデータ領域とファイル名のレコード領域が、同じ記憶領域を指していてはならない。
4. (OSVS)(VSC2)(MF)一意名-1は、浮動小数点数項目であってもよい。

一般規則

1. 1つのファイルに論理レコードのレコード記述がいくつもある場合、これらのレコードは自動的に同じ記憶領域を共有する。これは、暗黙的にレコードを再定義することと同じである。現在のデータ・レコードの範囲を超えるデータ項目は、RETURN文の実行が終了した時点でその内容は、どうなっているかわからない。
2. RETURN文の実行が終了すると、ファイル名のファイル中に存在する次のレコードが、そのレコード領域において利用可能とされる。このレコードの順序は、SORT文またはMERGE文中に指定されているキーによって決定される。ファイル名のファイル中に次の論理レコードが存在しないとファイル終了条件が発生し、AT END指定の無条件文-1に制御が移される。この場合、無条件文-1の中の各文に関する規則に従って、処理は続行される。その中で、制御を明示的に移行する手続きの分岐または条件文が実行された場合は、その文に関する規則に従って制御が移される。 そうでなければ、無条件文-1の実行が終了した時点で、制御は RETURN文の末尾に移される。
   (ANS85)この場合、NOT AT END指定があっても、無視される。
   ファイル終了条件が発生すると、RETURN文の実行は不成功に終わり、ファイル名のレコード領域の内容はどうなるかわからない。AT END指定中の無条件文-1の実行が終了した後では、現在の出力手続きの一環としてそれ以上RETURN文を実行することはできない。
3. (ANS85)RETURN文の実行中にファイル終了条件が発生しなかった場合、レコードが利用可能にされる。次いでINTO指定に基づく暗黙の転記が行われた後で、無条件文-2を指定してあれば、制御はそちらに移される。無条件文-2を指定してなければ、制御はRETURN文の末尾に移される。
4. (ANS85)END-RETURN指定は、RETURN文の範囲を区切る。
5. RETURN文中に INTO指定を行えるのは、下記のどちらかの場合である。
   a. 整列併合ファイル記述のもとに、レコードが1件だけ記述されている。
   b. ファイル名のファイルのすべてのレコードおよび一意名のデータ項目が、集団項目または基本英数字項目である。
6. INTO指定を伴うRETURN文を実行した結果は、下記の規則を順に適用することと同じである。
   a. INTO指定を伴わない同じ内容のRETURN文を実行する。

b. 現在のレコード領域の内容を、一意名のデータ領域に転記する。転記の仕方はCORRESPONDING指定のないMOVE文の規則に従う。現在のレコードの大きさは、RECORD句の指定によって決まる。ファイル記述項にRECORD IS VARYING句が指定してある場合は、暗黙的に集団項目転記が行われるが、この転記は不成功に終わる。一意名に添字付けがされていれば、レコードが読み出された後、一意名のデータ項目に転記される直前に評価される。レコード領域および 一意名-1のデータ項目の、 両方のデータが利用できる。

15.1.5 REWRITE（書き換え）文

REWRITE（書き換え）文は、ディスク・ファイル中に存在するレコードを論理的に置き換える。

一般形式

書き方 1 (行順ファイル)

```
REWRITE レコード名 [FROM 一意名]                    MF
    [END-REWRITE]
```

書き方 2 (レコード順ファイル)

```
REWRITE レコード名 [FROM 一意名]
    [END-REWRITE]
```

書き方 3 (相対ファイルおよび索引ファイル)

```
REWRITE レコード名 [FROM 一意名]
    [INVALID KEY 無条件文-1]
    [NOT INVALID KEY 無条件文-2]                    ANS85
    [END-REWRITE]
```

指令とランタイム・スイッチ

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - WRITE-LOCK - 複数のレコード・ロックを使用するときに、REWRITE文用のレコード・ロックを取得する。
2. 下記のランタイム・スイッチによって、この項に記述した意味が影響を受ける可能性がある。
    - N - 行順レコードを書くときに、制御文字の前の空文字の挿入を制御する。

- T - 行順レコードを書くときに、タブ文字の挿入を制御する。

構文規則

すべての書き方（すべてのファイル）

1. (ANS85)一意名が関数一意名である場合、英数字関数を参照しなければならない。一意名が関数一意名でない場合、
   レコード名と一意名が、同じ記憶領域を指してはならない。
2. レコード名は、データ部のファイル節の中の論理レコードの名前である。レコード名は修飾してもよい。
3. (OSVS)(VSC2)(MF)レコード名は、浮動小数点数項目または2バイト文字項目であってもよい。
4. (OSVS)(VSC2)(MF)一意名は、USAGE DISPLAY-1（DBCS）項目または浮動小数点数項目であってもよい。

書き方 3（相対ファイルおよび索引ファイル）

5. 順呼出し法でファイルを処理している場合には、REWRITE文には INVALID KEYを指定できない。
   (OSVS)(VSC2)ただし、順呼出し法で索引ファイルを処理している場合には、REWRITE文にINVALID KEYを指定してもよい。
6. 乱呼出し法または動的呼出し法でファイルを処理しているが、そのファイルに適用できるUSE手続きを指定していない場合には、REWRITE文にINVALID KEYを指定する。

一般規則

すべての書き方（すべてのファイル）

1. 該当するレコード名のレコードが含まれるファイルは、ディスク・ファイルとする。この文を実行する時点では、 入出力両用モードで開いておく。(前述の*OPEN*文の節を参照。)
2. ファイルを順呼出し法で処理している場合、REWRITE文を実行する前に実行した直前の入出力文は、正常に終了したREAD文でなければならない。オペレーティングシステムは、READ文によって呼び出されたレコードを論理的に置き換える。
3. FROM指定をしたREWRITE文を実行することは、
   MOVE 一意名 TO レコード名
   を実行し、それからFROM指定をしないREWRITE文を実行することと同じである。暗黙のMOVE文が実行される前のレコード領域の内容は、REWRITE文の実行に影響を及ぼさない。
4. ファイル位置指示子は、REWRITE文の実行の影響を受けない。
5. REWRITE文を実行すると、更新対象のファイルに対応するFILE STATUSデータ項目を指定してあれば、その値が更新される。 (前述の*入出力状態*の節を参照。)
6. (ANS85)END-REWRITE指定は、REWRITE文の範囲を区切る。
7. (MF)書き換えようとするレコードが他の実行単位によってロックされていると、REWRITE文の実行は不成功に終わる。

書き方 1 および 2 (順ファイル)

8. レコード名のレコードの文字数は、新しい書き換え用のレコードの文字数と等しくする。

   注: 圧縮された順ファイルにはREWRITE文を使用しないように勧める。その理由は、圧縮された新しいレコードの長さが圧縮された古いレコードの長さと同じでないと、REWRITE処理は正常に終了しないからである。

9. REWRITE文の実行が正常に終了すると、レコード領域中の論理レコードは解放され、利用できなくなる。ただし、対応するファイルを SAME RECORD AREA句に指定しておけば、利用できる。この場合、その論理レコードは 、そのファイルの論理レコードとしてだけではなく、そのSAME RECORD AREA句に指定した他のファイルのレコードとしても、利用できる。

書き方 1 (行順ファイル)

10. (MF)書き込もうとするレコードの圧縮された長さが、元の圧縮されたレコードの長さ以下のとき、 REWRITE文を使用して正常に終了させることができる。ここで、圧縮された長さとは、後行の空白を削除しタブを圧縮し空文字を挿入したものを指す。レコードを読み込むと、空文字とタブ文字が拡張されて、レコードの長さが長くなる。

書き方 3 (相対ファイルおよび索引ファイル)

11. REWRITE文の実行が正常に終了すると、レコード領域中の論理レコードは解放され、利用できなくなる。ただし、対応するファイルを SAME RECORD AREA句に指定しておけば、利用できる。この場合、その論理レコードは、 そのレコード名を含むファイルの論理レコードとしてだけではなく、そのSAME RECORD AREA句に指定した他のファイルのレコードとしても利用できる。

書き方 3 (相対ファイル)

12. 乱呼出し法または 動的呼出し法で処理しているファイルに関しては、オペレーティングシステムは、そのファイルの RELATIVE KEYデータ項目の内容によって指定されるレコードを論理的に置き換える。ファイル中に該当するレコードが存在しないと、無効キー条件が発生する(前述の *無効キー条件*の節を参照。) この場合、更新処理は行われず、レコード領域中のデータは元のまま残る。

書き方 3 (索引ファイル)

13. 順呼出し法で処理しているファイルに関しては、書き換える対象のレコードは 主レコードキーの値によって指定する。 REWRITE文を実行するとき、置き換えようとするレコードの主レコードキー・データ項目中の値は、ファイルから最後に読み込んだレコードの主レコードキーの値と等しくなければならない。

14. 乱呼出し法または動的呼出し法で処理しているファイルに関しては、書き換える対象のレコードは主レコードキー・データ項目の値によって指定する。
15. これから書き換えられるレコードの副レコードキー・データ項目の内容は、置き換える側のものと違っていても構わない。オペレーティングシステムは、更新後にどのレコードキーを使用してもレコードを呼び出すことができるように、REWRITE文の実行中にレコードキー・データ項目の内容を使用する。
16. 下記のどれかの場合に、無効キー条件が発生する。
    a. 順呼出し方式でファイルを処理しているときに、新しく置き換える側のレコードの主レコードキー・データ項目の値が、ファイルから最後に読み込んだレコードの主レコードキーの値と等しくない。
    b. 主レコードキー・データ項目の値と主レコードキーの値が一致するレコードが、ファイル中に存在しない。
    c. 副レコードキー・データ項目の値と副レコードキーの値が一致するレコードが、 DUPLICATES句を指定していないファイルに関して、既にファイル中に存在する。

    上記の場合、更新処理は行われず、レコード領域中のデータは元のまま残る。(前述の*無効キー条件*の節を参照。)

15.1.6 ROLLBACK（更新取消し）文

(MF)

The ROLLBACK（更新取消し）文は、実行単位によって押さえられているすべてのファイル中のすべてのレコードロックを解除する。SELECT文のWITH...ROLLBACK句を単なる注記ではなく実際に機能できるCOBOLシステムでは、ROLLBACK文は現在のトランザクションの終わりを示し、そのトランザクションに関して行われた処理をすべて取り消す。

一般形式

```
ROLLBLACK
```

一般規則

1. ROLLBACK文を実行すると、実行単位によって押さえられているすべてのファイル中のすべてのレコードロックが解除される。
2. 排他ファイル中のファイルロックは、ROLLBACK文の影響を受けない。
3. 使用しているCOBOLシステムにおいて、SELECT文のWITH...ROLLBACK句が単なる注記ではなく実際に機能する場合、ROLLBACK文は下記の処理を行う。
    a. 現在のトランザクションを終了させる。
    b. 現在のトランザクションの処理中に行われたすべての変更を取り消す。

第16章: 手続き部 - SEARCH - WRITE

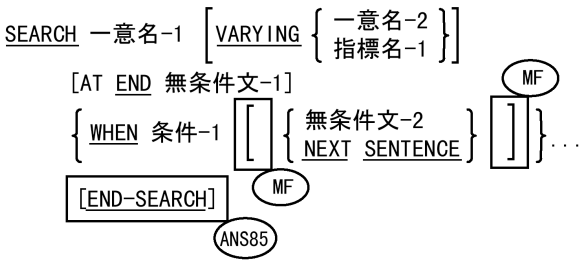
16.1 COBOLの文

16.1.1 SEARCH（表引き）文

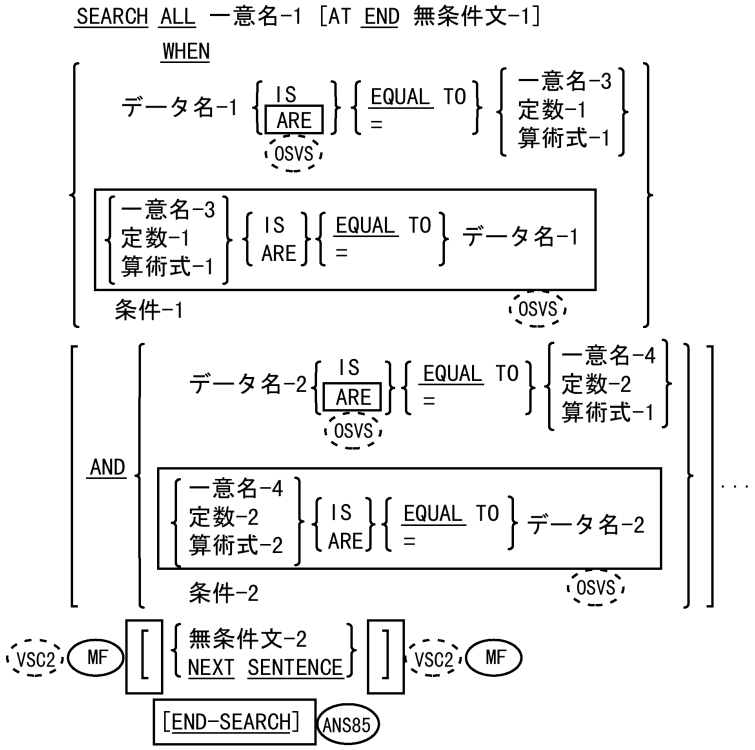
SEARCH（表引き）文は、 表を検索して、指定された条件を満足する表要素を探し出し、対応する指標名がその表要素を指すようにするために使用する。

一般形式

書き方 1



書き方 2



比較文字の"=" は必須であるが、他の記号と混同することを避けるために、下線を引いていないことに注意。

構文規則

すべての書き方（すべてのファイル）

1. 一意名-1には添字または指標を付けてはならない。このデータ記述には、 OCCURS句と INDEXED BY BY句が含まれていなければならない。書き方2では、一意名-1のデータ記述のOCCURS句内に、KEY IS指定も含まれていなければならない。
2. 一意名-2を指定する場合、指標データ項目（USAGE IS INDEX）または想定される小数点の右に何もない整数データ項目とする。
3. (ANS85)END-SEARCH指定を書いた場合、NEXT SENTENCE指定を書いてはならない。
   (VSC2)(MF)END-SEARCH指定とNEXT SENTENCE指定を同時に書いてもよい。NEXT SENTENCE指定が実行されると、制御はEND-SEARCH指定の後ろに続く次の文に移されるのではなく、最も近い終止符（.）の後ろの文に移される。
4. (MF)無条件文-1および無条件文-2の代わりに、条件文を使用してもよい。

書き方 1

5. 条件文-1、条件文-2 などには、前述の*条件式*の節で説明した任意の条件を書ける。

6. (OSVS)(VSC2)(MF)一意名-1（つまり表引きの対象とする表要素）は、内部浮動小数点数項目または外部浮動小数点数項目であってもよい。

書き方 2

7. 使用する条件名は、すべて単一の値をもつものとして定義しておく。条件名に関連するデータ名は、一意名-1の KEY句の中に記述しておく。データ名-1およびデータ名-2は、それぞれ修飾できる。データ名-1およびデータ名-2は、それぞれ他の必要な指標や定数とともに、一意名-1用の最初の指標名によって指標付けしておき、一意名-1のKEY句の中に記述しておく。一意名-3、一意名-4、算術式-1または算術式-2の中で使用する一意名は、一意名-1のKEY句の中に記述してあってはならず、一意名-1用の最初の指標名によって指標付けしてあってもならない。
   一意名-1のKEY句の中のデータ名を使用するとき、または一意名-1のKEY句の中のデータ名に関連する条件名を使用するとき、一意名-1のKEY句の中のそのデータ名の前にあるすべてのデータ名またはそれらに関連する条件名を使用する。
8. (OSVS)(VSC2)(MF)一意名-1（つまり表引きの対象とする表要素）は、浮動小数点数であってはならない。
9. (OSVS)(VSC2)(MF)キー・データ項目であるデータ名-1およびデータ名-2は浮動小数点数であってならない。しかし、キー・データ項目と比較する対象の一意名-3、一意名-4、定数-1、定数-2は、浮動小数点数であってもよい。

一般規則

すべての書き方

1. SEARCH文の範囲は、下記のどれかによって区切られる。
   a. (ANS85)同じ入れ子レベルにあるEND-SEARCH指定
   b. 区切り文字の終止符
   c. 前のIF文に対応するELSE指定
      (ANS85)またはEND-IF指定
2. 実行された無条件文-1または無条件文-2にGO TO文が含まれていなければ、制御は次の実行完結文に移される。
3. 2次元または 3次元の表を作成するためのOCCURS句が含まれるデータ項目の下位に一意名-1が属する場合、OCCURS句のINDEXED BY句を通じて、表の各次元ごとに指標名を付ける。SEARCH文が実行されると、一意名-1用の指標名（および指定されていれば一意名-2または指標名-1）だけが、値を変えられる。2次元または3次元の表の全体を表引きするためには、SEARCH文を数回実行する必要がある。指標の値を調整する必要がある場合には、SEARCH文を実行する前に、SET文を実行しておく。

書き方 1

4. 書き方1のSEARCH文を用いると、 逐次表引き処理が行われる。 このときの指標の値が、その開始点となる。

a. SEARCH文の実行が開始される時点で、一意名-1用の指標データ項目の値が出現番号の最大値よりも大きいと、そのSEARCH文の実行は直ちに停止される。一意名-1の出現番号の最大値の詳細については、OCCURS句に説明してある。(前述の*OCCURS句*の節を参照。) このとき、AT END指定が書いてあれば、無条件文-1が実行される。AT END指定が書いてなければ、制御は次の実行完結文に移される。

b. SEARCH文の実行が開始される時点で、一意名-1用の指標データ項目の値が出現番号の最大値を超えないと、そのSEARCH文の処理が進められる。(一意名-1の出現番号の最大値の詳細については、OCCURS句に説明してある。) まず、条件が書かれている順に評価される。この際、指定されていれば、検査の対象となる要素の出現番号を判定するために、指標データ項目の値が使用される。条件が満足されていない場合は、指標データ項目の値が増やされて、次の要素が参照できるようにされる。この処理が順次繰り返される。一意名-1用の指標データ項目の値が出現番号の最大値を超えると、上記の一般規則1aに示すとおり、その表引き処理は終了する。この間にどこかの時点で条件が満足されると、表引き処理は直ちに停止され、その条件に付随する無条件文が実行される。このとき、その指標データ項目には、条件を満足させた要素の出現番号が記録されている。

5. VARYING指定を書かなかった場合、表引き処理に使用される指標名は、一意名-1のINDEXED BY句の中の最初の（または唯一の）指標名だけである。一意名-1用のその他の指標名の指標データ項目の内容は変わらない。
6. VARYING指標名-1指定を書いた場合、一意名-1のINDEXED BY句の中に指標名-1が書いてあれば、その指標名がこの表引き処理に使用される。そうでない場合、またはVARYING一意名-2指定を書いた場合、一意名-1のINDEXED BY句の中の最初の（または唯一の）指標名が表引きに使用される。この場合、さらに下記の処理も行われる。

a. VARYING指定で指標名-1を書いた場合、この指標名-1が別の表の INDEXED BY指定に書いてあると、指標名-1の指標データ項目が表わす出現番号は、一意名-1用の指標名の指標データ項目が表わす出現番号と、同時に同じだけ増加される。

b. VARYING指定で一意名-2を書いた場合、この一意名-2が指標データ項目であると、この指標データ項目が表わす出現番号は、一意名-1用の指標名の指標データ項目が表わす出現番号と、同時に同じだけ増加される。一意名-2が指標データ項目でないと、この値は一意名-1用の指標が増加されるときに、1だけ増加される。

書き方 2

7. SEARCH文で、 SEARCH ALLの処理結果が正しく得られるためには、下記の条件が共に満たされる必要がある。

a. 表の中のデータは、一意名-1のデータ記述のASCENDING KEY句またはDESCENDING KEY句に指定された順番に並んでいること。

b. WHEN句中に書いたキーの内容が、表の要素を一意に識別するのに十分であること。

8. 書き方2のSEARCH文を用いると、 非逐次表引き処理が行われる。このとき、データ名-1用の指標データ項目の初期値は無視され、表引き処理につれて指標データ項目の値は変化する。ただし、その値が許される最大値よりも大きくなることも、許される最小値よりも小さくなることもない。 表の大きさについては、OCCURS句に説明してある。許容範囲内のどの指標値に関してもWHEN句に指定された条件のうちで満たされないものがある場合、AT END指定が書いてあれば、その無条件文-1に制御が移される。AT END指定が書いてなければ、次の実行完結文に制御が移される。どちらの場合も、指標データ項目の最後の値はどうなるかわからない。すべての条件が満たされたときは、そのときの要素の出現番号が指標データ項目に記録され、制御は無条件文-2に移される。
9. 書き方2では、表引き処理に使用される指標名は、一意名-1のINDEXED BY句の中の最初の（または唯一の）指標名だけである。一意名-1用のその他の指標名の指標データ項目の内容は変わらない。
10. (OSVS)(VSC2)(MF)無条件文-2もNEXT SENTENCE指定も必要ない。これらが指定されていないと、SEARCH文は表の中の条件に一致した値に指標を設定する。

図16-1に、WHEN指定が2つ含まれる書き方1の表引き処理の流れ図を示す。

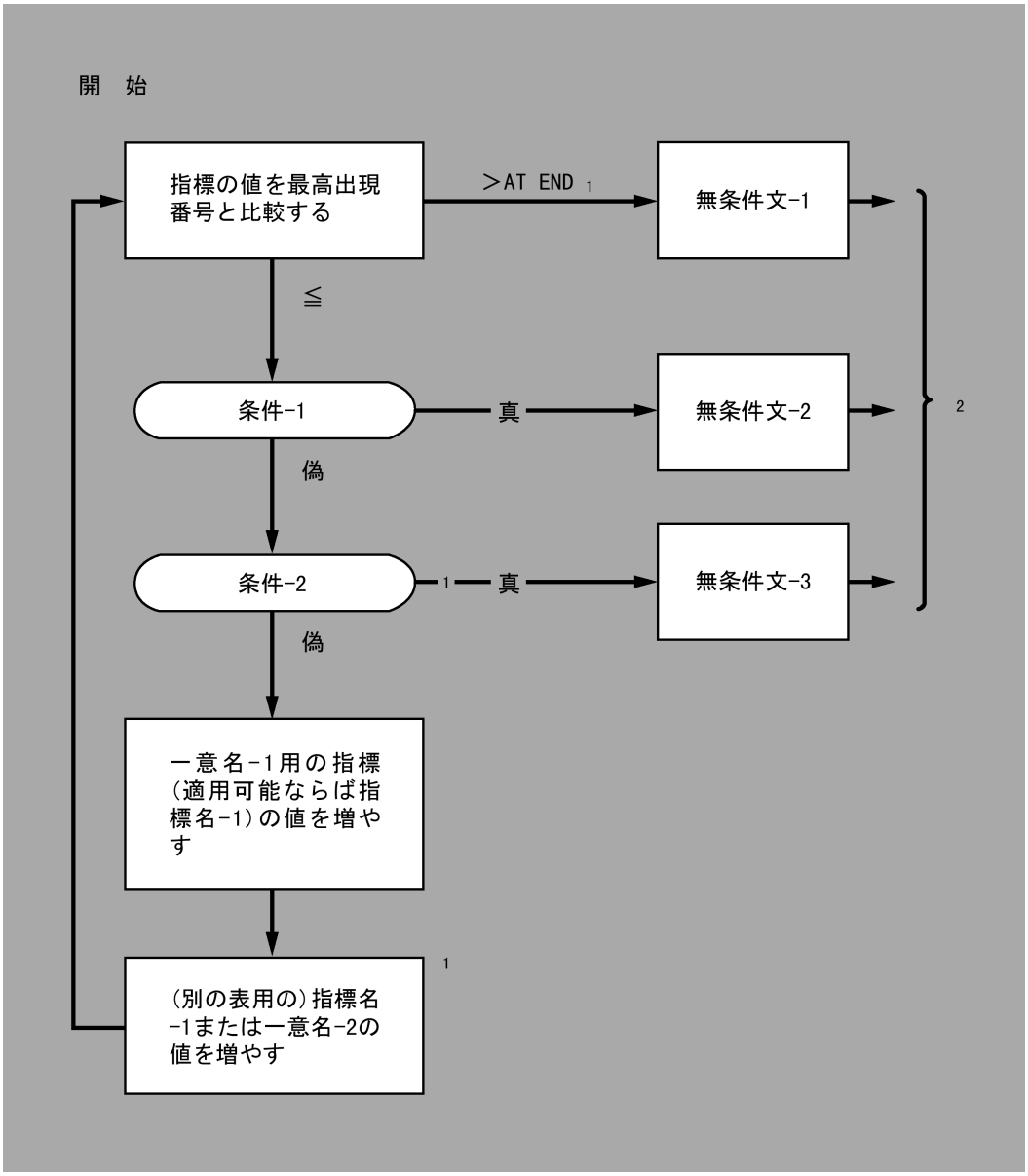


図16-1: WHEN 指定が2つ含まれる表引き処理の流れ図

1. これらの処理は、SEARCH文中に指定されたときにだけ行われる。
2. 各無条件文にGO TO文が含まれていなければ、制御は次の実行完結文に移される。

16.1.2 SERVICE（サービス）文

(OSVS)(VSC2)

SERVICE（サービス）文は、 通常、CICSプログラム内の連絡節中の項目の番地参照性を確立するために使用する。

一般形式

```
SERVICE { LABEL          }
        { RELOAD 一意名  }
```

構文規則

1. SERVICE LABELは、メインフレームのCICSプリプロセッサによって、制御の流れを示すために生成される。これは、一般的な用途に向けたものではない。
2. SERVICE LABEL文は手続き部にだけ書ける。宣言節には書けない。
3. 一意名は、連絡節内のBLLセル・ブロックで示される最初の01レベルの項目か、EXEC CICS文に続いて番地参照性を再確立する必要があるその他の連絡節内の01レベルの項目とする。

一般規則

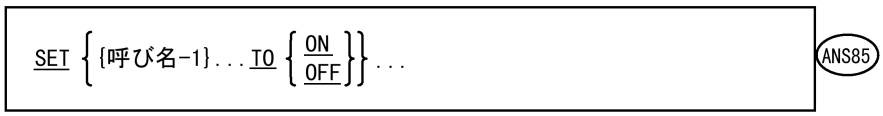
1. SERVICE LABEL文の次の文のところで、有効性を失っているレジスタがすべて再ロードされる。
2. CICSプログラム中でOS/VS COBOL BLL機構を使用する場合と、手続き部の冒頭でパラメータ・リストへの番地参照性が確立されなければならない。このために、手続き部の冒頭に「SERVICE RELOAD一意名」文を追加する。ここで、一意名は連絡節内の最初の項目であり、連絡節内の他のすべての記述項に対するポインタを保持している。
3. OSVS指令を適用して翻訳したプログラム中に、局所モードのEXEC CICS文が含まれている場合、各CICSコマンドの後ろにSERVICE RELOAD文を続けて、一意名はCICSコマンド中に使用されたのと同じポインタ（BLLセル・リファレンス）とする。
4. OSVS指令を適用して翻訳したCICSプログラムにおいては、大きさが4096バイト以上ある連絡節を参照するたびに、SERVICE RELOAD文を使用して、4096バイトを超える部分への番地参照性を再確立させる。
5. VS2指令を適用して翻訳したCICSプログラムにおいては、SERVICE RELOAD文は注記の役割を果たすだけである。

16.1.3 SET（設定）文

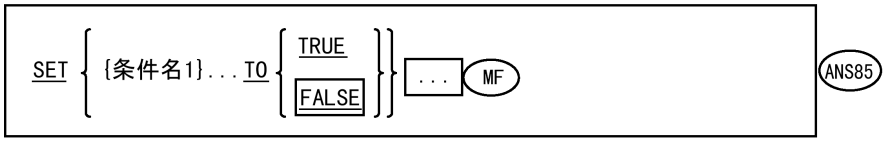
1. (ANS85)SET文は、 外部スイッチの状態を変更するために使用する。
2. (ANS85)SET文は、 条件変数の値を変更するために使用する。
3. (VSC2)(MF)SET文は、ポインタ変数にデータ項目の番地を割り当てるために使用する。SET文は、ポインタ変数内容を調整するために使用する。
4. SET文は、 表操作処理のための 参照点を決めるために、表要素の指標を設定する。
5. (MF)SET文は、プログラムの番地または手続きポインタ・データ項目へのプログラムの入口点を割り当てるために使用する。
6. (ISO2000)(MF)SET文は、オブジェクト参照を割り当てるために使用する。

一般形式

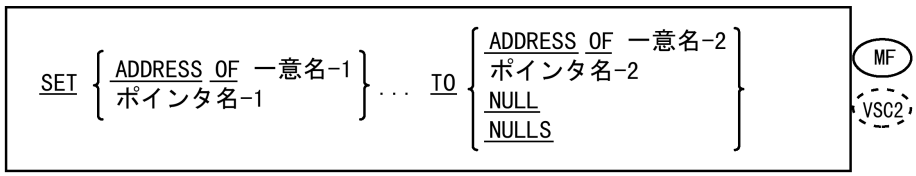
書き方 1



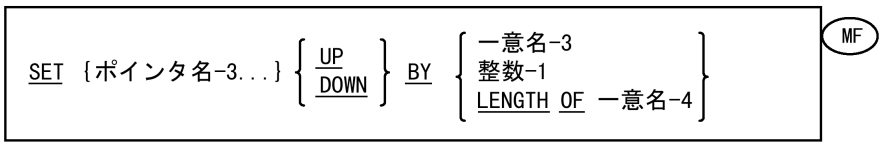
書き方 2



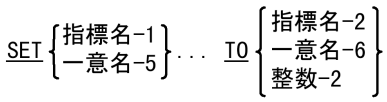
書き方 3



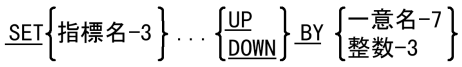
書き方 4



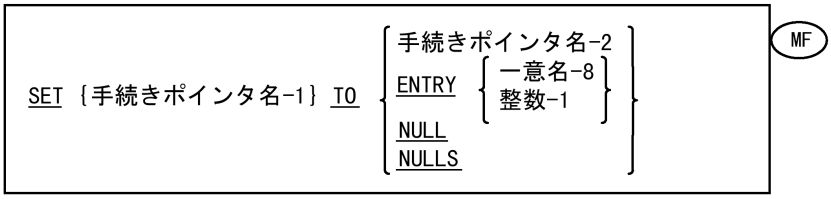
書き方 5



書き方 6



書き方 7



書き方 8

SET {一意名-9} ... TO 一意名-10

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - STICKY-LINKAGE - SET文によってポインタ・データ項目に入れられたデータ項目の番地を、副プログラムの呼出しの間に保持しておくか否かを決定する。

構文規則

すべての書き方

1. 表意定数のNULLやNULLSあるいはポインタ名-2や手続きポインタ名-2によって参照されるデータ項目は送出し側を表す。「ADDRESS OF 一意名-2」、「ENTRY 一意名-8」、「ENTRY 定数-1」、または送出し側に保持されている値は送出し値を表す。
   ポインタ名-1やポインタ名-3や手続きポインタ名-1によって参照されるデータ項目あるいは「ADDRESS OF 一意名-1」によって暗黙的に示されるCOBOLシステム領域は受取り側を表す。
   一意名-3、整数-1、「LENGTH OF 一意名-4」は増分値を表す。

書き方 1

2. (ANS85)呼び名-1は、状態を変更できる外部スイッチに関するものとする。
   (MF)SET文で参照できる外部スイッチについては、*特殊名段落*の節を参照。

書き方 2

3. (ANS85)条件名-1は、条件変数に関するものとする。
4. (MF)FALSE指定を書いた場合、条件名-1のデータ記述項のVALUE句にFALSEを指定しておく。
5. (MF)各FALSE指定またはTRUE指定は、それらの前に来る条件名-1の反復要素および前のFALSE指定またはTRUE指定に続く条件名-1の反復要素（もしあれば）に適用される。

書き方 3

6. (VSC2)(MF)一意名-1は、連絡節内の01レベルまたは77レベルの項目とする。
7. (VSC2)(MF)一意名-2は
   (VSC2)連絡節内で宣言されたレベルが77または01以上49以下のデータ項目を参照しなければならない。
8. (VSC2)(MF)ポインタ名-1とポインタ名-2はそれぞれ、USAGE IS POINTERを伴う基本データ項目を参照する一意名でなければならない。

書き方 4

9. (MF)ポインタ名-3はUSAGE IS POINTERを伴う基本データ項目を参照する一意名でなければならない。
10. (MF)一意名-3は整数の基本項目でなければならない。
11. (MF)整数-1は符合付きであってもよい。

書き方 5

12. 一意名-5と一意名-6はそれぞれ、指標データ項目または整数として記述されている基本項目を参照しなければならない。両方を指定した場合は、同一の基本項目を参照してはならない。

書き方 5 および 6

13. 整数-2と整数-3は、符号付きであってもよい。整数-2は、正の数とする。

書き方 6

14. 一意名-7は整数の基本項目を参照しなければならない。

書き方 7

15. (MF)手続きポインタ名-1と手続きポインタ名-2はそれぞれ、USAGE IS PROCEDURE-POINTERを伴う基本データ項目を参照する一意名でなければならない。
16. (MF)一意名-8は英数字のデータ項目として定義されていなければならない。その値がCOBOLのプログラム名であってもCOBOL以外のプログラム名であってもよいようにするためである。
17. (MF)定数-1は文字定数でなければならない。

書き方 8

18. 一意名-9は字類がオブジェクトであり、受取り側項目として許されている項目でなければならない。
19. 一意名-10はクラス名またはオブジェクト参照でなければならない。定義済みオブジェクト参照であるSUPERを指定してはならない。
20. 一意名-9によって参照されるデータ項目が一般的オブジェクト参照である場合、一意名-10に指定できる定義済みオブジェクト参照はSELFとNULLだけである。
21. 一意名-9によって参照されるデータ項目の記述にインタフェース-1を指すインタフェース名がある場合、一意名-10によって参照されるデータ項目は下記のどれかでなければならない。
    a. インタフェース-1に適合するインタフェースを指すインタフェース名を用いて記述されたオブジェクト参照
    b. 下記の規則に従うクラス名を用いて記述されたオブジェクト参照
        1. FACTORY指定を書いた場合は、指定したクラスのファクトリ・オブジェクトのインタフェースはインタフェース-1に適合しなければならない。
        2. FACTORY指定を書かなかった場合は、指定したクラスのオブジェクトのインタフェースはインタフェース-1に適合しなければならない。
    c. ACTIVE-CLASS指定を用いて記述されたオブジェクト参照。ただし、下記の規則に従う。
        1. FACTORY指定を書いた場合は、一意名-10によって参照されるデータ項目を含むクラスのファクトリ・オブジェクトのインタフェースはインタフェース-1に適合しなければならない。
        2. FACTORY指定を書かなかった場合は、一意名-10によって参照されるデータ項目を含むクラスのオブジェクトのインタフェースはインタフェース-1に適合しなければならない。
    d. ファクトリ・オブジェクトのインタフェースがインタフェース-1に適合するクラス名
    e. SET文を含むオブジェクトのインタフェースがインタフェース-1に適合する、定義済みオブジェクト参照のSELF
    f. 定義済みオブジェクト参照のNULL
22. 一意名-9によって参照されるデータ項目の記述にクラス名がある場合、一意名-10によって参照されるデータ項目は下記のどれかでなければならない。
    a. クラス名を用いて記述されたオブジェクト参照。ただし、下記の規則に従う。
        1. 一意名-9によって参照されるデータ項目にONLY指定が書いた場合、一意名-10によって参照されるデータ項目はONLY指定を用いて記述されていなければならず、一意名-10によって参照されるデータ項目の記述中に指定されているクラス名は一意名-9によって参照されるデータ項目の記述中に指定されているクラス名と同じである。

2. 一意名-9によって参照されるデータ項目にONLY指定を書かなかった場合、一意名-10によって参照されるデータ項目は一意名-9によって参照されるデータ項目の記述中に指定されているのと同じクラスまたはそのサブクラスを参照しなければならない。
3. FACTORY指定の有無は一意名-9によって参照されるデータ項目の記述と同じである。

b. ACTIVE-CLASS指定を用いて記述されたオブジェクト参照。ただし、下記の規則に従う。
1. 一意名-9によって参照されるデータ項目の記述にONLY指定があってはならない。
2. 一意名-10によって参照されるデータ項目を含むクラスは一意名-9によって参照されるデータ項目の記述中に指定されているのと同じクラスまたはそのサブクラスでなければならない。
3. FACTORY指定の有無は一意名-9によって参照されるデータ項目の記述と同じでなければならない。

c. 定義済みオブジェクト参照のSELF。ただし、下記の規則に従う。
1. 一意名-9によって参照されるデータ項目の記述ONLY指定があってはならない。
2. SET文を含むオブジェクトのクラスは一意名-9によって参照されるデータ項目の記述中に指定されているのと同じクラスまたはそのサブクラスでなければならない。
3. 一意名-9によって参照されるデータ項目の記述にFACTORY指定がない場合、SET文を含むオブジェクトはインスタンス・オブジェクトとする。
4. 一意名-9によって参照されるデータ項目の記述にFACTORY指定がある場合、SET文を含むオブジェクトはファクトリ・オブジェクトとする。

d. クラス名。ただし、一意名-9によって参照されるデータ項目の記述にFACTORY指定があるものとして、そのクラス名は一意名-9によって参照されるデータ項目の記述中に指定されているのと同じクラスまたはそのサブクラスを参照しなければならない。

e. 定義済みオブジェクト参照のNULL

23. 一意名-9によって参照されるデータ項目ACTIVE-CLASS指定を書いた場合、一意名-10によって参照されるデータ項目は下記のどちらかでなければならない。

a. ACTIVE-CLASS指定を用いて記述されたオブジェクト参照。ただし、FACTORY指定の有無は一意名-9によって参照されるデータ項目の場合と同じでなければならない。

b. 定義済みオブジェクト参照のSELF。ただし、下記の規則に従う。
1. 一意名-9によって参照されるデータ項目の記述にFACTORY指定がない場合、SET文を含むオブジェクトはインスタンス・オブジェクトでなければならない。

2. 一意名-9によって参照されるデータ項目の記述にFACTORY指定がある場合、SET文を含むオブジェクトはファクトリ・オブジェクトでなければならない。

c. 定義済みオブジェクト参照のNULL

一般規則

書き方 1

1. (ANS85)指定した呼び名-1に関する各外部スイッチの状態は、ONを指定した場合はオンに、OFFを指定した場合はオフに、設定される。(前述の*スイッチ状態条件*の節を参照。)

書き方 2

2. (ANS85)条件名-1に関するVALUE句中の定数が、VALUE句の規則に従って、条件変数に入れられる。(前述の*VALUE句*の節を参照。) VALUE句の中に複数の定数を指定してある場合には、条件変数の値はVALUE句内の最初の定数に等しく設定される。
3. (ANS85)複数の条件名を指定した場合、そのSET文を実行した結果は、その中の各条件名-1用にその順番どおりに別々のSET文を書いて実行したのと等しい。
4. (MF)FALSE指定を書くと、条件名-1に関するVALUE句のFALSE指定の中の定数が、VALUE句の規則に従って条件変数に入れられる。(前述の*VALUE句*の節を参照。)

書き方 3

5. (VSC2)(MF)送出し値はデータ項目の番地を表す。ポインタ名-2を指定した場合、送出し値はポインタ名-2によって参照されるデータ項目に関して保持されている値である。「ADDRESS OF 一意名-2」を指定した場合、送出し値は一意名-2の番地を表す。
6. (VSC2)(MF)ポインタ名-1を指定した場合、送出し値はポインタ名-1によって参照されるデータ名に転記される。
7. (VSC2)(MF)「ADDRESS OF 一意名-1」を指定した場合、送出し値はCOBOLのシステム領域に転記される。それ以降、 ランタイム要素は一意名-1によって参照される記憶領域が送出し値によって表される番地にあるものとして動作する。
   (MF)副プログラムを呼び出すときにリンクを維持するか否かは、STICKY-LINKAGE指令の影響を受ける。

書き方 4

8. (MF)SET文が実行される前に、ポインタ名-3によって参照されるデータ項目の値は論理レコード内のデータ項目の番地である当初の番地を表していなければならない。SET文が実行された後では、ポインタ名-3によって参照されるデータ項目の値は新しい番地を表す。当初の番地と新しい番地が共に同じ論理レコード（あるいは、番地空間がセグメント化れている環境では、同じセグメント）内に収まらない場合、ポインタ名-3によって参照されるデータ項目の値を使用すると、結果はどうなるか分からない。
9. (MF)UP句を指定した場合、増分値に指定された値を当初の番地に加えて、新しい番地が算出される。

10. (MF)DOWN句を指定した場合、増分値に指定された値を当初の番地から引いて、新しい番地が算出される。

書き方 5

11.
    a. 指標名-2、一意名-6、整数-2によって参照される表要素に出現番号が対応する表要素を参照するように指標名-1の値が設定される。一意名-6が指標データ項目を参照する場合、または指標名-2が一意名-1と同じ表に関連している場合、変換は行われない。
    b. 一意名-5が指標データ項目を参照する場合、それを指標名-2または同様に指標項目を参照する一意名-6のどちらかの内容と等しく設定できる。どちらの場合も、変換は行われない。
    c. 一意名-5が指標データ項目を参照しない場合、指標名-2の値に対応する出現番号にのみ設定できる。この場合、一意名-6も整数-2も使用できない。
    d. この処理が指標名-1または一意名-5（指定した場合）のすべての繰返しに対して繰り返えされる。そのたびに、指標名-2または一意名-6の値には、この文を実行を開始したときの値が使用される。一意名-5などに関する添字付けまたは指標付けは、それぞれのデータ項目の値を変更する直前に評価される。
12. SET文における各種の作用対象の有効な組合わせを、表16-1に示す。この中に、適用される一般規則の番号も示す。

表16-1: 指標を作用対象とするSET 文の有効な作用対象の組合わせ

送出し側項目	受取り側項目		
	整数データ項目	指標名	指標データ項目
整数定数	無効/11c	有効/11a	無効/11b
整数データ項目	無効/11c	有効/11a	無効/11b
指標名	有効/11c	有効/11a	有効/11b[1]
指標データ項目	無効/11c	有効/11a[1]	有効/11b[1]

[1] = 変換は行われない

書き方 5 および 6

13. 指標名は対応する表を定義する際に、 OCCURS句のINDEXED BY指定を用いて定義しておく。
14. 指標名-1を指定する場合、SET文を実行した後の指標の値は、対応する表の要素の数に見合っていなければならない。 SEARCH文またはPERFORM文を実行した後の指標名の指標の値は、どうなっているかわからない。(前述の*PERFORM*文および*SEARCH*文の節を参照。)

    指標名-2を指定する場合、SET文を実行する前の指標の値は、対応する表の要素の数に見合っていなければならない。

指標名-3を指定する場合、SET文を実行する前と後の指標の値は共に、対応する表の要素の数に見合っていなければならない。

書き方 6

15. 指標名-3の内容が、整数-3または一意名-7によって表わされる出現番号の値だけ、増分（UP BY）または減分（DOWN BY）される。指標名-3に複数の項目を指定すると、そのすべてについて上記の処理が繰り返される。その際、初回と同じように、一意名-7の値が使用される。

書き方 7

16. ㊍送出し値はCOBOLまたはCOBOL以外のプログラム内の手続きの開始番地を表す。
17. ㊍送出し値は手続きポインタ-1によって参照されるデータ項目に転記される。
18. ㊍手続きポインタ名-2を指定した場合、送出し値は手続きポインタ-2によって参照されるデータ項目内に収められている値となる。
19. ㊍定数-1、または一意名-8によって参照されるデータ項目の内容は、参照される手続きの名前である。参照される手続きがCOBOLの手続きである場合、参照される手続きの名前にはプログラム名または入口名が収められていなければならない。前者は参照されるプログラムのプログラム名段落に記述されているものである。後者は参照される手続きのENTRY文に記述されているものである。
    ㊍呼ばれるプログラムがCOBOLのプログラムではない場合、プログラムまたは手続きの名前を形成するための規則がインタフェースに関するCOBOLシステム・ドキュメンテーションに記載されている。
    ㊍参照される手続きが前に利用可能にされており、SET文を実行する時点でもまだ利用可能な場合には、送出し値は参照される手続きの番地を表す。
    ㊍参照される手続きがSET文を実行する時点でもまだ利用可能ではない場合には、送出し値はCOBOLシステムのエラー手続きの番地を表す。

書き方 8

20. 一意名-10がオブジェクト参照である場合、一意名-10によって識別されるオブジェクトへの参照が、一意名-9によって参照される各データ項目に指定された順に収められる。
21. 一意名-10がクラス名である場合、一意名-10によって識別されるクラスのファクトリ・オブジェクトへの参照が、一意名-9によって参照される各データ項目に指定された順に収められる。

16.1.4 SORT（整列）文

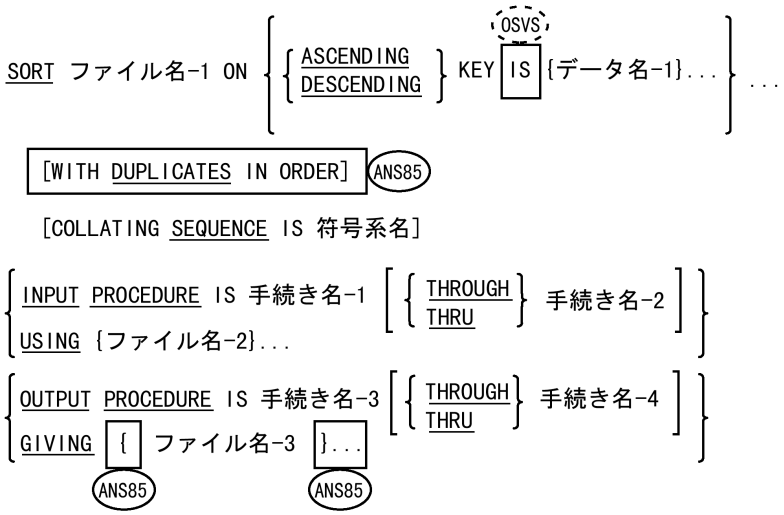
SORT（整列）文は、 ファイル中のレコードを整列させる。この処理は3つの段階に分かれる。最初の段階では、入力手続きを実行するかまたは他のファイルからレコードを転送するかして、整列用ファイルを作成する。次の段階では、指定されたキーの組に基づいて、整列用ファイル中のレコードの順序をそろえる。最後の段階では、 整列させたレコードをその順番に出力手続きまたは出力ファイルに引き渡す。
(MF)表の中の要素を整列させるためにも、SORT文を使用できる。

例：

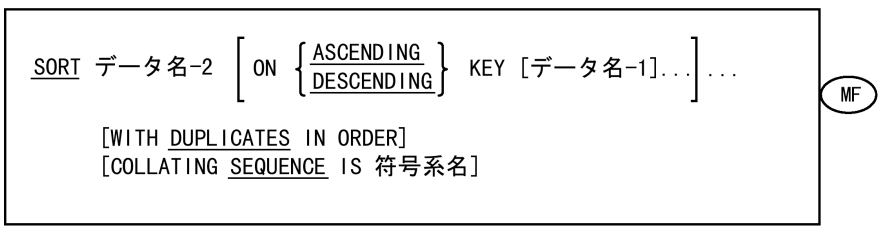
1. ファイル中の要素の順序を整えるための入出力手続きを伴うSORT動詞の使用例が『言語リファレンス - 追加トピック』の「例」の章に記載されている。
2. 表の要素の順序を整えるためのSORT動詞の使用例が『言語リファレンス - 追加トピック』の「例」の章に記載されている。

一般形式

書き方 1



書き方 2



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - CALLSORT - SORTおよびMERGEの処理を行うために使用するプログラムを指定する。

構文規則

すべての書き方

1. 書き方1の SORT文は、手続き部の宣言部分、SORT文またはMERGE文に関連する入力手続き、出力手続きの場所には置くことができない。
   (MF)書き方2のSORT文を宣言節に置いてもよい。
2. データ名-1は キーのデータ名であり、下記の規則に従う。
    a. キー・データ名は修飾してもよい。
    b. キー・データ名のデータ項目は、可変長項目であってはならない。
    c. キー・データ項目は、浮動小数点数項目でもよい。
    d. (OSVS)(VSC2)(MF)キーが外部浮動小数点数であると、コンパイラはそのデータ項目を数字データとしてではなく文字データとして扱う。レコードがそろえられる順序は、使用する文字の大小順序に左右される。
    e. (OSVS)(VSC2)(MF)キー・データ項目が内部浮動小数点数である場合、キーの順序は数値順となる。

書き方 1

3. ファイル名-1は、データ部の整列併合ファイル記述項に記述しておく。
4. データ名-1はキーのデータ名であり、下記の規則に従う。
    a. キーとするデータ項目は、ファイル名-1のレコード中に記述しておく。
    b. ファイル名-1に複数のレコード記述を含める場合、キー・データ項目は1つのレコード記述の中だけに指定する。
    c. キー・データ項目には、OCCURS句を含む記述項は記述できない。また、OCCURS句を含む記述項の下位にキー・データ項目を属させてはならない。
    d. ファイル名-1のファイルのレコードが可変長である場合、データ名をキーに指定するデータ項目はすべて、そのファイルの最初の x 文字位置のレコードの範囲内に入っていること。ここで x は、ファイル名-1によって参照されるファイルに指定された最小レコードサイズを示す。
5. 語THRUとTHROUGHは同義語である。
6. ファイル名-2とファイル名-3はデータ部の中の整列併合ファイル記述項にではなく、ファイル記述項に記述しておく。
7. (ANS85)ファイル名-2のファイルとファイル名-3のファイルは、同じマルチ・ファイル・リールに収録されていてもよい。

8. (ANS85)ファイル名-3が索引ファイルを指す場合、最初のデータ名-1には ASCENDINGを指定する。また、データ名-1がレコード上で占める文字位置は、そのファイルの主レコードキーと同じにする。
9. 1つの SORT文の中で使用しているファイル名の組は、SAME SORT AREA句またはSAME SORT-MERGE AREA句には指定できない。 GIVING指定の対象となっている複数のファイルは、同じSAME句には指定できない。
10. (ANS85)GIVINGを指定したとき、ファイル名-3のファイルに含まれるレコードが可変長である場合、ファイル名-1のファイル中に含まれるレコードの大きさは、ファイル名-3のレコードの最小のもの以上かつ最大のもの以下にする。ファイル名-3のファイルに含まれるレコードが固定長である場合、ファイル名-1のファイル中に含まれるレコードの大きさは、ファイル名-3のレコード最大のものを超えてはならない。
    (ANS85)これらのレコードを構成する基本データ項目のデータ記述が異なる場合、対応するレコード同士の文字数が等しくなるように記述することは、プログラマの責任である。
11. (ANS85)USINGを指定し、ファイル名-1のファイルに含まれるレコードが可変長である場合、 ファイル名-2のファイル中に含まれるレコードの大きさは、ファイル名-1のレコードの最小のもの以上かつ最大のもの以下にする。ファイル名-1のファイルに含まれるレコードが固定長である場合、ファイル名-2のファイル中に含まれるレコードの大きさは、ファイル名-1のレコードの最大のものを超えてはならない。
12. 手続き名-1は、入力手続きの名前を表わす。手続き名-3は、出力手続きの名前を表わす。
13. 手続き名-1、手続き名-2、手続き名-3、手続き名-4は、節名とする。
    この制限は解除された。
14. USING句またはGIVING句に10個以上のファイル名を指定したい場合、コンパイラ指令のCALLSORT"EXTSM"を使用しなければならない。この指令を使用すると、255個までのファイルを指定できる。

書き方 2

15. (MF)データ名-2は修飾してもよい。データ名-2のデータ記述項には、OCCURS句を記述しておく。
16. (MF)データ名-1のデータ項目は、データ名-2のデータ項目と同じであるか、またはデータ名-2のデータ項目の下位に属する項目とする。
17. (MF)データ名-1とデータ名-2が同じでない場合、データ名-1のデータ項目のデータ記述項に OCCURS句が記述してあってはならない。また、データ名-1のデータ項目は、データ名-2の下位に属しOCCURS句を伴うデータ項目の下位にも属してはならない。
18. (MF)データ名-2の表の記述中にKEY指定がある場合にだけ、KEY指定を省略できる。
19. (MF)OCCURS句を伴うデータ項目の下位にデータ名-2のデータ項目が属す場合、データ名-2のデータ項目の親であるOCCURS句を伴うデータ項目には、指標名を付ける。

一般規則

すべての書き方

1. 語ASCENDINGおよびDESCENDINGは、別のASCENDINGまたはDESCENDINGが出てくるまで、後ろに繰り返し続くすべてのデータ名-1に効力を及ぼす。
2. データ名-1のデータ項目は、キー・データ項目である。このキー・データ項目によって、ファイル名-1のファイルからレコードが引き取られる順序または 整列が終了した後で 表の要素が格納される順序が決まる。キーの強さの順序は、 SORT文の中に記述した順である。 ASCENDINGまたは DESCENDING の指定は、キーの強さには関係ない。
3. DUPLICATES指定をした場合、あるレコードまたは表要素のすべてのキー・データ項目の内容が他のいくつかのレコードまたは表要素の対応するキー・データ項目の内容と等しいとき、引き取られるレコードの順序は下記のようになる。
    a. SORT文の中または実行時スイッチに指定された入力ファイルの順序。 入力ファイルからレコードを読み込んだ場合は、その順序が保たれる。
    b. 入力手続きが用いられているときは、入力手続きからレコードが引き渡された順序。
    c. 整列処理が実行される前の、表要素の内容の相対順序。
4. DUPLICATES指定をしなかった場合、あるレコードまたは表要素のすべてのキー・データ項目の内容が他のいくつかのレコードまたは表要素の対応するキー・データ項目の内容と等しいとき、引き取られるレコード順序または表要素の内容の相対順序はどうなるかわからない。
5. 文字のキー・データ項目の比較に適用される文字の大小順序は、SORT文の実行開始時に、下記の優先順序で決定される。
    a. 最初は、指定してあれば、 SORT文中のCOLLATING SEQUENCEに指定した文字の大小順序。
    b. 次は、プログラム用に設定してある文字の大小順序。

書き方 1

6. SORT文は、ファイル名-2のファイル中のすべてのレコードまたは入力手続きによって、ファイル名-1のファイルに引き渡されるレコードを受け取る。その後、データ名-1のデータ項目の値と ASCENDING または DESCENDING指定によって定まる順に、それらのレコードを並べ変え、出力手続きまたはファイル名-3のファイルに引き渡す。
7. ファイル名-1のファイルから返された2つのレコードの相対順序を決めるために、比較条件の規則に従い、対応するキー・データ項目の内容が、最も強いキー・データ項目から始まって、順に比較される。
    a. 対応するキー・データ項目同士の内容が異なり、キーにASCENDING指定がなされている場合、キーの値が小さい方のレコードが先に返される。
    b. 対応するキー・データ項目同士の内容が異なり、キーにDESCENDING指定がなされている場合、キーの値が大きい方のレコードが先に返される。
    c. 対応するキー・データ項目同士の内容が等しい場合、次に強いキー・データ項目の内容に基づいて、順序が決定される。

8. SORT文の実行は、下記の3つの段階に区分される。
    a. ファイル名-1のファイル中で、レコードを利用できるようにする。このためには、入力手続き中のRELEASE文を実行するか、またはファイル名-2に対して暗黙的にREAD文を実行する。この段階の処理が始まるときに、ファイル名-2のファイルは開かれていてはならない。この段階の処理が終わるとき、ファイル名-2のファイルは閉じられている。
    b. ファイル名-1のファイル中のレコードの順序がそろえられる。この段階では、ファイル名-2およびファイル名-3のファイルの処理は、何も行われない。
    c. ファイル名-1のファイルのレコードがそろえられた順序に従って、利用できるようにされる。 このために、順序をそろえられたレコードがファイル名-3のファイルに書き出されるか、RETURN文を実行することによって出力手続きで利用できるようにされる。この段階の処理が始まるときに、ファイル名-3のファイルは開かれていてはならない。この段階の処理が終わるとき、ファイル名-3のファイルは閉じられている。
9. 入力手続きでは、RELEASE文によって1件ずつファイル名-1のファイルに引き渡されるレコードを、必要に応じて選択したり修正したり複写したりする。この入力手続きの範囲には、その中からCALL文、PROGRAM指定のないEXIT文、GO TO文、PERFORM文によって制御を移された結果実行されるすべての文が含まれる。この入力手続きの範囲内で、MERGE文、RETURN文、SORT文を起動するようなことはできない。
10. 入力手続きを指定すると、SORT文によってファイル名-1のファイル中のレコードの並べ替えが開始される前に、入力手続きに制御が渡される。コンパイラは、入力手続きの最後の文の末尾に復帰機構を組み込む。入力手続きの最後の文に制御が渡ると、ファイル名-1のファイルに引き渡されているレコードの整列処理が開始される。
11. USINGを指定すると、ファイル名-2のファイル中のすべてのレコードが、ファイル名-1のファイルに書き出される。SORT文を実行すると、ファイル名-2の各ファイルに対して、下記の処理が行われる。
    a. ファイルの処理が開始される。これはINTPUT指定およびWITH LOCK指定をしたOPEN文を実行するように行われる。 .
    b. 論理レコードが、入力ファイルから整列用ファイルに引き渡される。これはNEXTおよびAT END指定をしたREAD文が実行されるように行われる。ファイル名-1のファイルに含まれるレコードが固定長である場合、ファイル名-2のファイル中にファイル名-1の固定長レコードよりも短いものがあると、ファイル名-2のレコードがファイル名-1のファイルに引き渡されるときに、ファイル名-2のレコードの後部に空白が埋められる。ファイル名-1のファイルのレコードが可変長であると、ファイル名-1のファイルに書き出されるレコードの大きさは、ファイル名-2またはファイル名-3から読み込んだときのレコードの大きさとなる。このことは、ファイル名-1のファイルのファイル記述項に指定したRECORD VARYING DEPENDING ON句またはOCCURS DEPENDING ON句のDEPENDING ON指定の対象となっているデータ項目の内容にかかわらず、適用される。

c. ファイルの処理が終了される。これは任意指定の何もないCLOSE文が実行されたように行われる。この終了処理が済んだ後で、SORT文によるファイル名-1のファイルの整列処理が開始される。
相対ファイルに関しては、GIVING指定の対象にファイル名-2を指定しないと、SORT文の実行が終了したときに、RELATIVE KEYデータ項目の内容はどうなっているかわからない。
USE AFTER EXCEPTION手続きを指定してあれば、上記の暗黙の処理においてそれも対象となる。しかし、そのUSE手続きの中から、ファイル名-2のファイルを操作したり、そのレコード領域をアクセスしたりするような文が実行されるようなことがあってはならない。
ファイル名-2のファイル記述項中に指定したRECORD VARYING DEPENDING ON句の対象となっているデータ項目の値は、SORT文が完了した時点でどのようになっているかはわからない。

12. 出力手続きでは、RETURN文によって1件ずつファイル名-1のファイルから引き取られる整列済みのレコードを、必要に応じて選択したり修正したり複写したりする。この出力手続きの範囲には、その中からCALL文、PROGRAM指定またはMETHOD指定のないEXIT文、GO TO文、PERFORM文によって制御を移された結果実行されるすべての文が含まれる。また、この出力手続きの範囲内の文を実行した結果として実行される宣言手続き中のすべての文も、この出力手続きの範囲に含まれる。この出力手続きの範囲内で、MERGE文、RELEASE文、SORT文を起動するようなことはできない。
13. 出力手続きを指定すると、SORT文によってファイル名-1のファイル中のレコードの並べ替えが終了した後で、出力手続きに制御が渡される。コンパイラは、出力手続きの最後の文の末尾に復帰機構を組み込む。出力手続きの最後の文に制御が渡ると、復帰機構によって SORT文の処理が終了され、SORT文の次の実行文に制御が移される。出力手続きに入る前に、レコードの並べ替えは済んでいる。要求されれば次のレコードを引き渡せる状態に達している。出力手続き中のRETURN文は、次のレコードを求める要求である。
14. GIVINGを指定すると、順序をそろえられたすべてのレコードがファイル名-3のファイルに書き出される。これはSORT文に暗黙的に出力手続きを使用することである。SORT文を実行すると、ファイル名-3の各ファイルに対して下記の処理が行われる。
    a. ファイルの処理が開始される。これはOUTPUT指定をしたOPEN文を、暗黙的に実行することである。この開始処理は、入力手続きがあれば、その実行が済んだ後で行われる。

b. 論理レコードが、整列用ファイルから出力ファイルに書き出される。これは何も任意指定をしないWRITE文を、暗黙的に実行することである。ファイル名-3のファイルに含まれるレコードが固定長である場合、ファイル名-1のファイル中にファイル名-3の固定長レコードよりも短いものがあると、ファイル名-1のレコードがファイル-3のファイルに書き出されるときに、ファイル名-1のレコードの後部に空白が埋められる。ファイル名-3のファイルのレコードが可変長であると、ファイル名-3のファイルに書き出されるレコードの大きさは、ファイル名-1から読み込んだときのレコードの大きさとなる。このことは、ファイル名-3のファイルのファイル記述項に指定したRECORD VARYING DEPENDING ON句またはOCCURS DEPENDING ON句の対象となっているデータ項目の内容にかかわらず、適用される。

出力ファイルが相対ファイルである場合、最初に引き取られるレコードのRELATIVE KEYデータ項目の値は "1" 、2番目に引き取られるレコードのRELATIVE KEYデータ項目の値は "2" 、という具合いになる。SORT文の実行が終了した後のRELATIVE KEYデータ項目の内容は、どうなっているかわからない。

c. ファイルの処理が終了される。これは任意指定の何もないCLOSE文を、暗黙的に実行することである。

USE AFTER EXCEPTION手続きを指定してあれば、上記の暗黙の処理において、それも対象となる。しかし、そのUSE手続きの中から、ファイル名-3のファイルを操作したりそのレコード領域を呼び出したりするような文が実行されるようなことがあってはならない。外部的に定義されているファイル境界を超えて最初に書き出しが試みられたときに、そのファイルに関してUSE AFTER EXCEPTION手続きを指定してあれば、その手続きが実行される。USE文の規則に従って制御が戻されると、さらに暗黙の書き出し処理が行われることはない。上記のように、そのファイルの処理は終了される。

ファイル名-1のファイル記述項中に指定したRECORD VARYING DEPENDING ON句の対象となっているデータ項目の値は、 GIVING指定をしたSORT文が完了した時点でどのようになっているかはわからない。

15. SORT文を含むプログラムを区分化できる。（***言語リファレンス - 追加トピック***の*区分化*の節を参照。）ただし、下記の制限がある。

a. 独立区分ではない区分中の節内に SORT文が存在する場合、そのSORT文から呼び出される入力手続きまたは出力手続きは、下記のどちらかの状態で組み込まれていなければならない。

- 非独立区分中に完全に含まれる。
- 単一の独立区分中に完全に含まれる。

b. 独立区分中にSORT文が存在する場合、そのSORT文から呼び出される入力手続きまたは出力手続きは、下記のどちらかの状態で組み込まれていなければならない。

- 非独立区分中に完全に含まれる。
- SORT文と同じ独立区分中に完全に含まれる。

16. (OSVS)(VSC2)(MF)SORT 実行時要素用に、特殊レジスタのSORT-RETURNが用意されている。この特殊レジスタには、整列処理が終わったときに0（正常終了）または16（不成功）が設定される。入力/出力手続きの中で、この 特殊レジスタに値16を設定して、 途中で整列処理を終わらせることができる。この場合、整列処理は次のRETURN文またはRELEASE文のところで終了される。

書き方 2

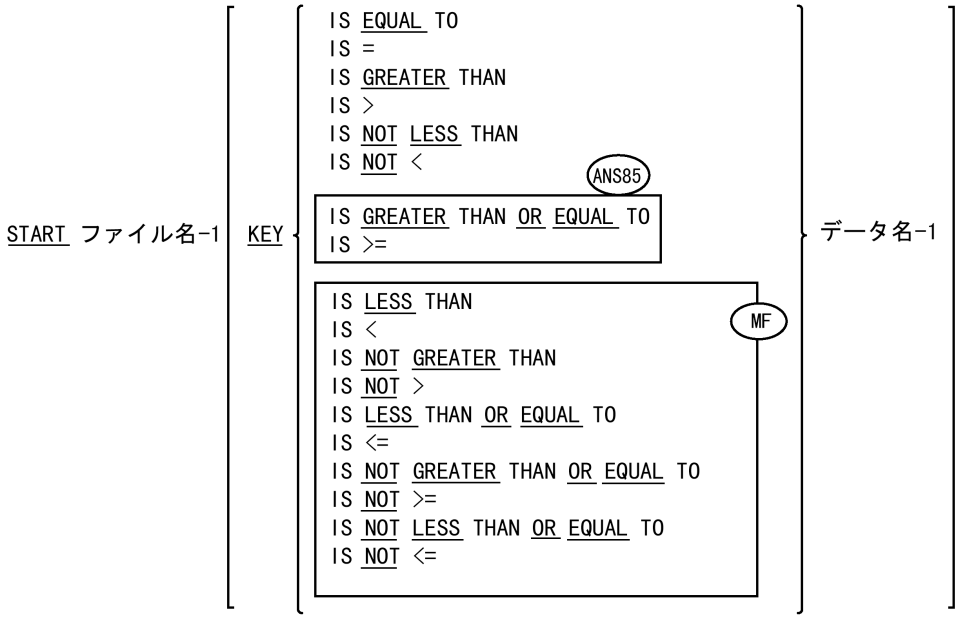
17. (MF)SORT文は、データ名-2の表の要素の順序をそろえる。どのような順序にそろえるかは、 ASCENDING または の指定および KEYの指定によって決まる。
18. (MF)順序をそろえた後の表要素の相対順序を決めるために、比較条件の規則に従い、対応するキー・データ項目の内容が、最も強いキー・データ項目から始まって、順に比較される。
    a. 対応するキー・データ項目同士の内容が異なり、キーにASCENDING指定がなされている場合、キーの値が小さい方の表要素に小さい出現番号が割り当てられる。
    b. 対応するキー・データ項目同士の内容が異なり、キーにDESCENDING指定がなされている場合、キーの値が大きい方の表要素に小さい出現番号が割り当てられる。
    c. 対応するキー・データ項目同士の内容が等しい場合、次に強いキー・データ項目の内容に基づいて、順序が決定される。
19. (MF)データ名-2の表の要素の出現番号は、OCCURS句の規則に従って決定される。
20. (MF)OCCURS句を伴うデータ項目の下位に属するデータ項目をデータ名-2が指す場合、親の表の最初または唯一の指標名を、希望する値に設定してからSORT文を実行する。
21. (MF)KEY指定をしないと、データ名-2の表のデータ記述項中のKEY指定によって、順序が決定される。
22. (MF)KEY指定をすると、この指定の方が、データ名-2の表のデータ記述項中のKEY指定よりも優先する。
23. (MF)KEY指定をしてもデータ名-1を省略すると、データ名-2のデータ項目がキー・データ項目とされる。
24. (MF)データ名-2の表の要素を 整列した結果は、同じ表に書き戻される。
25. (MF)COLLATING SEQUENCE指定をしても、注記にとどまる。

16.1.5 START（位置決め）文

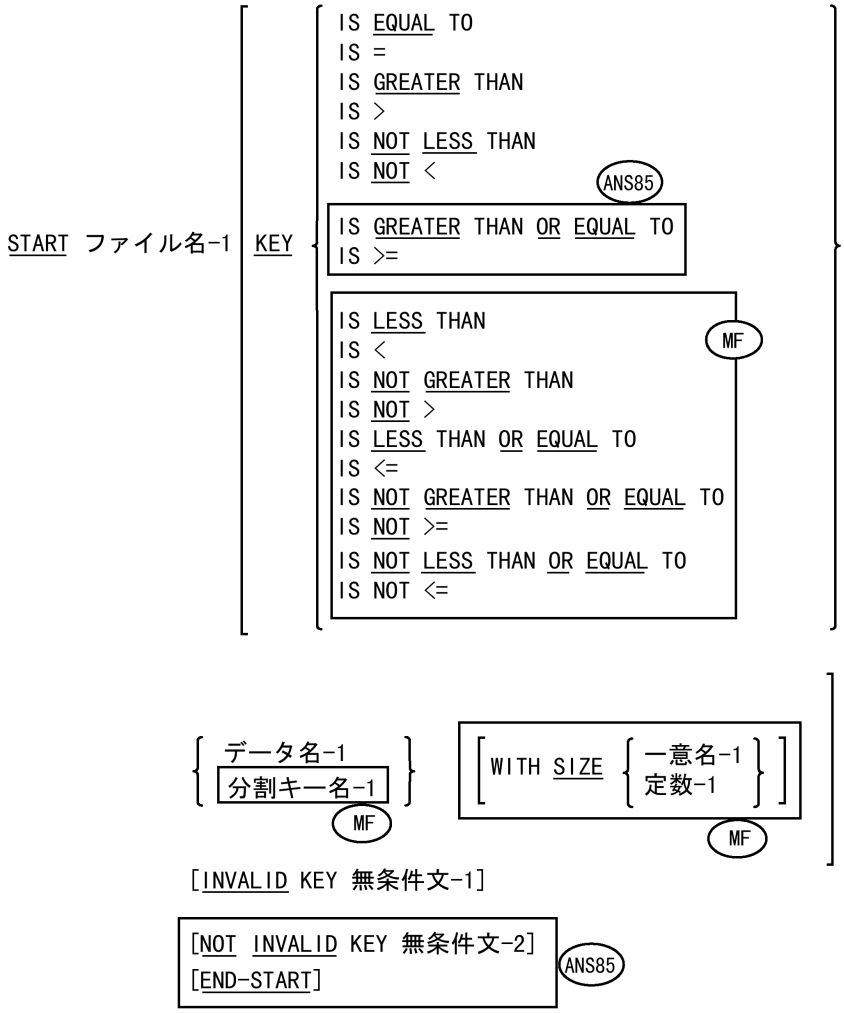
START（位置決め）文は、相対ファイルまたは索引ファイルの中で、以降に検索するレコードに論理的に位置付ける。この文は順編成のファイルには適用できない。

一般形式

書き方 1（相対ファイル）



書き方 2 (索引ファイル)



必須の比較文字 ">"、"<"、">="、"<="、"=" には下線を引いていないことに注意。これは、 "≥" (以上)のような他の記号と混同することを避けるためである。

構文規則

すべての書き方 (相対ファイルおよび索引ファイル)

1. ファイル名-1は、相対ファイルまたは索引ファイルの名前にする。
2. ファイル名-1は、順呼出し法または動的呼出し法のファイルの名前にする。
3. データ名-1は修飾してもよい。
4. ファイル名-1に適用できるUSE手続きを指定していないときは、INVALID KEY句を指定する。

   (OSVS)(VSC2)(MF)この規則は強制しない。

書き方 1（相対ファイル）

5. KEY指定をする場合、データ名-1は対応するファイル管理記述項のRELATIVE KEYに指定したデータ項目とする。

書き方 2（索引ファイル）

6. KEY指定をする場合、ファイル名-1のレコードキーとして指定したデータ項目を、データ名-1のデータ項目として使用できる。さらに、ファイル名-1のレコードキーとして指定したデータ項目の下位に属し、その レコードキー・データ項目と左端の文字位置が同じで項類が英数字のデータ項目を、データ名-1のデータ項目として使用できる。
7. (MF)分割キー名-1のデータ項目には、ファイル名-1のレコードキーとして指定したデータ項目のいくつかを使用できる。
8. (MF)WITH SIZE指定は、位置決め処理で使用するキー中の文字数を示す。
9. (MF)WITH SIZE指定の対象とした場合、一意名-1は基本整数データ項目の名前とする。

一般規則

すべての書き方（すべてのファイル）

1. START文を実行するときには、ファイル名-1のファイルはINPUTモードまたはI-0モードで開いておく。(前述の*OPEN*文の節を参照。)
2. KEY指定をしないと、比較演算IS EQUAL TOを暗黙的に指定したものとみなされる。
3. START文を実行すると、ファイル名-1に対応する FILE STATUSデータ項目を指定してあれば、その値が更新される。(前述の*入出力状態*の節を参照。)
4. (MF)START文が、レコードロックを取得したり検出したり解除したりすることはない。

書き方 1（相対ファイル）

5. KEY指定中の比較演算子によって表わされる比較は、ファイル名-1のファイル中のレコードのキー（下記の6項を参照）とデータ名-1との間で行われる。
   a. (ANS85)比較演算子で、キーをデータ名-1と等しいか、より大きいか、以上と指定すると、
      ファイル中に現存する論理レコードのうちで、その比較条件を満足する最初のものに ファイル位置指示子が位置付けられる。
   b. (MF)比較演算子で、キーをデータ名-1より小さいか、以下と指定すると、ファイル中に現存する論理レコードのうちで、その比較条件を満足する 最後のものにファイル位置指示子が位置付けられる。
   c. 比較条件を満たすレコードがファイル中に存在しないと、 無効キー条件が発生する。この場合、 START文の実行は不成功に終わり、ファイル位置指示子の内容はどうなるかわからない。(前述の*無効キー条件*の節を参照。)
6. 上記の5項の比較では、ファイル名-1のRELATIVE KEY句に指定したデータ項目が使用される。したがって、ファイル名-1にはRELATIVE KEY句を指定しておく。

書き方 2 (索引ファイル)

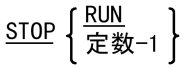
7. KEY指定中の比較演算子によって表わされる比較は、ファイル名-1のファイル中のレコードのキー（下記の8項を参照）とデータ名-1との間で行われる。ファイル名-1が索引ファイルであり、作用対象の大きさが等しくないと、短い方を超える長い方の右側の部分が切捨てられたように比較が進められる。文字の比較に関するその他の規則が、すべてこの比較に適用される。ただし、 PROGRAM COLLATING SEQUENCE句は、この比較には効力をもたない。(前述の*文字作用対象の比較*の節を参照。)
    a. (ANS85)比較演算子で、キーをデータ名-1と等しいか、より大きいか、以上と指定すると、
       ファイル中に現存する論理レコードのうちで、その比較条件を満足する最初のものに ファイル位置指示子が位置付けられる。
    b. (MF)比較演算子で、キーをデータ名-1より小さいか、以下と指定すると、ファイル中に現存する論理レコードのうちで、その比較条件を満足する最後のものに ファイル位置指示子が位置付けられる。 キーの値が重複するものが存在すると、そのうちの最後のものにファイル位置指示子が位置付けられる。
    c. 比較条件を満たすレコードがファイル中に存在しないと、 無効キー条件が発生する。この場合、START文の実行は不成功に終わり、ファイル位置指示子の内容はどうなるかわからない。（前述の*無効キー条件*の節を参照。）
8. 上記の7項の比較においてKEY指定をした場合は、データ名のデータ項目（キー）が使用される。
9. 上記の7項の比較において KEY指定をしなかった場合は、ファイル名-1のRECORD KEY句に指定したデータ項目（キー）が使用される。
10. START文の実行が正常に終了すると、下記のように 参照キーが設定され、後続の書き方3のREAD文で使用できるようになる。 (前述の*READ*文の節を参照。)
    a. KEY指定をしなかった場合は、ファイル名-1の主レコードキーが参照キーとなる。
    b. KEY指定をし、ファイル名-1のレコードキーとしてデータ名-1
       (MF)または分割キー名-1
       を指定した場合は、そのレコードキーが参照キーとなる。
    c. KEY指定をし、ファイル名-1のレコードキーとしてデータ名-1
       (MF)または分割キー名-1
       を指定しなかった場合は、データ名-1
       (MF)または分割キー名-1
       と左端の文字位置が同じであるレコードキーが参照キーとなる。
11. START文の実行が不成功に終わったときは、参照キーはどうなっているかわからない。

16.1.6 STOP（停止）文

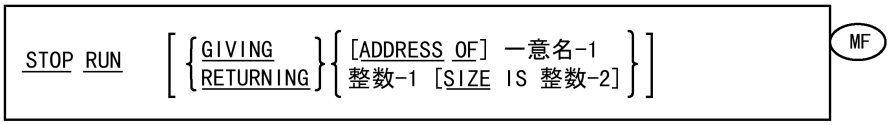
STOP（停止）文は、実行単位の実行を、永久的にまたは一時的に停止させる。
(ANS85)「STOP定数」文は、ANSI'85標準では廃要素に分類されており、ANSI標準の次回の全面改訂の際に、削除される予定である。
(MF)この構文は、Micro Focusに組み込まれているすべての方言で全面的に使用できる。FLAGSTD指令を使用すると、この構文が使われているすべての箇所を見つけ出すことができる。
(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、「STOP定数」の書き方は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内ではこの文を使用するべきではない。

一般形式

書き方 1



書き方 2



構文規則

1. 定数-1は語ALLで始まる表意定数であってはならない。
2. 定数-1が数字定数の場合、符号なしの整数とする。
   (OSVS)(VSC2)符号付きの整数を使用できる。
3. (MF)整数-1は 符号を付けてもよい。
4. 完結文の中の一連の無条件文の中にSTOP RUN文を書く場合、その中の最後の文にする。
   (OSVS)(VSC2)(MF)必ずしも、一連の無条件文の最後にSTOP RUN文を書かなくてもよい。しかし、一連の文の中でSTOP RUN文よりも後ろにあるものは実行されない。
5. (MF)GIVING と RETURNING は同義語であり、どちらを書いてもよい。
6. (MF)一意名-1は8バイト以下のサイズとする。

一般規則

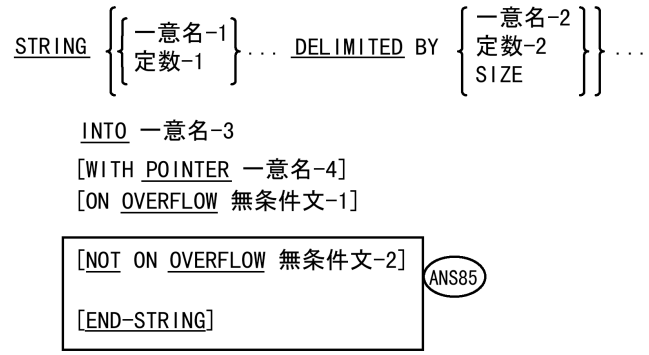
1. RUN指定を書くと、実行単位の実行が停止され、制御はオペレーティングシステムに移される。

2. STOP RUN文を実行中に、実行単位内の開かれている各ファイルに関して、選択指定の何もないCLOSE文が暗黙的に実行される。それらのファイルに関連するUSE手続きがあっても、実行されない。
3. STOP RUN文を実行したときに、実行単位がメッセージにアクセスしている最中であり、部分的に受信済みのメッセージがあると、メッセージ制御システム（MCS）はそれを待ち行列から削除する。SEND文を通じて実行単位から送信されたメッセージのうちで、EMIまたはEGIによって終了されていない部分があると、すべてシステムから除去される。
4. (MF)STOP RUN文を実行すると、戻り値がシステム領域に設定される。その領域は一般に、COBOL以外のランタイム要素が値を戻すために使用できる。実行されたプログラムからオペレーティング・システム環境へ値を戻す機能がサポートされている場合には、システム領域から戻り値が返される。
   (MF)GIVING指定を書かないと、実行単位の動作は下記のようになる。具体的には、システム領域がCOBOLの数字データ項目であり、それにUSAGE COPM-5が指定されていて、そのサイズはCOBOLシステムの外の操作環境によって決められているものとして宣言されている状態で、RETURN-CODEを送出し側としシステム領域を受取り側として、MOVE文を実行したかのように動作する。（RETURN-CODE詳細については、「COBOL言語の概念」の章の「特殊レジスタ」の項を参照。）
   (MF)「GIVING 一意名-1」指定を書いた場合、一意名-1はシステム領域に戻り値を収めるのにちょうど必要な長さであり、オペレーティング・システムによって期待される型と用途のものでなければならない。通常、一意名-1はPIC S9 (9) USAGE COMP-5を明示的または暗黙的に指定して宣言する必要がある。一意名-1を送出し側項目とし、システム領域を受取り側項目として、MOVE文を実行したかのように、実行単位は動作する。
   (MF)「GIVING 整数-1」指定を指定した場合、整数-1の値はシステム領域に保持できるよりも大きくてはならない。実行単位は、整数-1を送出し側としそのシステム領域を受取り側として、MOVE文を実行したかのように動作する。
5. STOP定数-1」を指定すると、操作員に定数-1が通知される。実行用プログラムの再開が行われると、次の実行文から処理が継続される。実行キーまたはそれと機能的に等しいキーを押すと、実行が再開される。

16.1.7 STRING（連結）文

STRING（連結）文は、いくつかのデータ項目の内容の一部または全部をつなぎ合わせて、1つのデータ項目を作る。

一般形式



構文規則

1. 各定数には、任意の表意定数を使用できる。ただし、ALLを除く。
2. 定数は、すべて文字定数とする。一意名-4を除く一意名はすべて、明示的または暗黙的にUSAGE IS DISPLAYと記述されていなければならない。
3. 一意名-3は、基本英数字データ項目とする。編集記号またはJUSTIFIED句を伴っていてはならない。
4. 一意名-4は、整数の基本データ項目とする。その大きさは、一意名-3によって参照される領域の大きさに、1を加えた値を収められるほどでなければならない。一意名-4のPICTURE文字列には、"P" を含めることはできない。
5. 一意名-1または一意名-2が基本数字データ項目である場合、PICTURE文字列には、"P" が含まれない整数として記述されていなければならない。
6. (ANS85)一意名-3を、部分参照してはならない。
7. (MF)一意名-3を、部分参照できる。

一般規則

1. 一意名-1と定数-1は、送出し側項目を表わす。一意名-3は、受取り側項目を表わす。
2. 一意名-2と定数-2は、転記の区切り文字を示す。SIZE指定を書くと、一意名-1または定数-1に指定したデータ項目の内容のすべてが転記される。区切り文字として表意定数を指定すると、それは1文字として扱われる。
3. 定数-1または定数-2として表意定数を指定すると、暗黙的に用途がDISPLAYである1文字として扱われる。
4. STRING文が実行されると、下記の規則に従ってデータが転記される。
    a. 英数字間の転記規則に従って、定数-1または一意名-1のデータ項目の内容が一意名-3に転記される。ただし、空白文字の穴埋めは行われない。(前述の*MOVE*文の節を参照。)

b. SIZE指定を書かずにDELIMITED指定を書くと、一意名-1によって参照されるデータ項目の内容または定数-1の値が受取り側データ項目に移送される。その順序はSTRING文に指定されたとおりであり、左端の文字から始まって左から右へ順に進み、データ項目の末尾に達するか、定数-2に指定された文字が出てくるか、一意名-2の内容によって指定された文字が出て来た時点で終了する。定数-2に指定された文字または一意名-2によって参照されたデータ項目に指定された文字は移送されない。

c. DELIMITED BY指定にSIZEと書くと、定数-1の値または一意名-1のデータ項目の内容の全体が、一意名-3のデータ項目に転記される。送出し側のすべてのデータが転記されるか、または一意名-3のデータ項目の末尾に達した時点で、この転記処理は終了する。

5. POINTER指定を書くと、プログラマが一意名-4を明示的に利用できるようになる。この初期値を設定するのは、プログラマの役割である。一意名-4の初期値は、1以上とする。
6. POINTER指定を書かないと、暗黙的に一意名-4を指定し、その初期値を1としたものとして、以下の一般規則が適用される。
7. 送出し側項目から受取り側項目に文字列が転記される処理は、文字が1つずつ取り出されて、一意名-3のデータ項目の内の一意名-4の値で示される文字位置に送られるように処理される。次の文字が送られる前に、一意名-4の値が1繰り上げられる。一意名-4の値は、STRING文の上記の処理以外では変えられることはない。
8. STRING文の実行が終了した時点で、一意名-3の内容が変わっているのは、直接的に転記の対象となった部分だけである。一意名-3のそれ以外の部分には、STRING文を実行する前のデータ内容が残っている。
9. 各文字を転記する前の時点で、一意名-4のデータ項目の値が1より小さいか一意名-3のデータ項目の文字数よりも大きいと、それ以上転記は行われない。この場合、
(ANS85)NOT ON OVERFLOW指定はあっても無視されて、
STRING文の末尾に制御が移される。ただし、ON OVERFLOW指定があれば、無条件文-1に制御が移される。無条件文-1に制御が移された場合は、その中に記述されている文に関する規則に従って、プログラムの実行が進められる。手続き分岐文あるいは明示的に制御を移行させる条件文が実行されると、その文に関する規則に従って、制御が移される。そうでなければ、無条件文-1の実行が終わると、STRING文の末尾に制御が移される。
10. (ANS85)STRING文の一般規則に従ってデータの転記が行われた後で、上記の一般規則9に示した状況が発生していなければ、 ON OVERFLOW指定はあっても無視されて、STRING文の末尾に制御が移される。ただし、NOT ON OVERFLOW指定があれば、無条件文-2に制御が移される。無条件文-2に制御が移された場合は、その中に記述されている文に関する規則に従って、プログラムの実行が進められる。手続き分岐文あるいは明示的に制御を移行される条件文が実行されると、その文に関する規則に従って、制御が移される。そうでなければ、無条件文-2の実行が終わると、STRING文の末尾に制御が移される。

11. (ANS85)END-STRING指定はSTRING文の範囲を区切る。(*COBOL言語の概念*の章の*明示範囲符と暗黙範囲符*の節を参照。)

16.1.8 SUBTRACT（減算）文

SUBTRACT（減算）文は、いくつかの数字データ項目から1つの数字データ項目または2つ以上の数字データ項目の和を差し引いて、その結果をいくつかのデータ項目に入れる。

一般形式

書き方 1

```
SUBTRACT { 一意名-1 }...FROM{一意名-2 [ROUNDED]}...
         { 定数-1   }

    [ON SIZE ERROR 無条件文-1]
    [NOT ON SIZE ERROR 無条件文-2]   (ANS85)
    [END-SUBTRACT]
```

書き方 2

```
SUBTRACT { 一意名-1 }...FROM { 一意名-2 }
         { 定数-1   }       { 定数-2   }

GIVING{一意名-3 [ROUNDED]}...
    [ON SIZE ERROR 無条件文-1]
    [NOT ON SIZE ERROR 無条件文-2]   (ANS85)
    [END-SUBTRACT]
```

書き方 3

```
SUBTRACT { CORRESPONDING } 一意名-1 FROM 一意名-2 [ROUNDED]
         { CORR          }

    [ON SIZE ERROR 無条件文-1]
    [NOT ON SIZE ERROR 無条件文-2]   (ANS85)
    [END-SUBTRACT]
```

構文規則

すべての書き方

1. 各一意名は、数字の基本項目とする。ただし、書き方2では、語GIVINGの後ろの一意名は、数字編集の基本項目であってもよい。書き方3の一意名は、集団項目でなければならない。
2. 作用対象を合成したものは、 18桁を超えてはならない。(前述の*算術文*の節を参照。)
    a. 書き方1では、指定したすべての作用対象が作用対象の合成に用いられる。
    b. 書き方2では、指定したすべての作用対象のうちの語GIVINGの後ろのデータ項目を除いたものが作用対象の合成に用いられる。
    c. 書き方3では、対応するデータ項目の組ごとに別々に作用対象の合成が行われる。

書き方 1 および 2

3. 各定数は、数字定数とする。
4. (OSVS)(VSC2)(MF)数字データ項目および定数を使えるところでは、どこでも浮動小数点数のデータ項目および定数を使用できる。

書き方 3

5. CORRは、CORRESPONDINGの省略形である。

一般規則

すべての書き方

1. 前述の*ROUNDED指定*、*ON SIZE ERROR指定*、*算術文*、*作用対象の重なり*、*算術文における複数個の答*の各節を参照。
2. (VSC2)(MF)計算過程で意味のある桁が失われることのないように、COBOLコンパイラによって十分な作業領域が確保される。

書き方 1

3. 語FROMの前にあるすべての定数または一意名の和が取られ、一意名-2の現在の値からその和が差し引かれ、その結果が直ちに一意名-2に入れられる。語FROMの後ろに続く各作用対象に対して、それぞれ上記の処理が繰り返される。

書き方 2

4. 書き方2では、語FROMの前にあるすべての定数または一意名の和が取られ、定数-2または一意名-2の現在の値からその和が差し引かれ、その結果が一意名-3の各データ項目の新しい値として入れられる。

書き方 3

5. 書き方3では、一意名-1に属するデータ項目が一意名-2に属する対応するデータ項目から引かれて、その結果が一意名-2に属する対応するデータ項目に入れられる。

16.1.9 TRANSFORM（変形）文

(OSVS)

TRANSFORM（変形）文は、変形規則に従って文字を変更する。

一般形式

```
TRANSFORM 一意名-3 CHARACTERS
          FROM { 表意定数-1 } TO { 表意定数-2 }
               { 文字定数-1 }    { 文字定数-2 }
               { 一意名-1   }    { 一意名-2   }
```

構文規則

1. 一意名-3は、基本項目でも集団項目でもよい。ただし、用途がDISPLAY以外の数字基本項目を除く。
2. 一意名-1と一意名-2は、英字または英数字の基本項目にする。
3. 文字定数-2または一意名-2の長さは、1文字であるか、または文字定数-1または一意名-1と同じにする。

一般規則

1. 表意定数-1または表意定数-2を使用することは、同じ値の1文字の文字定数を使用することと等しい。
2. 一意名-1または一意名-2が、一意名-3と同じ記憶領域を指していると、TRANSFORM処理の結果はどうなるかわからない。(前述の*REDEFINES句*の節を参照。)
3. 文字定数-1または一意名-1の中で文字が繰り返されると、TRANSFORM処理の結果はどうなるかわからない。
4. TRANSFORM文が実行されると、一意名-3の内容が調べられて、一意名-1または文字定数-1に指定した文字が探される。該当する文字が見つかると、一意名-2または文字定数-2の中の対応する文字（または1文字からなる項目の場合はその文字）によって、一意名-3のその部分が置き換えられる。一意名-1または文字定数-1と、一意名-2または文字定数-2との対応関係は、データ項目内の文字位置によって付けられる(左から始まる)。

16.1.10 UNLOCK（開錠）文

(MF)

UNLOCK（開錠）文は、 実行単位によってロック（施錠）されている、指定されたファイル中のすべてのレコードのロックを解除（開錠）する。

一般形式

```
UNLOCK ファイル名 { RECORD  }
                  { RECORDS }
```

構文規則

ファイル名は、ファイル管理記述項のSELECT文中に定義しておく。

一般規則

1. ファイル名のファイルは、OPEN文によって既に開かれていること。
2. UNLOCK文は、実行単位によってロックされている、指定されたファイル中のすべてのレコードのロックを解除する。

16.1.11 UNSTRING（分解）文

UNSTRING（分解）文は、送出し側項目の連続するデータを分解していくつかの受取り側項目に入れる。

一般形式

```
UNSTRING 一意名-1
    [DELIMITED BY [ALL] { 一意名-2 }[ OR [ALL] { 一意名-3 }]...]
                        { 定数-1   }          { 定数-2   }
        INTO {一意名-4 [DELIMITER IN 一意名-5][COUNT IN 一意名-6]}...
        [WITH POINTER 一意名-7]
        [TALLYING IN 一意名-8]
        [ON OVERFLOW 無条件文-1]
        [NOT ON OVERFLOW 無条件文-2]
        [END-UNSTRING]                                    (ANS85)
```

構文規則

1. 各定数は文字定数とする。各定数は表意定数であってもよい。ただし、ALLは除く。

2. 一意名-1、一意名-2、一意名-3、一意名-5は、明示的または暗黙的に、項類が英数字のデータ項目として記述されていなければならない。
3. 一意名-4は、英字でも英数字でも数字でもよい。ただし、英字の場合、PICTURE文字列の中に記号文字"B" を含めることはできない。数字の場合は、PICTURE文字列の中に記号文字"P" を含めることはできない。一意名-4の用途は、DISPLAYとする。
4. (MF)構文規則2と3は適用しない。代わって、下記の規則を適用する。
    - 一意名-1は、英数字とする。
    - 一意名-2と一意名-3は、用途をDISPLAYとし、編集されていてはならない。
    - 一意名-5の用途は、DISPLAYとする。
    - 一意名-4の用途は、DISPLAYにしてもよい。
    - 一意名-4は、数字として定義されており正しく転記できるならば、どのような用途のものであってもよい。
5. (OSVS)(VSC2)(MF)一意名-4は、浮動小数点数項目であってはならない。
6. 一意名-6と一意名-8は、整数項目とする。このPICTURE文字列の中に、記号文字" P" を含めることはできない。
7. 一意名-7は整数の基本項目とし、一意名-1のデータ項目の文字数に1を加えた値をもつことができなければならない。このPICTURE文字列の中に、記号文字"P" を含めることはできない。
8. どの一意名も、88レベルの記述項を指すことはできない。
9. DELIMITER IN指定およびCOUNT IN指定は、DELIMITED BY指定を書いたときだけ指定できる。
10. (MF)UNSTRING処理のソースは修正された参照であってもよい。

一般規則

1. 一意名-2と定数-1に関する記述はすべて、それぞれ一意名-3と定数-2およびそれらを反復指定したものに、等しく当てはまる。
2. 一意名-1は、送出し側を表わす。
3. 一意名-4は、受取り側を表わす。一意名-5は、区切り文字用の受取り側を表わす。
4. 定数-1または一意名-2のデータ項目は区切り文字を表わす。
5. 一意名-6のデータ項目は、一意名-1のデータ項目のうちで、区切り文字によって分離されて一意名-4のデータ項目に転記された文字数を表わす。区切り文字は、この文字数に入らない。
6. 一意名-7のデータ項目は、一意名-1のデータ項目中の相対文字位置を表わす。
7. 一意名-8のデータ項目は、UNSTRING文の実行中に処理の対象となった受取り側データ項目の数を記録するカウンタである。
8. 区切り文字に表意定数を使用した場合、1文字の文字定数として扱われる。
   ALL指定を書くと、定数-1（文字定数または表意定数）または一意名-2のデータ項目の内容が一意名-1の文字列中に連続して何回か現れても、それが1つのものとして扱われる。そして、一般規則13dの規則に従って、その内容が受取り側に転記される。
9. 検査中に区切り文字が2つ連続して検出されると、このときの受取り側の項目はそのデータ記述に従って、英字または英数字ならば空白で、数字ならばゼロで満たされる。

10. 定数-1または一意名-2のデータ項目には、計算機文字集合中の任意の文字を含めることができる。
11. 各定数-1または一意名-2のデータ項目は、1つの区切り文字を表わす。1つの区切り文字が何文字かで構成されている場合、送出し側の文字列の中で区切り文字として認識されるためには、まったく同じに並んでいなければならない。
12. DELIMITED BY指定に複数の区切り文字を書くと、各区切り文字は論理和条件"OR"で結ばれる。各区切り文字が、送出し側項目中の文字列と比較される。両者が一致すると、送出し側項目中の文字列のその部分が、1つの区切り文字とみなされる。送出し側項目中のどの文字も、2つ以上の区切り文字の一部とみなされることはない。
    各分離符はUNSTRING文に指定された順序で送出し側フィールドに適用される。
13. UNSTRING文の実行が開始されると、下記の規則に従って、送出し側の一意名-1のデータ項目から受取り側の一意名-4のデータ項目に、データが転記される。
    a. POINTER指定を書くと、一意名-7のデータ項目の内容によって示される相対文字位置から、一意名-1のデータ項目の文字列の検査が開始される。POINTER指定を書かないと、一意名-1の左端の文字位置から、データ項目の文字列の検査が開始される。
    b. DELIMITED BY指定を書くと、送出し側の文字列の検査は左から右に進められて、定数-1または一意名-2のデータ項目によって示される区切り文字が出てきたところで終了する（一般規則11を参照）。DELIMITED BY指定を書かないと、検査される文字数はそのときの受取り側の大きさと等しくなる。しかし、受取り側の項目の符号が独立であると定義されている場合は、検査される文字数はそのときの受取り側の大きさよりも1つ小さくなる。
    区切り文字が出てくる前に一意名-1のデータ項目の末尾に達すると、最後の文字が検査された時点で検査は終了する。
    c. 検査の結果、対象と判定された文字列（区切り文字が指定されている場合それは含まれない）が、MOVE文の規則に従って、英数字の基本項目として、このときの受取り側に転記される。(前述の*MOVE*文の節を参照。)
    d. DELIMITER IN指定を書くと、区切り文字（列）が英数字の基本データ項目として扱われて、MOVE文の規則に従って一意名-5のデータ項目に転記される。(前述の*MOVE*文の節を参照。）ただし、送出し側の文字列の検査が一意名-1のデータ項目の末尾に達した場合は、一意名-5のデータ項目は空白で埋められる。
    e. COUNT IN指定を書くと、検査の対象となった文字の数（区切り文字が指定されている場合、それは含まれない）が、基本項目転記の規則に従って、一意名-6のデータ項目に転記される。
    f. DELIMITED BY指定を書くと、文字列の検査は区切り文字のすぐ右側の文字から再開される。DELIMITED BY指定を書かないと、文字列の検査は転記の済んだ部分のすぐ右側の文字から再開される。

g. 一意名-4のデータ項目にデータが転記されると、一意名-4の次の繰り返しデータ項目が新たな受取り側となる。そして、上記の13bから13fの処理が繰り返される。一意名-1のデータ項目中の文字がすべて検査され終わるか、または受取り側の項目がなくなった時点で、この繰り返し処理は終了する。

14. POINTER指定または TALLYING指定に関連するデータ項目の内容を初期化しておくことは、プログラマの役割である。
15. 一意名-7のデータ項目の内容は、一意名-1のデータ項目中で各文字が検査されるごとに、1繰り上げられる。POINTER指定のあるUNSTRING文の実行が終了した時点では、一意名-7のデータ項目の値は、その初期値に一意名-1のデータ項目の検査の対象となった文字数を加えたものになっている。
16. TALLYING指定のあるUNSTRING文の実行が終了した時点では、一意名-8のデータ項目の値は、その初期値に処理の対象となった受取り側項目の数を加えたものになっている。
17. 下記のどちらかの場合には、あふれ条件が発生する。
    a. UNSTRING文の実行開始時に、一意名-7のデータ項目の値が1より小さいか、または一意名-1のデータ項目の大きさよりも大きい場合
    b. UNSTRING文の実行中に、受取り側がすべて使われたのに、一意名-1のデータ項目中にまだ検査されていない文字が残っている場合
18. あふれ条件が発生したときのUNSTRING文の終了処理は、次のようになる。
    (ANS85)NOT ON OVERFLOW指定がある場合は、その指定は無視されて、UNSTRING文の末尾に制御が移される。ON OVERFLOW指定がある場合は、無条件文-1に制御が移される。無条件文-1に制御が移された場合は、その中に記述されている文に関する規則に従って、プログラムの実行が進められる。手続き分岐文あるいは明示的に制御を移行させる条件文が実行されると、その文に関する規則に従って、制御が移される。そうでなければ、無条件文-1の実行が終わると、UNSTRING文の末尾に制御が移される。
19. (ANS85)END-UNSTRING指定は、UNSTRING文の範囲を区切る。(*COBOL言語の概念*の章の*明示範囲符と暗黙範囲符*の節を参照。)
20. UNSTRING文の実行中に一般規則17に記述した条件が発生しなかった場合、他の一般規則に従ってデータの転記が済んだ後、UNSTRING文の末尾に制御が移される。 ON OVERFLOW指定は、あっても無視される。
    (ANS85)NOT ON OVERFLOW指定がある場合は、無条件文-2に制御が移される。無条件文-2に制御が移された場合は、その中に記述されている文に関する規則に従って、プログラムの実行が進められる。手続き分岐文あるいは明示的に制御を移行させる条件文が実行されると、その文に関する規則に従って、制御が移される。そうでなければ、無条件文-2の実行が終わると、UNSTRINGの末尾に制御が移される。
21. 一意名に付けられている添字および指標は下記のように評価される。
    a. 一意名-1と一意名-7と一意名-8に関する添字または指標があれば、UNSTRING文を実行した結果としてデータの転記が行われる直前に、1回だけ評価される。

b. 一意名-2、一意名-3、一意名-4、一意名-5、一意名-6に関する添字または指標があれば、それぞれのデータ項目にデータの転記が行われる直前に評価される。

22. (ANS85)DELIMITED BY, INTO, DELIMITER IN, COUNT INの各指定に関連する一意名に添字付けがされている場合、区切り文字を探して送出し側項目を検査する直前に1回だけ添字が評価される。

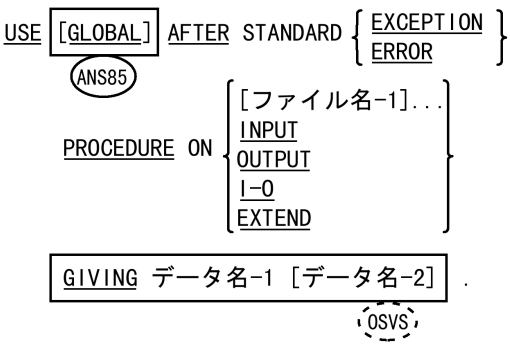
23. 一意名-1、一意名-2、一意名-3のどれかが、一意名-4、一意名-5、一意名-6、一意名-7、一意名-8のどれかと同じ記憶領域を共有する場合、または一意名-4、一意名-5、一意名-6のどれかが、一意名-7、一意名-8のどれかと同じ記憶領域を共有する場合、または一意名-7と一意名-8が同じ記憶領域を共有する場合、UNSTRING文の実行結果はどうなるかわからない。これらのデータ項目が同じデータ記述に項定義されていたとしても、そのことは変わらない。

16.1.12 USE（使用）文

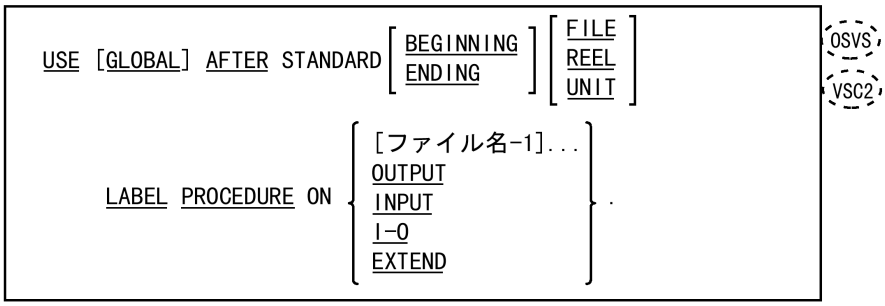
USE（使用）文は、入出力誤りを処理するための手続きを指定する。この手続きは、入出力管理システムによって行われる標準の手続きに、更に追加して行うものである。

一般形式

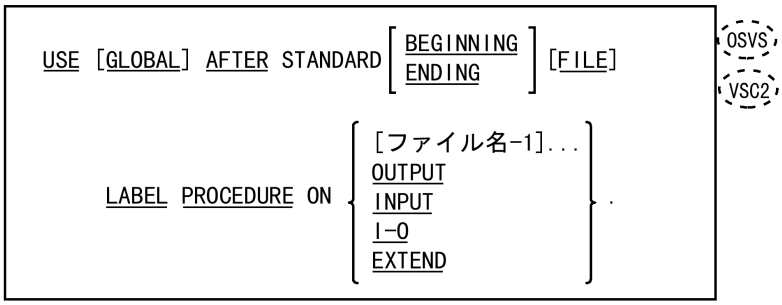
書き方 1 (順ファイル、相対ファイル、索引ファイル)



書き方 2 (レコード順ファイル)



書き方 3 (相対ファイルおよび索引ファイル)



構文規則

すべての書き方 (すべてのファイル)

1. 書き方1は、 誤りに関する宣言である。
   (OSVS)(VSC2)書き方2と3は、 ラベルに関する宣言である。
2. USE文を記述する場合は、宣言節の見出しの直後から書き始め、終止符 (.) に続く空白で止める。
3. USE文自体は、実行されることはない。USE文は、ただ USE手続きの実行を求める条件を定義するだけである。
4. USE文の中で明示的または暗黙的に参照されるファイルは、すべて同じ編成または呼出し法のものである必要はない。

書き方 1 (順ファイル、相対ファイル、索引ファイル)

5. この書き方に従った別々の形の記述中に、同じファイル名が現れてもよい。USE文中に指定したファイル名によって、複数のUSE手続きの実行が同時に要求されるようなことがあってはならない。
   (ANS85)同じ手続き部内の複数のUSE AFTER EXCEPTION手続き中に、同じファイル名が現れてはならない。

6. ERRORとEXCEPTIONは同義語であり、どちらを使用してもよい。

書き方2および3（レコード順ファイル、相対ファイル、索引ファイル）

7. (OSVS)(VSC2)BEGINNINGとENDINGの指定を両方とも省略すると、BEGINNINGとENDINGが共に指定されたものとみなされる。

書き方 2（レコード順ファイル）

8. (OSVS)(VSC2)REELとUNITは同義語として扱われる。
9. (OSVS)(VSC2)FILEとREEL/UNITの指定を両方とも省略すると、FILEとREEL/UNITが共に指定されたものとみなされる。
10. (OSVS)(VSC2)BEGINNING/ENDINGとFILE/REELの下記の組合わせのそれぞれに対して、任意の1つのファイル名および任意の1つのOPENモードを指定できる。
    BEGINNING FILE
    BEGINNING REEL/UNIT
    ENDING FILE
    ENDING REEL/UNIT

一般規則

すべての書き方（すべてのファイル）

1. USE手続きの実行が終わると、呼出し元の手順に制御が戻される。
2. USE手続き内では、非宣言手続きを参照してはならない。逆に、非宣言部分では、宣言部分中の手続き名を参照してはならない。ただし、PERFORM文で USE文を参照することはできる。
   (OSVS)(VSC2)(MF)この制限は無視してもよい。
3. 呼び出されたがまだ呼び出し元の手順に制御が戻されていないUSE手続きを実行させるような文を、実行してはならない。

書き方 1（順ファイル、相対ファイル、索引ファイル）

4. 指定した手続きは、標準入出力誤り手順が終了した後で、または入出力文中にAT END指定をしなかった場合はAT END条件が検出された時点で、入出力システムによって実行される。
5. (ANS85)ファイル名-1を明示的に指定した場合は、他のUSE手続きはファイル名-1に適用されない。
6. GIVINGは注記の目的だけに使用する。

書き方 2 および 3（レコード順ファイル、相対ファイル、索引ファイル）

7. (OSVS)(VSC2)BEGINNING指定を明示的または暗黙的に指定すると、適用可能なOPEN文の実行中に、下記の措置が取られる。

OPEN文のモード	措置
INPUT	• 見出しラベルを読む • 開始宣言を実行する
OUTPUT	• 開始宣言を実行する • 見出しラベルを書く
I/O	• 見出しラベルを読む • 開始宣言を実行する • 見出しラベルを書く
EXTEND	• 見出しラベルを読む • 開始宣言を実行する（終わりファイル・ラベルは見出しとして扱われる） • 見出しラベルを書く

8. (OSVS)(VSC2)ENDING指定を明示的または暗黙的に指定すると、適用可能なCLOSE文の実行中に、下記の措置が取られる。

OPEN文のモード	措置
INPUT	• 終わりファイル・ラベルを読む • 終了宣言を実行する
OUTPUT	• 終了宣言を実行する • 終わりファイル・ラベルを書く
I/O	• 終わりファイル・ラベルを読む • 終了宣言を実行する • 終わりファイル・ラベルを書く
EXTEND	• 終了宣言を実行する • 終わりファイル・ラベルを書く

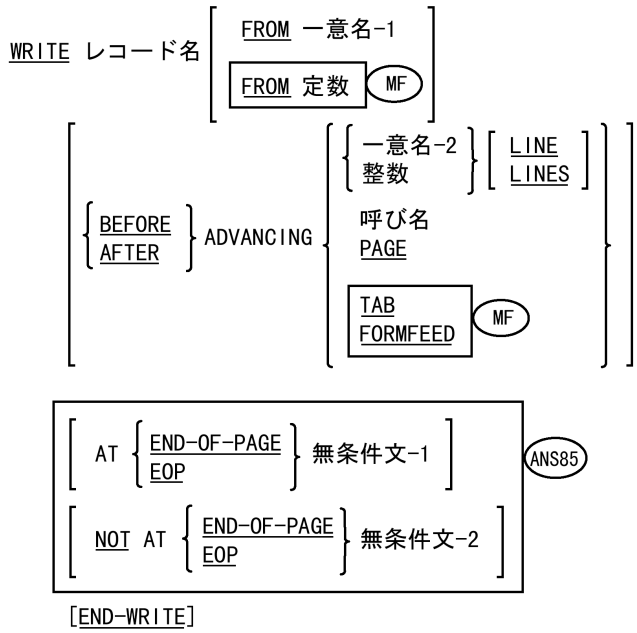
9. (OSVS)(VSC2)GO TO MORE-LABELS文はそれが現れる宣言手続きの先頭への単純なジャンプとして扱われる。

16.1.13 WRITE（書き出し）文

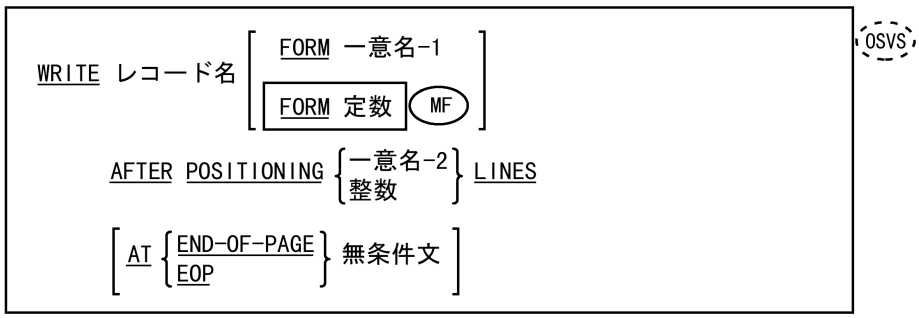
WRITE（書き出し）文は、論理レコードを出力ファイルまたは入出力両用ファイルに書き出す。順ファイルに関しては、論理ページ内での縦方向の行の位置付けにもWRITE文を使用できる。(XOPEN)標準COBOL定義の一部を構成するにもかかわらず、X/OpenのCOBOL言語定義では、 ADVANCING指定中の呼び名は明示的に除外されている。したがって、X/OpenのCOBOLに準拠する原始プログラム内では呼び名を使用するべきではない。

一般形式

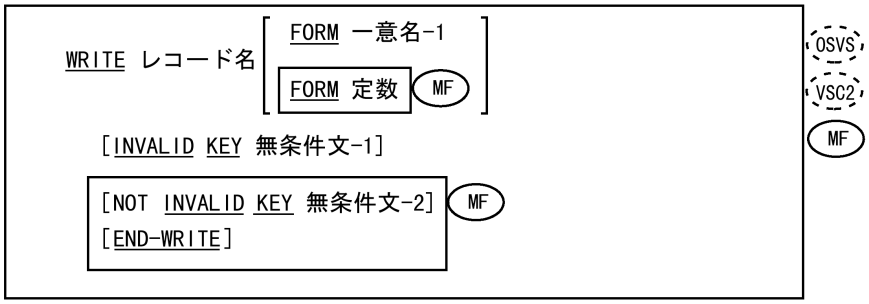
書き方 1（レコード順ファイル(MF)および行順ファイル）



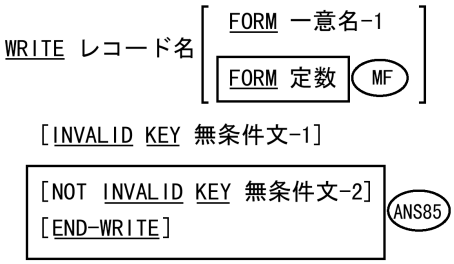
書き方 2（レコード順ファイル）



書き方 3 (レコード順ファイル)



書き方 4 (相対ファイルおよび索引ファイル)



指令とランタイム・スイッチ

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - WRITE-LOCK - 複数のレコード・ロックを使用しているときに、WRITE文にレコード・ロックを取得させる
    - FDCLEAR - WRITE文の後で、レコード領域の内容を予知できるようにする。
2. 下記のランタイム・スイッチによって、この項に記述した意味が影響を受ける可能性がある。
    - N - 行順レコードを書くときに、制御文字の前への空文字の挿入を制御する。
    - T - 行順レコードを書くときに、タブ文字の挿入を制御する。

構文規則

すべての書き方 (すべてのファイル)

1. (ANS85)一意名-1が関数一意名である場合は、英数字関数を参照すること。一意名-1が関数一意名でない場合は、レコード名と一意名-1が同じ記憶領域を指してはならない。
2. レコード名は、データ部のファイル節内の論理レコードの名前であり、修飾してもよい。
3. (OSVS)(VSC2)(MF)レコード名に、浮動小数点数項目を指定してもよい。
4. (OSVS)(VSC2)(MF)一意名-1は、浮動小数点数項目として定義してもよい。

書き方 1（レコード順ファイル）

5. (MF)TABを表わす呼び名を指定すると、縦の標準タブ位置まで用紙が飛ばされる。特殊名段落に記述すると、TABの代わりに、利用者が定義した呼び名を使用できる。(前述の*特殊名段落*の節を参照。)
6. ADVANCING指定中に一意名-2を使用する場合、一意名-2は基本整数データ項目の名前とする。
7. 整数または一意名-2のデータ項目の値は、ゼロであってもよい。
8. (VSC2)整数は符号付きでもよい。
9. END-OF-PAGE指定をした場合、該当するファイルのファイル記述項中にLINAGE句を指定する。
10. 語END-OF-PAGEと EOPは同義語である。
11. ファイル記述項に LINAGE句を指定したファイルにレコードを書き出すときは、ADVANCING呼び名指定をすることはできない。
12. (ANS85)1つのWRITE文中に、ADVANCING PAGEとEND-OF-PAGEを共に指定してはならない。
    (MF)この制限は無視してよい。
    (OSVS)(VSC2)(MF)呼び名の代わりに、それに相当する機能名を使用してもよい。

書き方 2（レコード順ファイル）

13. (OSVS)ファイル記述項にLINAGE句を指定したファイルにレコードを書き出すときは、この書き方は使用できない。
14. (OSVS)あるファイルにレコードを書き出すためにこの書き方の WRITE文を使用する場合、そのファイルに適用するWRITE文はすべてこの書き方にする。
15. (OSVS)AFTER POSITIONING指定に使用する一意名-2は、1文字の英数字項目として定義しておく。（指定できる値については、一般規則18を参照）

書き方 4（相対ファイルおよび索引ファイル）

16. 該当するファイルに適用できるUSE手続きを指定しなかった場合、INVALID KEY指定をする。
    (OSVS)(VSC2)(MF)この規則は強制しない。

一般規則

すべての書き方（すべてのファイル）

1. FROM指定をした WRITE文を実行することは、下記の順に処理を行うことと同じである。
    a. MOVE文の規則に従って、下記の文を実行する。
        MOVE 一意名-1 TO レコード名
    b. FROM指定をしない同じ WRITE文を実行する。

    暗黙のMOVE文を実行する前のレコード領域の内容は、このWRITE文の実行には影響を及ぼさない。

    WRITE文の実行が終了した後で、レコード領域のデータは利用できなくなることがあるが、一意名-1のデータはそのまま残っている。（一般規則13を参照）

2. ファイル位置指示子は、WRITE文の実行の影響を受けない。
3. WRITE文を実行すると、その対象のファイルに対応するFILE STATUSデータ項目を指定してあれば、それが更新される。(前述の*入出力状態*の節を参照。)
4. ファイル中のレコードの最大の大きさは、ファイルを作成したときに決定される。この大きさを後で変更してはならない。
5. ファイル中の論理レコードを格納する大記憶装置上の文字位置の数は、そのレコードの論理記述に定義した文字位置の数と、等しくてもよいし異なっていてもよい。 .
6. WRITE文が実行されると、論理レコードが書き出される。
7. (ANS85)WRITE文の実行が不成功に終わると、レコード名に対応するファイルの入出力状態が更新される。次いで、そのファイルに適用できるUSE AFTER EXCEPTION手続きが指定してあればその手続きが実行され、その後でUSE文の規則に従って制御が移される。(前述の*USE*文の節を参照。)

書き方 1 (レコード順ファイル)

8. END-OF-PAGE指定をしたWRITE文の実行中に印刷プログラムの論理的な末尾に達すると、END-OF-PAGE指定中の無条件文-1が実行される。 論理ページの大きさは、該当ファイルに関するファイル記述中の、該当レコードのLINAGE句に指定する。
9. LINAGE句を指定してあると、WRITE文を実行した結果LINAGE句の整数-2またはデータ名-2のデータ項目に定義した値以上にLINAGE-COUNTERの値が達すると、END-OF-PAGE指定をしたWRITE文が実行されたときにページ終了条件が発生する。 この場合、書き出し操作に続いて、そのWRITE文のEND-OF-PAGE指定中の無条件文が実行される。
   WRITE文（END-OF-PAGE指定があってもなくても）の実行結果が現在のページ本体に収まりきらないときは、自動的にページあふれ条件が発生する。
   ページあふれ条件とは、WRITE文を実行すると、LINAGE句の整数-2に指定した値またはデータ名-2に指定したデータ項目の値をLINAGE-COUNTERの値が超えることである。
   ページあふれが発生する場合、レコードの書き出しと改ページのどちらを先にするかは、BEFOREとAFTERのどちらを指定したかによる。改ページされると、装置はLINAGE句の指定に従って、次の論理ページの最初の行に位置付け直される。END-OF-PAGE句中に無条件文を指定してあると、レコードの書き出しと装置の位置付け直しが済んだ後で、その無条件文が実行される。
   LINAGE句に整数-2もデータ名-2にも指定していないと、ページあふれ条件とは別にページ終了条件が発生することはない。この場合、ページあふれ条件とページ終了条件は同時に発生する。
   LINAGE句に整数-2またはデータ名-2を指定してある場合で、WRITE文を実行したときに、LINAGE句の整数-2に指定した値またはデータ名-2に指定したデータ項目の値、およびLINAGE句の整数-1に指定した値またはデータ名-1に指定したデータ項目の値の両方を、LINAGE-COUNTERの値が同時に超えるときは、整数-2またはデータ名-2は指定されてないように処理は進められる。

書き方 1（行順ファイル）

10. (MF)ADVANCING指定をしないと、使用しているオペレーティングシステムのテキスト・エディタの方式に従って、自動的に1行が行送りされる（通常はBEFORE ADVANCING 1 LINEが想定される）。
11. (MF)順ファイルの外部的に定義されている境界を超えて書き出しを行おうとすると、例外条件が発生する。このとき、レコード領域の内容は影響を受けず、下記の例外処理が行われる。
    a. 該当するファイルに対応する FILE STATUSデータ項目を指定してあると、その値が区域外書き出しを表わすように設定される。(前述の*入出力状態*の節を参照。)
    b. 該当するファイルに適応できる USE AFTER STANDARD EXCEPTION宣言を明示的または暗黙的に指定してあると、その宣言手続きが実行される。
    c. 該当するファイルに適応できるUSE AFTER STANDARD EXCEPTION宣言を明示的にも暗黙的にも指定してないと、結果はどうなるかわからない。
12. (MF)複数の物理リール/ユニットにまたがる出力ファイルのリール/ユニットの終わりが検出されると、WRITE文は下記の操作を行う。
    a. 標準のリール/ユニット終了手続き
    b. リール/ユニットの交換手続き
    c. 標準のリール/ユニット開始手続き
13. (MF)(XOPEN)レコード長が固定のファイルに長さの異なるレコードを再定義して収めている場合、バッファ領域全体がファイルに書き込まれることを知っている必要がある。したがって、現在のレコードが前のレコードよりも短い場合には、空白を埋める必要がありうる。
14. (ANS85)NOT END-OF-PAGE指定をした WRITE文の実行が正常に進み、その間にページ終了条件が発生しないと、入出力処理の後で制御は無条件文-2に移される。

書き方 1（レコード順ファイル(MF)および行順ファイル）

15. ADVANCING指定および END-OF-PAGE指定は、共に印刷ページ上での各行の縦方向の位置を制御する働きをする。
    ADVANCING指定をしないと、リスト装置（PRINTERまたはPRINTER-1）に向けて出力されたときに、AFTER ADVANCING 1 LINEを指定してあったかのように、自動的に行送りされる。 ADVANCING指定をすると、下記のように行送りされる。
    a. 一意名-2を指定すると、その現在の値に等しい行数だけ、印刷ページ上で行送りされる。
    b. 整数を指定すると、その値に等しい行数だけ、印刷ページ上で行送りされる。
    c. 呼び名を指定すると、特殊名段落に指定した内容に従って、行送りされる。
    d. BEFOREを指定すると、該当する行が出力された後に、上記のa，b，cの規則に従って行送りされる。
    e. AFTERを指定すると、上記のa，b，cの規則に従って行送りされた後で、該当する行が出力される。

f. PAGEを指定すると、装置が次の論理ページに位置付け直される。該当する行の出力と改ページのファイル記述項中にLINAGE句を指定してあるレコード順ファイルにレコードを書き出すと、 LINAGE句の指定に従って、装置は次の論理ページ上に書き出し可能な最初の行に位置付け直される。

16. 1つのWRITE文中に、 ADVANCINGとEND-OF-PAGEを共に指定してはならない。

書き方 1、2、3 (順ファイル)

17. WRITE文を実行するときは、対象ファイルをOUTPUTモードまたはEXTENDモードで開いておく。(前述の*OPEN*文の節を参照。)

18. WRITE文の実行によって書き出された論理レコードは、レコード領域内で利用できなくなる。ただし、 SAME RECORD AREA句中に該当するファイルを指定してある場合、または区域外書き出しのためにWRITE文の実行が不成功に終わった場合は、そのレコード領域は元のまま残っている。.

SAME RECORD AREA句を指定してある場合、論理レコードは レコード名に対応するファイルのものとしてだけではなく、SAME RECORD AREA句中に指定した他のファイルのレコードとしても使用できる。

書き方 2 (レコード順ファイル)

19. (OSVS)WRITE文中に AFTER POSITIONING指定をすると、 システムはレコードの先頭の文字位置に適切な文字を転記してから、レコードをファイルに書き出す。このため、この先頭の文字位置は空けておくこと。AFTER POSITIONINGに一意名-2を指定をすると、その値がレコードの先頭の文字位置に転記される。その値は下記のどれかとする。

一意名-2	意味
(空白)	1行改行
0	2行改行
−	3行改行
+	改行なし
1-9	それぞれ、チャネル1-9へ飛ぶ
A, B, C	それぞれ、チャネル10, 11, 12へ飛ぶ
V, W	ポケット1または2を選択

AFTER POSITIONINGに整数を指定をすると、出力レコードの先頭に転記される文字は下記のようになる。

整数	出力文字	意味
0	1	チャネル1へ飛ぶ
1	(空白)	1行改行
2	0	2行改行
3	−	3行改行

20. (OSVS)END-OF-PAGE指定は、指定しても注記になり、実行されることはない。

書き方 3 (レコード順ファイル)

21. (OSVS)(VSC2)(MF)順ファイルの外部的に定義されている境界を超えて書き出しを行おうとすると、 無効キー条件が発生する。COBOLシステムによって無効キー条件が検出されると、 WRITE文の実行は不成功に終わる。このとき、レコード領域の内容は影響を受けない。該当するファイルに対応する FILE STATUSデータ項目を指定してあると、その原因を表わす値がその項目に設定される。実行は、 無効キー条件に関する規則に従って続行される。(前述の*入出力状態*の節を参照。)

書き方 4 (相対ファイルおよび索引ファイル)

22. WRITE文を実行するとき、対象ファイルをOUTPUT, I-O, EXTENDのどれかのモードで開いておく。索引ファイルを順呼出し法で処理する場合、 I-Oモードで開いてはならない。(前述の*ファイル管理記述項*および*OPEN文*の節を参照。)
23. WRITE文の実行によって書き出された論理レコードは、 SAME RECORD AREA句中に該当するファイルを指定してある場合、または無効キー条件のためにWRITE文の実行が不成功に終わった場合にだけ、引続きレコード領域内のものが利用できる。
    SAME RECORD AREA句を指定してある場合、論理レコードは、 レコード名に対応するファイルのものとしてだけではなく、 SAME RECORD AREA句中に指定した他のファイルのレコードとしても使用できる。
24. 無効キー条件が検出されると、WRITE文の実行は不成功に終わる。このとき、レコード領域の内容は影響を受けない。該当するファイルに対応するFILE STATUSデータ項目を指定してあると、その原因を表わす値がその項目に設定される。 実行は、 無効キー条件に関する規則に従って続行される。(前述の*入出力状態*の節を参照。)

書き方 4 (相対ファイル)

25. 出力モードで開いたファイルに関しては、下記のどちらかの方法でレコードを書き出すことができる。
    a. 順呼出し法を採る場合、 WRITE文によってレコードが書き出される。最初に書き出されるレコードの相対レコード番号は1番とされ、以降レコードが書き出される順に相対レコード番号が1つずつ繰り上げられる。該当するファイルのファイル管理記述項に RELATIVE KEYデータ項目を指定してあると、WRITE文の実行中に書き出されたレコードの相対レコード番号がオペレーティングシステムによってRELATIVE KEYデータ項目に収められる。
    b. 乱呼出し法または 動的呼出し法を採る場合、WRITE文を実行するのに先立って、プログラム中でRELATIVE KEYデータ項目に相対レコード番号を設定しておく。つまり、レコード領域中のレコードをRELATIVE KEYデータ項目に関係付けておく。次いで、WRITE文を実行することによって、そのレコードがファイルに書き出される。

26. 入出力両用モードで開き、乱呼出し法または動的呼出し法で処理するファイルに関しては、レコード領域中のレコードを書き出す先の相対レコード番号を実行時要素中でRELATIVE KEYデータ項目に設定しておく。次いで、WRITE文を実行すると、そのレコード領域の内容がファイルに書き出される。
27. 下記の場合に、 無効キー条件が発生する。
    a. 乱呼出し法または 動的呼出し法を採る場合、RELATIVE KEYデータ項目が指すレコードが既に存在する。
    b. 外部的に定義されているファイル境界を超えて、レコードの書き出しが試みられた。

書き方 4 (索引ファイル)

28. WRITE文を実行すると、 レコード領域の内容が書き出される。オペレーティングシステムは レコードキーの内容を使用して、以降どのレコードキーを通じてでもそのレコードを呼び出すことができるようにする。
29. 主レコードキーの値は、ファイル中の全レコードを通じて一意とする。
30. 主レコードキーに指定したデータ項目は、 WRITE文を実行するのに先立って、実行時要素中で希望する値に設定しておく。
31. 順呼出し法でファイルを処理する場合、主レコードキーの値の昇順に順次レコードを書き出すこと。
32. 乱呼出し法または 動的呼出し法を採る場合、任意の順序でレコードを書き出せる。
33. 索引ファイルのファイル管理記述項に ALTERNATE RECORD KEY句を指定した場合、併せてDUPLICATES指定をしてあれば、 副レコードキーの値は一意でなくてもよい。 副レコードキーの値が重複する場合、オペレーティングシステムは、それらが書き出された順に呼び出されるようにする。
34. 下記のどれかの場合に、 無効キー条件が発生する。
    a. 出力モードで開いたファイルに順呼出し法を適用しているとき、前に書き出したレコードの主レコードキーの値よりも、書き出そうとするレコードの主レコードキーの値が大きくない。
    b. 出力モードまたは 入出力両用モードで開いたファイルの場合、書き出そうとするレコードと主レコードキーの値が等しいレコードが、ファイル中に既に存在する。
    c. 出力モードまたは入出力両用モードで開いたファイルの場合、キーの重複を許していないのに、書き出そうとするレコードと副レコードキーの値が等しいレコードが、ファイル中に既に存在する。
    d. 外部的に定義されたファイル境界を超えて、レコードの書き出しが試みられた。
35. (ANS85)WRITE文の実行が正常にまたは不成功に終わった後、制御がどのように移されるかは、INVALID KEY指定とNOT INVALID指定の有無に左右される。(前述の*無効キー条件*の節を参照。)

第17章：オブジェクトCOBOL言語拡張

(MF)

本章ではオブジェクト指向をサポートするために追加されたCOBOLの文法を説明する。この文法はすべて、ANS X3.23 - 1985に対するMicro Focus COBOLに固有の拡張である。したがって、本章の周囲のボックスを省略した。オブジェクト指向機能の詳細については***オブジェクト指向プログラミング***を参照。

17.1 指令

予約語リストにフラグを付けて修正する機能を提供するコンパイラ指令に加えて、下記の指令が本節の構文または意味のいずれかに影響を与える可能性がある。

- ALIGN - 01レベルまたは77レベルのデータ項目を割り付けるメモリの境界を指定する。
- IBMCOMP - 語格納モードをオンにする。
- MAPNAME - メソッド名内のアルファベット以外の文字の取扱いに影響を与える。
- MF-OO - オブジェクト指向の構文を使えるようにする
- REPOSITORY - クラスに関するリポジトリ記述項を作成するか、無視するか、チェックするかをコンパイラに知らせる
- OOCTRL - オブジェクト指向オプションに関する特別なスイッチを提供する。設定をオンにするには各オプションを表す文字の前に + 記号を付け、オフにするには - 記号を付ける。省略時の解釈はOOCTRL(-G-N-P+Q-W)である。各オプションの意味は下記のとおり。

+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。

注意：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。

+N CLASS-OBJECTおよびEND CLASS-OBJECTの構文を使用できるようにする。

+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・インタフェー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・インタフェー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（「COBOLの概念」の章の「文の種類」の項を参照）。

-Q　メソッド・インタフェー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・インタフェース定義の動詞シグネチャ中で動詞も使えるようにする（「COBOLの概念」の章の「文の種類」の項を参照）。

+W　オブジェクトおよびクラス・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。

注意：ISO COBOL標準に基づく開発との互換性を持たせる。

17.2 クラス定義

クラスはオブジェクト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、DATA句もINHERITS句も使用できない。
3. 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。
4. クラス名-2はクラス名-1と同じであってはならない。
5. 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。
6. 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. SPRIVATE指定はクラス名-1から継承したデータにサブクラスが直接アクセスできないようにする。
6. INHERITS指定はクラス-2がクラス-1の親クラスであることを指定する。
7. INHERITS指定を指定しないと、クラス名-1は分類スキーマ内で新しいルートを形成する。
8. WITH DATA指定は、クラス名-2から継承したデータにクラス名-1が直接アクセスできることを指定する。

17.3 クラス拡張

クラス拡張を行うと、元のソース・コードを変更せずに、オブジェクトCOBOLに機能を追加できる。
継承を行うのではなく、クラス拡張によってクラスを拡張することの違いは、クラス拡張は既存のすべてサブクラスによって継承されることである。たとえば、クラスAのサブクラスにクラスBがあるとする（つまり、 B INHERITS FROM A）。クラスAのサブクラスとしてクラスCを生成することによって、クラスAにメソッドを追加できる。しかし、その場合はクラスBはクラスCのメソッドを継承しない。クラス拡張Xによってクラスを拡張した場合は、実行時の効果はクラスAを変更してコンパイルし直したのと同じである。クラスXによって追加されたすべてのメソッドをクラスBも継承する。

一般形式

[IDENTIFICATION DIVISION.]

CLASS-ID.拡張名-1 EXTEND クラス名-1 [WITH DATA].

クラス本体

オブジェクトプログラム

END CLASS 拡張名-1

構文規則

1. 拡張名-1はクラス名-1と同じでなければならない
2. クラス終了見出し中の拡張名-1は先行するクラス名段落中に指定されている拡張名-1と同じでなければならない。

3. クラス名-1はクラス管理段落に指定されているクラスの名前でなければならない。
4. クラス本体のデータ部に空のオブジェクト記憶節を含めてもよい。その他にクラス拡張のデータ部に指定してもよい節は、作業場所節と連絡節だけである。
5. 下記の両方に該当する場合にのみ、クラス拡張中の文からクラス名-1内に宣言されているデータを参照できる。
    - クラス名-1のクラス名段落中にDATA IS PROTECTED指定またはDATA IS RESTRICTED指定がある。
    - クラス拡張のクラス名段落中にWITH DATA指定がある。

一般規則

1. EXTEND句はクラス拡張を指定する。クラス拡張はオブジェクト・クラスにメソッドを追加する。拡張名-1に指定されたメソッドはクラス名-1の新旧のすべてのサブクラスによって継承される。
2. 実行単位の実行中に、クラス拡張中のメソッドを呼び出すのに先立って、拡張名-1に対してCOBOLのCALL文を実行しなければならない。それによって、クラス拡張中のメソッドがOOランタイム・システムに登録される。

17.4 クラス本体

クラス本体には、クラスのデータとメソッドを定義した、すべてのコードが含まれる。

一般形式

書き方1

（コンパイラ指令のOOCTRL(+N)を指定したときに使用される。）

注意：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]
```

書き方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

すべての書き方

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。

書き方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。

書き方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

すべての書き方

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。

書き方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。

書き方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。

   注意：作業場所節内のデータはクラスまたはインスタンスを初期化するためのデータとして役に立つ。

10. オブジェクト記憶節内で定義されたデータ項目は、クラス・メソッドの呼出しの間は、最後に使用されたときの状態が保たれる。

17.5 クラス・オブジェクト

クラス・オブジェクトはオブジェクトを生成する働きをするオブジェクトである。

一般形式

```
[IDENTIFICATION DIVISION.]
 CLASS-OBJECT.
[データ部]
[PROCEDURE DIVISION.]
[メソッド-1]...
 END CLASS-OBJECT.
```

一般規則

1. クラス・オブジェクトのオブジェクト記憶節内で宣言されたデータ項目はクラス・データである。クラス・データはサブクラスによって継承されうる。
2. データ項目は実行単位の開始時に初期化され、メソッド呼出しの間は最後に使用されたときの状態が保たれる。
3. メソッド-1はクラス・メソッドである。

17.6 オブジェクト・プログラム

オブジェクト・プログラムにはクラスのすべてのインスタンスに関するデータとメソッドの定義が含まれる。

一般形式

[IDENTIFICATION DIVISION.]

OBJECT.

[データ部]

[PROCEDURE DIVISION.]

[メソッド-1]...

END OBJECT.

一般規則

1. オブジェクト・プログラムのオブジェクト記憶節内で宣言されたデータ項目はインスタンス・データである。インスタンス・データはインスタンス・メソッド内でのみ参照できる。
2. データ項目は実行単位の開始時に初期化され、メソッド呼出しの間は最後に使用されたときの状態が保たれる。
3. メソッド-1はインスタンス・メソッドである。

17.7 メソッド

書き方

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。

注意：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書き方は、プログラムに関する手続き部の見出しの書き方1と同じである。手続き部の見出しの書き方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、「手続き部の見出し」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。

   注意： 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識別子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識別子内に、その結果が入れられる。

17.8 メソッド・インタフェース定義

メソッド・インタフェース定義は、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。

書き方

METHOD-ID.″メソッド名-1″.

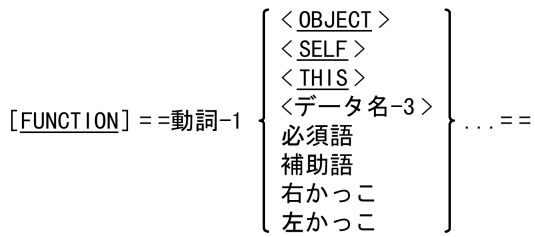
[連絡節]

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″.

動詞シグネチャは下記のとおり。



構文規則

1. メソッド・インタフェー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書き方2と同様である。ただし、呼び名もREPEATED指定も指定できない。上の書き方に示した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、「手続き部の見出し」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 "]" で囲んだもの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。

   注意 : それらによって、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

17.9 データ部

オブジェクト記憶節はオブジェクト・データを宣言するために使用する。オブジェクトへの参照を宣言するためには、OBJECT REFERENCEを使用する。

17.9.1 オブジェクト記憶節

データ部内でオブジェクト記憶節を使用すると、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・データを定義できる。この節を指定するかしないかは任意である。

書き方

```
OBJECT-STORAGE SECTION.
{データ記述項}...
```

構文規則

1. クラス・オブジェクトのオブジェクト記憶節内で宣言されたデータはクラス・メソッド内でのみアクセスできる。
2. 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はインスタンス・メソッド内でのみアクセスできる。
3. クラス名段落内でDATA IS RESTRICTED指定またはDATA IS PROTECTED指定を指定した場合、オブジェクト記憶節内で宣言されたデータをサブクラス内で直接アクセスできる。そうでない場合、このデータにアクセスできるのは呼び出したメソッドのみである。
4. オブジェクト記憶節内で宣言するデータに関する構文規則は、連絡節の場合と同様である。ただし、ODOSLIDEコンパイラ指令を指定した場合には、OCCURS句のDEPENDING ON指定を指定してはならない。

一般規則

1. オブジェクト記憶節内で宣言されたデータはサブクラスによって継承されうる。

2. 新しいインスタンス・オブジェクトはそれぞれ、オブジェクト記憶節内で宣言されたデータ項目について、独自の固有な記憶域を割り当てられる。
3. コンパイラ指令のOOCTRL(+W)を指定すると、作業場所節はオブジェクト記憶節として扱われる。

17.9.2 USAGE（用途）句

USAGE IS OBJECT REFERENCEを指定して定義されたデータ項目はオブジェクトへの参照を収めるために使用される。

一般形式

```
[USAGE IS] OBJECT REFERENCE
```

構文規則

1. USAGE IS OBJECT REFERENCE句を指定するデータ項目には、REDEFINES句を指定してはならない。
2. データ項目の宣言内にUSAGE IS OBJECT REFERENCE句が含まれる場合、そのデータ項目はREDEFINES句内に指定されてはならない。
3. 基本データ項目にUSAGE IS OBJECT REFERENCE句が指定されている場合、そのデータ項目にPICTURE句が含まれていてはならない。
4. 集団項目のデータ記述項内にUSAGE IS OBJECT REFERENCE句が指定されていてはならない。
5. USAGE IS OBJECT REFERENCE句が指定されされているデータ項目にKEY IS句を指定してはならない。
6. USAGE IS OBJECT REFERENCE句が指定されされているデータ項目にVALUE句が指定されている場合、有効な値はNULLだけである。

一般規則

1. 宣言内にUSAGE IS OBJECT REFERENCE句が指定されているデータ項目には、オブジェクトへの参照が収められる。そのオブジェクトのクラスは任意である。
2. 宣言内にUSAGE IS OBJECT REFERENCE句が指定されているデータ項目のサイズと形式は各オブジェクト参照ごとに異なりうる。

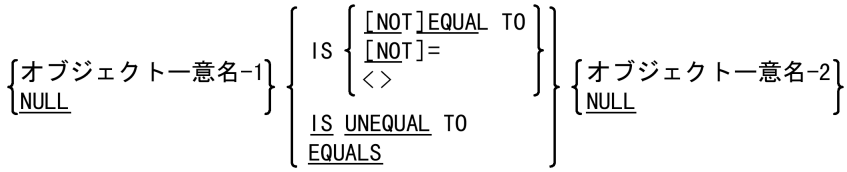
17.10 手続き部

メソッドの手続き部には実行すべき手続きが収められる。オブジェクト定義およびクラス定義の手続き部には、それぞれ、インスタンス・オブジェクトまたはクラス・オブジェクトに関して呼び出されるメソッドが収められる。

17.10.1 条件式

２つのオブジェクト参照を比較して、両者が同じオブジェクトを指しているかどうかをチェックできる。

一般形式



構文規則

1. 比較の一方の作用対象だけに表意定数のNULLを指定できる。

一般規則

1. 同じオブジェクトを指す場合に、２つの作用対象は等しい。そうでない場合には、両者は等しくない。

17.10.2 EXIT（出口）文

EXIT METHOD文は呼び出されたメソッドの論理的な末尾を示す。

一般形式

```
EXIT METHOD
```

構文規則

1. EXIT METHOD文はメソッドの手続き部内にのみ指定できる。

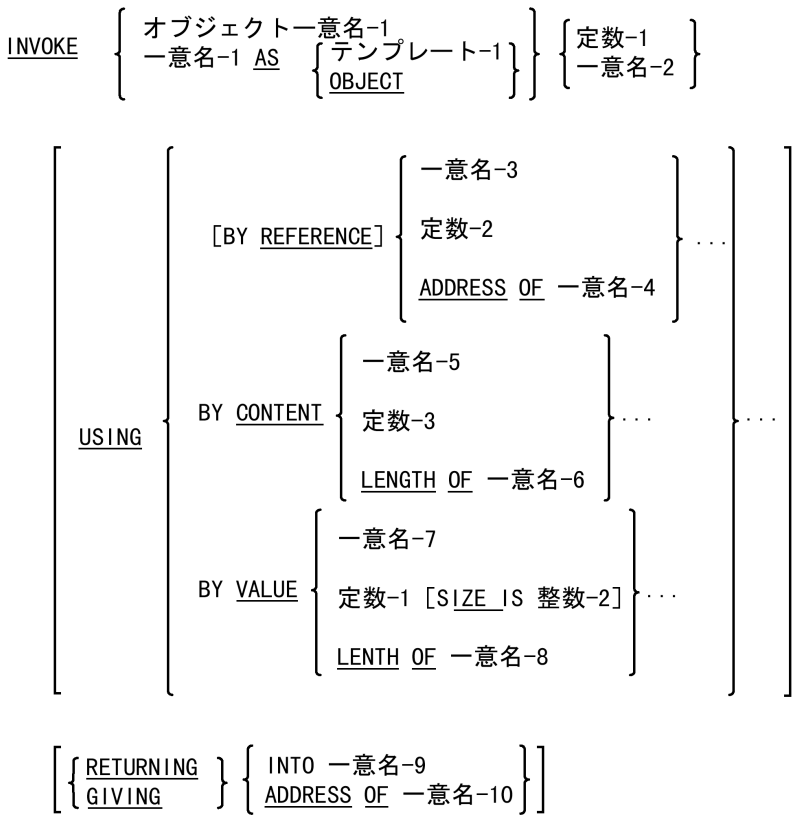
一般規則

1. EXIT METHOD文を実行すると、INVOKE文に続く次の実行可能文に制御が移る。元のメソッド定義中にRETURNING指定があると、RETURNING指定によって参照されるデータ項目中の値が、そのメソッド呼出しの結果となる。

17.10.3 INVOKE（呼出し）文

INVOKE文はメソッドを呼び出す。

一般形式



構文規則

1. オブジェクト識別子-1はオブジェクト識別子でなければならない。
2. 識別子-1は４バイトのデータ項目でなければならない。
3. 定数-1は英数字であり、かつ有効なメソッド名でなければならない。
4. 定数-1、定数-2、定数-3のいずれも表意定数であってはならない。
5. 識別子-2は英数字のデータ項目として定義されていなければならない。その値として、COBOLのメソッド名でもCOBOL以外のメソッド名でも取れるようにするためである。
6. 識別子-3、識別子-4、識別子-5、識別子-6、識別子-7のいずれも関数識別子数であってはならない。
7. 整数-1は符合付きでもゼロでもよい。
8. GIVINGとRETURNINGは同等である。
9. 識別子-9はサイズが４バイトでなければならない。.

10. 識別子-10は連絡節内でレベル番号が01または77のデータ項目として定義されていなければならない。
11. INVOKE文に定数-1が指定され（ただし識別子-2が指定された場合を除く）、現在のコンパイル・ユニット内にメソッド・インタフェース定義が含まれていてその名前が定数-1に合致する場合は、構文チェックの間に下記の項目の妥当性が検査される。
    - 必要なパラメータの数
    - パラメータのタイプ
    - 呼出し方式

一般規則

1. オブジェクト識別子-1および識別子-1はメソッドを呼び出す対象のオブジェクトを識別する。
2. オブジェクト識別子-1によって参照されるデータ項目の内容がクラス・オブジェクトである場合、クラス・メソッドが呼び出される。そうでない場合は、インスタンス・メソッドが呼び出される。
3. 定数-1または識別子-2によって参照されるデータ項目の内容は呼び出す対象のメソッドの名前である。それがCOBOLのメソッドである場合、定数-1または識別子-2によって参照されるデータ項目の内容は、呼び出されるメソッドのメソッド名段落内のメソッド名でなければならない。呼出し対象のメソッドがCOBOLのメソッドでない場合には、メソッド名の構成は該当分野の規則に従う。
4. AS指定を使用すると、COBOLのデータ項目を対象にしてメソッドを呼び出せる。テンプレート-1を指定すると、それはクラス・テンプレートとして使用される。テンプレートの作成は提供されるクラス・ライブラリ中のオブジェクトに付随する機能である（詳細については、書籍***「オブジェクト指向プログラミング」***を参照）。
5. INVOKE文を実行すると、指定したメソッドが実行可能な状態にされて、そのメソッドに制御が移される。そのメソッドから制御が戻されると、INVOKE文の末尾に制御が移される。
6. 実行単位には、呼出し対象のメソッドがCOBOLのものであるか否かは、最初は分からない。それが判明するのは、実行に先立って、呼出し対象のメソッドの所在が突き止められたときである。メソッド名の形式によって、対象のメソッドがCOBOLのものであるか否かを判定はできない。
7. メソッドの呼出しまたは終了のプロセスによって、外部ファイル結合子に関連するファイルの状態または位置づけが変更されることはない。
8. 呼出し対象のメソッドがCOBOLものである場合、そのメソッドの手続き部の見出し内にUSING指定がある場合にのみ、INVOKE文にUSING指定が含められる。その場合、USING指定内の作用対象の数は同じでなければならない。INVOKE文内にRETURNING指定がある場合、呼出し対象のメソッドの手続き部の見出し内にRETURNING指定がなければならない。

9. 呼出し対象のメソッドがCOBOLものではない場合、そのメソッドに対していくつかのパラメータが宣言された場合にのみ、INVOKE文にUSING指定が含められる。その場合、USING指定内の作用対象の数は呼出し対象のメソッドのパラメータの数と同じでなければならない（COLOL以外の言語の処理系の中には、作用対象の数が同じでないことを認めているものがある）。

   注意：COBOL言語ではデータ項目のアドレスの桁寄せについては制約を課していない。他方、COBOL以外の言語では通常はアドレスについて前提を設けており、それに反するデータ項目を参照すると何らかのエラーを起こす。桁寄せを行うには、下記のひとつまたは複数の措置を取る。

   - 集団項目を変更して、無名項目を追加する
   - ALIGNコンパイラ指令を使用するとともに、USING指定内の作用対象のデータ項目を01レベルまたは77レベルとする
   - IBMCOMPコンパイラ指令とSYNCHRONIZED句を併用する

   INVOE文内でRETURNING指定を指定した場合、COBOL以外のメソッドは適切な形式で結果を返さなければならない。
10. INVOKE文のUSING指定および呼出し対象のメソッドの手続き部の見出しのUSING指定の中に現れる作用対象の順序によって、INVOKE文によって使用されるデータ項目と呼び出されるメソッドとの間の通信が決まる。つまり、この通信は位置によって対応が取られるであって、名前が等しいことによって対応が取られるのではない。一方のUSING指定内の最初のパラメータは他方の最初のパラメータに、一方の２番目のパラメータは他方の２番目のパラメータに、それぞれ対応するといったぐあいである。
    パラメータが指標名であると、そのような対応関係は確立されない。呼出し元のプログラムまたはメソッドと呼出し対象のメソッドでは指標名は別々の指標を指すからである。
11. INVOKE文のUSING指定内で参照されるパラメータの値は、INVOKE文が実行されたときに、呼び出されたメソッドで使用できるようになる。
12. BY CONTENT、BY REFERENCE、BY VALUEの各指定は後続のパラメータにも効力を及ぼす。他のBY CONTENT、BY REFERENCE、BY VALUEの指定が出てくると、新しい指定が有効になる。
    最初のパラメータの前にBY CONTENT、BY REFERENCE、BY VALUEのどの指定もないと、BY REFERENCEが指定されているものとみなされる。
13. 識別子-3にBY REFERENCEが明示的または暗示的に指定された場合、その中の対応するデータ項目が呼出し元のプログラムまたはメソッド内のデータ項目と同じ記憶領域を占めるものとして、呼び出されたメソッドは処理を行う。呼び出されるメソッド内のデータ項目は、呼出し元のプログラム内の対応するデータ項目と文字数が同じであるように、記述されていなければならない。

BY REFERENCE ADDRESS OF指定が明示的にまたは暗黙的に指定された場合、USAGE POINTERを用いて追加のデータ項目が宣言され、「SET データ項目 TO ADDRESS OF 識別子-4」文によって取得された値とともにそのデータ項目がBY REFERENCEとして渡されたものとして、呼び出されたメソッドは処理を行う。

識別子-4が連絡節内にあってそのレベル番号が01または77以外であるか、または作業場所節内にある場合、そのデータ項目をBY CONTENTとして渡すことに相当する。その場合、識別子-4のアドレスを呼び出されたメソッドで変更することはできない。定数-2にBY REFERENCE指定が明示的または暗黙的に指定された場合、呼び出されたメソッドは定数-2を定数-3のために記述されたものとして処理する。

14. パラメータに対してBY CONTENT指定が明示的または暗黙的に指定された場合、INVOKE文のUSING指定内で参照されているこのパラメータの値を、呼び出されたメソッドは変更できない。ただし、呼び出されたメソッドは、その手続き部の見出し内の対応するデータ名によって参照されるデータ項目の値を変更することはできる。INVOKE文のBY CONTENT指定内の各パラメータのデータ記述は、手続き部の見出しのUSING指定内の対応するパラメータのデータ記述と同じでなければならない。つまり、変換も拡張も切詰めもあってはならない。

    パラメータに対してBY CONTENT指定が明示的または暗黙的に指定された場合、追加のデータ項目が宣言され、それがBY REFERENCE指定内でパラメータとして使用されているものとして、オブジェクト・プログラムは処理を行う。識別子-5が指定された場合、追加されたデータ項目の暗黙的なデータ記述とその内容は、識別子-5のものと同じである。定数-3が指定された場合、追加されたデータ項目の暗黙的なデータ記述は定数-3と同じサイズの英数字データ項目と同等であり、その内容は定数-3の値に設定される。LENGTH OF 識別子-6が指定された場合、追加されたデータ項目のデータ記述はPIC S9(9) USAGE BINARYと同等であり、その内容は識別子-6に割り当てられた記憶域のバイト数に設定される。

15. パラメータに対してBY VALUE指定が明示的または暗黙的に指定された場合、INVOKE文のUSING指定内で参照されているこのパラメータの値を、呼び出されたメソッドは変更できない。ただし、呼び出されたメソッドは、その手続き部の見出し内の対応するデータ名によって参照されるデータ項目の値を変更することはできる。概念的には、COBOL以外の言語でパラメータの受け渡しに使用されるシステム領域（通常は「スタック」）内に追加データが宣言されており、そのデータ項目は呼び出されたメソッド内の対応するデータ項目と同じ記憶領域を占めるもとして、呼び出されたメソッドは処理を行う。識別子-7が指定された場合、追加されたデータ項目の暗黙的なデータ記述とその内容は、識別子-7のものと同じである。定数-1が指定された場合、追加されたデータ項目の暗黙的なデータ記述は符合付きでUSAGE COMP-5の数字項目となる。ただし、そのサイズは、定数-2が指定されればその値のバイト数、そうでなければ４バイトとなる。

16. LENGTH OF 識別子-8が指定された場合、追加データ項目のデータ記述はPIC S9(9) USAGE BINARYとなり、その値は識別子-8に割り当てられた記憶域のバイト数に設定される。
    INVOKE文のBY VALUE指定内の各パラメータに概念的なデータ項目が追加され、呼び出されるメソッドの手続き部の見出しのUSING指定内に対応するパラメータが宣言される。呼び出されるメソッド内のそれらのパラメータのデータ記述は同じでなければならない。つまり、対応する概念的な追加データ項目を変換したり拡張したり切り詰めたりするものであってはならない。それに加えて、暗黙的な概念データ項目のサイズはシステム領域の最大サイズ（通常は４バイト）を超えてはならない。そうしないと、システムが重大な障害を起こす可能性がある。
    呼出し対象のメソッドがCOBOLのものである場合、INVOKE文のBY VALUE指定内の各パラメータに対応するパラメータが、手続き部の見出しのUSING指定内に存在し、かつBY VALUE指定が明示的または暗黙的に指定されていなければならない。
    呼出し対象のメソッドがCOBOLのものではない場合、BY VALUE指定をいつ使用する必要があるかについて詳細は、対象のシステムによって決まる。
17. 識別子-9が指定された場合、そのデータ記述は呼出し対象のメソッドの手続き部の見出しのRETURNING指定内に指定されているデータ項目と同じでなければならない。メソッドの結果は識別子-9に代入される。そのさい、識別子-9の用途がオブジェクト参照または指標であれば、SET文の規則が適用される。そうでない場合は、MOVE文の規則が適用される。呼出し元のプログラムまたはメソッドに制御が戻されたときには、識別子-9に戻り値が入っている。
18. 識別子-10が指定された場合、呼び出されたメソッドは値を返さなければならない。そのさい、USAGE POINTERを明示的または暗黙的に指定しなければならない。メソッドの結果は、SET文の規則に基づいて、識別子-10に代入される。
19. 呼び出されたメソッドの中にINVOKE文が含まれていてもよい。呼び出されたメソッドは、呼出し元のメソッドを直接または間接的に呼び出す、INVOKE文を実行できる。

17.10.4 SET（設定）文

SET文はオブジェクト参照を代入するために使用する。

一般形式

SET {オブジェクト一意名-1}... TO { オブジェクト一意名-2 / NULL }

構文規則

1. オブジェクト識別子-1およびオブジェクト識別子-2の用途はオブジェクト参照でなければならない。

一般規則

1. SET文はオブジェクト識別子-2によって識別されるオブジェクトのオブジェクト参照を各オブジェクト識別子-1に関連する記憶域に指定された順序で設定する。

第18章：翻訳指示文

翻訳指示文（compiler-directing statement）は、翻訳集団の処理を制御する。以下の翻訳指示機能が用意されている。

- 原始文操作(source text manipulation)を指定するための翻訳指示文
- (ISO2000)標準翻訳オプション(standard compilation options)を指定するための翻訳指示文
- (OSVS)(VSC2)BASIS機構
- (OSVS)(VSC2)++INCLUDE/-INC機構
- (MF)条件付き翻訳(conditional compilation)
- (OSVS)(VSC2)(MF)リスト作成制御文（listing control statement）
- (MF)実行オプション(implementor options)を指定するコンパイラ指令

18.1 原始文操作

原始文操作(source text manipulation) は、原始利用者登録集（source user-library）から複写してくる原文を指定する機能を提供する。この原始利用者登録集は、通常、何らかの適切な原始文エディタ（source text editor）を使用して作成する。
(ANS85)また、原始プログラム中の原文を置き換える機能を提供する。
COBOLライブラリ（library）は、COBOLシステムで利用するための原始文を含むファイルから構成されている。COPY（複写）文の働きは、原始プログラム中に原文を挿入して、COBOLシステムがそれを原始プログラムの一部として扱うようにすることである。登録集原文（library text）中の定数や一意名や語や語の集団を指定して、複写処理中にそれらが出てくると、すべて別の原文に置き換えることができる。 原始文操作では、実行用プログラムを作成するときに、複数のCOBOL登録集を使えるようにすることもできる。
(ANS85)REPLACE（置換）文の働きは、原始文中に現れる原文を新しい原文に置き換えて、COBOLシステムがそれを原始文の一部として扱うようにすることである。

18.1.1 COPY（複写）文

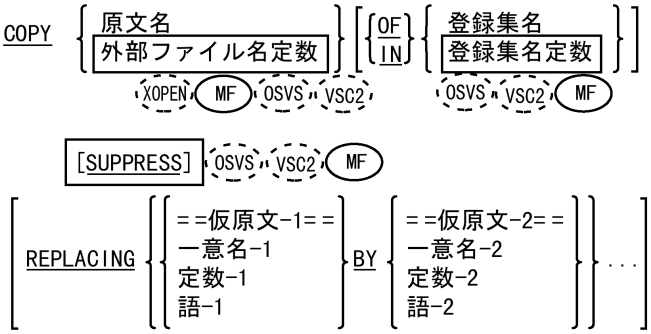
機能

COPY（複写）文は、COBOL翻訳集団中に原文を組み入れる。

例

1. ANSI'68またはLANGLVL(1) の動作を実現するためのOLDCOPYコンパイラ指令の使用例が『言語リファレンス - 追加トピック』中の「例」の章に掲げられている。
2. ANSI'85の規則でサポートされている部分的な語の置換を伴うCOPY文の使用例が『言語リファレンス - 追加トピック』中の「例題」の章に掲げられている。

一般形式



指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - COPYEXT - コピー・ファイルの所在を突き止めるために使用する、ファイル名の拡張子を指定する。
    - COPYLBR - ライブラリ名が**.lbr**に相当することを指定する。
    - COPYLIST - コピー文の結果をリスティング・ファイルに含めさせる。
    - FOLD-COPY-NAME - ディスク上のライブラリ名とCOPY文中の原文名の間で登録集メンバー名の大文字と小文字の指定が違っている場合に、コンパイラ・システムがコピー登録集メンバーを見つけ出せるようにする。
    - OLDCOPY - ANS'68およびOS/VS COBOL LANGLVL(1)の規則に合うように、COPY文の取扱いを変更する。

構文規則

1. 原始プログラムをCOBOLシステムに渡すときに複数のCOBOL登録集が利用できる場合は、原文名を登録集名で修飾して、原文名の原文が登録されているCOBOL登録集を一意に識別しなければならない。登録集についての詳細は、COBOLシステムのマニュアルを参照。

   (OSVS)(VSC2)(MF)この制限は廃止された。
2. The COPY文は、前に空白を1つ置き、後ろを分離符の終止符（.）で止める。
3. 仮原文-1は、空でも、空白文字だけから成っても、注記行だけから成ってもならない。
4. 仮原文-2は、空であってもよい。
5. 仮原文-1および仮原文-2の中の文字列は、行をまたがって継続させることができる。ただし、仮原文の区切り記号（psuedo-text delimiter）は、2つとも同じ行にあること。（*COBOLプログラムの概念*の章の*行のつなぎ*の節を参照。）
6. 語-1または語-2は、任意の単一のCOBOLの語であってよい。

7. 文字列または分離符が書けるところならば、原始プログラム中のどこにでもCOPY文を書くことができる。ただし、COPY文の中にはCOPY文を書けない。
8. 原文名は、一意の外部ファイル名を定義する。このファイル名は、利用者語の作成規則に準拠していること。
   (OSVS)(VSC2)(MF)(COB370)外部ファイル名定数は英数字定数であり、ファイル名に関するオペレーティングシステムの規則に準拠していること。外部ファイル名定数を指定する時は、引用符で囲んでもよいし囲まなくてもよい。
   原文名および引用符で囲んでいない外部ファイル名定数は、常に大文字に変換されることに注意。
9. (MF)登録集名定数は文字定数であり、ファイル名に関するオペレーティングシステムの規則、または装置一意名に関するオペレーティングシステムの規則に準拠していること。登録集名定数を指定する時は、引用符で囲んでもよいし囲まなくてもよい。
   引用符で囲んでいない登録集名定数は、常に大文字に変換されることに注意。
10. (OSVS)(VSC2)(MF)SUPPRESSは、原始リスト上にコピー・メンバの内容を印刷しないようにするために指定する。
11. (ANS85)注記項の中または注記項があってもよい場所に語COPYがある場合、注記項の一部とみなされる。
    (OSVS)注記項中にあるCOPY文は処理される。
12. (OSVS)(MF)外部ファイル名定数および登録集名定数は、引用符で囲まれている場合、文字$、#、@を含んでもよい。

一般規則

1. COPY文を含む翻訳集団をコンパイルすることは、論理的には、すべてのCOPY文を処理してからその結果の翻訳集団をコンパイルすることと同じである。
2. COPY文を処理した結果は、予約語COPYに始まり句読文字の終止符で終わるCOPY文全体を論理的に置き換える形で、原文名によって表わされる登録集原文を原始文中に複写したようになる。
3. REPLACING指定をしないと、登録集原文はそのままで複写される。
   REPLACING指定をすると、複写される登録集原文のうちで、仮原文-1、一意名-1、定数-1、語-1に一致するそれぞれの部分が、対応する仮原文-2、一意名-2、定数-2、語-2によって置き換えられる。
4. 登録集と一致するかどうか比較する目的では、一意名-1と定数-1と語-1は、それぞれ一意名-1または定数-1または語-1だけを含む仮原文として扱われる。
5. 原文を置換するかどうかを判定するための比較は、以下のように行われる。
   a. 分離符のコンマでもセミコロンでもない登録集原文語の左端の語が、比較の最初の対象となる。この原文語の前にある原文語または空白は、原始文中に複写される。比較対象の最初の原文語以降が、REPLACINGの作用対象全体と連続する同じ文字数だけ比較される。ここで、REPLACINGの作用対象とは、予約語BYの前までに指定した仮原文-1、一意名-1、定数-1、語-1を指す。

b. 仮原文-1、一意名-1、定数-1、語-1は、その語を構成する文字の並びが登録集原文語の文字の並びと、逐一同じであるときにだけ、等しいと判定される。照合の目的では、仮原文-1または登録集原文の中の分離符のコンマとセミコロンと空白は、これらが出てくるたびに単一の空白とみなされる。分離符の空白がいくつか並んで出てきたものは、単一の空白とみなされる。
(ANS85)小文字は、COBOL文字集合に指定されている対応する大文字に等しいものとして扱われる。

c. 照合結果が一致しなかった場合、REPLACING指定の続きがあれば、次の各仮原文-1、一意名-1、定数-1、語-1を対象にして比較が繰り返される。このようにして一致するものが見つかるか、 REPLACING指定の作用対象の続きがなくなるまで、比較は繰り返し続けられる。

d. REPLACING指定の作用対象のすべてが比較されたが、一致するものが見つからなかった場合、左端の登録集原文語は原始文中に複写される。次いで、その次に続く登録集原文語が左端の登録集原文語であるとみなされて、REPLACING指定中の最初の仮原文-1、一意名-1、定数-1、語-1から始まって、比較周期が再び開始される。

e. 仮原文-1、一意名-1、定数-1、語-1と登録集原文との間で比較結果が一致した場合、対応する仮原文-2、一意名-2、定数-2、語-2が原始文中に入れられる。比較の対象とされた登録集原文部分のうちの右端の原文語のすぐ後ろの語が、左端の登録集原文語であるとみなされて、REPLACING指定中の最初の仮原文-1、一意名-1、定数-1、語-1から始まって、比較周期が再び開始される。

f. この処理は、登録集原文中の右端の原文語が左端の登録集原文語とみなされて比較され、照合結果が一致するかまたは比較周期を完了したところで、終了する。

6. 比較の目的では、登録集原文および仮原文-1中に出てくる注記行は、単一の空白とみなされる。仮原文-2および登録集原文の中に出てくる注記行は、原始文中に元のまま複写される。

7. 登録集原文内および仮原文-2には、デバッグ行があってもよい。仮原文-1の中では、デバッグ行は認められない。 デバッグ行の中の原文語が比較されるときは、標識領域中に "D" が記されていないものとして扱われる。
(ANS85)原始文中で、左側の仮原文区切り記号の後ろで、かつ、それに対応する右側の仮原文区切り記号の前でデバッグ行が始まっているならば、そのデバッグ行はその仮原文中に指定されているものとする。

8. COPY文の処理を完了した結果の原文中に、 COPY文が含まれていてはならない。
(VSC2)結果の原文中にCOPY文が含まれていてもよい。ただし、そのCOPY文および既に展開済みのCOPY文に、REPLACING指定が含まれていないことが条件である。
(MF)任意のレベルまで、COPY文を入れ子にしてもよい。入れ子になった任意のCOPY文の中に、REPLACING指定が含まれていてもよい。下位レベルのすべての COPY文に対して、REPLACING指定は効力を発揮する。

9. 登録集原文の構文の正しさは、独立して判定することはできない。
   (ANS85)ただし、COPY文は例外である。
   すべてのCOPY文の処理が完了するまで、COBOL原始プログラム全体の構文の正しさは判定できない。
10. 登録集から複写されたが置換はされなかった各原文語は、結果としてできあがるプログラムの中でも、登録集内の行の中でその語が占めていたのと同じ領域から始まるように複写される。しかし、登録集から複写してきたA領域から始まる原文語の後ろに別のやはりA領域から始まる原文語があって、前の方の原文語がそれよりも長いものに置換された場合、A領域に余裕がなければ、後ろの原文語はB領域から開始されることがある。結果としてできあがる原始文の中に入れられる仮原文-2の中の各原文語は、仮原文-2の中でその語が占めていたのと同じ領域から始まるように配置される。各一意名-2、定数-2、語-2は、結果としてできあがるプログラムの中で、比較の対象となった左端の登録集原文語が置換されなかったならば、位置していたのと同じ領域から開始される。
    登録集原文は、COBOLの正書法に適合させること。
    COPY文を処理した結果として原始文中に行が追加される場合、COPY文がデバッグ行上から始まるか、または複写される原文語が登録集原文中のデバッグ行上にあると、その原文語はデバッグ行上に置かれる。BY指定の対象の原文語が組み入れられるときに、置換される最初の登録集原文語がデバッグ行上にあると、置き換わる語もデバッグ行上に配置される。上記の場合を除いて、結果としてできあがる原始文中のデバッグ行上に配置されるのは、仮原文-2の中のデバッグ行上に指定されている原文語だけである。定数-2としてまたは仮原文-2か登録集原文の中に指定されている定数が1行に収まりきらないほど長く、かつその定数がデバッグ行上にはない場合、あふれる分を収容する継続行が後ろに追加される。置換の結果、定数の続きがデバッグ行にかかる場合は、翻訳集団はエラーとなる。
11. 比較の目的では、置換後の原文語は、*COBOLプログラムの概念*の章に記述されている正書法の規則に従って、原始文中に配置される。
12. 装置一意名を明示的に指定しないと、省略時解釈の装置が使用される。省略時解釈値は、オペレーティングシステムによって異なる。
13. (OSVS)OLDCOPYコンパイラ指令を設定すると、COPY文の内容が原文名の複写ファイルの内容で置き換えられる際に、 COPY文の前にあるデータ名が複写ファイル中の対応するデータ名の代わりに用いられる。
14. (ANS85)登録集原文中で何らかの適用規則が守られている場合、データ名の接頭辞のような名前の一部をREPLACING指定を用いて変更できる。
    (ANS85)この方式の「部分的な語の置換」機能を利用するためには、変更の対象とする部分の前後を下記のように囲まなければならない。
    a. 一組の左かっこと右かっこで。たとえば、 **(ABC)**。
    b. コロンで。たとえば、**:XYZ:**は有効な候補である。

18.1.2 REPLACE（置換）文

(ANS85)

機能

REPLACE（置換）文は、原文を翻訳単位に置き換えるために使用する。

一般形式

書き方 1

```
REPLACE{ ==仮原文-1== =BY== ==仮原文-2== }...
```

書き方 2

```
REPLACE OFF
```

構文規則

1. REPLACE文は、翻訳集団中で文字列を書くことができる場所ならば、どこにでも書くことができる。翻訳集団中の先頭の文でない場合は、REPLACE文の前に分離符の終止符（.）を打つこと。
2. REPLACE文は、分離符の終止符（.）によって止める。
3. 仮原文-1には、1語以上の原文語を含める。
4. 仮原文-2には、原文語を含めなくても、1語以上の原文語を含めてもよい。
5. 仮原文-1および仮原文-2の中の文字列は、次の行へ継続できる。
6. 仮原文中の原文語の長さは、1文字から322文字の間とする。
7. 仮原文-1は、そのすべてが分離符のコンマまたはセミコロンから構成されていてはならない。
8. 注記項の中または注記項を記してもよい場所に語REPLACEがあると、注記項の一部とみなされる。

一般規則

1. 仮原文-1には、仮原文-2によって置き換える原文を指定する。
2. 書き方2のREPLACE文は、現在働いている原文置換機能を停止することを指定する。
3. いったん指定した REPLACE文の効力は、次のREPLACE文の指定が出てくるか、そのプログラムの終わりに達するまで持続する。
4. 原始単位中に含まれるREPLACE文の処理は、その原始単位中に含まれるCOPY文があればその処理の後で行われる。
5. REPLACE文を処理した結果の原文には、COPY文もREPLACE文も含まれていてはならない。
6. 原文を置換するか否かを判定するための比較は、以下のように行われる。

a. 原始文中の左端の原文語と最初の仮原文-1から始まって、仮原文-1が原始文中の連続する同じ文字数の原文語と比較される。
b. 仮原文-1は、それを構成する原文語の並びが原始文中の原文語の並びと同じであるときにだけ、等しいと判定される。照合の目的では、仮原文-1または原始文の原文の中の分離符のコンマとセミコロンと空白は、それらが出てくるたびに単一の空白とみなされる。分離符の空白がいくつか並んで出てきたものは、単一の空白とみなされる。
   小文字は、COBOL文字集合に指定されている対応する大文字に等しいものとして扱われる。
c. 照合結果が一致しなかった場合、以降の各仮原文-1を対象にして比較が繰り返される。このようにして、一致するものが見つかるか、仮原文-1の続きがなくなるまで、比較は繰り返し続けられる。
d. 仮原文-1と原始文中の原文との間で比較結果が一致した場合、原始文中のその原文が対応する仮原文-2によって置き換えられる。比較の対象とされた原始文中の原文部分のうちの右端の原文語のすぐ後ろの語が、原始文中の左端の原文語であるとみなされて、最初の仮原文-1から始まって、比較周期が再び開始される。
e. この処理は、原始文の原文中の右端の原文語が REPLACE文の範囲内の左端の原文語とみなされて比較され、照合結果が一致するかまたは比較周期を完了したところで、終了する。

7. 原始文中の原文および仮原文-1の中に現れる注記行または空白行は、照合の目的上は無視される。原始文中の原文および仮原文-1の中の原文語の順序は、正書法に関する規則に従って判定される。詳細については、*COBOLプログラムの概念*の章の*正書法の表現*の節を参照。原始文中の原文が仮原文-2によって置換される場合は、仮原文-2の中の注記行または空白行は結果の原始文にそのまま入れられる。原始文中にあって仮原文-1に一致する一連の原文語の中に現れる注記行または空白行は、結果の原始文には入れられない。
8. 仮原文内には、デバッグ行があってもよい。デバッグ行の中の原文語が比較されるときは、標識領域中に "D" が記されていないものとして扱われる。
9. すべてのCOPY文とREPLACE文の処理が完了した後でだけ、残りの原始コードの構文の正しさを判定できる。
10. REPLACE文を処理した結果として原始文中に挿入される原文語は、正書法に関する規則に従って原始文中に配置される。詳細については、 *COBOLプログラムの概念*の章の*正書法の表現*の節を参照。仮原文-2の原文語を原始文に挿入するときに、追加の空白を組み込むことができるのは、既に空白が空いている原文語の間だけである（原始行の間に想定される空白を含む）。
11. REPLACE文を処理した結果として原始文中に行が追加される場合、追加される行の標識領域には置換される行の先頭の文字が入れられる。ただし、置換される行の先頭の文字がハイフン（-）である場合は、空白が入れられる。

仮原文-2の中の定数が1行に収まりきらないほど長く、かつその定数がデバッグ行上にない場合、あふれる分を収容する継続行が後ろに追加される。置換の結果、定数の続きがデバッグ行にかかる場合は、翻訳単位はエラーとなる。

18.2 コンパイラ指令

(ISO2000)

コンパイラ指令はコンパイラによって使用される選択機能や変数を指定するためのものである。

一般形式

>>翻訳指令

構文規則

1. コンパイラ指令は１行に指定しなければならない。ただし、EVALUATEとIFは別である。この２つの指令には別の規則が適用される。
2. コンパイラ指令の前に置けるのは１つ以上の空白文字だけである。空白がなくともかまわない。
3. 固定方式の正書法を採る場合、コンパイラ指令はプログラム原文領域に記すものとし、その後ろには空白文字だけを記してよい。
4. 自由方式の正書法を採る場合、コンパイラ指令の後ろに記してよいのは空白文字と書いても書かなくてもよい行内注記だけである。
5. コンパイラ指令は２つの連続したCOBOLの文字>>とそれに続くCOBOLの空白文字と翻訳指令から構成される。ただし、空白文字は書かなくともかまわない。>>の後ろに空白文字を書かなかった場合、>>の後ろに空白が１つあるように扱われる。
6. 翻訳指令は、いくつかのコンパイラ指令語から構成される。各コンパイラ指令語にはそれなりの構文の定義がある。
7. コンパイラ指令語は該当のコンパイラ指令の文脈内で予約されている。それを、任意の型のCOBOLの語として、どこでも使用できる。
8. コンパイラ指令は翻訳集団内のどこにでも指定できる。ただし、下記を除く。
    a. 個々のコンパイラ指令の規則で制限されている場所
    b. 原始文操作文の中
    c. 継続されている文字列の行の間
    d. デバッグ行の上
9. 登録集原文内にコンパイラ指令の行を指定できる。
10. コンパイラ指令内に、連結式または表意定数として、定数を指定してはならない。

一般規則

1. COPY文またはREPLACE文の照合処理の間は、コンパイラ指令は単一の空白行として扱われる。コンパイラ指令はいかなる仮原文あるいは部分語とも合致しない。したがって、置換処理の影響を受けない。
2. COPY文およびREPLACE文の処理にさいしては、コンパイラ指令は該当の指令の規則に従って、処理の前、間、後のいずれかの時点で処理される。
3. >>IF指令および>>EVALUATE指令の原始文に指定された場合、コンパイラ指令の>>EVALUATEと>>IFは条件付き翻訳の間に出てきたときに効力を発揮する。それ以外のコンパイラ指令はすべて、条件付き翻訳の結果として得られた原始文が処理されるときに処理される。

18.2.1 条件付き翻訳

コンパイラ指令を使用すると、原始文の範囲を選択して対象に含めたり対象から除外したりすることができる。そのような翻訳方法を条件付き翻訳と呼ぶ。そのために使用するコンパイラ指令はEVALUATEとIFである。EVALUATE指令およびIF指令を使用して、原始文の行を選択し、翻訳の対象に含めたり対象から除外したりする。

18.2.2 コンパイラ指令中の算術式

EVALUATE指令および定数の条件式の中に算術式を指定できる。それらの算術式の構成と優先順位と評価規則 を以下に示す。

1. 作用対象はすべて、固定小数点の数字定数、または使用されるすべての作用対象が固定小数点の数字定数である算術式のどちらかでなければならない。

   注： この中には固定小数点の数字定数に等しい定数を含む。

2. 各算術演算の後で、結果の値の小数部分が切り捨てられて、結果は整数とみなされる。
3. 算術式が評価された後で、結果の値は数字定数とみなされる。
4. 算術演算の結果、桁あふれが発生してはならない。

18.2.3 定数条件式

定数条件式とは、すべての作用対象が定数であるか、定数項だけが含まれる算術式である、条件式である。定義済み条件として知られる特殊な形の条件も、定数条件式の一部として使用できる。

構文規則

1. 定数条件式は下記のどれかに該当しなければならない。

a. 単純比較条件であって、両方の作用対象が定数、または定数項だけを含む算術式であり、 下記の規則に当てはまる。
    1. 比較演算子の両側の作用対象の項類が同じでなければならない。算術式の項類は数字である。
    2. 数字以外の定数を指定した場合、比較演算子は"IS [NOT] EQUAL TO"または"IS [NOT] ="でなければならない。
b. 定義済みの条件
c. 上記の単純条件を組み合わせて構成した複合条件。 省略した形の比較条件の組合せを指定してはならない。

一般規則

1. 複合条件は 10.1.4.3「複合条件」の規則に従って評価される。
2. 作用対象が数字ではない単純比較条件に関しては、文字の大小順序は適用されない。１文字ずつ比較して、等しいか否かが判定される。

注： したがって、大文字と小文字は等しくないとみなされる。

18.2.4 定義済み条件

一般形式

翻訳変数名-1 IS [NOT] DEFINED

一般規則

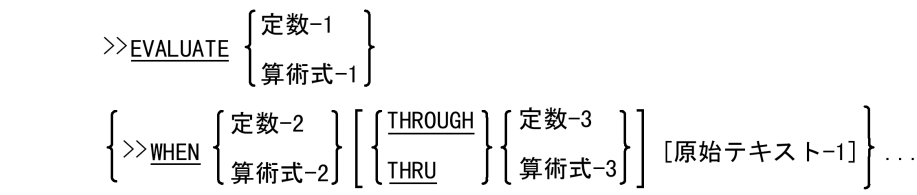
1. IS DEFINED構文を使用した定義済み条件は、翻訳変数名-1が現在定義されている場合にTRUEと評価される。
2. IS NOT DEFINED構文を使用した定義済み条件は、翻訳変数名-1が現在定義されていない場合にTRUEと評価される。

18.2.5 EVALUATE指令

EVALUATE指令を使用すると、複数の分岐のある条件付き翻訳を行える。

一般形式

書き方 1



書き方 2

```
>> EVALUATE TRUE
{>> WHEN 定数条件式-1 [原始テキスト-1]}...
[>> WHEN OTHER [原始テキスト-2]]
>> END-EVALUATE
```

構文規則

すべての書き方

1. 連続したCOBOLの文字>>およびそれに続くコンパイラ指令語および原始文-1と原始文-2の直前までの一組の記述を、新しい行にすべて収まるように指定しなければならない。原始文-1の最初の原文語および原始文-2の最初の原文語はそれぞれ新しい行の先頭から記入しなければならない。
2. 原始文-1および原始文-2はどのような種類のものであってもよい。その中には、コンパイラ指令も含む。原始文-1および原始文-2は複数の行にまたがってよい。

書き方 1

3. EVALUATE指令の作用対象の項類はすべて同じでなければならない。この規則に関しては、算術式の項類は数字である。
4. THROUGH指定を書いた場合、選択の左辺および選択の右辺はすべて、項類が数字でなければならない。
5. 語THROUGHとTHRUは同等である。

一般規則

すべての書き方

1. EVALUATE指令はCOPY文およびREPLACE文の処理の間に処理される。

書き方 1

2. 選択の左辺である定数-1または算術式-1が各WHEN指定中の値と比較される。その結果は下記のようになる。
    a. THROUGH指定がない場合は、選択の左辺が定数-2または算術式-2と等しいと、TRUEが返される。
    b. THROUGH指定がある場合は、選択の左辺が定数-2または算術式-2以上であり定数-3または算術式-3以下であると、TRUEが返される。

   結果がTRUEとなるWHEN指定が見つかると、対応する原始文-1が翻訳されて、>>END-EVALUATEに至るまでの残りの行は>>END-EVALUATEも含めて無視される。

書き方 2

3. 各WHEN指定に関して、順々に、定数条件式が評価される。
   結果がTRUEとなるWHEN指定が見つかった場合、関連する原始文-1が翻訳されて、>>END-EVALUATEに至るまでの残りの行は>>END-EVALUATEも含めて無視される。結果がTRUEとなるWHEN指定が見つからなかった場合、>>WHEN OTHERに対応する原始文-2が指定されていれば、それが翻訳される。

18.2.6 IF指令

IF指令は選択肢が１つまたは２つの
条件付き翻訳を行えるようにする。

一般形式

```
>>IF 定数条件式-1 [原始文-1]
  [ >>ELSE [原始文-2] ]
   >>END-IF
```

構文規則

1. 連続したCOBOLの文字>>およびそれに続くコンパイラ指令語および原始文-1と原始文-2の直前までの一組の記述を、新しい行にすべて収まるように指定しなければならない。原始文-1の最初の原文語および原始文-2の最初の原文語はそれぞれ新しい行の先頭から記入しなければならない。
2. 原始文-1および原始文-2はどのような種類のものであってもよい。その中には、コンパイラ指令も含む。原始文-1および原始文-2は複数の行にまたがってよい。

一般規則

1. IF指令はCOPY文およびREPLACE文の処理の間に処理される。
2. 定数条件式-1の評価結果がTRUEであると、原始文の一部として原始文-1が翻訳されて、原始文-2は無視される。

3. 定数条件式-1の評価結果がFALSEであると、原始文-1は無視されて、原始文-2が指定されていると、原始文の一部として翻訳される。

18.2.7 REPOSITORY指令

REPOSITORY指令は外部リポジトリに情報を追加するかどうかを指定する。また、外部リポジトリ中のインタフェース情報に照らして、プロトタイプおよび定義をチェックするかどうかも指定する。

一般形式

```
>> REPOSITORY UPDATE { ON                          }
                    { OFF [WITH CHECKING]         }
```

指令

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - RDFPATH - リポジトリ・ファイル用の登録集の所在を指定する。

構文規則

1. REPOSITORY指定を書ける場所は、翻訳単位の最初の見出し部の前だけである。翻訳単位の中にREPOSITORY指定を書いてはならない。
2. REPOSITORY指令の効力は後続の翻訳集団に及ぶ。他のREPOSITORY指定が出てくると、新しい指定が発効する。

一般規則

1. 省略時のREPOSITORY指定は下記のとおり。

```
>>REPOSITORY UPDATE OFF WITH CHECKING.
```

2. ON指定を明示的または暗黙的に指定すると、翻訳単位に関する情報が外部リポジトリに追加される。それから、次の翻訳単位の翻訳が行われる。次の翻訳単位がない場合には、外部リポジトリに情報を追加した後で、翻訳が終了する。
3. OFF指定を書くと、外部リポジトリは更新されない。
4. CHECKING指定を書くと、外部リポジトリ中のソース単位に関する情報と異なるクラス定義またはインタフェース定義に警告のフラグが付けられる。外部リポジトリ中の情報の詳細については、3.3「外部リポジトリ」に指定されている。

18.3 BASIS機構

OSVS VSC2

BASIS機構は、COBOLの原始ファイルを非対話的に（一時的に）編集し、その結果をCOBOLシステムに投入できるようにする。 一時的編集の源として参照されるCOBOLの原始ファイル（"編集元プログラム"）は、変更されることはない。また、COBOLシステムによって出力されたファイル（リストなど）以外には、編集結果の記録は残らない。結果として出力されるコード・ファイルは、参照する原始ファイルが存在しないため、アニメートすることはできない。
BASIS機構は固定形式の原始ファイルにだけ有効である。

COBOLシステムに投入されるプログラムは、2つのファイルから構成される。1つは編集制御ファイル（edit control file）（BASIS文と編集情報を記録している）であり、もう1つはCOBOL原始ファイル（上記の"編集元プログラム"）である。
BASIS機構の中で特別な働きをする文が3つある。それらは下記のとおり。

1. BASIS
2. INSERT
3. DELETE

これらのBASIS機構用の文は、COBOL言語の一部を構成するものではない。これらの文は1行に完全に収まっていなければならず、大文字で記さなければならない。
INSERT文またはDELETE文を使用して、BASIS文によって用意されたCOBOLの原始プログラムを変更する場合、COBOL原始プログラムの一連の項目は、昇順に一連番号を持っていなくてはならない。

BASIS機構文に関する一般的注意

1. 原始プログラム行のB領域に完全に含まれ、その後ろに有効な編集元順序番号または編集元順序番号の範囲が続かないDELETE文は、COBOLシステムによってCOBOLのDELETE文として扱われる。
2. BASIS機構のDELETE文で、編集元順序番号または編集元順序番号の範囲は、すべて数字の昇順になっていなければならない。
3. 編集制御ファイル内では、編集元順序番号または編集元順序番号の範囲は、すべて数字の昇順になっていなければならない。

18.3.1 BASIS文

OSVS VSC2

機能

BASIS文は、該当するプログラム（編集制御ファイルとCOBOL原始ファイル）をBASIS機構の規則に従って、COBOLシステムに投入することを示す。

一般形式

```
[順序番号] BASIS { 原文名
                 { 外部ファイル名定数
```

構文規則

1. BASIS文は、編集制御ファイルの最初の行に置く。
2. 予約語BASISは、その文を記述する行のカラム1から66の間のどこから書き始めてもよい。
3. 順序番号は、その文を記述する行のカラム1から6の間のどこに書いてもよい。その後ろには空白文字を1つ置く。
4. 原文名は、一意の外部ファイル名を定義する。このファイル名は、利用者語の作成規則に準拠していること（小文字は大文字に変換されることに注意）。外部ファイル名定数は、引用符で囲んだ文字定数である。ファイル名に関するオペレーティングシステムの規則に準拠していること。

一般規則

1. 原文名または外部ファイル名定数は、編集制御ファイルによって編集するCOBOL原始ファイルを指名する。
2. COBOL原始ファイルの編集は、編集制御ファイル中のINSERT文およびDELETE文によって行われる。

18.3.2 DELETE（削除）文 - BASIS制御

(OSVS) (VSC2)

機能

DELETE（削除）文は、（BASIS機構のもとで）COBOLシステムに無視させるCOBOL原始ファイル中の行を指定する。 DELETE文の後ろに続くCOBOL文は、（編集制御ファイル中に次のBASIS機構のINSERT文またはDELETE文が出てくるまで）原始ファイル中に残される。

一般形式

```
[順序番号] DELETE

{編集元順序番号-1          }[ , {編集元順序番号-2          }]...
{編集元順序番号の範囲-1    }[   {編集元順序番号の範囲-2    }]

    [COBOL原始プログラムの文]...
```

構文規則

1. 予約語DELETEは、その文を記述する行のカラム1から6の間のどこから書き始めてもよい。
2. 順序番号は、その文を記述する行のカラム1から6の間のどこに書いてもよい。その後ろには空白文字を1つ置く。
3. 編集元順序番号-1や編集元順序番号-2などは、引用符で囲まない6文字の正の整数とする（整数数字定数に関する規則に従う）。
4. 編集元順序番号の範囲-1や編集元順序番号の範囲-2などは、2つの編集元順序番号（上記）をハイフン（-）でつないだものとする。
5. 編集元順序番号または編集元順序番号の範囲の間のコンマは、必須である。

一般規則

1. 編集元順序番号-1や編集元順序番号-2などは、COBOL原始ファイル中の文のうちCOBOLシステムに無視させるものの順序番号を表わす。
2. 編集元順序番号の範囲-1や編集元順序番号の範囲-2などは、COBOL原始ファイル中の文のうち中間コードを生成するときに、COBOLシステムに無視させるものの順序番号の範囲を表わす。この範囲は両端を含む。
3. 編集制御ファイル中に次のBASIS機構のINSERT文またはDELETE文が出てくるまで、DELETE文の後ろに続くCOBOL文は、COBOLシステムに投入される原始ファイル中に残される。それらのCOBOL原始プログラムの文は、BASIS機構のDELETE文によって削除された最後の文に代わって、挿入される。

18.3.3 INSERT（挿入）文 - BASIS制御

OSVS VSC2

機能

INSERT（挿入）文は、（BASIS機構のもとで）COBOLシステムに投入するプログラム中に挿入する、COBOL原始プログラムの行をリストする。

一般形式

```
[順序番号]INSERT編集元順序番号
COBOL 文[COBOL 文]
```

構文規則

1. 予約語のINSERTは、その文を記述する行のカラム1から66の間のどこからでも書き始めることができる。.
2. 編集元順序番号は、引用符で囲まない正の整数とする（整数数字定数に関する規則に従う）。

一般規則

1. 編集元順序番号は、COBOL原始ファイル中のどの文の後ろにCOBOL文を挿入するかを示す。
2. 編集制御ファイル中のINSERT文の後ろに続くすべてのCOBOL文は、次のBASIS機構のDELETE文またはINSERT文が出てくるまで、COBOLシステムに投入される原始プログラム中に含められる。BASIS機構のINSERT文の直後には、COBOL文が少なくとも1行はあること。

18.4 ++INCLUDE機構および-INC機構

OSVS VSC2

++INCLUDE機構および-INC機構は、COBOLプログラムの原始ファイルまたは原始コードの一部をコンパイル対象のファイルに組み込めるようにする、もう1つの（IBMメインフレーム互換の）手段である。

これらの++INCLUDE機構および-INC機構の文は、COBOL言語の一部を構成するものではない。これらの文は1行に完全に収まっていなければならず、大文字で記さなければならない。

18.4.1 -INC文

OSVS VSC2

機能

-INC文は、コンパイル時に、ある原始ファイルのすべてのデータ・レコードを別の原始ファイルに組み込むために使用する。

一般形式

```
-INC 原文名
```

指令およびランタイム・スイッチ

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - LIBRARIAN - -INCの構文を使用できるようにする。

構文規則

1. -INCはカラム1から書き始め、後ろに1つ以上の空白文字を置く。
2. 原文名は、一意の外部ファイル名を定義する。このファイル名は利用者語の作成規則に準拠していること。
3. -INC文は、COBOL言語の一部を構成するものではない。これらの文は1行に完全に収まっていなければならず、大文字で記さなければならない。

一般規則

1. 原文名は、原始コード中でこの文を書いた場所に組み込む、COBOL原始ファイルを指名する。
2. この行に原文名以外のテキストを書くと、注記として扱われる。

18.4.2 ++INCLUDE文

OSVS VSC2

機能

++INCLUDE文は、コンパイル時に、ある原始ファイルのすべてのデータ・レコードを別の原始ファイルに組み込むために使用する。

一般形式

```
++INCLUDE 原文名
```

指令およびランタイム・スイッチ

1. 予約語リストにフラグを付けたり修正したりするコンパイラ指令に加えて、下記の指令によって、この項に記述した構文または意味が影響を受ける可能性がある。
    - PANVALET - ++INCLUDEの構文を使用できるようにする。

構文規則

1. ++INCLUDEはカラム8から書き始め、 すべて大文字とし、後ろに1つ以上の空白文字を置く。
2. 原文名は、一意の外部ファイル名を定義する。このファイル名は、利用者語の作成規則に準拠していること。
3. ++INCLUDE文は、COBOL言語の一部を構成するものではない。これらの文は1行に完全に収まっていなければならず、大文字で記さなければならない。

一般規則

1. 原文名は、原始コード中でこの文を書いた場所に組み込む、COBOL原始ファイルを指名する。
2. この行に原文名以外のテキストを書くと、注記として扱われる。

18.5 条件付き翻訳

(MF)

COBOLシステムには、COBOL原始コードの一部またはすべてを選択してコンパイルする機能が備わっている。条件付き翻訳の利点を十分に活かすためには、 78レベル(*データ部 - データ記述*の章を参照) およびCONSTANT コンパイラ指令を使用するのがよい。条件付き翻訳を行うには、$IF、$ELSE、$ENDの構造を用いる。この構造は、COBOL IF 構造と同様の働きをする。

条件付き翻訳文に関する一般規則

1. 条件付き翻訳の文は、固定形式原始コードのカラム7にドル記号($)を書き、その後ろにIF、DISPLAY、ELSE、ENDのどれかを続けたものである。
2. 条件付き翻訳は、COBOLの文字列を分ける形で使用してはならない。つまり、継続する行の間に、条件付き翻訳制御を割り込ませることはできない。

18.5.1 $DISPLAY文

(MF)

機能

原始行の処理中に$DISPLAY文が出て来ると(条件に従って除外された原始行を含まない)、 標準の出力装置上にテキスト・データが表示される。

一般形式

```
$DISPLAY テキストデータ
```

構文規則

1. 文全体を1行に収めなければならない。

18.5.2 $ELSE文

(MF)

機能

最も近い$IF文条件の逆を適用する。最も近い$IF文が真と評価されると、$ELSE文の後ろに続く原始行が処理される。最も近い$IF文が偽と評価されると、次の条件付翻訳行が出て来るまで、COBOL原始行は無視される。

一般形式

```
$ELSE
```

構文規則

1. 文全体を1行に収めなければならない。

18.5.3 $END文

(MF)

機能

最も内側の$IF文条件を終了させる。次いで、その時点で働いている$IF文に従って、処理が行われる。この条件が真と評価されると、$END文の後ろに続く原始行が処理される。この条件が偽と評価されると、次の条件翻訳行が出て来るまで、COBOL原始行は無視される。

一般形式

```
$END
```

構文規則

1. 文全体を1行に収めなければならない。

18.5.4 $IF文

(MF)

機能

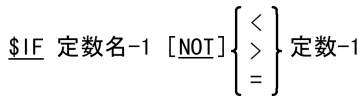
$IF文は原始文の選択した部分を翻訳に含めないようにする機能を提供する。

例

1. 条件付き翻訳の使用例が『言語リファレンス - 追加トピック』中の「例」の章に掲げられている。

一般形式

書き方 1



書き方 2

```
$IF 定数名-2[NOT]DEFINED
```

構文規則

1. 定数名-1はレベル78の記述項またはCONSTANTコンパイラ指令によって定義される。
2. 定数-1が数字である場合、正の整数またはゼロでなければならない。
3. 文全体を1行に収めなければならない。
4. $IF文は、他の$IF文の中に入れ子にすることができる。

一般規則

1. レベル78の記述項またはCONSTANTコンパイラ指令の左辺である場合、定数名-2は「定義されており」、そうでない場合は「定義されていない」。［?］
2. 条件が真と評価されると、$IF文の後ろに続く原始行が処理される。 条件が偽と評価されると、次ぎの条件付き翻訳行が出て来るまで、COBOL原始行は無視される。

18.6 リスト制御文

(OSVS)(VSC2)(MF)

リスト制御文（listing control statement）は、コンパイル処理中に出力ファイルのリストの作成を制御する。

リスト制御文には下記の3つがある。

1. EJECT
2. SKIP1, SKIP2 AND SKIP3
3. (VSC2)(MF)TITLE

18.6.1 EJECT（改ページ）文

(OSVS)(VSC2)(MF)

機能

EJECT（改ページ）文は、原始コードの次の行を次のページの冒頭に印刷するように、COBOLシステムに指示する。

一般形式

```
EJECT [.]
```

構文規則

1. EJECT文は、A領域またはB領域から書き始めてよいが、1行に単独で書く。最後に終止符を付けなくてもよい。

一般規則

1. EJECT文自体は印刷されない。

18.6.2 SKIP1，SKIP2，SKIP3文

(OSVS)(VSC2)(MF)

機能

SKIP1，SKIP2，SKIP3の各文は、COBOLシステムによって作成される 原始コードのリストの縦の間隔を制御する。これらは、原始コードのリスト中で空ける行間隔を指定する。

一般形式

```
{SKIP1}
{SKIP2} [.]
{SKIP3}
```

構文規則

1. これらの文は、A領域またはB領域から書き始めてよいが、1行に単独で書かなければならない。
   (VSC2)(MF)最後に終止符を付けても付けなくてもよい。

一般規則

1. SKIP1は、COBOLシステムに1行間隔を空けるように(2行改行)指示する。SKIP2は、COBOLシステムに2行間隔を空けるように（3行改行）指示する。SKIP3は、COBOLシステムに3行間隔を空けるように（4行改行）指示する。
2. SKIP文自体は印刷されない。

18.6.3 TITLE（標題）文

VSC2 MF

機能

TITLE（標題）文は、リストの後続のすべての先頭行に印刷する標題を、COBOLシステムに知らせる。

一般形式

```
TITLE 定数-1[.]
```

構文規則

1. 定数-1は文字とする。後ろに終止符を付けてもよい。表意定数であってはならない。
2. 語TITLEは、A領域またはB領域から書き始めてよいが、1行に単独で書かなければならない。
3. TITLE文は、翻訳集団中のどこに書いてもよい。

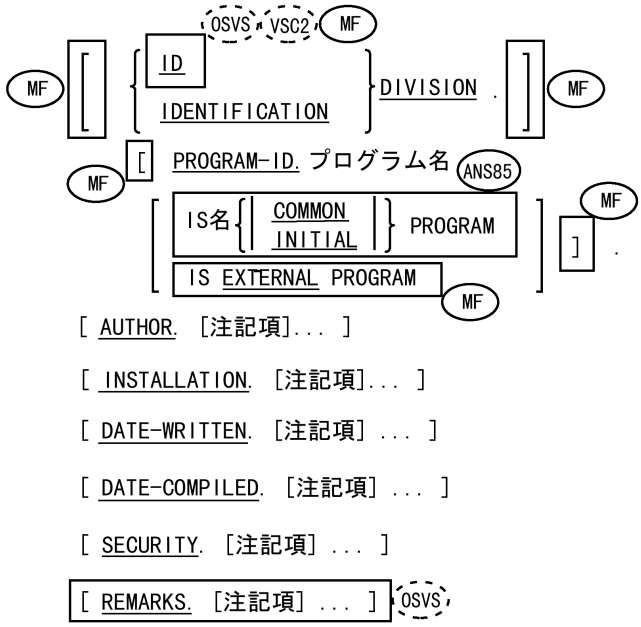
一般規則

1. 定数-1は、リストの後続のすべてのページに付ける標題として使用される。TITLE指令が出てくるまでは、省略時解釈の標題が使用される。省略時解釈の標題には、使用されているCOBOLシステムの識別情報と現在のリリース番号が示される。
2. 指定した標題または省略時解釈の標題は、各ページの先頭行の左側に寄せて印刷される。その行の右側には、中間コードが生成された日付と時刻およびページ番号が表示される。
3. 2番目の標題行も出力される。この行には、主原始ファイルおよび最新の複写ファイルの名前が表示される。
4. TITLE文を指定すると、すぐに改ページが行われる。
5. TITLE文自体は印刷されない。

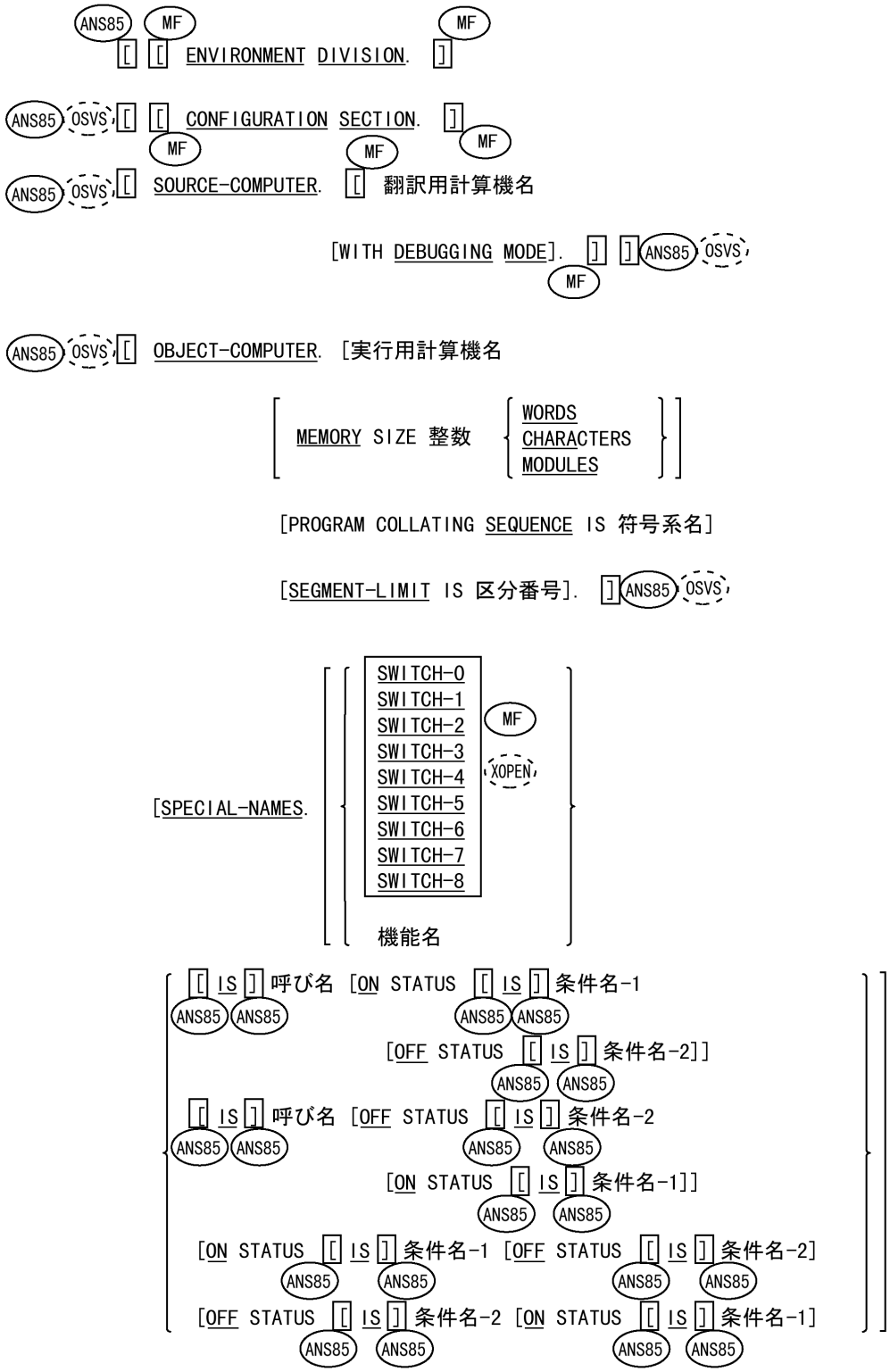
付録A: COBOL標準機能用の一般形式の一覧

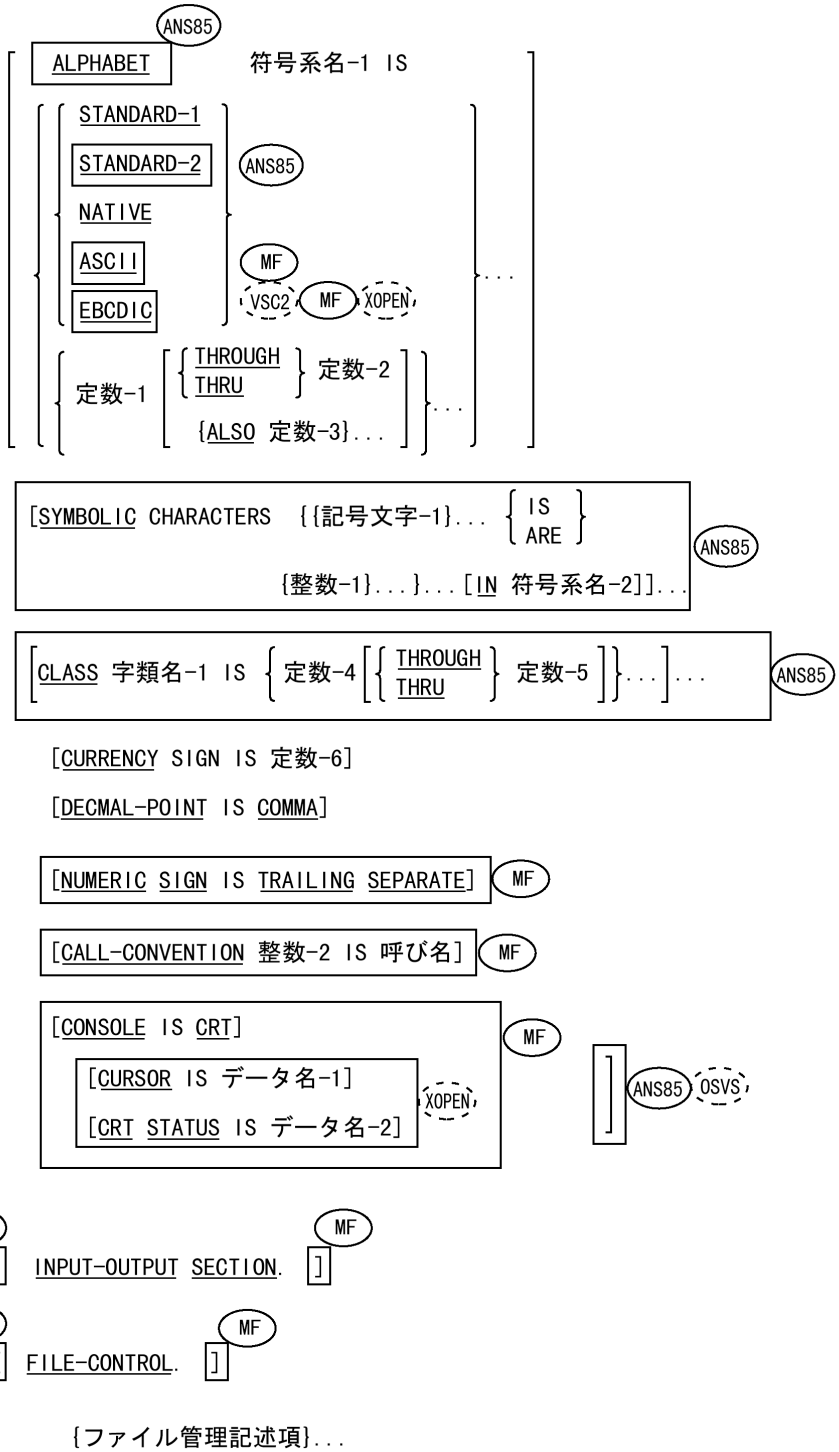
この付録では、このCOBOLシステムの構文を列挙する。この中には、Data General Interactive COBOL、 Ryan McFarland COBOL、Microsoft COBOLと互換性をもたせることだけを目的として用意されている構文は含まない。 これらの情報に関しては、***COBOL言語リファレンス - 追加トピック***の *Data General Interactive COBOL V1.3 構文支援機能*、 *Ryan McFarland COBOL V2.0 構文支援機能* 及び *Microsoft COBOL V1.0およびV2.0の構文支援機能* を参照。

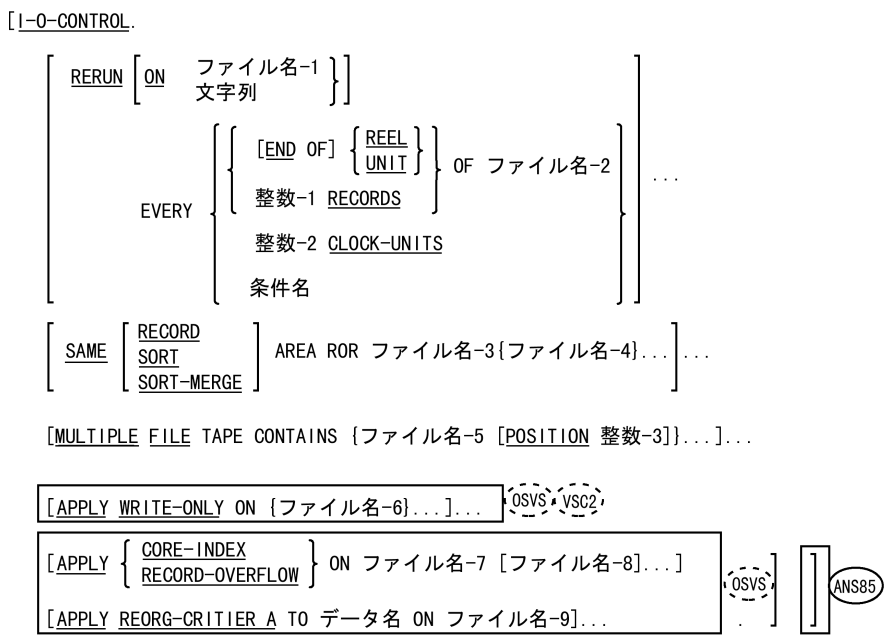
A.1 見出し部の一般形式



A.2 環境部の一般形式

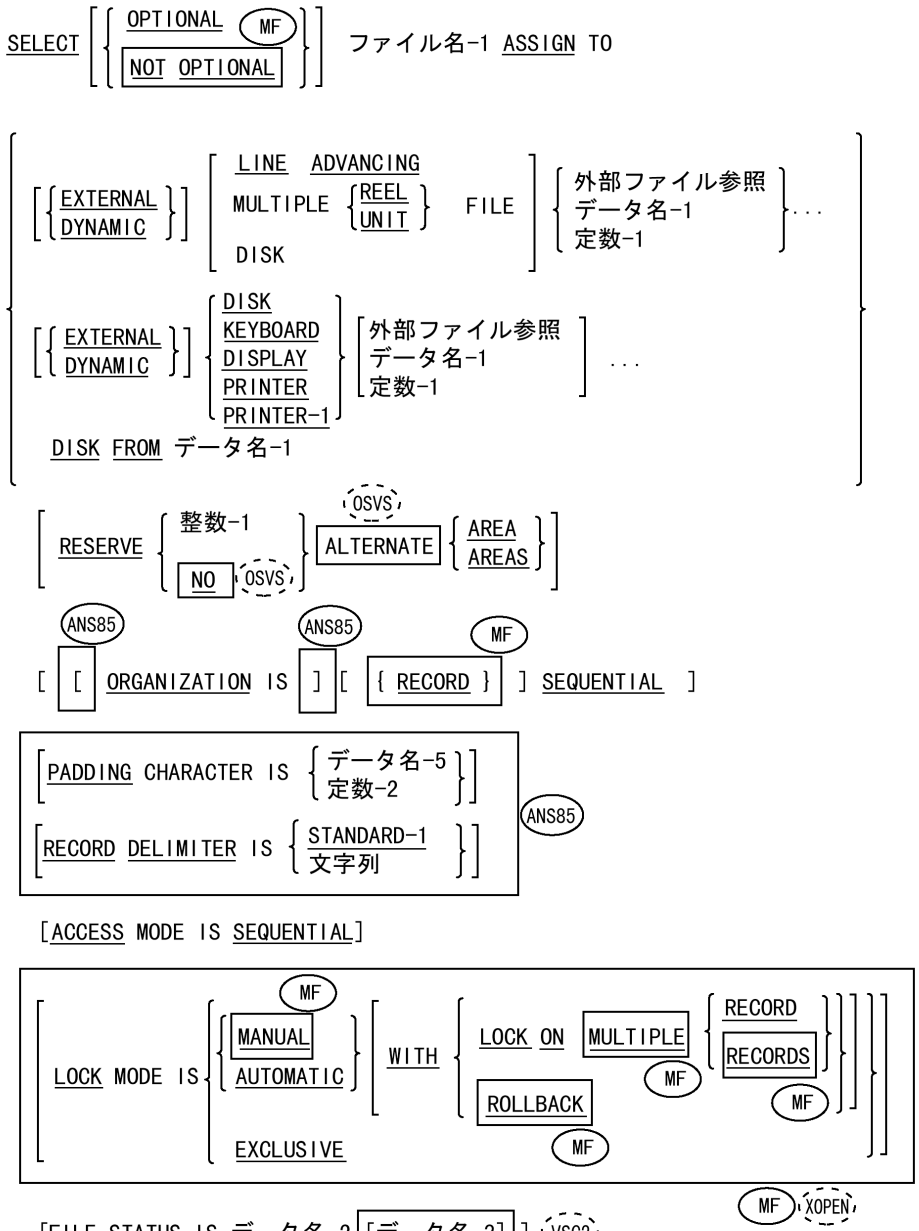

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. 翻訳用計算機名
[WITH DEBUGGING MODE].
OBJECT-COMPUTER. [実行用計算機名
MEMORY SIZE 整数
WORDS
CHARACTERS
MODULES
[PROGRAM COLLATING SEQUENCE IS 符号系名]
[SEGMENT-LIMIT IS 区分番号].
[SPECIAL-NAMES.
SWITCH-0
SWITCH-1
SWITCH-2
SWITCH-3
SWITCH-4
SWITCH-5
SWITCH-6
SWITCH-7
SWITCH-8
機能名
[IS] 呼び名 [ON STATUS [IS] 条件名-1
[OFF STATUS [IS] 条件名-2]]
[IS] 呼び名 [OFF STATUS [IS] 条件名-2
[ON STATUS [IS] 条件名-1]]
[ON STATUS [IS] 条件名-1 [OFF STATUS [IS] 条件名-2]
[OFF STATUS [IS] 条件名-2 [ON STATUS [IS] 条件名-1]


ANS85
ALPHABET
符号系名-1 IS
STANDARD-1
STANDARD-2
ANS85
NATIVE
ASCII
EBCDIC
MF
VSC2
MF
XOPEN
定数-1
THROUGH
THRU
定数-2
{ALSO 定数-3}...
[SYMBOLIC CHARACTERS {{記号文字-1}... IS ARE
{整数-1}...}...[IN 符号系名-2]]...
ANS85
[CLASS 字類名-1 IS 定数-4 THROUGH THRU 定数-5]...
ANS85
[CURRENCY SIGN IS 定数-6]
[DECMAL-POINT IS COMMA]
[NUMERIC SIGN IS TRAILING SEPARATE]
MF
[CALL-CONVENTION 整数-2 IS 呼び名]
MF
[CONSOLE IS CRT]
[CURSOR IS データ名-1]
[CRT STATUS IS データ名-2]
XOPEN
MF
ANS85
OSVS
MF
[INPUT-OUTPUT SECTION.
MF
MF
FILE-CONTROL.
MF
{ファイル管理記述項}...

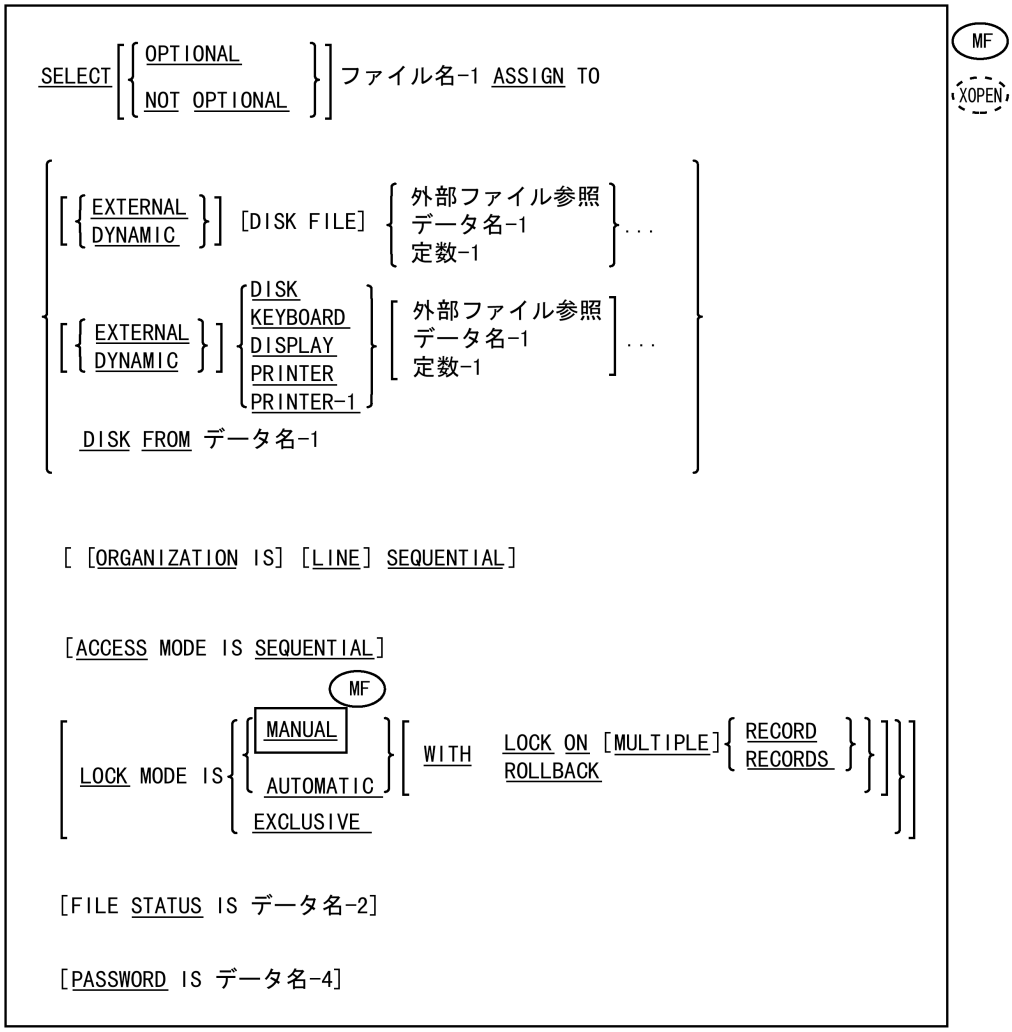

[I-O-CONTROL.
RERUN [ON ファイル名-1 / 文字列]
EVERY { [END OF] {REEL / UNIT} OF ファイル名-2 / 整数-1 RECORDS / 整数-2 CLOCK-UNITS / 条件名 } ...
SAME [RECORD / SORT / SORT-MERGE] AREA ROR ファイル名-3{ファイル名-4}... ...
[MULTIPLE FILE TAPE CONTAINS {ファイル名-5 [POSITION 整数-3]}...]...
[APPLY WRITE-ONLY ON {ファイル名-6}...]...
OSVS
VSC2
[APPLY {CORE-INDEX / RECORD-OVERFLOW} ON ファイル名-7 [ファイル名-8]...]
[APPLY REORG-CRITIER A TO データ名 ON ファイル名-9]...
OSVS
ANS85

A.2.1 ファイル管理記述項の一般形式

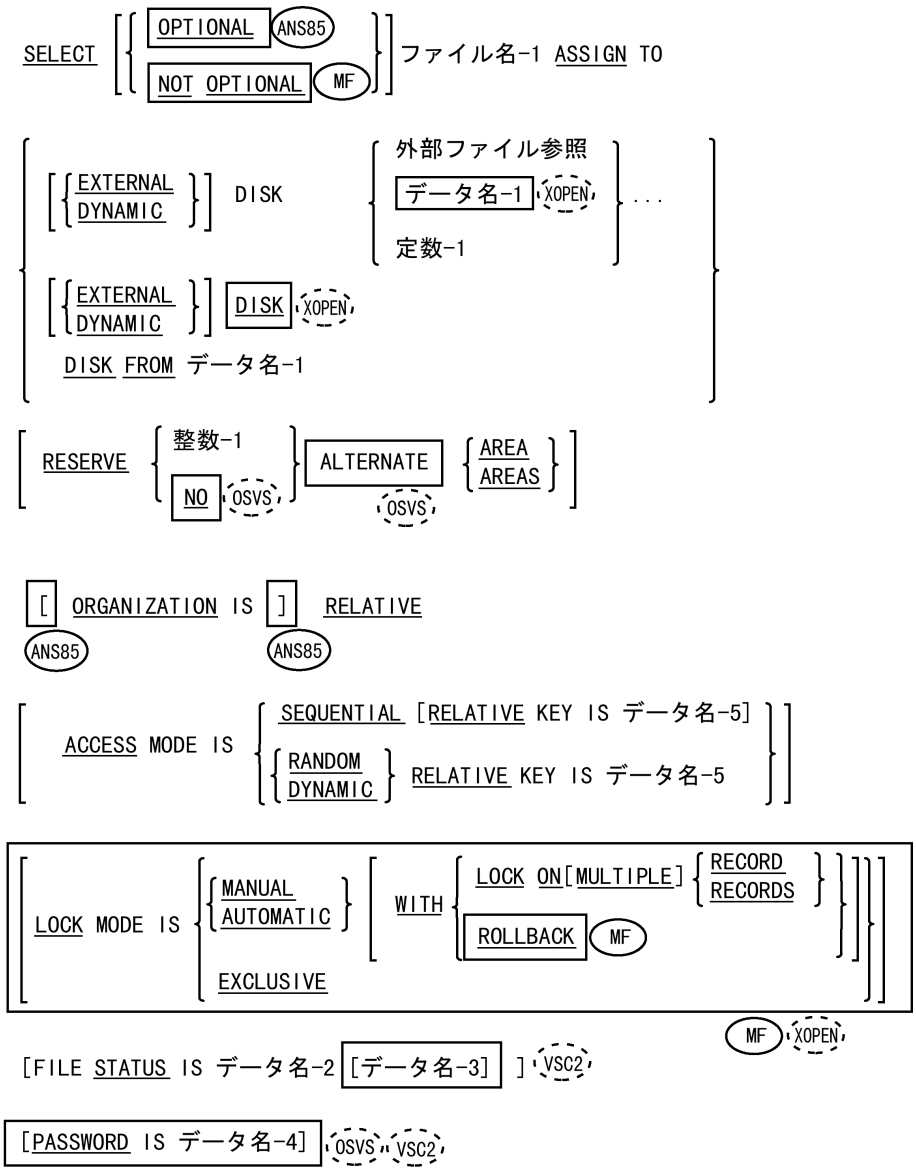
A.2.1.1 順ファイル



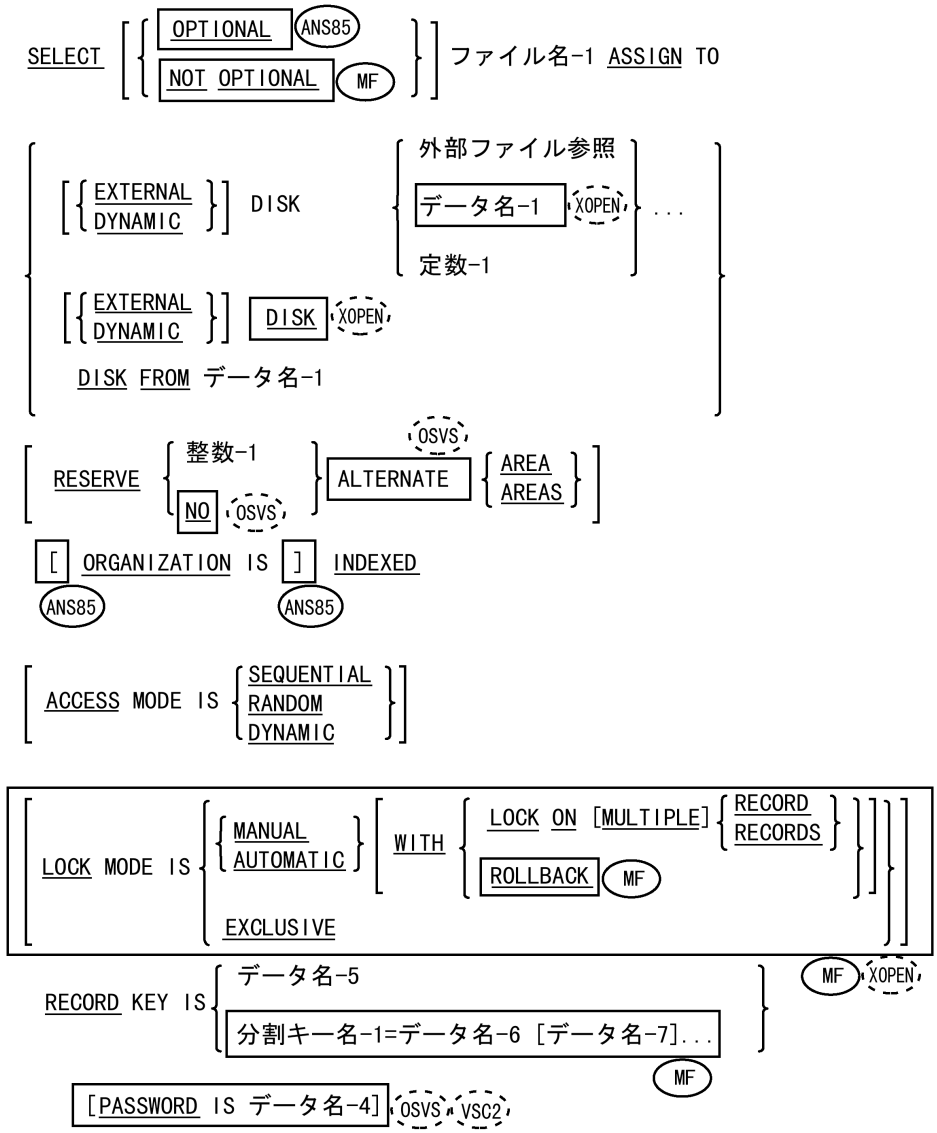
A.2.1.2 行順ファイル

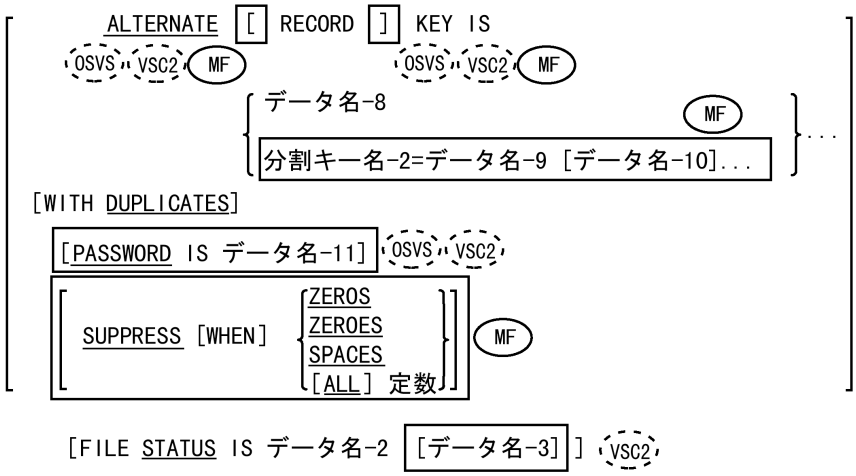

MF
XOPEN
SELECT [OPTIONAL / NOT OPTIONAL] ファイル名-1 ASSIGN TO
[EXTERNAL / DYNAMIC] [DISK FILE] 外部ファイル参照 / データ名-1 / 定数-1 ...
[EXTERNAL / DYNAMIC] DISK / KEYBOARD / DISPLAY / PRINTER / PRINTER-1 [外部ファイル参照 / データ名-1 / 定数-1] ...
DISK FROM データ名-1
[[ORGANIZATION IS] [LINE] SEQUENTIAL]
[ACCESS MODE IS SEQUENTIAL]
MF
LOCK MODE IS MANUAL / AUTOMATIC / EXCLUSIVE [WITH LOCK ON [MULTIPLE] RECORD / RECORDS ROLLBACK]
[FILE STATUS IS データ名-2]
[PASSWORD IS データ名-4]

A.2.1.3 相対ファイル

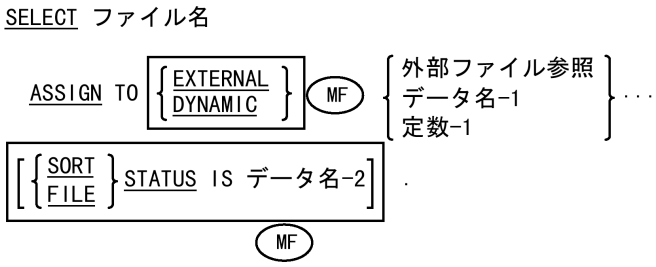

SELECT
OPTIONAL
ANS85
NOT OPTIONAL
MF
ファイル名-1 ASSIGN TO
EXTERNAL
DYNAMIC
DISK
外部ファイル参照
データ名-1
XOPEN
定数-1
DISK FROM データ名-1
RESERVE
整数-1
NO
OSVS
ALTERNATE
AREA
AREAS
[ORGANIZATION IS] RELATIVE
ACCESS MODE IS
SEQUENTIAL [RELATIVE KEY IS データ名-5]
RANDOM
DYNAMIC
RELATIVE KEY IS データ名-5
LOCK MODE IS
MANUAL
AUTOMATIC
EXCLUSIVE
WITH
LOCK ON[MULTIPLE]
RECORD
RECORDS
ROLLBACK
MF
[FILE STATUS IS データ名-2 [データ名-3]]
VSC2
[PASSWORD IS データ名-4]
OSVS
VSC2

A.2.1.4 索引ファイル

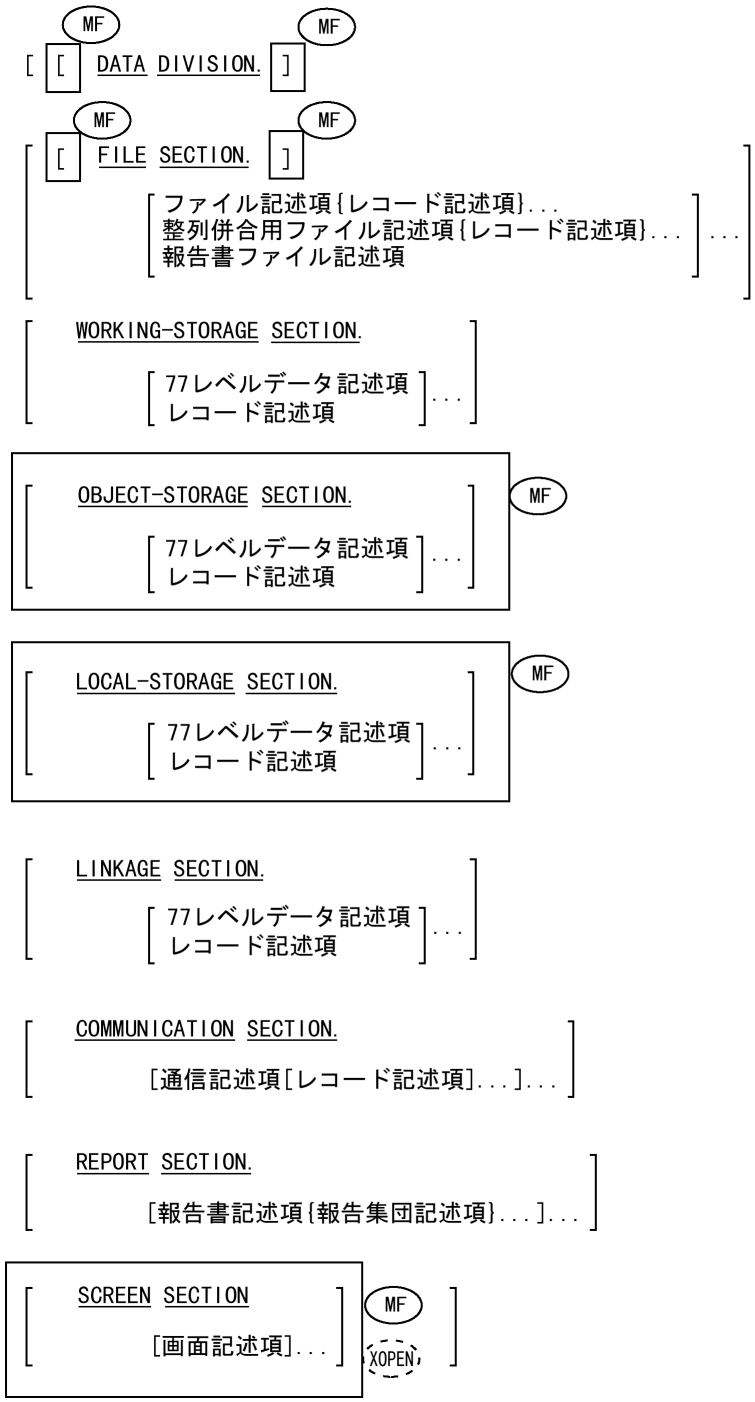

SELECT
OPTIONAL
ANS85
NOT OPTIONAL
MF
ファイル名-1 ASSIGN TO
EXTERNAL
DYNAMIC
DISK
外部ファイル参照
データ名-1
XOPEN
定数-1
DISK
DISK FROM データ名-1
RESERVE
整数-1
NO
OSVS
ALTERNATE
AREA
AREAS
ORGANIZATION IS
INDEXED
ACCESS MODE IS
SEQUENTIAL
RANDOM
DYNAMIC
LOCK MODE IS
MANUAL
AUTOMATIC
EXCLUSIVE
WITH
LOCK ON [MULTIPLE]
RECORD
RECORDS
ROLLBACK
RECORD KEY IS
データ名-5
分割キー名-1=データ名-6 [データ名-7]...
[PASSWORD IS データ名-4]
VSC2



A.2.1.5 整理併合ファイル

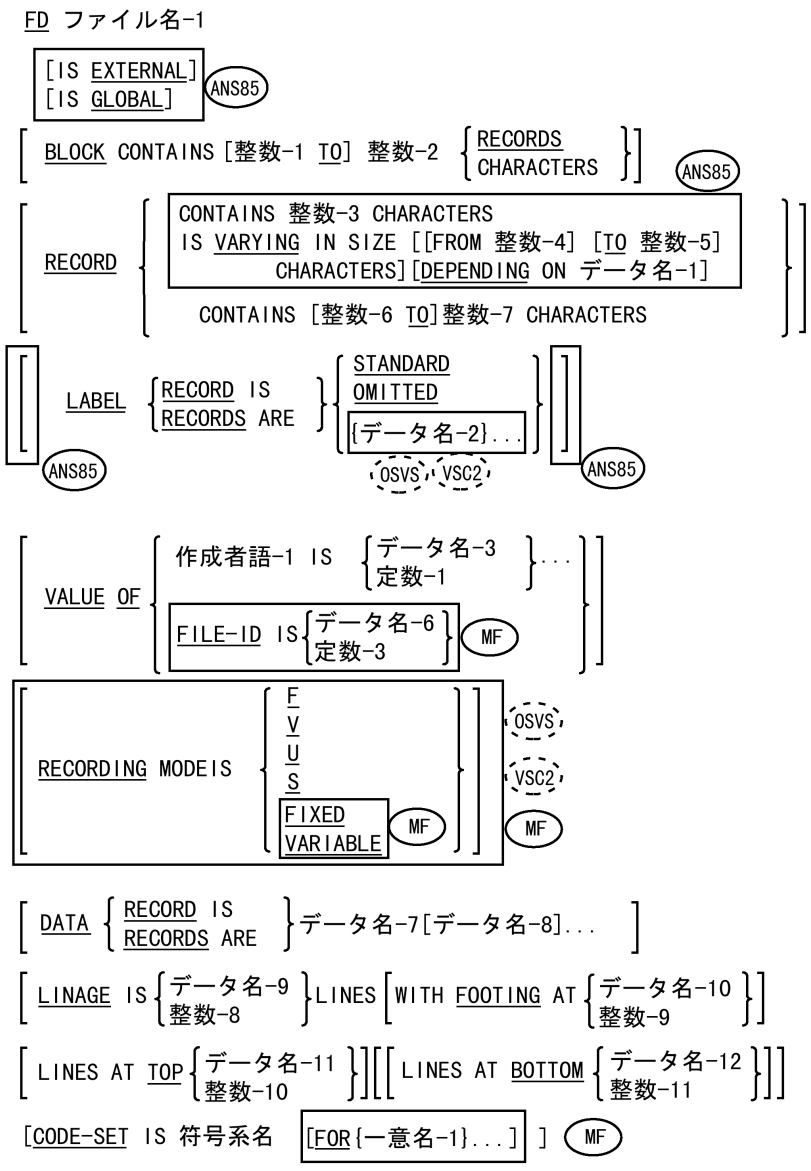


A.3 データ部の一般形式

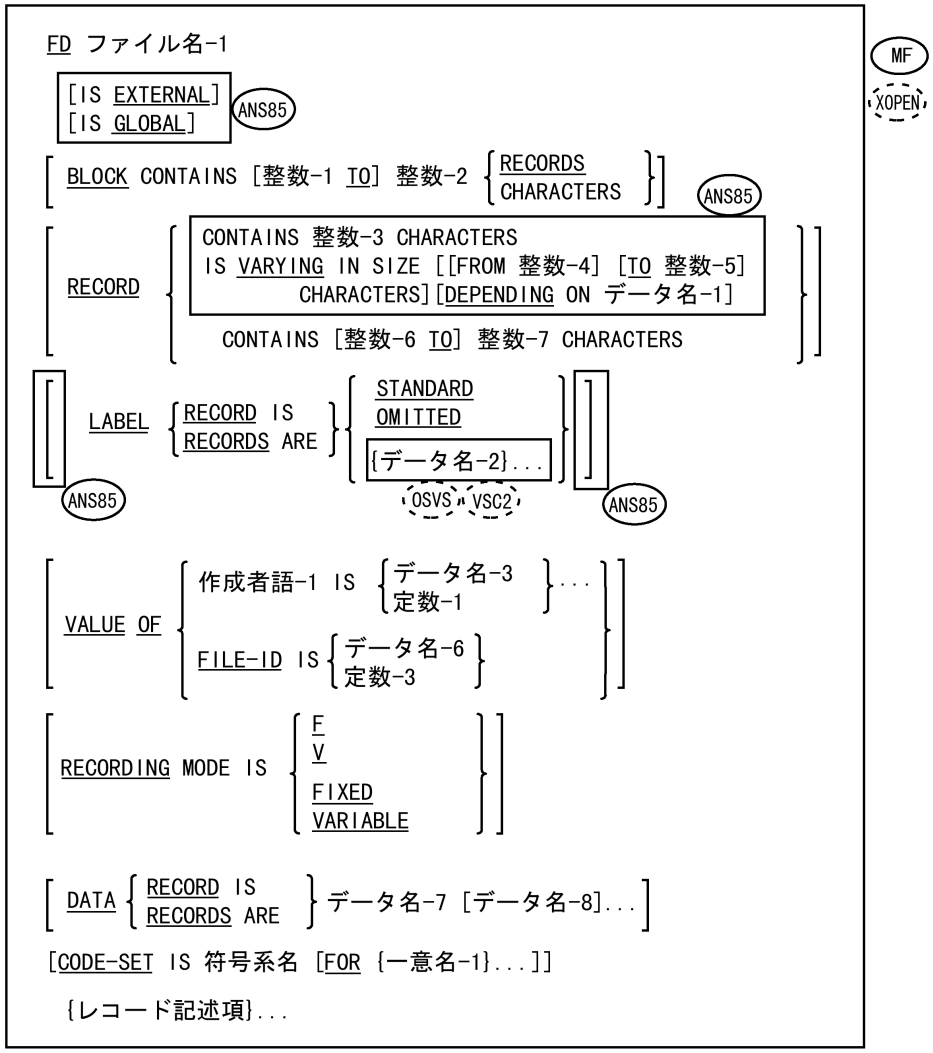


A.3.1 ファイル記述項の一般形式

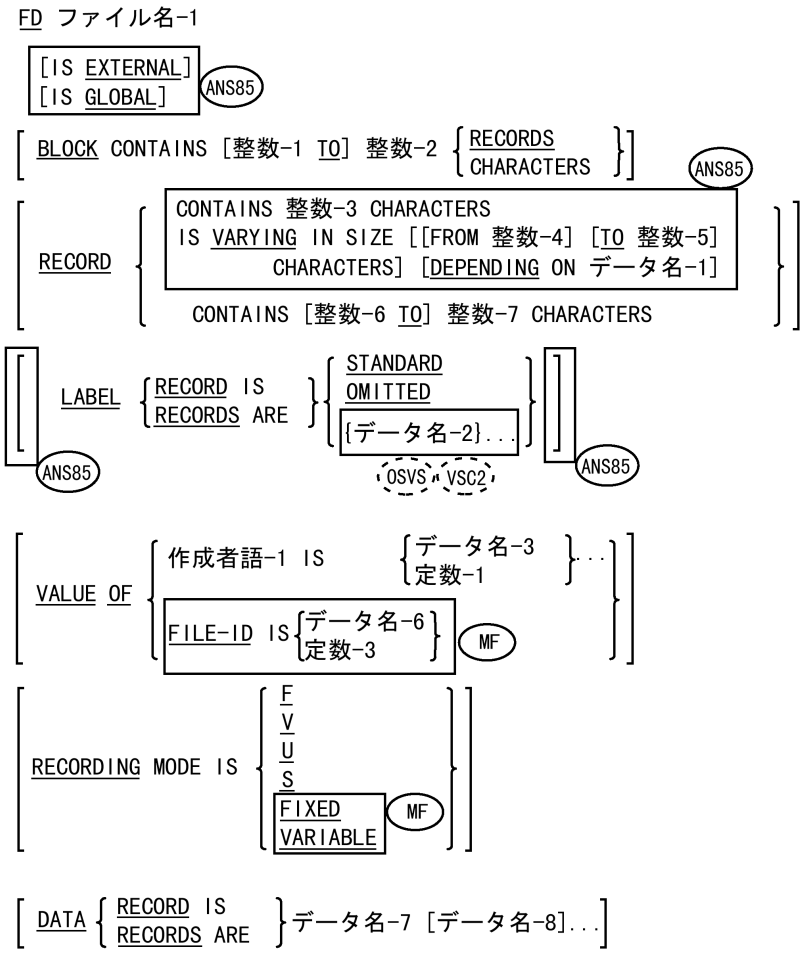
A.3.1.1 順ファイル



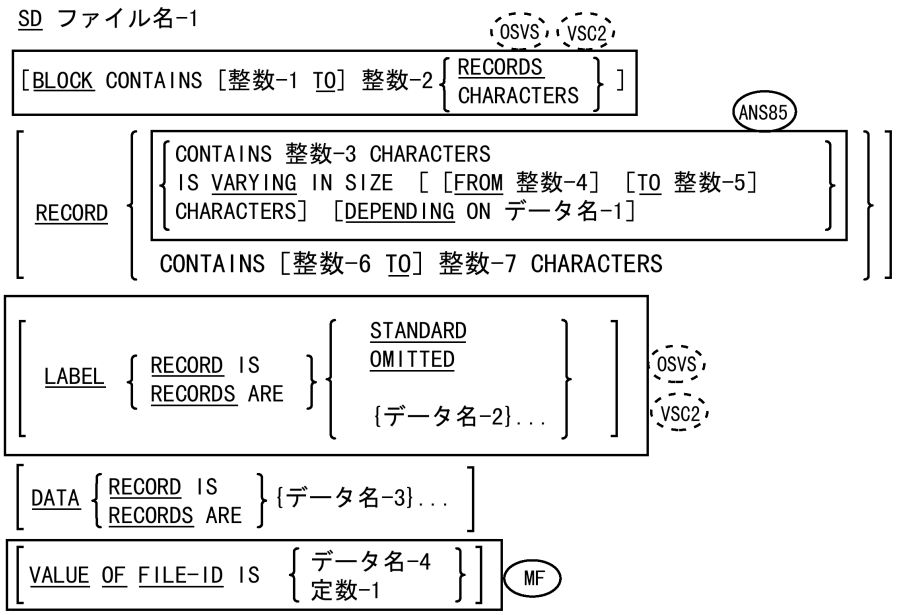
A.3.1.2 行順ファイル


FD ファイル名-1
[IS EXTERNAL]
[IS GLOBAL]
ANS85
MF
XOPEN
BLOCK CONTAINS [整数-1 TO] 整数-2 {RECORDS / CHARACTERS}
ANS85
RECORD
CONTAINS 整数-3 CHARACTERS
IS VARYING IN SIZE [[FROM 整数-4] [TO 整数-5] CHARACTERS][DEPENDING ON データ名-1]
CONTAINS [整数-6 TO] 整数-7 CHARACTERS
LABEL {RECORD IS / RECORDS ARE}
STANDARD
OMITTED
{データ名-2}...
OSVS
VSC2
ANS85
ANS85
VALUE OF
作成者語-1 IS {データ名-3 / 定数-1}...
FILE-ID IS {データ名-6 / 定数-3}
RECORDING MODE IS
F
V
FIXED
VARIABLE
DATA {RECORD IS / RECORDS ARE} データ名-7 [データ名-8]...
[CODE-SET IS 符号系名 [FOR {一意名-1}...]]
{レコード記述項}...

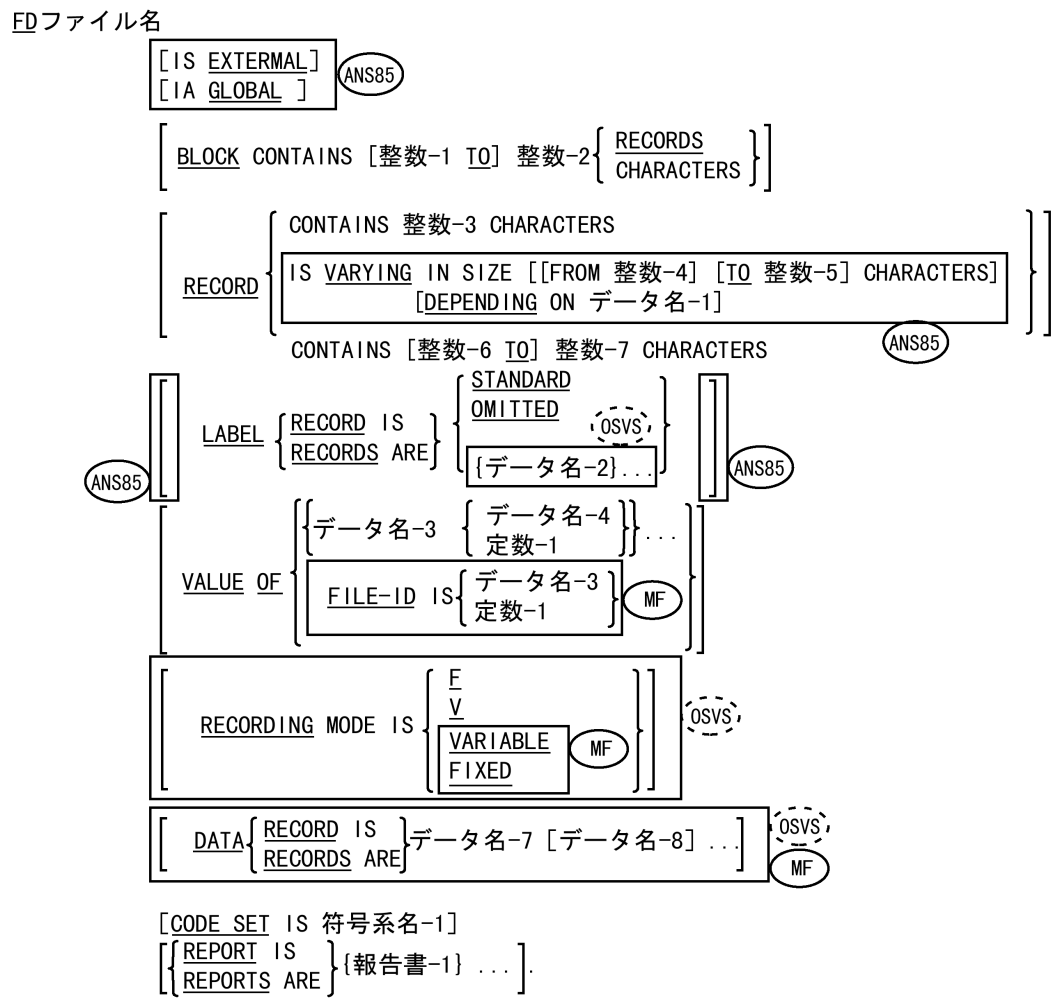
A.3.1.3 相対及び索引ファイル



A.3.1.4 整理併合ファイル

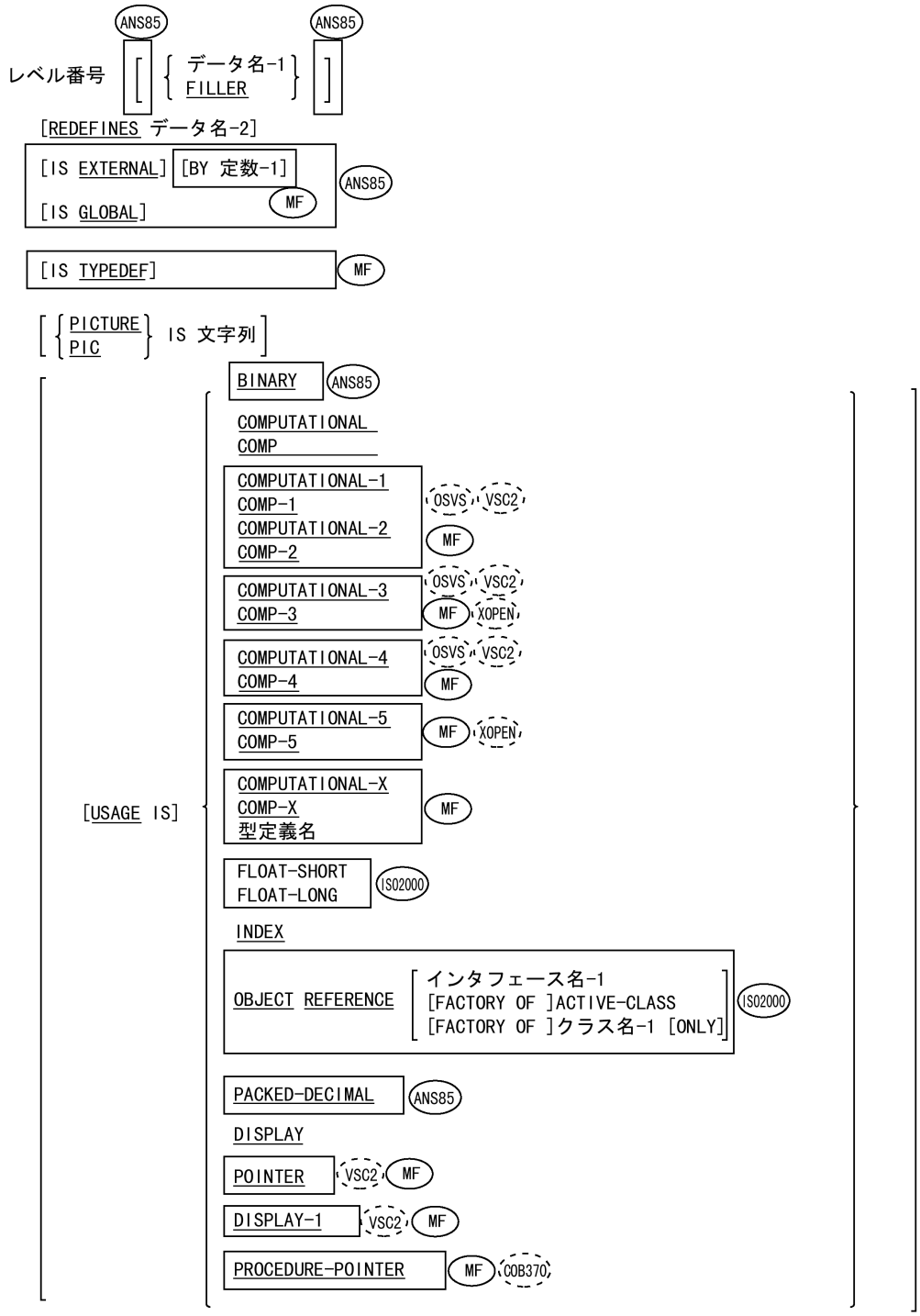


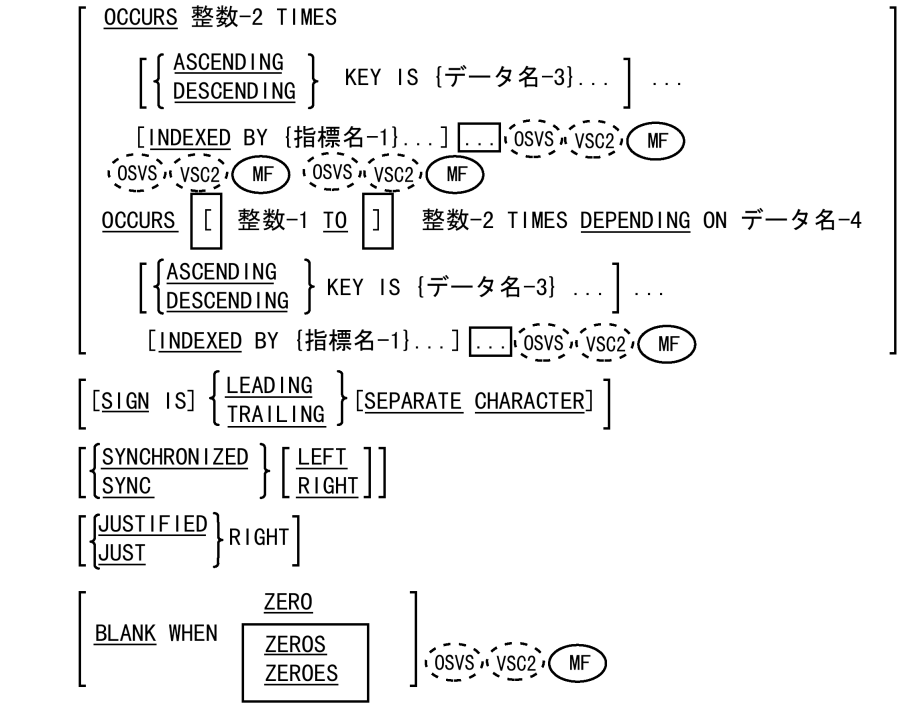
A.3.1.5 報告書ファイル



A.3.2 データ記述項の一般形式

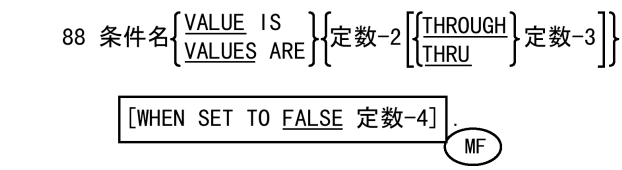
書き方 1



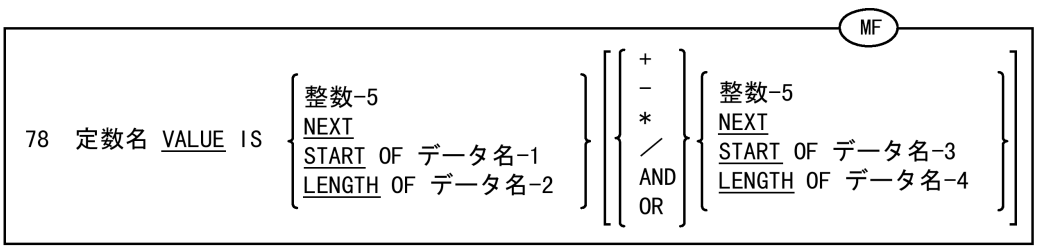


書き方 2

書き方 3



書き方 4



A.3.3 通信記述項の一般形式

書き方 1

```
CD 通信記述名
                                  [[SYNMBOLIC QUEUE IS データ名-1]
                                     [SYMBOLIC SUB-QUEUE-1IS データ名-2]
                                     [SYMBOLIC SUB-QUEUE-2IS データ名-3]
                                     [SYMBOLIC SUB-QUEUE-3IS データ名-4]
FOR[INITIAL]INPUT                    [MESSAGE DATE IS データ名-5]
                                     [MESSAGE TIME IS データ名-6]
                                     [SYMBOLIC SOURCE IS データ名-7]
                                     [TEXT LENGTH IS データ名-8]
                                     [END KEY IS データ名-9]
                                     [STATUS KEY IS データ名-10]
                                     [MESSAGE COUNT IS データ名-11]
                                  [データ名-1データ名-2...データ名-11]
```

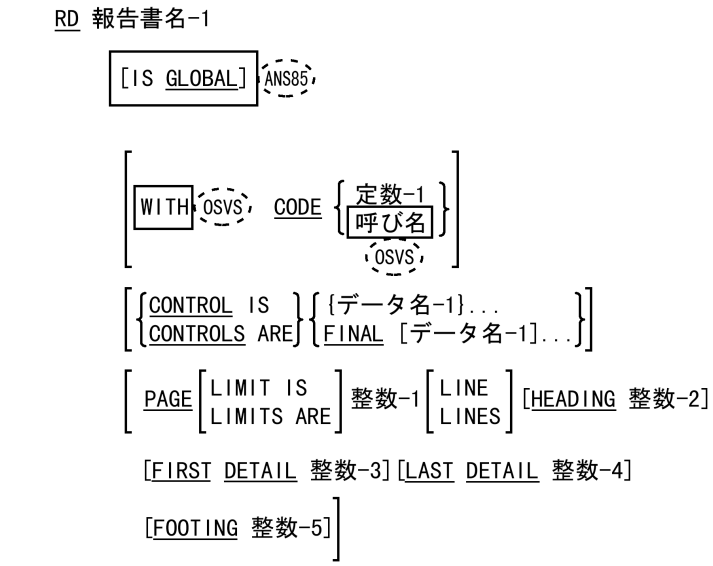
書き方 2

```
CD 通信記述名 FOR OUTPUT

[DESTINATION COUNT IS データ名-1]
      [TEXT LENGTH IS データ名-2 ]
      [STATUS KEY IS データ名-3]
      [DESTINATION TABLE OCCURS 整数-2 TIMES
          [INDEXED BY 指標名-1 [指標名-2]...] ]
[ERROR KEY IS データ名-4 ]
      [SYMBOLIC DESTINATION IS データ名-5 ].
```

書き方 3

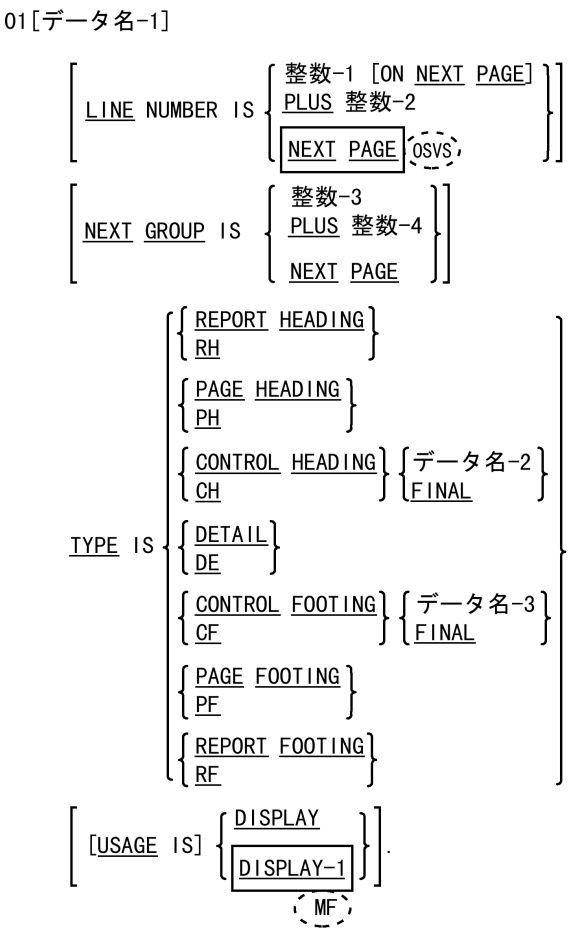
```
CD 通信記述名
                                  [MESSAGE DATE IS データ名-1]
                                  [MESSAGE TIME IS データ名-2]
                                  [SYMBOLIC TERMINAL IS データ名-3]
FOR [INITIAL]I-O                  [TEXT LENGTH IS データ名-4]          .
                                  [END KEY IS データ名-5]
                                  [STATUS KEY IS データ名-6]
                                [データ名-1 データ名-2...データ名-6]
```

A.3.4 報告書記述項の一般形式

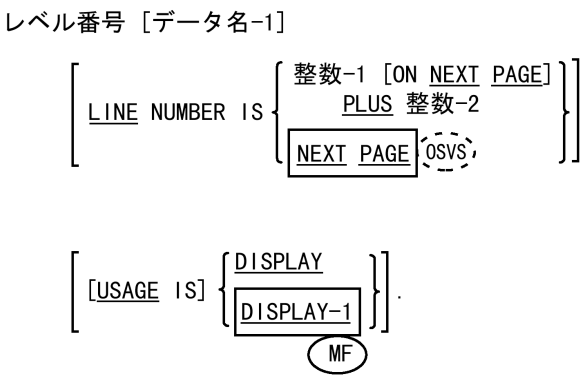


A.3.5 報告集団記述項の一般形式

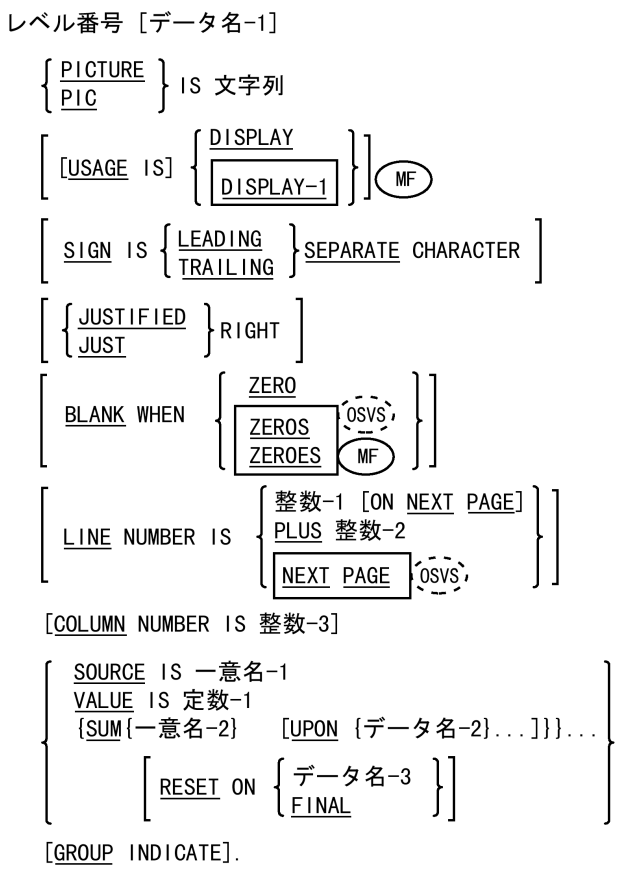
書き方 1



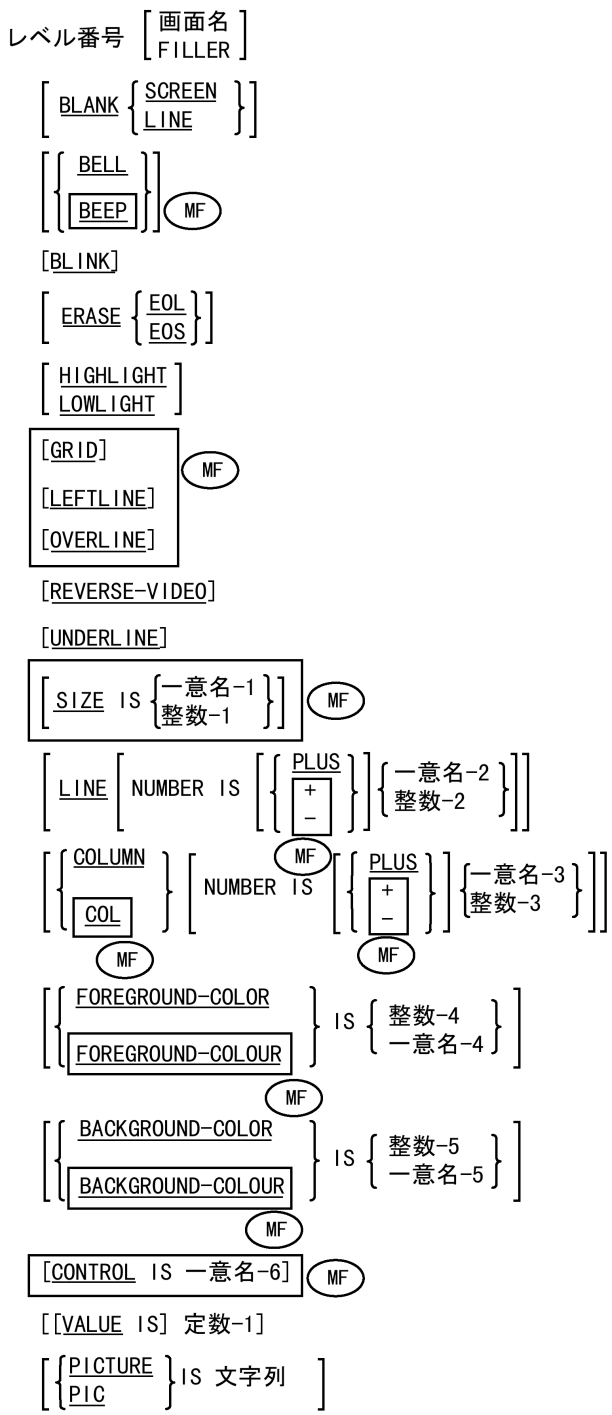
書き方 2

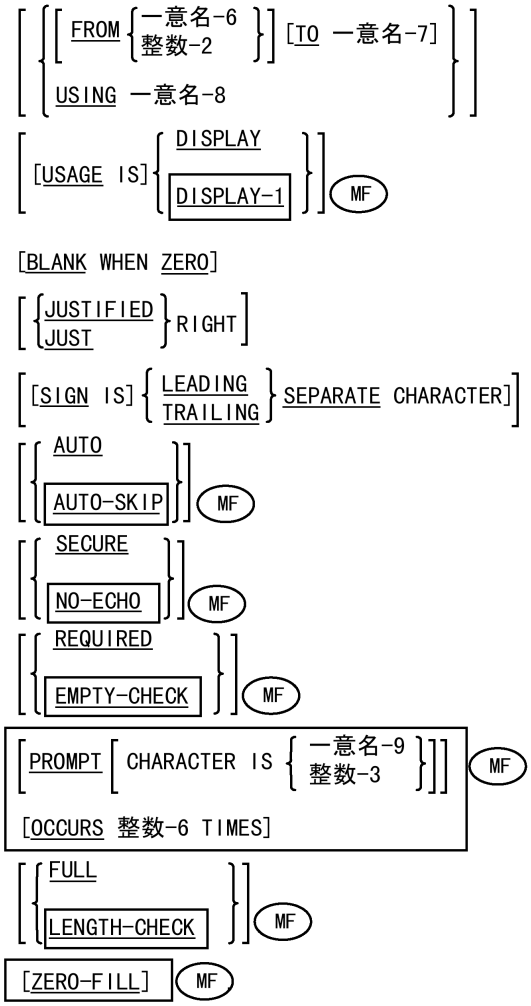


書き方 3

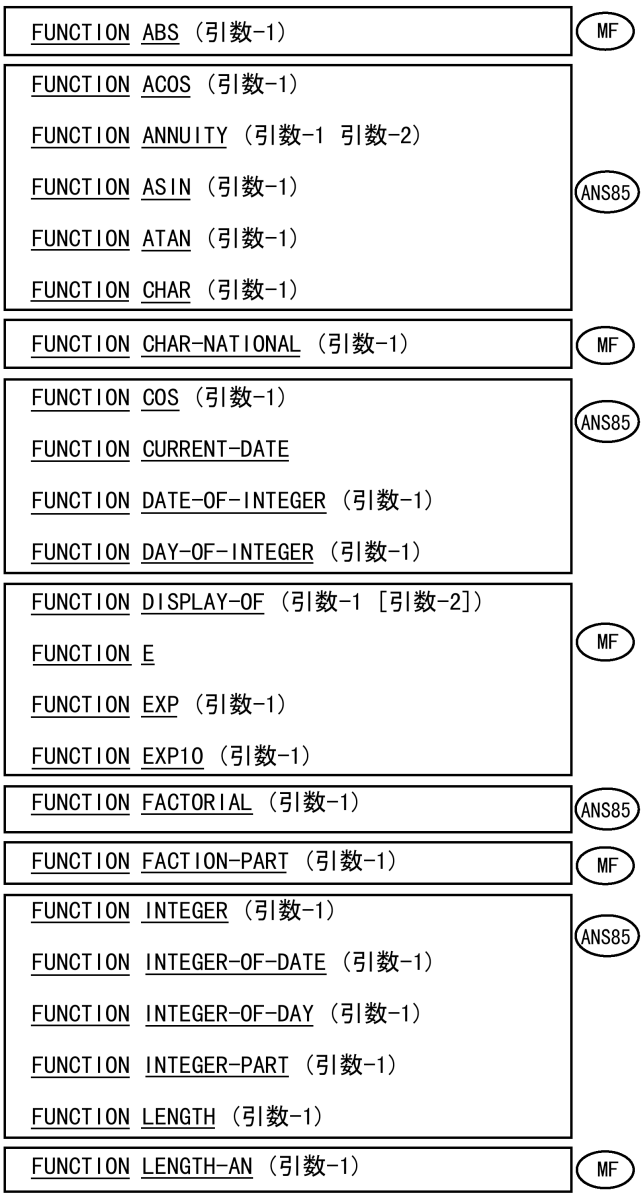

レベル番号 [データ名-1]
PICTURE
PIC
IS 文字列
[USAGE IS]
DISPLAY
DISPLAY-1
MF
SIGN IS
LEADING
TRAILING
SEPARATE CHARACTER
JUSTIFIED
JUST
RIGHT
BLANK WHEN
ZERO
ZEROS
ZEROES
OSVS
MF
LINE NUMBER IS
整数-1 [ON NEXT PAGE]
PLUS 整数-2
NEXT PAGE
OSVS
[COLUMN NUMBER IS 整数-3]
SOURCE IS 一意名-1
VALUE IS 定数-1
{SUM{一意名-2} [UPON {データ名-2}...]}...
RESET ON
データ名-3
FINAL
[GROUP INDICATE].

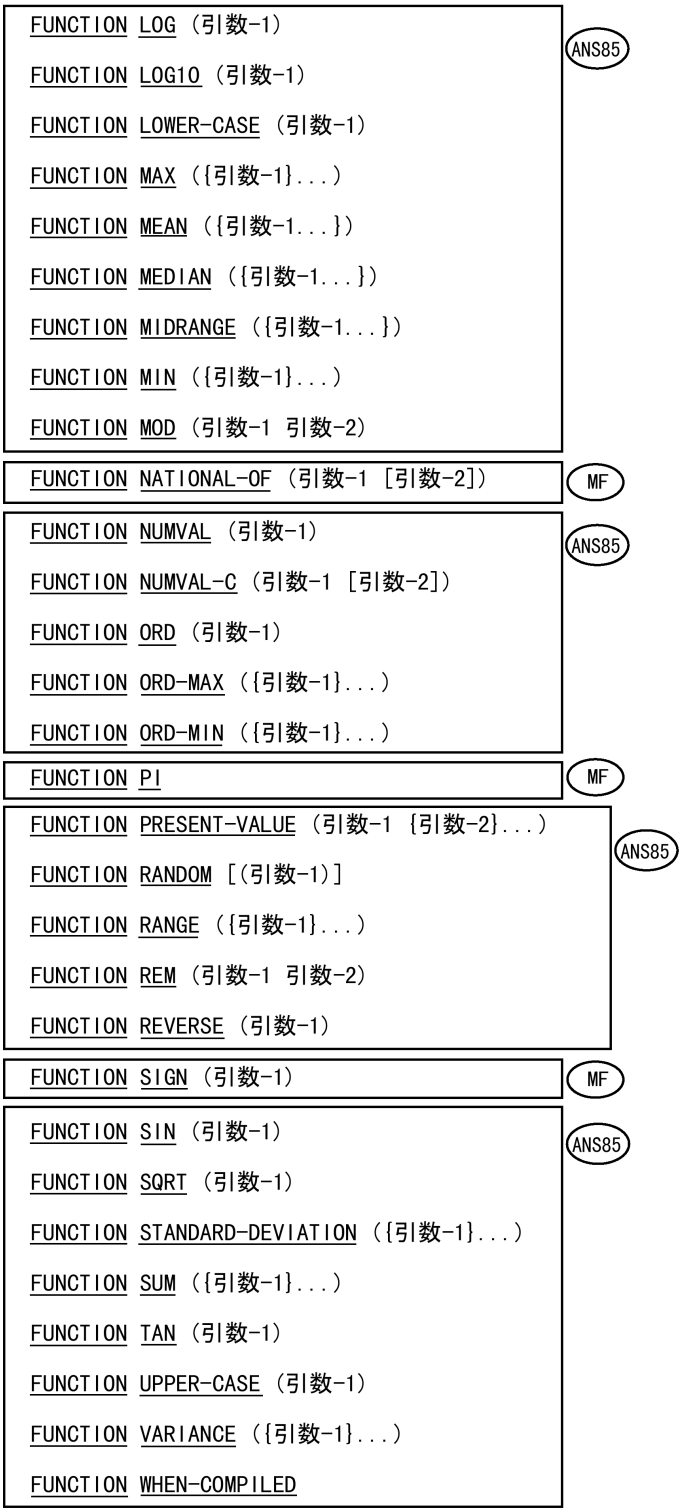
A.3.6 画面記述項の一般形式




FROM
一意名-6
整数-2
TO 一意名-7
USING 一意名-8
USAGE IS
DISPLAY
DISPLAY-1
MF
BLANK WHEN ZERO
JUSTIFIED
JUST
RIGHT
SIGN IS
LEADING
TRAILING
SEPARATE CHARACTER
AUTO
AUTO-SKIP
MF
SECURE
NO-ECHO
MF
REQUIRED
EMPTY-CHECK
MF
PROMPT
CHARACTER IS
一意名-9
整数-3
MF
OCCURS 整数-6 TIMES
FULL
LENGTH-CHECK
MF
ZERO-FILL
MF

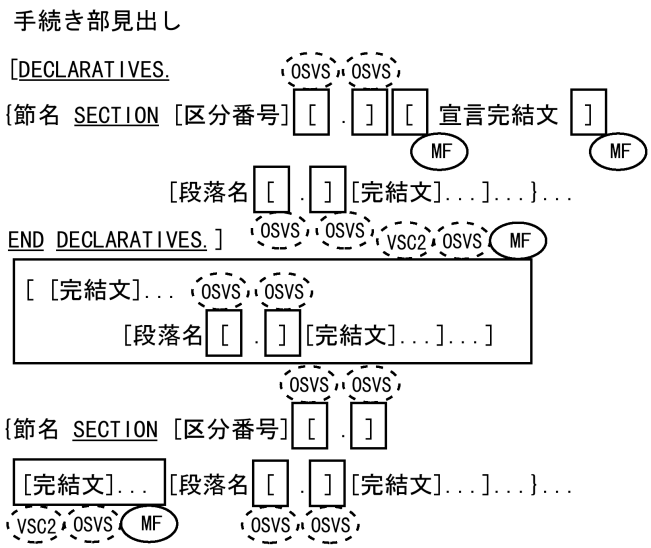
A.4 組込み関数の一般形式



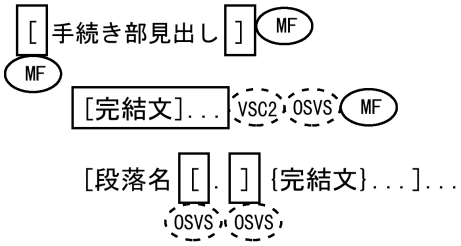

FUNCTION LOG (引数-1)
FUNCTION LOG10 (引数-1)
FUNCTION LOWER-CASE (引数-1)
FUNCTION MAX ({引数-1}...)
FUNCTION MEAN ({引数-1...})
FUNCTION MEDIAN ({引数-1...})
FUNCTION MIDRANGE ({引数-1...})
FUNCTION MIN ({引数-1}...)
FUNCTION MOD (引数-1 引数-2)
ANS85
FUNCTION NATIONAL-OF (引数-1 [引数-2])
MF
FUNCTION NUMVAL (引数-1)
FUNCTION NUMVAL-C (引数-1 [引数-2])
FUNCTION ORD (引数-1)
FUNCTION ORD-MAX ({引数-1}...)
FUNCTION ORD-MIN ({引数-1}...)
ANS85
FUNCTION PI
MF
FUNCTION PRESENT-VALUE (引数-1 {引数-2}...)
FUNCTION RANDOM [(引数-1)]
FUNCTION RANGE ({引数-1}...)
FUNCTION REM (引数-1 引数-2)
FUNCTION REVERSE (引数-1)
ANS85
FUNCTION SIGN (引数-1)
MF
FUNCTION SIN (引数-1)
FUNCTION SQRT (引数-1)
FUNCTION STANDARD-DEVIATION ({引数-1}...)
FUNCTION SUM ({引数-1}...)
FUNCTION TAN (引数-1)
FUNCTION UPPER-CASE (引数-1)
FUNCTION VARIANCE ({引数-1}...)
FUNCTION WHEN-COMPILED
ANS85

A.5 手続き部の一般形式

A.5.1 宣言部分の形式

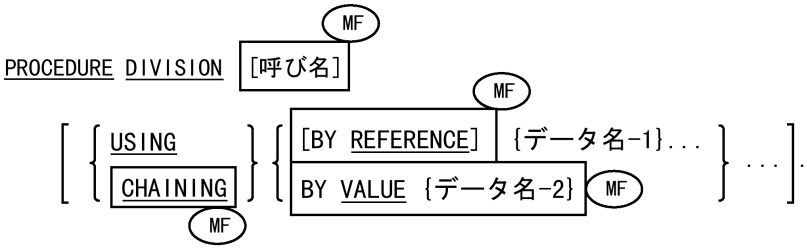


A.5.2 非宣言部分の形式

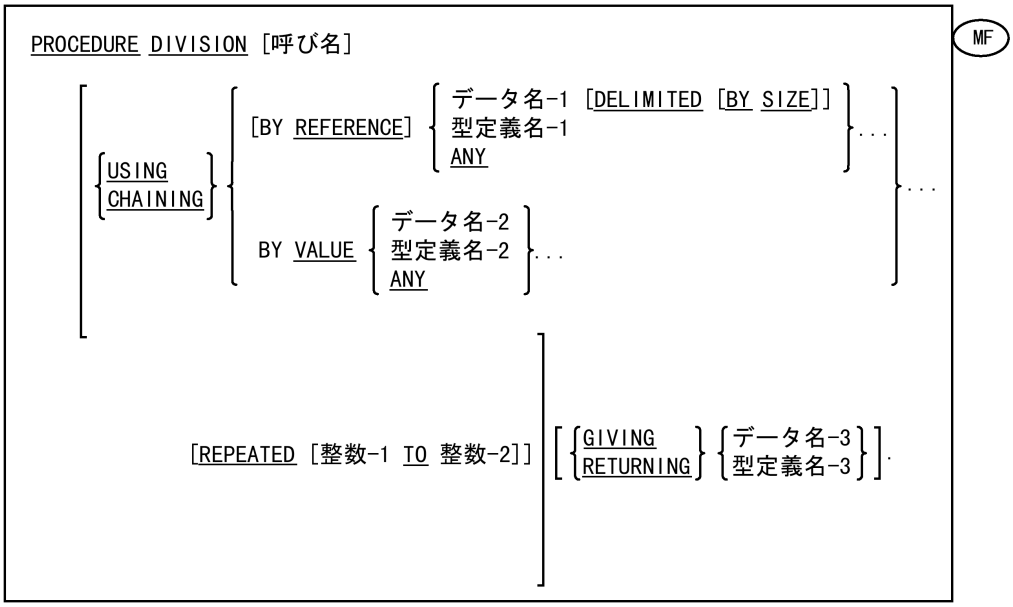


A.5.3 手続き部(PROCEDURE DIVISION)見出し

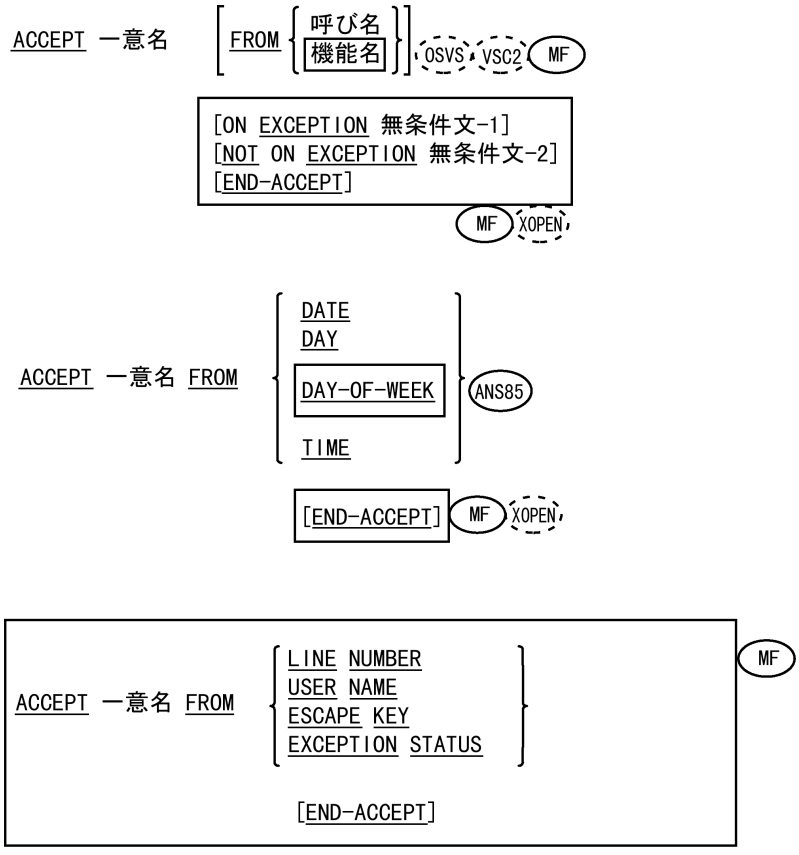
書き方 1

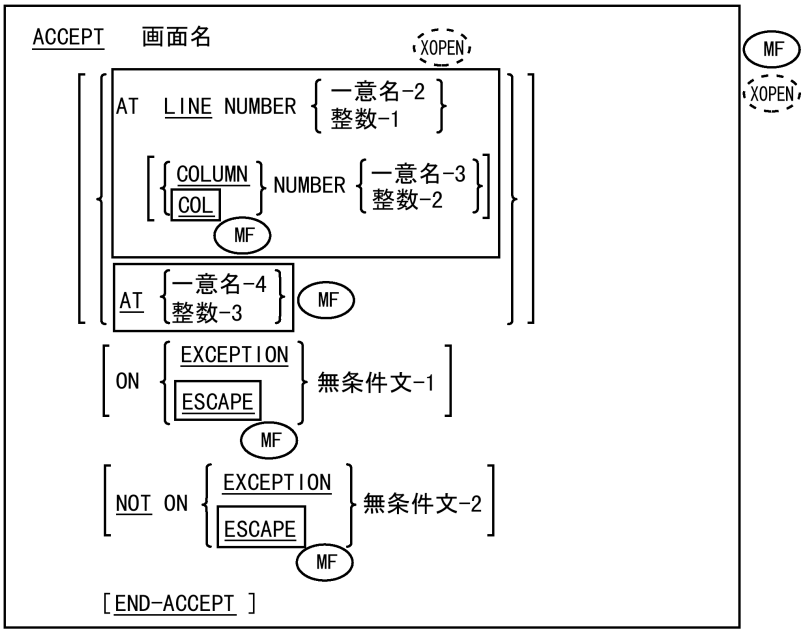


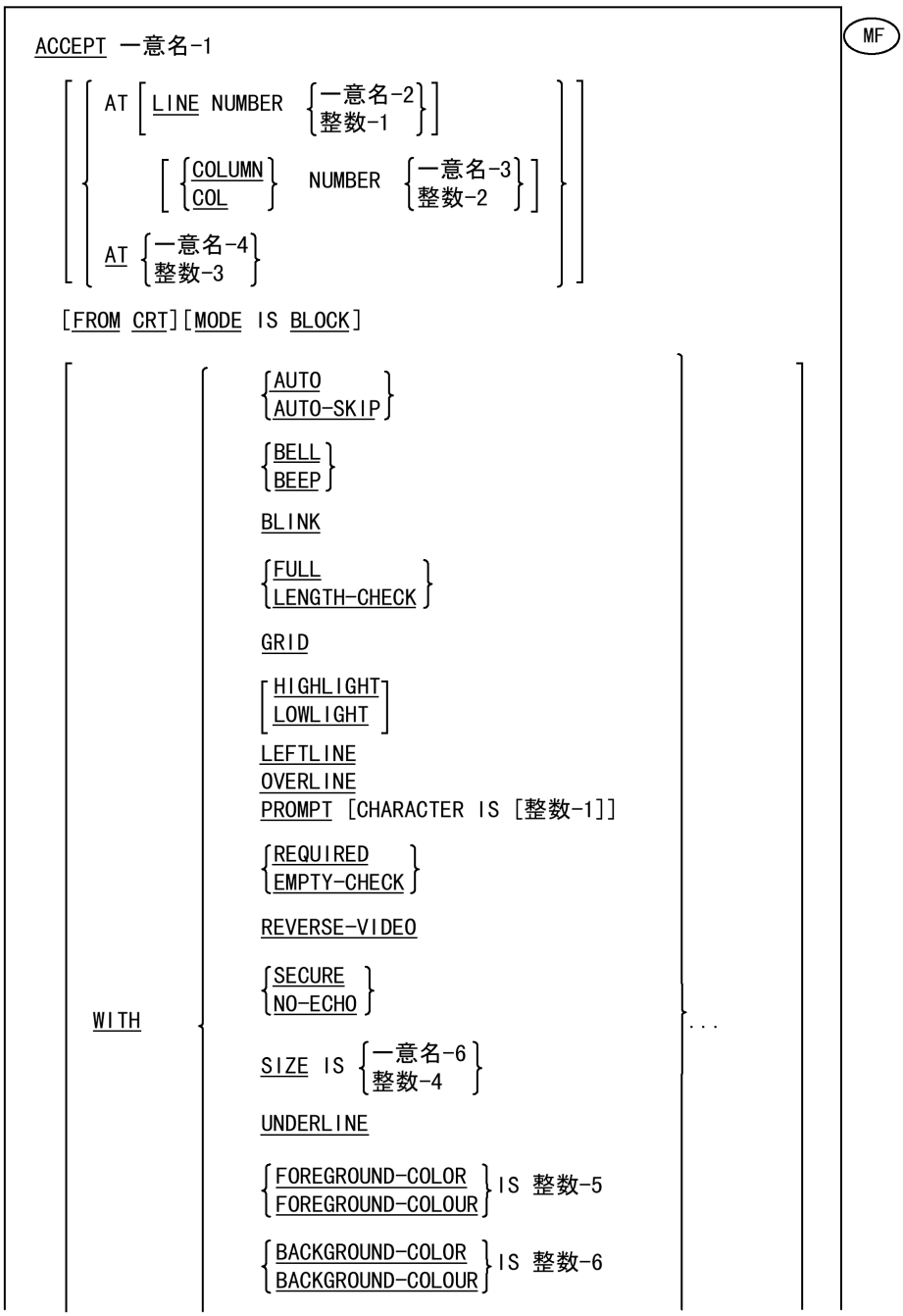
書き方 2

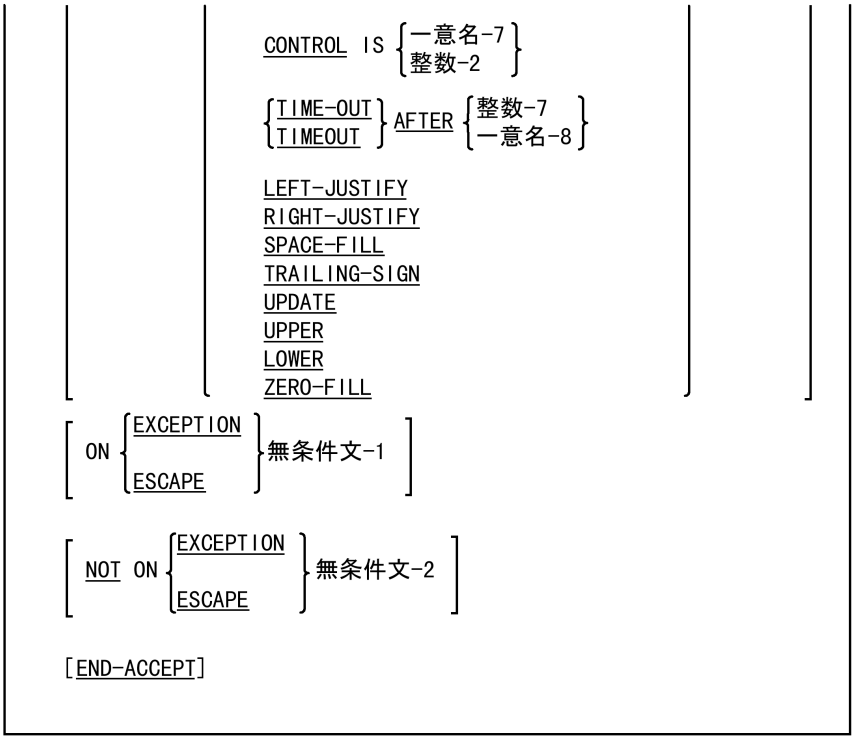

PROCEDURE DIVISION [呼び名]
MF
USING
CHAINING
[BY REFERENCE]
データ名-1 [DELIMITED [BY SIZE]]
型定義名-1
ANY
BY VALUE
データ名-2
型定義名-2
ANY
[REPEATED [整数-1 TO 整数-2]]
GIVING
RETURNING
データ名-3
型定義名-3

A.6 動詞の一般形式

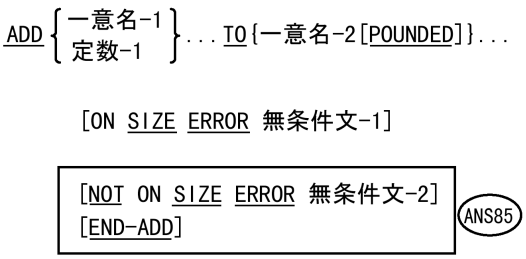

ACCEPT 一意名
FROM
呼び名
機能名
OSVS
VSC2
MF
[ON EXCEPTION 無条件文-1]
[NOT ON EXCEPTION 無条件文-2]
[END-ACCEPT]
MF
XOPEN
ACCEPT 一意名 FROM
DATE
DAY
DAY-OF-WEEK
TIME
ANS85
[END-ACCEPT]
MF
XOPEN
ACCEPT 一意名 FROM
LINE NUMBER
USER NAME
ESCAPE KEY
EXCEPTION STATUS
[END-ACCEPT]
MF

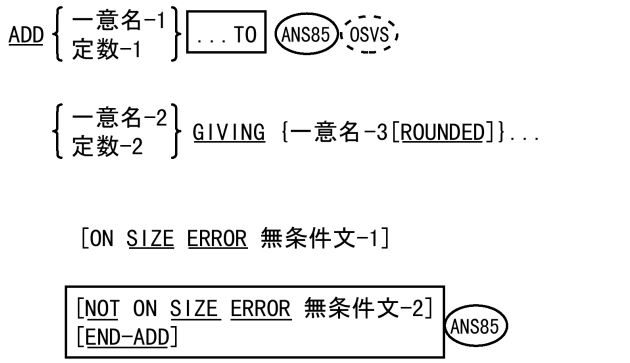

ACCEPT
画面名
XOPEN
AT LINE NUMBER
一意名-2
整数-1
COLUMN
COL
MF
NUMBER
一意名-3
整数-2
AT
一意名-4
整数-3
MF
ON
EXCEPTION
ESCAPE
MF
無条件文-1
NOT ON
EXCEPTION
ESCAPE
MF
無条件文-2
[END-ACCEPT]
MF
XOPEN


MF
ACCEPT 一意名-1
AT LINE NUMBER 一意名-2 整数-1
COLUMN COL NUMBER 一意名-3 整数-2
AT 一意名-4 整数-3
[FROM CRT][MODE IS BLOCK]
WITH
AUTO AUTO-SKIP
BELL BEEP
BLINK
FULL LENGTH-CHECK
GRID
HIGHLIGHT LOWLIGHT
LEFTLINE
OVERLINE
PROMPT [CHARACTER IS [整数-1]]
REQUIRED EMPTY-CHECK
REVERSE-VIDEO
SECURE NO-ECHO
SIZE IS 一意名-6 整数-4
UNDERLINE
FOREGROUND-COLOR FOREGROUND-COLOUR IS 整数-5
BACKGROUND-COLOR BACKGROUND-COLOUR IS 整数-6
...

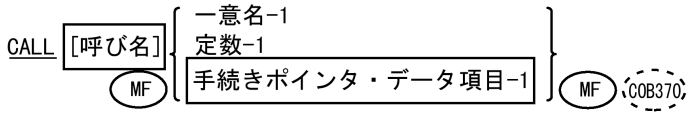


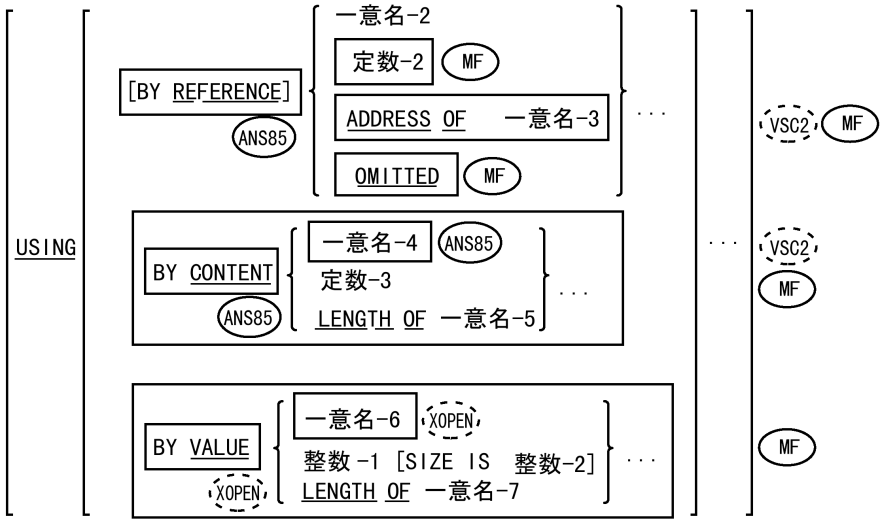
ACCEPT 通信記述名 MESSAGE COUNT



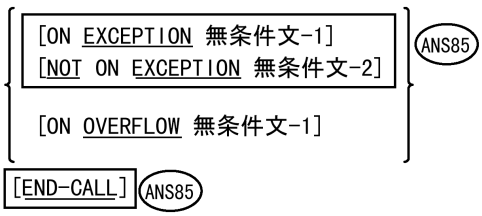


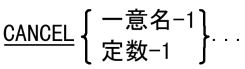
ALTER 手続き名-1 TO [PROCEED TO] 手続き名-2
[手続き名-3 TO [PROCEED TO] 手続き名-4]...

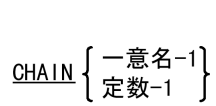

CALL
[呼び名]
MF
一意名-1
定数-1
手続きポインタ・データ項目-1
MF
COB370

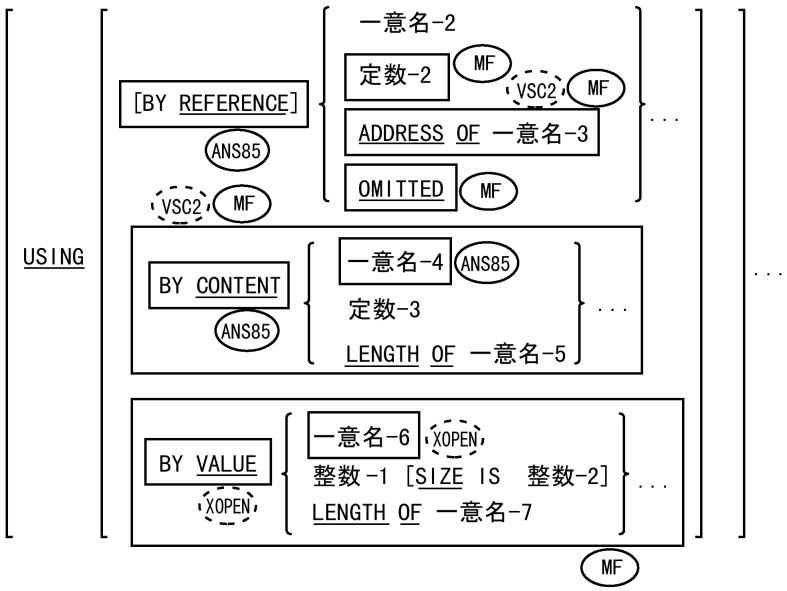

USING
[BY REFERENCE]
ANS85
一意名-2
定数-2
MF
ADDRESS OF 一意名-3
OMITTED
MF
VSC2
MF
BY CONTENT
ANS85
一意名-4
ANS85
定数-3
LENGTH OF 一意名-5
VSC2
MF
BY VALUE
XOPEN
一意名-6
XOPEN
整数-1 [SIZE IS 整数-2]
LENGTH OF 一意名-7
MF


GIVING
RETURNING
INTO 一意名-8
ADDRESS OF 一意名-9
MF

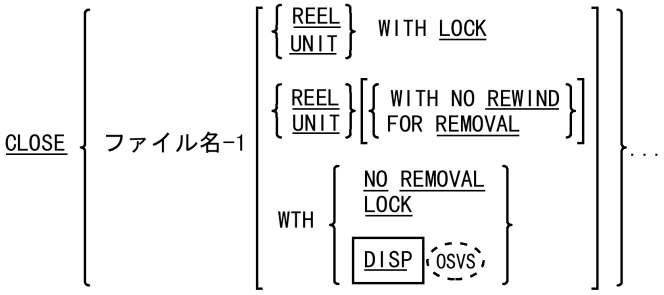

[ON EXCEPTION 無条件文-1]
[NOT ON EXCEPTION 無条件文-2]
ANS85
[ON OVERFLOW 無条件文-1]
[END-CALL]
ANS85


CANCEL
一意名-1
定数-1



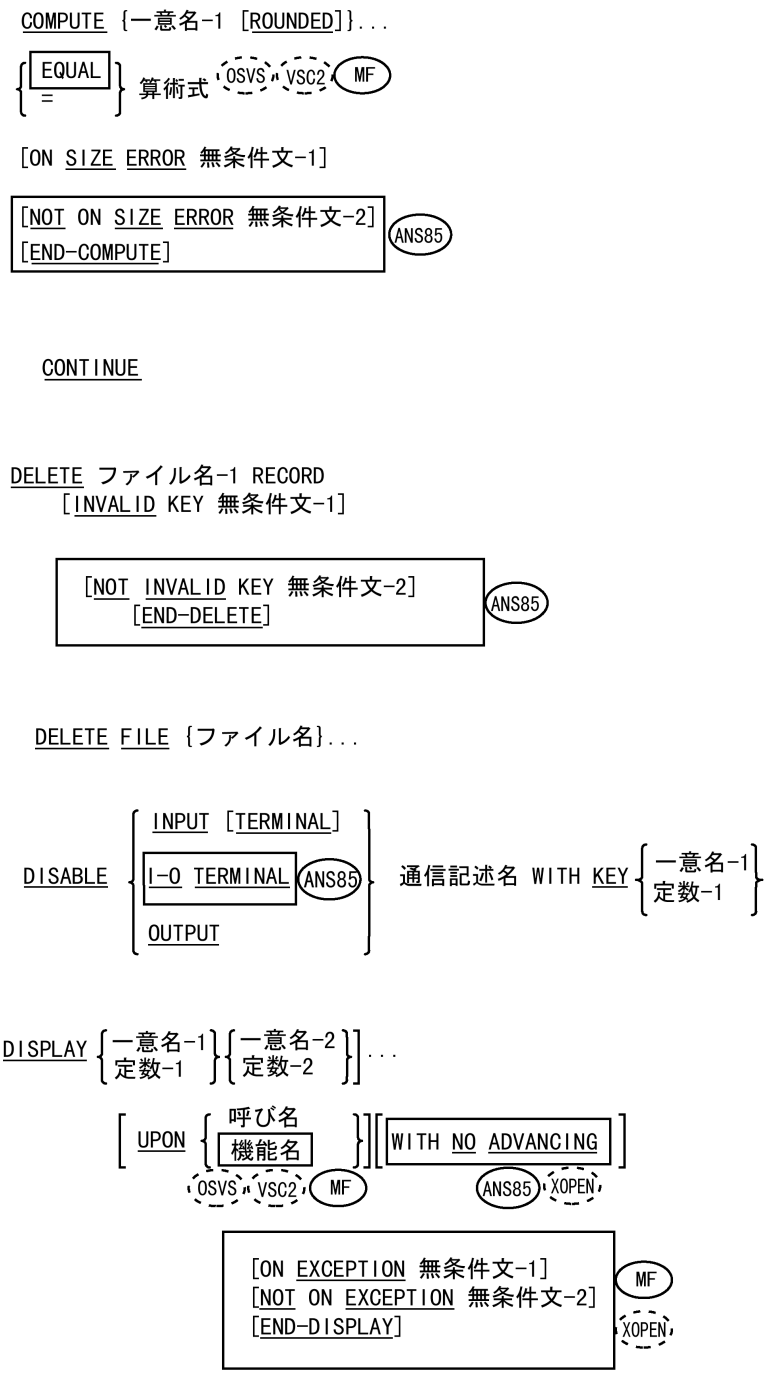


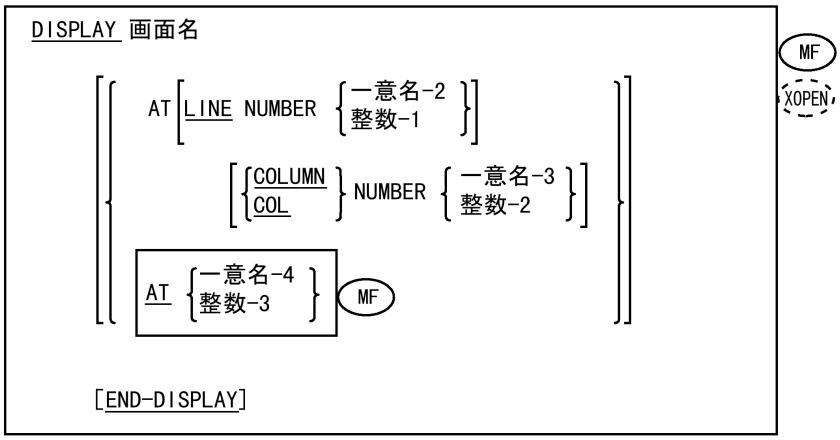
[END-CHAIN]

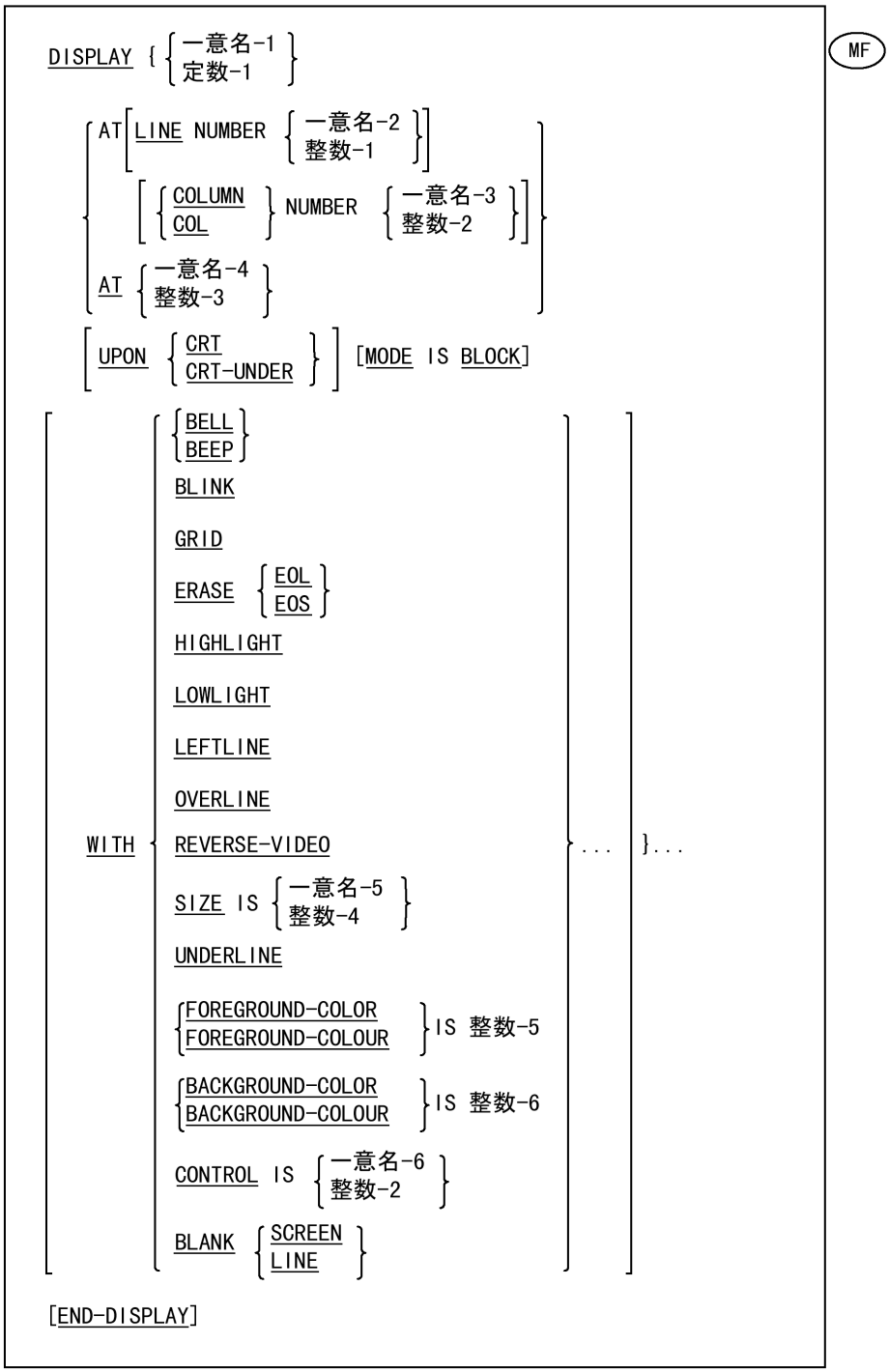


CLOSE ファイル名-1 [WITH LOCK] [ファイル名-2] [WITH LOCK]]...

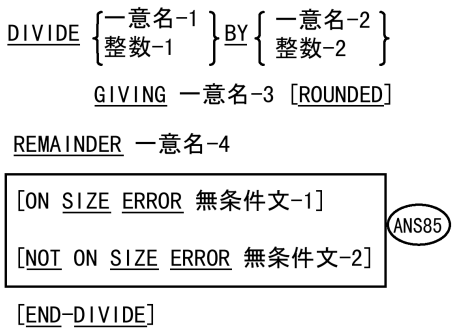
COMMIT

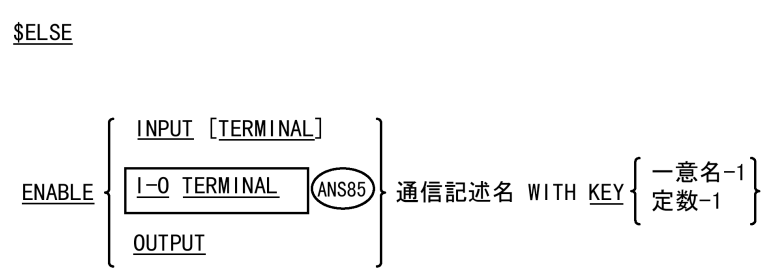

COMPUTE {一意名-1 [ROUNDED]}...
EQUAL
=
算術式
OSVS
VSC2
MF
[ON SIZE ERROR 無条件文-1]
[NOT ON SIZE ERROR 無条件文-2]
[END-COMPUTE]
ANS85
CONTINUE
DELETE ファイル名-1 RECORD
[INVALID KEY 無条件文-1]
[NOT INVALID KEY 無条件文-2]
[END-DELETE]
ANS85
DELETE FILE {ファイル名}...
DISABLE
INPUT [TERMINAL]
I-O TERMINAL
ANS85
OUTPUT
通信記述名 WITH KEY
一意名-1
定数-1
DISPLAY
一意名-1
定数-1
一意名-2
定数-2
UPON
呼び名
機能名
OSVS
VSC2
MF
WITH NO ADVANCING
ANS85
XOPEN
[ON EXCEPTION 無条件文-1]
[NOT ON EXCEPTION 無条件文-2]
[END-DISPLAY]
MF
XOPEN

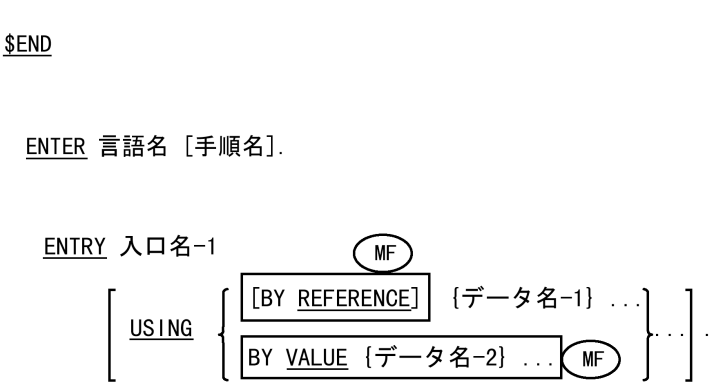

DISPLAY 画面名
AT
LINE NUMBER
一意名-2
整数-1
COLUMN
COL
NUMBER
一意名-3
整数-2
AT
一意名-4
整数-3
MF
[END-DISPLAY]
MF
XOPEN

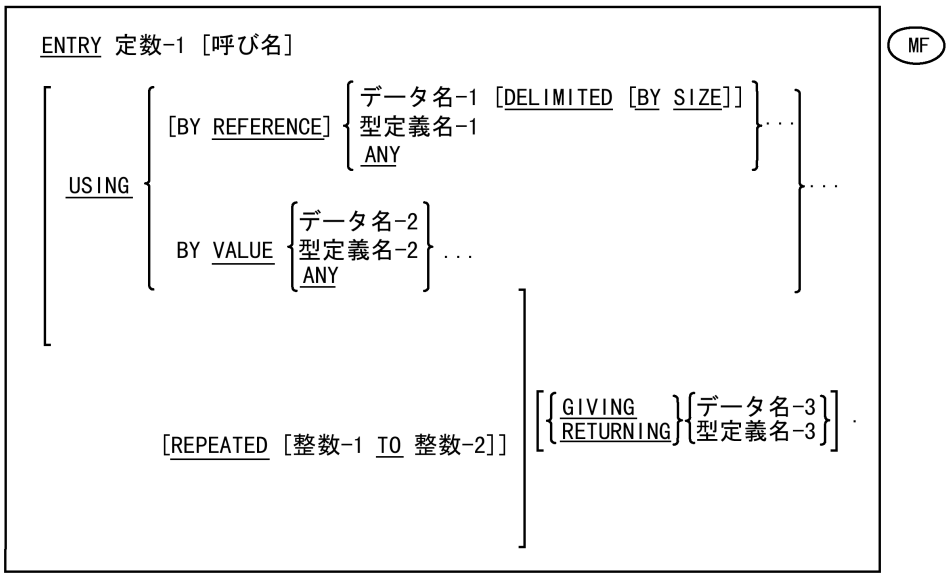

MF
DISPLAY
一意名-1
定数-1
AT
LINE NUMBER
一意名-2
整数-1
COLUMN
COL
NUMBER
一意名-3
整数-2
AT
一意名-4
整数-3
UPON
CRT
CRT-UNDER
[MODE IS BLOCK]
WITH
BELL
BEEP
BLINK
GRID
ERASE
EOL
EOS
HIGHLIGHT
LOWLIGHT
LEFTLINE
OVERLINE
REVERSE-VIDEO
SIZE IS
一意名-5
整数-4
UNDERLINE
FOREGROUND-COLOR
FOREGROUND-COLOUR
IS 整数-5
BACKGROUND-COLOR
BACKGROUND-COLOUR
IS 整数-6
CONTROL IS
一意名-6
整数-2
BLANK
SCREEN
LINE
[END-DISPLAY]

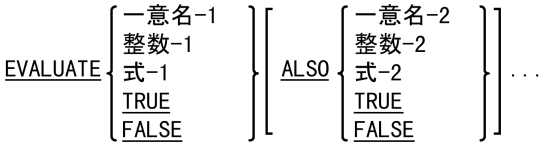
```
DIVIDE { 一意名-1 } INTO {一意名-2 [ROUNDED]} ...
       { 整数-1   }
 [ON SIZE ERROR 無条件文-1]
 [NOT ON SIZE ERROR 無条件文-2]
 [END-DIVIDE]                                    ANS85

DIVIDE { 一意名-1 } INTO { 一意名-2 }
       { 整数-1   }      { 整数-2   }
        GIVING{一意名-3 [ROUNDED]} ...
 [ON SIZE ERROR 無条件文-1]
 [NOT ON SIZE ERROR 無条件文-2]                  ANS85
 [END-DIVIDE]

DIVIDE { 一意名-1 } BY { 一意名-2 }
       { 整数-1   }    { 整数-2   }
        GIVING{一意名-3 [ROUNDED]} ...
 [ON SIZE ERROR 無条件文-1]                      ANS85
 [NOT ON SIZE ERROR 無条件文-2]
 [END-DIVIDE

DIVIDE { 一意名-1 } INTO { 一意名-2 }
       { 整数-1   }      { 整数-2   }
        GIVING 一意名-3 [ROUNDED]
  REMAINDER 一意名-4
 [ON SIZE ERROR 無条件文-1]
 [NOT ON SIZE ERROR 無条件文-2]                  ANS85
 [END-DIVIDE]
```

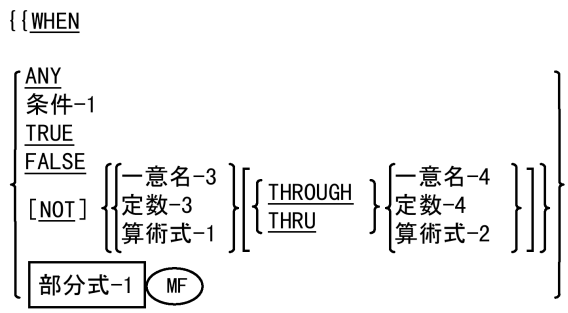

DIVIDE
一意名-1
整数-1
BY
一意名-2
整数-2
GIVING 一意名-3 [ROUNDED]
REMAINDER 一意名-4
[ON SIZE ERROR 無条件文-1]
[NOT ON SIZE ERROR 無条件文-2]
ANS85
[END-DIVIDE]

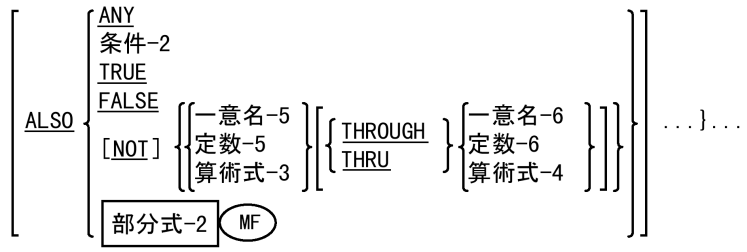

$ELSE
ENABLE
INPUT [TERMINAL]
I-O TERMINAL
ANS85
OUTPUT
通信記述名 WITH KEY
一意名-1
定数-1


$END
ENTER 言語名 [手順名].
ENTRY 入口名-1
MF
USING
[BY REFERENCE] {データ名-1} ...
BY VALUE {データ名-2} ...
MF


ENTRY 定数-1 [呼び名]
MF
USING
[BY REFERENCE]
データ名-1 [DELIMITED [BY SIZE]]
型定義名-1
ANY
BY VALUE
データ名-2
型定義名-2
ANY
[REPEATED [整数-1 TO 整数-2]]
GIVING
RETURNING
データ名-3
型定義名-3





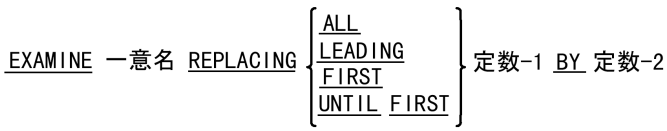


無条件文-1}...

[WHEN OTHER 無条件文-2]

[END-EVALUATE]

EXAMINE 一意名 TALLYING



EXEC [UTE] テキスト名 テキストデータ END-EXEC

```
EXHIBIT { NAMED                 } { 一意名-1 } ...
        { CHANGED [NAMED]       } { 整数-1   }
        { CHANGED               }
```

```
EXIT
```

```
EXIT METHOD
```

```
EXIT PERFORM [CYCLE]                                                    MF
```

```
EXIT { PARAGRAPH }                                                      MF
     { SECTION   }
```

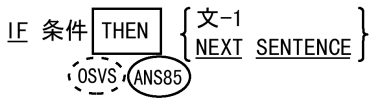
```
EXIT PROGRAM [ { GIVING    } { [ADDRESS OF] データ名-1 } ]
             { RETURNING } { 定数-1                  }
                                                                        MF
```

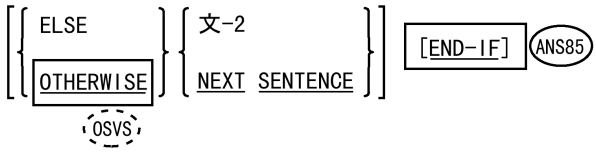
```
GENERATE { データ名-1 }
         { 報告書名-1 }
```

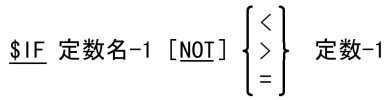
```
GOBACK [ { GIVING    } { [ADDRESS OF] データ名-1 } ]                    MF
       { RETURNING } { 定数-1                  }
```

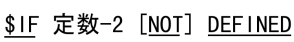
```
GO TO [手続き名-1]
```

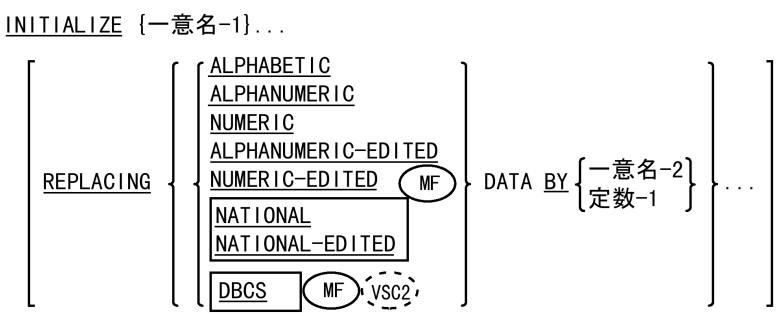
```
GO TO 手続き名-1 [手続き名-2]...DEPENDING ON 一意名
```

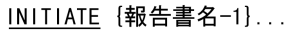

IF 条件
THEN
文-1
NEXT SENTENCE
OSVS
ANS85

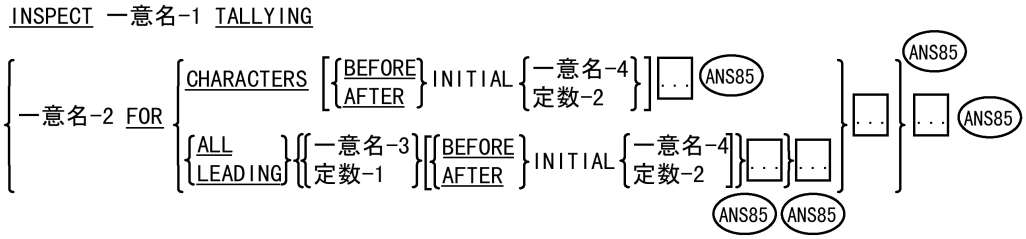

ELSE
OTHERWISE
文-2
NEXT SENTENCE
[END-IF]
ANS85
OSVS


$IF 定数名-1 [NOT]
<
>
=
定数-1


$IF 定数-2 [NOT] DEFINED


INITIALIZE {一意名-1}...
REPLACING
ALPHABETIC
ALPHANUMERIC
NUMERIC
ALPHANUMERIC-EDITED
NUMERIC-EDITED
MF
NATIONAL
NATIONAL-EDITED
DBCS
MF
VSC2
DATA BY
一意名-2
定数-1
...

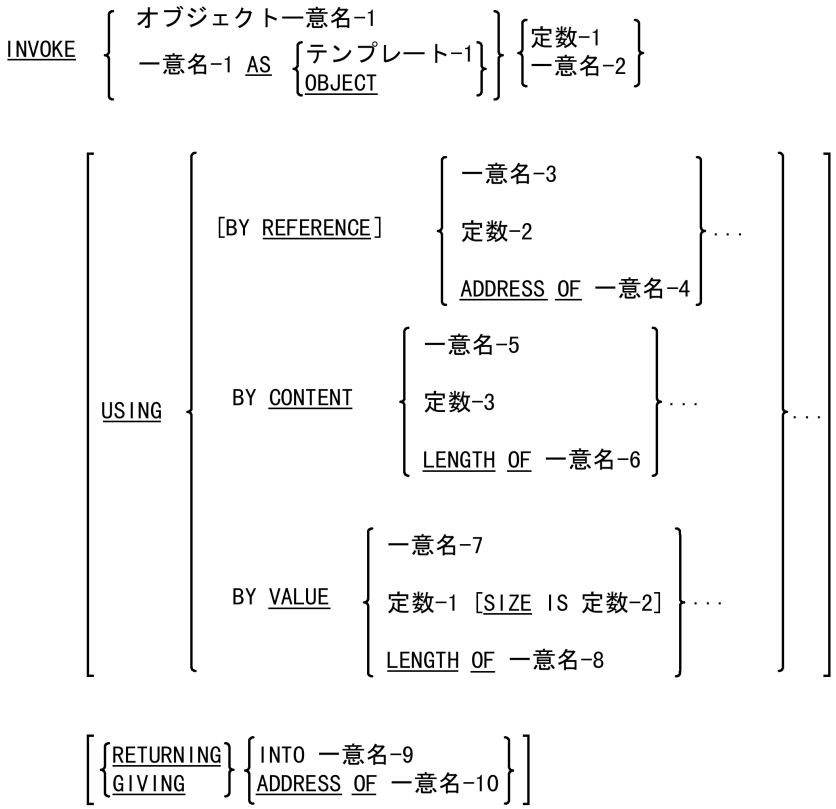

INITIATE {報告書名-1}...

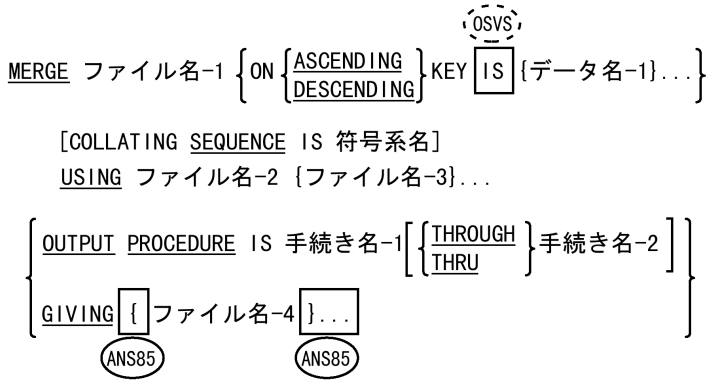

INSPECT 一意名-1 TALLYING
一意名-2 FOR
CHARACTERS
BEFORE
AFTER
INITIAL
一意名-4
定数-2
ANS85
ALL
LEADING
一意名-3
定数-1
ANS85
ANS85

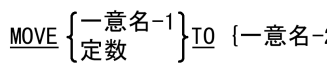
```
INSPECT 一意名-1 REPLACING

{ CHARACTERS BY {一意名-5 | 定数-3} [ {BEFORE | AFTER} INITIAL {一意名-4 | 定数-2} ] ... (ANS85)
{ {ALL | LEADING | FIRST} { {一意名-3 | 定数-1} BY {一意名-5 | 定数-3} [ {BEFORE | AFTER} INITIAL {一意名-4 | 定数-2} ] ... (ANS85) } ... (ANS85) } ... (ANS85)
```

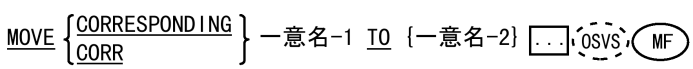
```
INSPECT 一意名-1 TALLYING

{ 一意名-2 FOR { CHARACTERS [ {BEFORE | AFTER} INITIAL {一意名-4 | 定数-2} ] ... (ANS85)
               { {ALL | LEADING} { {一意名-3 | 定数-1} [ {BEFORE | AFTER} INITIAL {一意名-4 | 定数-2} ] ... (ANS85) } ... (ANS85) } ... (ANS85) } ... (ANS85)

REPLACING

{ CHARACTERS BY {一意名-5 | 定数-3} [ {BEFORE | AFTER} INITIAL {一意名-4 | 定数-2} ] ... (ANS85)
{ {ALL | LEADING | FIRST} { {一意名-3 | 定数-1} BY {一意名-5 | 定数-3} [ {BEFORE | AFTER} INITIAL {一意名-4 | 定数-2} ] ... (ANS85) } ... (ANS85) } ... (ANS85)
```

```
INSPECT 一意名-1 CONVERTING {一意名-6 | 定数-4} TO {一意名-7 | 定数-5}        (ANS85)
    [ {BEFORE | AFTER} INITIAL {一意名-4 | 定数-5} ]
```


INVOKE
オブジェクト一意名-1
一意名-1 AS
テンプレート-1
OBJECT
定数-1
一意名-2
USING
[BY REFERENCE]
一意名-3
定数-2
ADDRESS OF 一意名-4
BY CONTENT
一意名-5
定数-3
LENGTH OF 一意名-6
BY VALUE
一意名-7
定数-1 [SIZE IS 定数-2]
LENGTH OF 一意名-8
RETURNING
GIVING
INTO 一意名-9
ADDRESS OF 一意名-10


MERGE ファイル名-1 ON ASCENDING DESCENDING KEY IS {データ名-1}... ...
OSVS
[COLLATING SEQUENCE IS 符号系名]
USING ファイル名-2 {ファイル名-3}...
OUTPUT PROCEDURE IS 手続き名-1 THROUGH THRU 手続き名-2
GIVING { ファイル名-4 }...
ANS85
ANS85


MOVE 一意名-1 定数 TO {一意名-2} ...


MOVE CORRESPONDING CORR 一意名-1 TO {一意名-2} ...
OSVS
MF

```
MULTIPLY {一意名-1} BY {一意名-2 [ROUNDED]} ...
         {定数-1  }

    [ON SIZE ERROR 無条件文-1]
   ┌──────────────────────────────┐
   │[NOT ON SIZE ERROR 無条件文-2]│ (ANS85)
   │[END-MULTIPLY]                │
   └──────────────────────────────┘
```

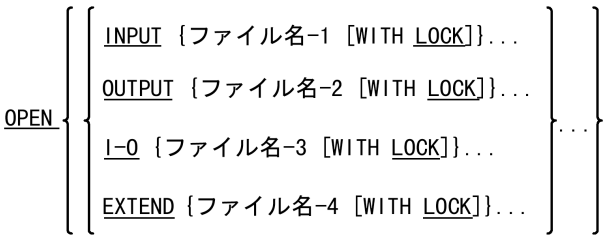
```
MULTIPLY {一意名-1} BY {一意名-2}
         {定数-1  }    {定数-2  }

                     GIVING {一意名-3 [ROUNDED]} ...

    [ON SIZE ERROR 無条件文-1]
   ┌──────────────────────────┐
   │[NOT ON SIZE ERROR 無条件-2]│ (ANS85)
   │[END-MULTIPLY]            │
   └──────────────────────────┘
```

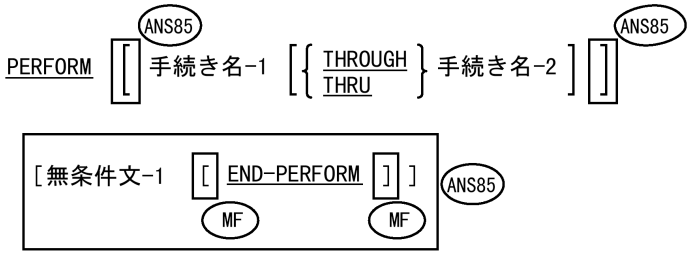
```
NEXT SENTENCE
```

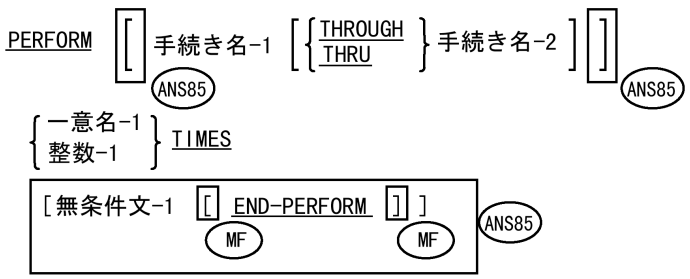
```
NOTE 文字列
```

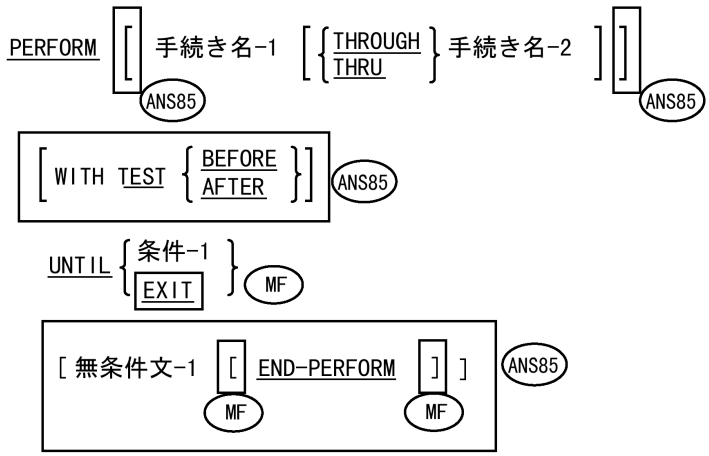
```
ON {定数-1  } [AND EVERY {定数-2  }]
   {一意名-1}            {一意名-2}

        [UNTIL {定数-3  }] {無条件文-1   }
               {一意名-3}  {NEXT SENTENCE}

              [{ELSE     } {無条件文-2   }]
              [{OTHERWISE} {NEXT SENTENCE}]
```

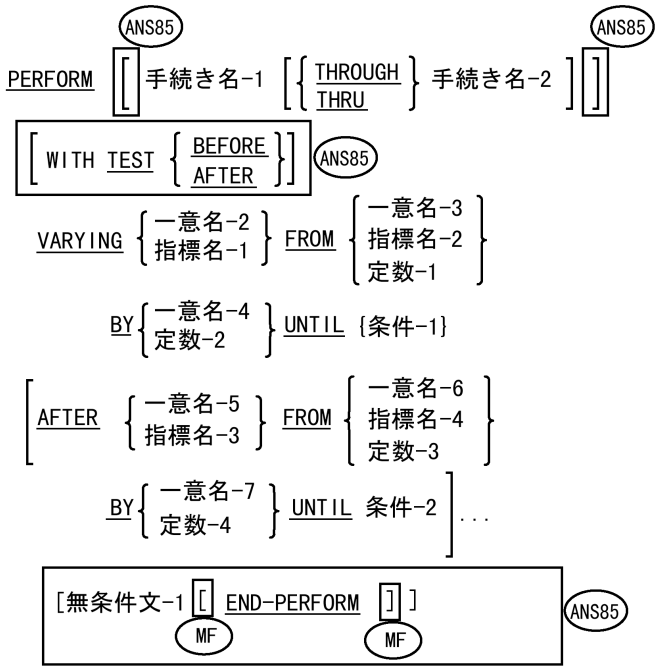
```
     { INPUT  {ファイル名-1 [REVERSED              ]} ...  }
     {                      [WITH {NO REWIND}      ]        }
     {                      [     {LOCK     }      ]        }
OPEN { OUTPUT {ファイル名-2 [WITH {NO REWIND}]} ...        } ...
     {                      [     {LOCK     }]              }
     { I-O {ファイル名-3 [WITH LOCK]}...                    }
     { EXTEND {ファイル名-4 [WITH LOCK]}...                 }
```


OPEN
INPUT {ファイル名-1 [WITH LOCK]}...
OUTPUT {ファイル名-2 [WITH LOCK]}...
I-O {ファイル名-3 [WITH LOCK]}...
EXTEND {ファイル名-4 [WITH LOCK]}...
...

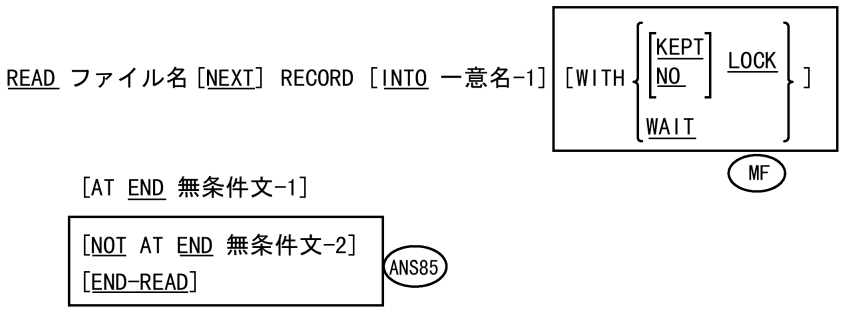

PERFORM
ANS85
手続き名-1
THROUGH
THRU
手続き名-2
ANS85
[無条件文-1
END-PERFORM
MF
MF
]
ANS85

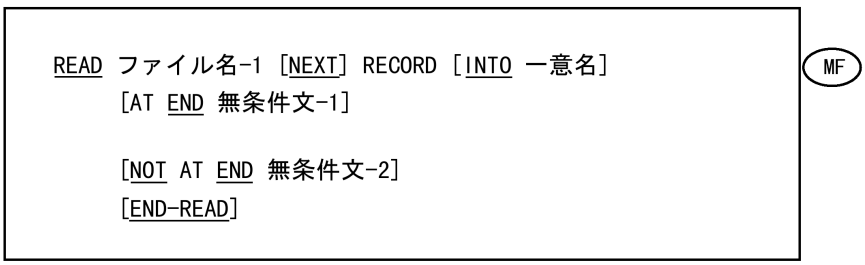

PERFORM
ANS85
手続き名-1
THROUGH
THRU
手続き名-2
ANS85
一意名-1
整数-1
TIMES
[無条件文-1
END-PERFORM
MF
MF
]
ANS85

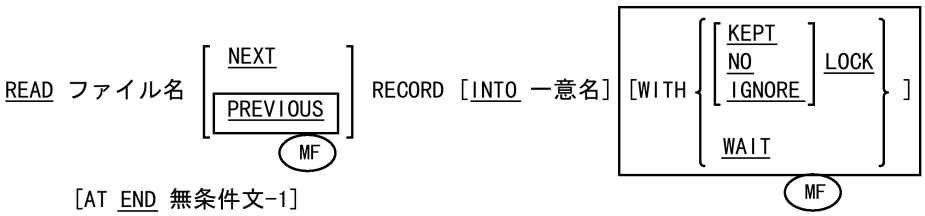

PERFORM
ANS85
手続き名-1
THROUGH
THRU
手続き名-2
ANS85
WITH TEST
BEFORE
AFTER
ANS85
UNTIL
条件-1
EXIT
MF
[無条件文-1
END-PERFORM
MF
MF
]
ANS85


ANS85
PERFORM
手続き名-1
THROUGH
THRU
手続き名-2
ANS85
WITH TEST
BEFORE
AFTER
ANS85
VARYING
一意名-2
指標名-1
FROM
一意名-3
指標名-2
定数-1
BY
一意名-4
定数-2
UNTIL {条件-1}
AFTER
一意名-5
指標名-3
FROM
一意名-6
指標名-4
定数-3
BY
一意名-7
定数-4
UNTIL 条件-2
...
[無条件文-1
END-PERFORM
]
MF
MF
ANS85

PURGE 通信記述名

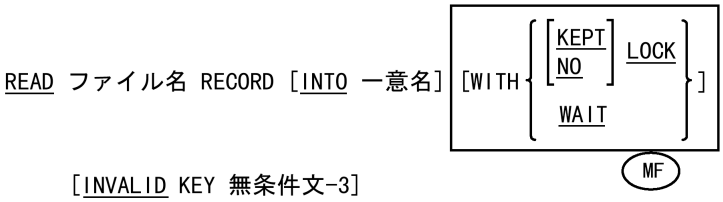

READ ファイル名 [NEXT] RECORD [INTO 一意名-1]
[WITH
KEPT
NO
LOCK
WAIT
]
MF
[AT END 無条件文-1]
[NOT AT END 無条件文-2]
[END-READ]
ANS85


READ ファイル名-1 [NEXT] RECORD [INTO 一意名]
[AT END 無条件文-1]
[NOT AT END 無条件文-2]
[END-READ]
MF



[AT END 無条件文-1]

[NOT AT END 無条件文-2]
[END-READ]
ANS85



[INVALID KEY 無条件文-3]

[NOT INVALID KEY 無条件文-4]
[END-READ]
ANS85

[INVALID KEY 無条件文-3]

[NOT INVALID KEY 無条件文-4]
[END-READ]
ANS85

READY TRACE.

RECEIVE 通信記述名{MESSAGE / SEGMENT} INTO 一意名-1

[NO DATA 無条件文-1]

[WITH DATA 無条件文-2]
[END-RECEIVE]
ANS85

```
RELEASE レコード名 [FROM 一意名]
```

```
RESET TRACE.
```

```
RETURN ファイル名 RECORD [INTO 一意名-1]
       AT END 無条件文-1

    [NOT AT END  無条件文-2]
    [END-RETURN]                    ANS85
```

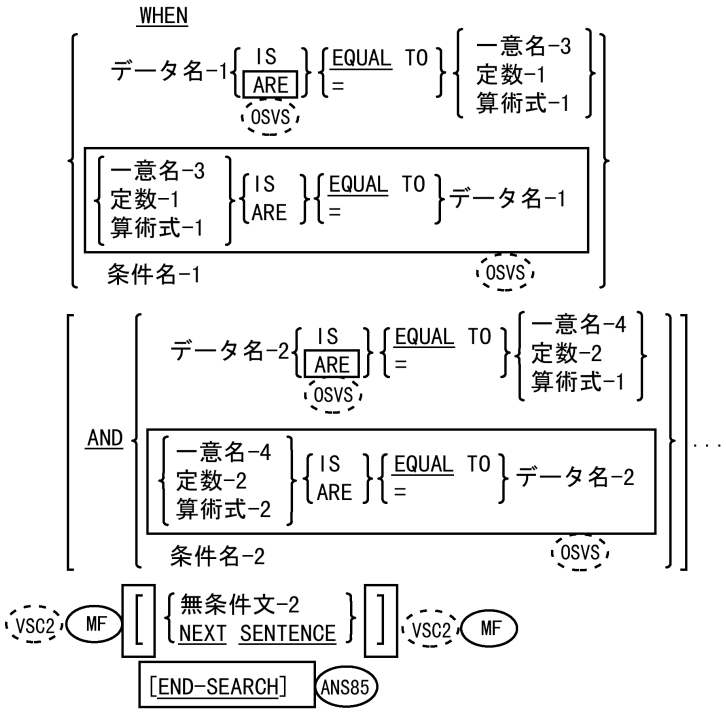
```
REWRITE レコード名[FROM 一意名]

        [END-REWRITE]
```

```
REWRITE レコード名 [FROM 一意名]
      [INVALID KEY 無条件文-1]

    [NOT INVALID KEY 無条件文-2]
    [END-REWRITE]                   ANS85
```

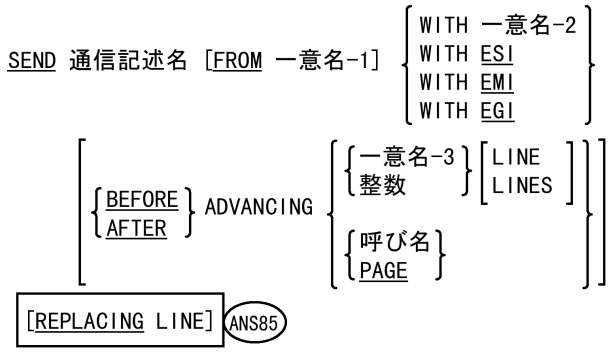
```
ROLLBACK
```

```
SEARCH 一意名-1 [ VARYING { 一意名-2 } ]
                          { 指標名-1 }
      [AT END 無条件文-1]                          MF
    { WHEN 条件-1 [ [ { 無条件文-2     } ] ] }...
                  [ [ { NEXT SENTENCE } ] ]
                    MF
    [END-SEARCH]
                ANS85
```

SEARCH ALL 一意名-1 [AT END 無条件文-1]

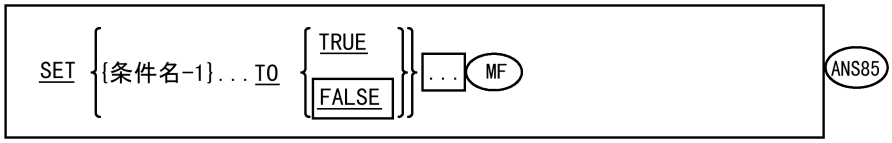


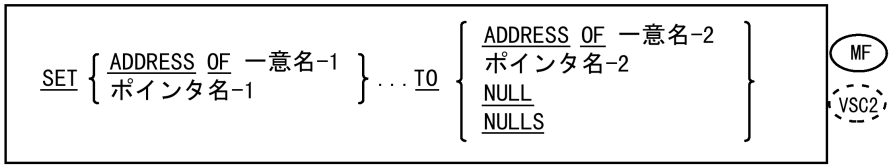
SEND 通信記述名 FROM 一意名-1

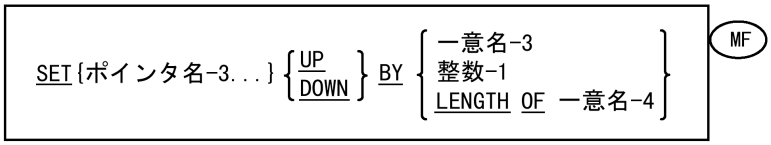


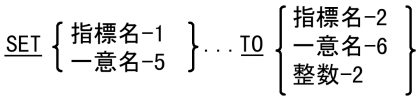
SERVICE { LABEL / RELOAD 一意名 }

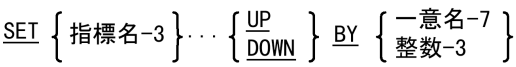
SET {呼び名-1}... TO { ON / OFF }... ANS85

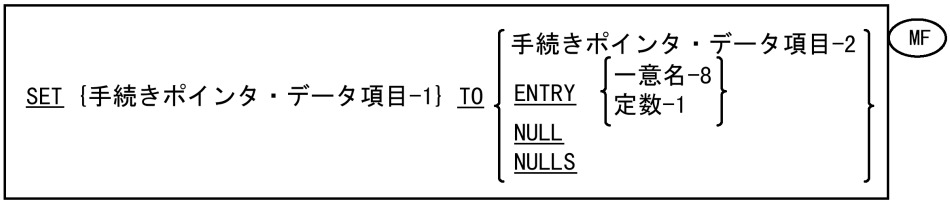

SET
{条件名-1}...TO
TRUE
FALSE
...
MF
ANS85


SET
ADDRESS OF 一意名-1
ポインタ名-1
...TO
ADDRESS OF 一意名-2
ポインタ名-2
NULL
NULLS
MF
VSC2


SET{ポインタ名-3...}
UP
DOWN
BY
一意名-3
整数-1
LENGTH OF 一意名-4
MF

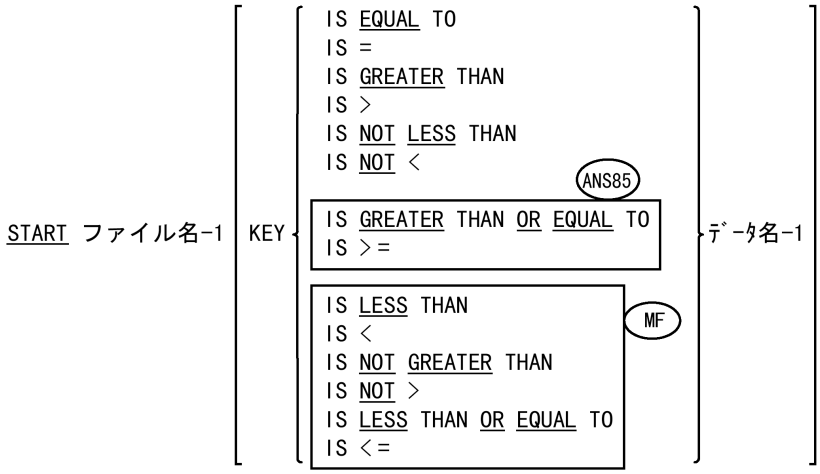

SET
指標名-1
一意名-5
...TO
指標名-2
一意名-6
整数-2


SET
指標名-3
...
UP
DOWN
BY
一意名-7
整数-3


SET {手続きポインタ・データ項目-1} TO
手続きポインタ・データ項目-2
ENTRY
一意名-8
定数-1
NULL
NULLS
MF

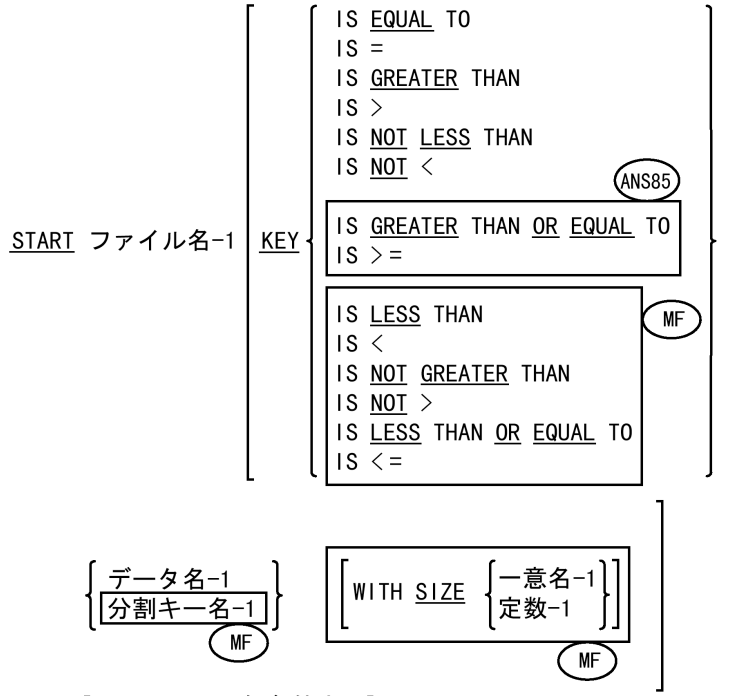
```
SET {オブジェクト一意名-1}... TO { オブジェクト一意名-2 }
                                  { NULL                 }

                                        (OSVS)
SORT ファイル名-1 ON { { ASCENDING  } KEY [IS] {ﾃﾞｰﾀ名-1}... }...
                     { { DESCENDING }                       }

    [WITH DUPLICATES IN ORDER] (ANS85)

    [COLLATING SEQUENCE IS 符号系名]

{ INPUT PROCEDURE IS 手続き名-1 [ { { THROUGH } 手続き名-2 } ] }
                                { { THRU    }               }
{ USING {ファイル名-2}...                                     }

{ OUTPUT PROCEDURE IS 手続き名-3 [ { { THROUGH } 手続き名-4 } ] }
                                 { { THRU    }               }
{ GIVING [{] ファイル名-3 [}]...                              }
         (ANS85)             (ANS85)
```

```
SORT データ名-2 [ ON { ASCENDING  } KEY [データ名-1]... ]... (MF)
                     { DESCENDING }

      [WITH DUPLICATES IN ORDER]
      [COLLATING SEQUENCE IS 符号系名]
```



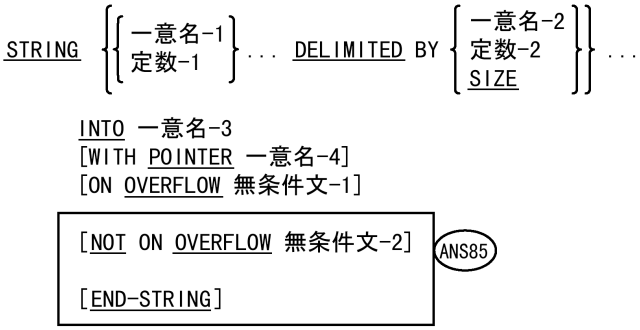
[INVALID KEY 無条件文-1]

[NOT INVALID KEY 無条件文-2]
[END-START]
ANS85


START ファイル名-1
KEY
IS EQUAL TO
IS =
IS GREATER THAN
IS >
IS NOT LESS THAN
IS NOT <
ANS85
IS GREATER THAN OR EQUAL TO
IS >=
IS LESS THAN
IS <
IS NOT GREATER THAN
IS NOT >
IS LESS THAN OR EQUAL TO
IS <=
MF
データ名-1
分割キー名-1
MF
WITH SIZE
一意名-1
定数-1
MF
[INVALID KEY 無条件文-1]
[NOT INVALID KEY 無条件文-2]
[END-START]
ANS85


STOP
RUN
定数-1


STOP RUN
GIVING
RETURNING
[ADDRESS OF] 一意名-1
整数-1 [SIZE IS 整数-2]
MF


STRING
一意名-1
定数-1
DELIMITED BY
一意名-2
定数-2
SIZE
INTO 一意名-3
[WITH POINTER 一意名-4]
[ON OVERFLOW 無条件文-1]
[NOT ON OVERFLOW 無条件文-2]
[END-STRING]
ANS85

```
SUBTRACT { 一意名-1 } ...FROM{一意名-2[ROUNDED]}...
         { 定数-1   }

  [ON SIZE ERROR 無条件文-1]

  [NOT ON SIZE ERROR 無条件文-2]   (ANS85)
  [END-SUBTRACT]
```

```
SUBTRACT { 一意名-1 } ...FROM { 一意名-2 }
         { 定数-1   }        { 定数-2   }

GIVING {一意名-3 [ROUNDED]}...

  [ON SIZE ERROR 無条件文-1]

  [NOT ON SIZE ERROR 無条件文-2]   (ANS85)
  [END-SUBTRACT]
```

```
SUBTRACT { CORRESPONDING } 一意名-1 FROM 一意名-2 [ROUNDED]
         { CORR          }

  [ON SIZE ERROR 無条件文-1]

  [NOT ON SIZE ERROR 無条件文-2]   (ANS85)
  [END-SUBTRACT]
```

```
SUPPRESS PRINTING
```

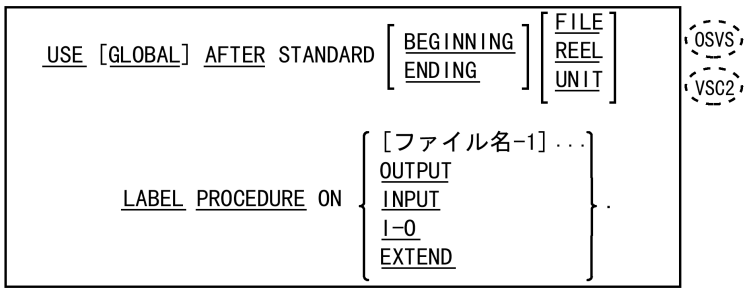
```
TERMINATE {報告書名-1}...
```

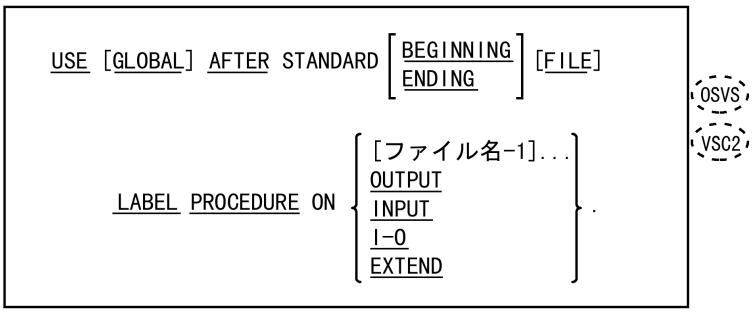
```
TRANSFORM 一意名-3 CHARACTERS
    FROM { 表意定数-1 } TO { 表意定数-2 }
         { 文字定数-1 }    { 文字定数-2 }
         { 一意名-1   }    { 一意名-2   }
```

```
UNLOCK ファイル名 { RECORD  }
                  { RECORDS }
```

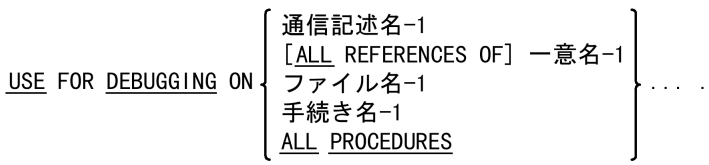
```
UNSTRING 一意名-1
    [ DELIMITED BY [ALL] { 一意名-2 } [ OR [ALL] { 一意名-3 } ]... ]
                         { 定数-1   }           { 定数-2   }
        INTO {一意名-4 [DELIMITER IN 一意名-5][COUNT IN 一意名-6]}...
        [WITH POINTER 一意名-7]
        [TALLYING IN 一意名-8]
        [ON OVERFLOW 無条件文-1]
    [NOT ON OVERFLOW 無条件文-2]
    [END-UNSTRING]                                   (ANS85)
```

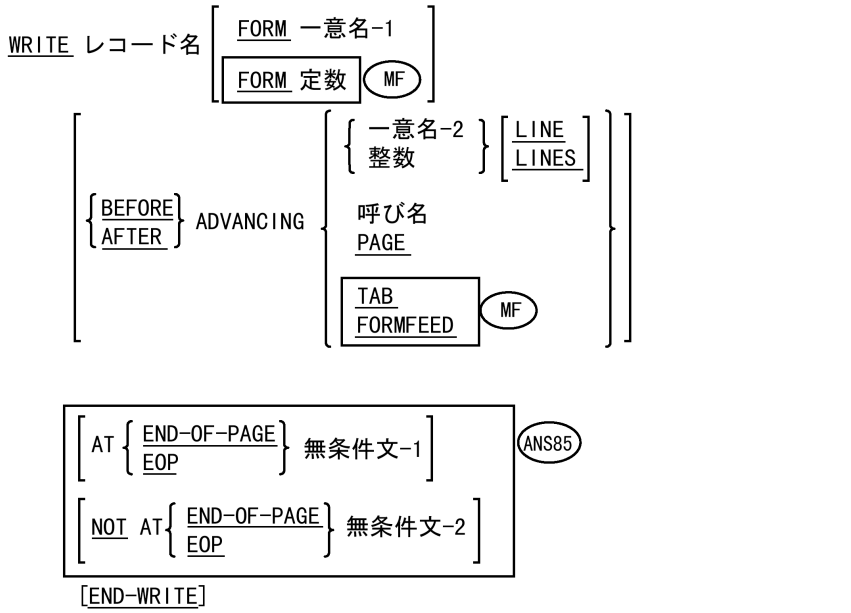
```
USE [GLOBAL] AFTER STANDARD { EXCEPTION }
    (ANS85)                 { ERROR     }
    PROCEDURE ON { [ファイル名-1]... }
                 { INPUT           }
                 { OUTPUT          }
                 { I-O             }
                 { EXTEND          }
    GIVING データ名-1 [データ名-2] .
                                     (OSVS)
```

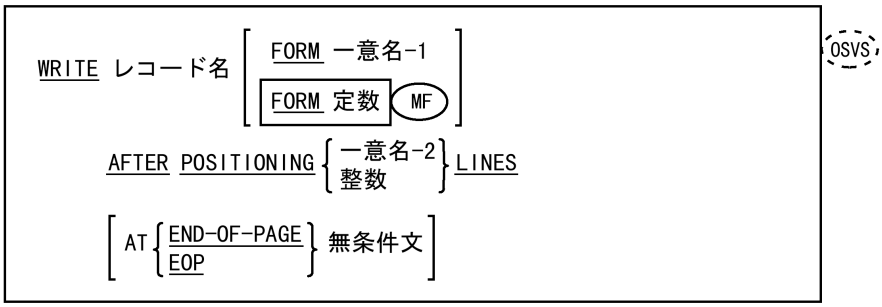

USE [GLOBAL] AFTER STANDARD [BEGINNING / ENDING] [FILE / REEL / UNIT]
OSVS
VSC2
LABEL PROCEDURE ON { [ファイル名-1]... / OUTPUT / INPUT / I-O / EXTEND } .

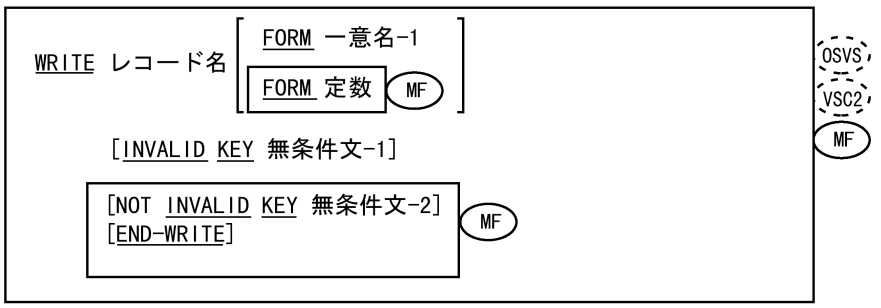

USE [GLOBAL] AFTER STANDARD [BEGINNING / ENDING] [FILE]
OSVS
VSC2
LABEL PROCEDURE ON { [ファイル名-1]... / OUTPUT / INPUT / I-O / EXTEND } .

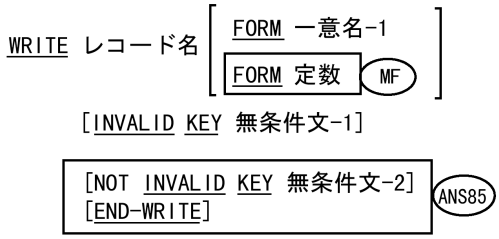

USE [GLOBAL] BEFORE REPORTING 一意名-1
ANS85


USE FOR DEBUGGING ON { 通信記述名-1 / [ALL REFERENCES OF] 一意名-1 / ファイル名-1 / 手続き名-1 / ALL PROCEDURES } ... .

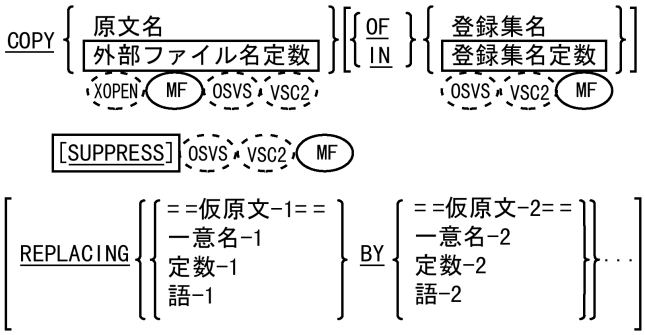

WRITE レコード名
FORM 一意名-1
FORM 定数
MF
BEFORE
AFTER
ADVANCING
一意名-2
整数
LINE
LINES
呼び名
PAGE
TAB
FORMFEED
MF
AT END-OF-PAGE
EOP
無条件文-1
ANS85
NOT AT END-OF-PAGE
EOP
無条件文-2
[END-WRITE]


WRITE レコード名
FORM 一意名-1
FORM 定数
MF
OSVS
AFTER POSITIONING
一意名-2
整数
LINES
AT END-OF-PAGE
EOP
無条件文


WRITE レコード名
FORM 一意名-1
FORM 定数
MF
OSVS
VSC2
MF
[INVALID KEY 無条件文-1]
[NOT INVALID KEY 無条件文-2]
[END-WRITE]
MF



A.6.1 COPY文の一般形式



A.6.2 REPLACE文の一般形式

REPLACE {==仮原文-1== BY ==仮原文-2==}...

REPLACE OFF

A.7 オブジェクトCOBOL言語拡張の一般形式

A.7.1 クラス定義

[IDENTIFICATION DIVISION.]

CLASS-ID. クラス名-1 [IS ABSTRACT | IS EXTERNAL]

[DATAIS {PRIVATE | PROTECTED | RESTRICTED}]

[INHERITS FROM クラス名-2 [WITH DATA]] .

クラス本体

オブジェクトプログラム

END CLASS クラス名-1

A.7.2 クラス拡張

書き方

[IDENTIFICATION DIVISION.]

CLASS-ID. 拡張名-1 EXTEND クラス名-1 [WITH DATA].

クラス本体

オブジェクトプログラム

END CLASS 拡張名-1

A.7.3 クラス本体

一般形式

書き方 1

OBJECT SECTION.

CLASS-CONTROL.

{クラス名-3 IS CLASS “外部名-1”} ... .

[データ部]

[クラスオブジェクト]

書き方 2

OBJECT SECTION.

CLASS-CONTROL.

{クラス名-3 IS CLASS “外部名-1”}... .
[データ部]

[手続き部]

[メソッド-1]...

A.7.4 クラスオブジェクト

書き方

[IDENTIFICATION DIVISION.]

CLASS-OBJECT.

[データ部]

[手続き部

[メソッド-1] ...]

END CLASS-OBJECT.

A.7.5 オブジェクトプログラム

書き方

[IDENTIFICATION DIVISION.]

OBJECT.

[データ部]

[PROCEDURE DIVISION.]

[メソッド-1] ...

END OBJECT.

A.7.6 メソッド

書き方

METHOD-ID.“メソッド名-1”.

［データ部］

［手続き部］

END METHOD“メソッド名-1”.

A.7.7 メソッドインタフェース定義

書き方

METHOD-ID.“メソッド名-1”.

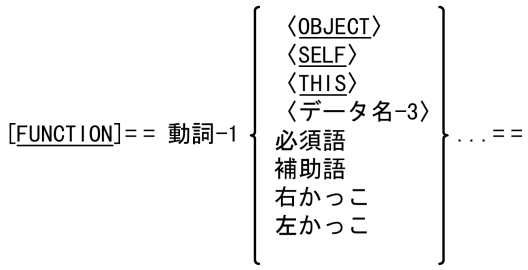
［連絡節］

手続き部見出し

［INVOKED AS 動詞シグネチャ ［OR AS 動詞シグネチャ］...］.

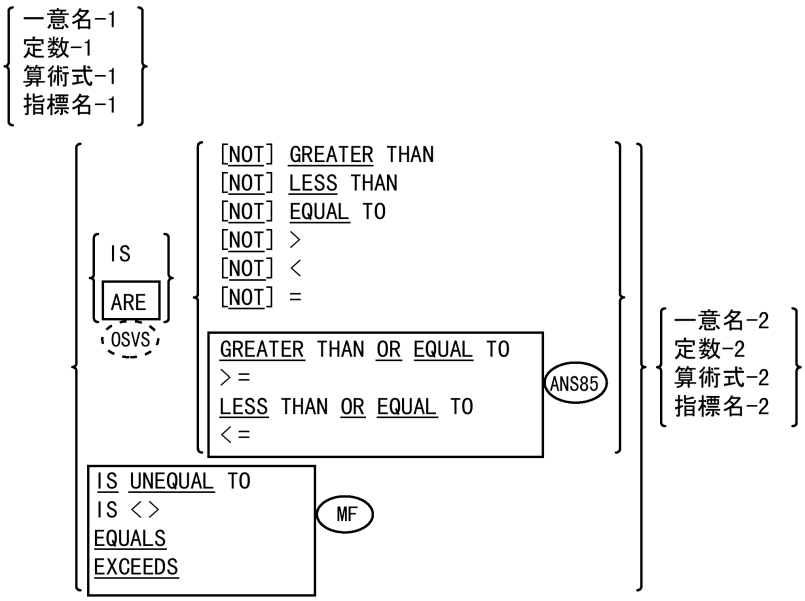
END METHOD“メソッド名-1”.

ここで動詞シグネチャは以下の通り。

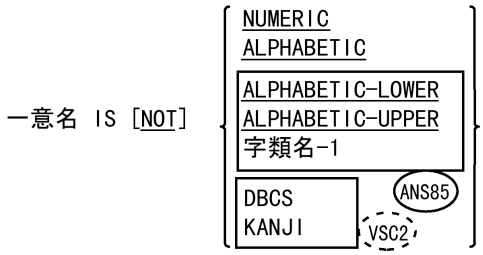


A.8 条件の一般形式

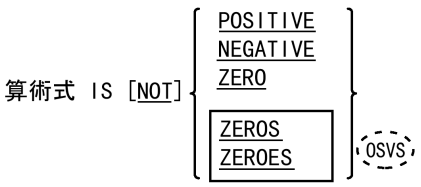
A.8.1 比較条件



A.8.2 字類条件



A.8.3 正負条件



A.8.4 条件名条件

条件名

A.8.5 スイッチ状態条件

条件名

A.8.6 否定単純条件

NOT 単純条件

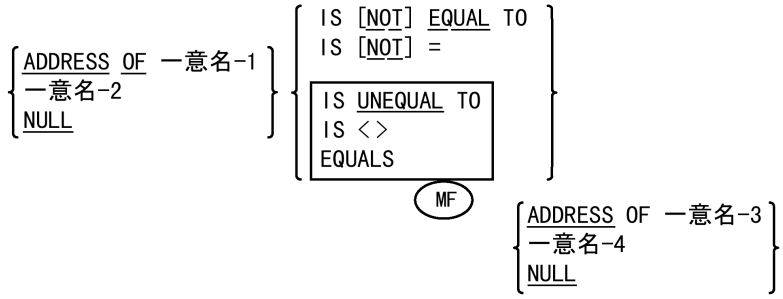
A.8.7 組合わせ条件

条件 {{AND / OR} 条件}...

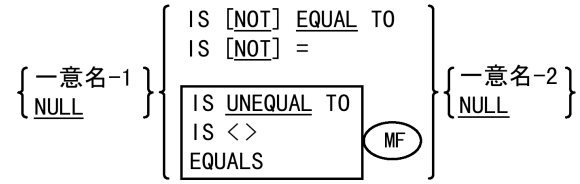
A.8.8 略記組合せ比較条件

比較条件 {{AND / OR} [NOT] [比較演算子] オブジェクト}...

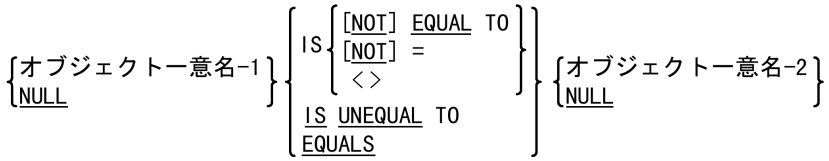
A.8.9 ポインタ条件



A.8.10 手続き部ポインタ(Procedure-Pointer)条件



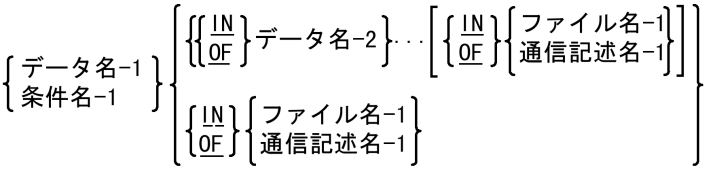
A.8.11 オブジェクト比較条件



A.9 その他の形式

A.9.1 修飾

書き方 1



書き方 2

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

書き方 3

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

書き方 4

```
LINAGE-COUNTER { OF } ファイル名-2
               { IN }
```

書き方 5

```
{ PAGE-COUNTER } { OF } 報告書名-1
{ LINE-COUNTER } { IN }
```

書き方 6

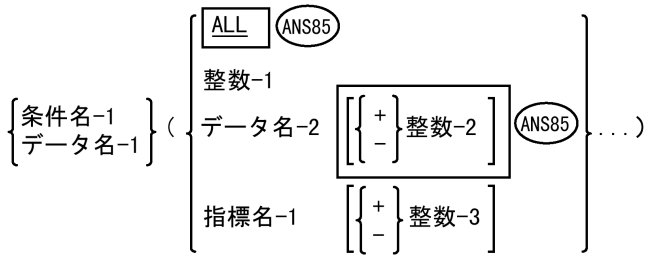
```
データ名-3 { { OF } データ名-4 [ { OF } 報告書名-2 ] }
           { { IN }             { IN }             }
           { { OF } 報告書名-2                     }
           { { IN }                                }
```

書き方 7

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

MF
XOPEN

A.9.2 添字付けと指標付け



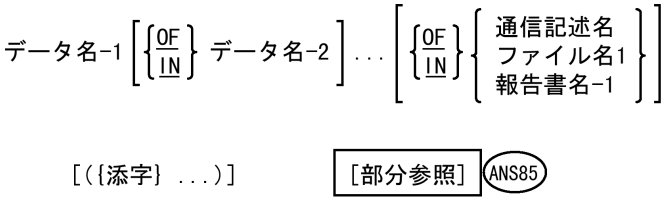
A.9.3 関数一意名

FUNCTION 関数名-1 [({引数-1}...)] [部分参照]

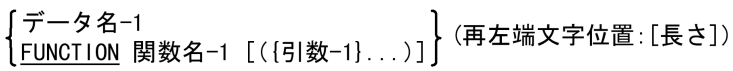
A.9.4 一意名

書き方 1

書き方 2



A.9.5 部分参照



付録B：文字集合と文字の大小順序

以下の表に、ASCIIおよびEBCDICの文字集合と文字の大小順序を示す。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大小順序の中の 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：文字集合と文字の大小順序

	ASCII		EBCDIC	
Hex	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

Hex	ASCII		EBCDIC	
	Character	COBOL	Character	COBOL
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

Hex	ASCII		EBCDIC	
	Character	COBOL	Character	COBOL
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

	ASCII		EBCDIC	
Hex	Character	COBOL	Character	COBOL
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	

Hex	EBCDIC	
	Character	COBOL
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

Hex	EBCDIC	
	Character	COBOL
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	

Hex	EBCDIC	
	Character	COBOL
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

Hex
FB
FC
FD
FE
FF

EBCDIC	
Character	COBOL
	X
	X
	X
	X
E0	X

付録C: 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。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。

C.1 凡例

74	ANSI 74標準の予約語
85	ANSI 85標準の予約語
I2	ISO 2000 標準(ドラフト)の予約語
OS	OSVS COBOL の予約語
VS(2)	VS COBOL II リビジョン 2の予約語
VS(3)	特に断わりがない限り、VS COBOL II rev 3、IBM LE COBOL/370 及び IBM COBOL for MVS & VM の予約語
VS(4)	特に断わりがない限り、VS COBOL II rev 4、IBM LE COBOL/370 及び IBM COBOL for MVS & VM の予約語
XO	X/Open定義の一部としての予約語
MFER	Early Release 構文の一部としての予約語
MF*n*	MF(*n*)指令を使用したときの予約語
MFOO	MFOO指令を使用した時の OO構文の一部としての予約語
MS*n*	MS(*n*)指令を使用したときの予約語
RM	RM指令を使用したときの予約語
C370	VS COBOL IIを含まない、IBM LE COBOL/370 リビジョン 2 及び IBM COBOL for MVS & VM の予約語
DVS	DOSVS COBOLの予約語

C.2 MF 互換性指令

MF(1)	LEVEL II COBOL、LEVEL II COBOL/ET 及び Professional COBOL
MF(2)	MF(1)にMicro Focus VS COBOL Workbenchバージョン1.2の追加機能を 加えたもの。
MF(3)	MF(2) に Micro Focus VS COBOL Workbench バージョン 1.3 と 2.0 、Professional COBOLバージョン2.0、Micro Focusバージョン1.5の追加機 能を加えたもの。
MF(4)	MF(3)にMicro Focus COBOL/2バージョン1.1、Microsoft COBOL™バー ジョン3.0、IBM COBOLバージョン2の追加機能を加えたもの
MF(5)	MF(4) に Micro Focus COBOL/2 バージョン 1.2 、 Micro Focus COBOL/2 Workbenchバージョン2.3の追加機能を加えたもの。

MF(6)	MF(5)にMicro Focus COBOL/2バージョン2.4、MicroFocus COBOL/2 Workbenchバージョン2.4の追加機能を加えたもの。
MF(7)	MF(6)にMicro Focus COBOL/2バージョン2.5、MicroFocus COBOL/2 Workbenchバージョン2.5の追加機能を加えたもの。
MF(8)	MF(7)にMicro Focus COBOLバージョン3.0、MicroFocus COBOL Workbenchバージョン3.0の追加機能を加えたもの。
MF(9)	MF(8)にMicro Focus COBOLバージョン3.1、MicroFocus COBOL Workbenchバージョン3.1の追加機能を加えたもの。
MF(10)	MF(9)にMicro Focus COBOLバージョン3.2、MicroFocus COBOL Workbenchバージョン3.2の追加機能を加えたもの。
MF(11)	MF(10)にMicro Focus NetExpress 3.0の 追加機能を加えたもの

C.3 予約語表

A

ABSENT			I2							
ABSTRACT							MF11			
ACCEPT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ACCESS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ACQUIRE	…	…		…	…	…	MF7	…	…	…
ACTIVE-CLASS			I2							
ACTUAL	…	…		OS	VS(2)	…	…	…	…	DVS
ADD	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ADDRESS	…	…	I2	OS	VS(2, 3, 4)	…	MF3	…	…	DVS
ADVANCING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
AFP-5A				OS						DVS
AFTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ALL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ALLOCATE			I2							
ALLOW			I2							
ALPHABET	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…
ALPHABETIC	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ALPHABETIC-LOWER	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…
ALPHABETIC-UPPER	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…
ALPHANUMERIC	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…
ALPHANUMERIC-EDITED	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…
ALSO	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

ALTER	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ALTERNATE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
AND	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ANY	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…
APPLY	…	…		OS	VS(2, 3, 4)	…	…	…	…	DVS
ARE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
AREA	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
AREAS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
AS	…	…	I2	…	…	…	MF00	…	…	…
ASCENDING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ASSIGN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
AT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
AUTHOR	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
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
BEGINNING	…	…		OS	VS(2, 3, 4)	…	…	…	…	DVS
BINARY	…	85	I2	…	VS(3, 4)	XO	MF7	…	RM	…
BINARY-CHAR			I2							
BINARY-DOUBLE			I2							
BINARY-LONG			I2							
BINARY-SHORT			I2							
BIT			I2							
BLANK	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
BLOCK	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

BOOLEAN			I2							
BOTTOM	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
BROWSING							MF11			
BY	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS

C

C01	...	...		0S	...	...	...	...	...	DVS
C02	...	...		0S	...	...	...	...	...	DVS
C03	...	...		0S	...	...	...	...	...	DVS
C04	...	...		0S	...	...	...	...	...	DVS
C05	...	...		0S	...	...	...	...	...	DVS
C06	...	...		0S	...	...	...	...	...	DVS
C07	...	...		0S	...	...	...	...	...	DVS
C08	...	...		0S	...	...	...	...	...	DVS
C09	...	...		0S	...	...	...	...	...	DVS
C10	...	...		0S	...	...	...	...	...	DVS
C11	...	...		0S	...	...	...	...	...	DVS
C12	...	...		0S	...	...	...	...	...	DVS
CALL	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
CALL-CONVENTION			I2							
CALLED							MF11			
CANCEL	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	...
CBL	...	...		0S	VS(2, 3, 4)	...	...	...	...	...
CD	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	...
CF	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
CH	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
CHAIN	...	...		...	...	...	MF3	MS2	...	...
CHAINING	...	...		...	...	...	MF3	MS2	...	...
CHANGED	...	...		0S	VS(2)	...	MF3	...	...	DVS
CHARACTER	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
CHARACTERS	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
CLASS	...	85	I2	...	VS(3, 4)	X0	MF7	...	...	...
CLASS-CONTROL							MF00			
CLASS-ID			I2		C370		MF11			
CLASS-OBJECT	...	...		...	...	...	MF00	...	...	...
CLOCK-UNITS	74	85		0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
CLOSE	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
COBOL	74	85		0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
CODE	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
CODE-SET	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS

COERCION							MF11			
COL	...	...	I2	...	...	...	MF3	MS2	...	...
COLLATING	74	85	I2	OS	VS(2, 3, 4)	XO	MF3	MS2	RM	DVS
COLS			I2							
COLUMN	74	85	I2	OS	VS(2, 3, 4)	XO	MF3	MS2	RM	DVS
COLUMNS			I2							
COM-REG	...	...		...	VS(2, 3, 4)	...	...	...	...	DVS
COMMA	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
COMMIT	...	...		...	...	...	MF1	...	...	...
COMMITMENT	...	...		...	...	...	MF7	...	...	...
COMMON	...	85	I2	...	VS(3, 4)	XO	MF7	...	...	...
COMMUNICATION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	...
COMP	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
COMP-0	...	...		...	...	...	MF3	MS2	...	...
COMP-1	...	...		OS	VS(2, 3, 4)	...	MF7	...	RM	DVS
COMP-2	...	...		OS	VS(2, 3, 4)	...	MF7	...	...	DVS
COMP-3	...	...		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
COMP-4	...	...		OS	VS(2, 3, 4)	...	MF7	MS2	...	DVS
COMP-5	...	...		...	...	XO	MF3	...	...	...
COMP-6	...	...		...	...	...	MF10	...	RM	...
COMP-X	...	...		...	...	...	MF2	...	...	...
COMPUTATIONAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
COMPUTATIONAL-0	...	...		...	...	...	MF3	MS2	...	...
COMPUTATIONAL-1	...	...		OS	VS(2, 3, 4)	...	MF7	...	RM	DVS
COMPUTATIONAL-2	...	...		OS	VS(2, 3, 4)	...	MF7	...	...	DVS
COMPUTATIONAL-3	...	...		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
COMPUTATIONAL-4	...	...		OS	VS(2, 3, 4)	...	MF7	MS2	...	DVS
COMPUTATIONAL-5	...	...		...	...	XO	MF3	...	...	...
COMPUTATIONAL-6	...	...		...	...	...	MF10	...	RM	...
COMPUTATIONAL-X	...	...		...	...	...	MF2	...	...	...
COMPUTE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
CONFIGURATION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
CONSOLE	...	...		OS	...	...	MF1	MS2	...	DVS
CONSTANT			I2							
CONTAINS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
CONTENT	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
CONTINUE	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
CONTROL	74	85	I2	OS	VS(2, 3, 4)	XO	MF5	MS2	RM	DVS
CONTROL-AREA	...	...		...	...	...	MF7	...	...	...
CONTROLS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
CONVERT	...	...		...	...	...	...	...	RM	...

CONVERTING	...	85	I2	...	VS(3, 4)	X0	MF7	...	...	...
COPY	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
CORE-INDEX	...	...		0S	VS(2)	...	...	...	...	DVS
CORR	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
CORRESPONDING	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
COUNT	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
CRT	...	...	I2	...	...	X0	MF1	...	...	...
CRT-UNDER	...	...		...	...	...	MF1	...	...	...
CSP	...	...		0S	...	...	...	...	...	DVS
CURRENCY	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
CURRENT-DATE	...	...		0S	VS(2)	...	...	...	...	DVS
CURSOR	...	...	I2	...	...	X0	MF1	...	...	...
CYL-INDEX	...	...		...	...	...	...	...	...	DVS
CYL-OVERFLOW	...	...		...	...	...	...	...	...	DVS

D

DATA	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DATE	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DATE-COMPILED	74	85		0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DATE-WRITTEN	74	85		0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DAY	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DAY-OF-WEEK	...	85	I2	...	VS(3, 4)	X0	MF7	...	...	...
DBCS	...	...		...	VS(3, 4)	...	MF7	...	...	...
DE	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DEBUG	...	...		0S	VS(2)	...	...	...	...	DVS
DEBUG-CONTENTS	74	85		0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DEBUG-ITEM	74	85		0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DEBUG-LINE	74	85		0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DEBUG-NAME	74	85		0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DEBUG-SUB-1	74	85		0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DEBUG-SUB-2	74	85		0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DEBUG-SUB-3	74	85		0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DEBUGGING	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DECIMAL-POINT	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DECLARATIVES	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DEFAULT			I2				MF11			
DEFINITION							MF11			
DELETE	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DELIMITED	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
DELIMITER	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS

DEPENDING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
DESCENDING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
DESTINATION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	...
DETAIL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
DISABLE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	...
DISK	...	...		...	...	...	MF3	MS2	...	...
DISP	...	...		OS	VS(2)	...	...	...	...	...
DISPLAY	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
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
DIVISION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
DOWN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
DROP	...	...		...	...	...	MF7	...	...	...
DUPLICATES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
DYNAMIC	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

E

ECHO	...	...		...	...	...	...	...	RM	...
EGCS	...	...		...	VS(2, 3, 4)	...	...	...	...	...
EGI	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	...
EJECT	...	...		OS	VS(2, 3, 4)	...	MF7	MS2	...	DVS
ELSE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
EMI	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	...
EMPTY-CHECK	...	...		...	...	...	MF3	MS2	...	...
ENABLE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	...
END	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
END-ACCEPT	...	...	I2	...	...	XO	MF4	...	...	...
END-ADD	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-CALL	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-CHAIN	...	...		...	...	...	MF5	...	...	...
END-COMPUTE	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-DELETE	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-DISPLAY	...	...	I2	...	...	XO	MF7	...	...	...
END-DIVIDE	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-EVALUATE	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-IF	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-INVOKE	...	...		...	C370	...	MF00	...	...	...
END-MULTIPLY	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-OF-PAGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

END-PERFORM	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-READ	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-RECEIVE	...	85	I2	...	VS(3, 4)	XO	MF7	...	...	...
END-RETURN	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-REWRITE	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-SEARCH	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-START	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-STRING	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-SUBTRACT	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-UNSTRING	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
END-WAIT							MF11			
END-WRITE	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
ENDING	...	...		OS	VS(2, 3, 4)	...	...	...	...	DVS
ENTER	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ENTRY	...	...		OS	VS(2, 3, 4)	...	MF5	...	...	DVS
ENVIRONMENT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
EOP	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
EQUAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
EQUALS	...	...		...	...	...	MF7	...	...	...
ERROR	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ESCAPE	...	...		...	...	...	MF3	MS2	...	...
ESI	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	...
EVALUATE	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
EVENT-POINTER							MF11			
EVERY	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
EXAMINE	...	...		OS	VS(2)	...	...	...	...	DVS
EXCEEDS	...	...		...	...	...	MF7	...	...	...
EXCEPTION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
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
EXTEND	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
EXTENDED-SEARCH	...	...		...	...	...	...	...	...	DVS
EXTERNAL	...	85	I2	...	VS(3, 4)	XO	MF2	...	...	...
EXTERNALLY-DESCRIBED-KEY	...	...		...	...		...	MF7	...	...

F

FACTORY	...	...	I2	...	...	...	MF00	...	...	
FALSE	...	85	I2	...	VS(2, 3, 4)	X0	MF7	...	...	...
FD	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
FH--FCD	...	...		...	...	...	MF5	...	...	...
FH--KEYDEF	...	...		...	...	...	MF5	...	...	...
FILE	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
FILE-CONTROL	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
FILE-ID	...	...		...	...	...	MF3	MS2	...	...
FILE-LIMIT	...	...		0S	VS(2)	...	...	...	...	DVS
FILE-LIMITS	...	...		0S	VS(2)	...	...	...	...	DVS
FILLER	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
FINAL	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
FIRST	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
FIXED	...	...		...	...	...	MF3	...	...	...
FLOAT-EXTENDED			I2							
FLOAT-LONG			I2							
FLOAT-SHORT			I2							
FOOTING	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
FOR	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
FORMAT			I2				MF7			
FREE			I2							
FROM	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
FUNCTION	...	85	I2	...	...	X0	MF7	...	...	
FUNCTION-ID			I2							

G

GENERATE	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
GIVING	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
GLOBAL	...	85	I2	...	VS(3, 4)	X0	MF7	...	...	...
GO	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
GOBACK	...	...	I2	0S	VS(2, 3, 4)	...	MF5	...	...	DVS
GREATER	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
GRID	...	...		...	...	...	MF4	...	...	...
GROUP	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS

H

HEADING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
HIGH-VALUE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
HIGH-VALUES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

I

I-O	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
I-O-CONTROL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ID	…	…		OS	VS(2, 3, 4)	…	MF7	…	…	DVS
IDENTIFICATION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
IDENTIFIED							MF11			
IF	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
IGNORE	…	…		…	…	…	MF8	…	…	…
IN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
INDEX	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
INDEXED	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
INDIC	…	…		…	…	…	MF7	…	…	…
INDICATE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
INDICATOR	…	…		…	…	…	MF7	…	…	…
INDICATORS	…	…		…	…	…	MF7	…	…	…
INHERITING	…	…		…	…	…	MF00	…	…	…
INHERITS			I2		C370		MF11			
INITIAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
INITIALIZE	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…
INITIATE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
INPUT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
INPUT-OUTPUT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
INSERT	…	…		OS	VS(2, 3, 4)	…	…	…	…	DVS
INSPECT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
INSTALLATION	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
INTEGER			I2							
INTERFACE			I2							
INTERFACE-ID			I2							
INTO	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
INVALID	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
INVOKE	…	…	I2		C370	…	MF00	…		…
INVOKED	…	…		…	…	…	MF00	…		…
IS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

J

JAPANESE	...	...		...	...	...	MF1	...	...	...
JUST	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
JUSTIFIED	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

K

KANJI	...	...		...	VS(2, 3, 4)	...	MF8	...	...	...
KEPT	...	...		...	...	...	MF1	...	...	...
KEY	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
KEYBOARD	...	...		...	...	...	MF3	...	...	

L

LABEL	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LAST	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LEADING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LEAVE	...	...		OS	VS(2)	...	...	...	...	...
LEFT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LEFT-JUSTIFY	...	...		...	...	...	MF3	MS2	...	...
LEFTLINE	...	...		...	...	...	MF4	...	...	...
LENGTH	74	85	I2	OS	VS(2, 3, 4)	XO	MF6	MS2	RM	...
LENGTH-CHECK	...	...		...	...	...	MF3	MS2	...	...
LESS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LIMIT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LIMITS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LIN	...	...		...	...	...	...	MS2	...	...
LINAGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LINAGE-COUNTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	...
LINE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LINE-COUNTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LINES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LINKAGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LOCAL-STORAGE	...	...	I2	...	C370	...	MF5	...	...	...
LOCK	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LOCKING	...	...		...	...	...	...	MS2	...	...
LOW-VALUE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LOW-VALUES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
LOWER	...	...		...	...	...	MF8	...	...	...

M

MASTER-INDEX	…	…		…	…	…	…	…	…	DVS
MEMORY	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
MERGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
MESSAGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…
METACLASS					C370					
METHOD	…	…	I2	…	C370	…	MF11	…	…	…
METHOD-ID			I2		C370		MF11			
MODE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
MODIFIED	…	…		…	…	…	MF7	…	…	…
MODULES	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
MONITOR-POINTER							MF11			
MORE-LABELS	…	…		OS	VS(2, 3, 4)	…	…	…	…	DVS
MOVE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
MULTIPLY	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
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
NCHAR	…	…		…	…	…	MF5	…	…	…
NEGATIVE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
NESTED			I2							
NEXT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
NO	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
NO-ECHO	…	…		…	…	…	MF3	MS2	…	…
NOMINAL	…	…		OS	VS(2)	…	…	…	…	DVS
NOT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
NOTE	…	…		OS	VS(2)	…	…	…	…	DVS
NSTD-REELS	…	…		…	…	…	…	…	…	DVS
NULL	…	…	I2	…	VS(2, 3, 4)	…	MF6	…	…	…
NULLS	…	…		…	VS(2, 3, 4)	…	MF6	…	…	…
NUMBER	74	85	I2	OS	VS(2, 3, 4)	XO	MF3	MS2	RM	DVS
NUMBERS			I2							
NUMERIC	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
NUMERIC-EDITED	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	

O

O-FILL	…	…		…	…	…	MF7	…	…	…
OBJECT	…	…	I2	…	C370	…	MF00	…	…	…
OBJECT-COMPUTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
OBJECT-ID							MF00			
OBJECT-STORAGE	…	…		…	…	…	MF00	…	…	…
OCCURS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
OF	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
OFF	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
OMITTED	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ON	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
OOSTACKPTR	…	…		…	…	…	MF00	…	…	…
OPEN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
OPTIONAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
OPTIONS			I2							
OR	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ORDER	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…
ORGANIZATION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
OTHER	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…
OTHERWISE	…	…		OS	VS(2)	…	…	…	…	DVS
OUTPUT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
OVERFLOW	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
OVERLINE	…	…		…	…	…	MF4	…	…	
OVERRIDE			I2		C370					

P

PACKED-DECIMAL	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…
PADDING	…	85	I2	…	VS(3, 4)	XO	MF7	…	…	…
PAGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PAGE-COUNTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PASSWORD	…	…		OS	VS(2, 3, 4)	…	…	…	…	DVS
PERFORM	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PF	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PH	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PIC	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PICTURE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PLUS	74	85	I2	OS	VS(2, 3, 4)	XO	MF3	MS2	RM	DVS
POINTER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
POS	…	…		…	…	…	…	…	RM	…

POSITION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
POSITIONING	...	...		OS	VS(2)	...	...	...	...	DVS
POSITIVE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PRESENT			I2							
PRINT	...	...		...	...	...	...	...	RM	...
PRINT-SWITCH	...	...		OS	...	...	...	...	...	DVS
PRINTER	...	...		...	...	...	MF3	MS2	...	...
PRINTER-1	...	...		...	...	...	MF3	...	...	...
PRINTING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PRIOR	...	...		...	...	...	MF7	...	...	...
PRIVATE	...	...		...	...	...	MFOO	...	...	...
PROCEDURE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PROCEDURE-POINTER	...	...		...	C370	...	MF5	...	...	...
PROCEDURES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PROCEED	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PROCESS	...	...		...	...	...	MF7	...	...	...
PROCESSING	...	...		OS	VS(2, 3, 4)	...	...	...	...	DVS
PROGRAM	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PROGRAM-ID	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
PROGRAM-POINTER			I2							
PROMPT	...	...		...	...	...	MF3	MS2	RM	...
PROPERTY			I2							
PROTECTED	...	...		...	...	...	MF3	...	...	...
PUBLIC	...	...		...	...	...	MFOO	...	...	...
PURGE	...	85	I2	...	VS(3, 4)	XO	MF7	...	...	

Q

QUEUE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	...
QUOTE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
QUOTES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

R

RAISE			I2							
RAISING			I2							
RANDOM	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RANGE	...	...		...	...	...	MF3	...	...	...
RD	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
READ	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

READING							MF11			
READY	...	...		OS	VS(2, 3, 4)	...	MF3	MS2	...	DVS
RECEIVE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	...
RECORD	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RECORD-OVERFLOW	...	...		OS	VS(2)	...	...	...	...	...
RECORDING	...	...		OS	VS(2, 3, 4)	...	MF3	...	...	DVS
RECORDS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RECURSIVE			I2		C370					
REDEFINES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
REDEFINITION							MF11			
REEL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
REFERENCE	...	85	I2	...	VS(2, 3, 4)	XO	MF7	...	...	...
REFERENCES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RELATIVE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RELEASE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RELOAD	...	...		OS	VS(2, 3, 4)	...	...	...	...	DVS
REMAINDER	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
REMARKS	...	...		OS	VS(2)	...	...	...	...	DVS
REMOVAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RENAMES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
REORG-CRITERIA	...	...		OS	VS(2)	...	...	...	...	...
REPEATED	...	...		...	...	...	MF10	...	...	...
REPLACE	...	85	I2	...	VS(3, 4)	XO	MF7	...	...	...
REPLACING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
REPORT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
REPORTING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
REPORTS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
REPOSITORY			I2		C370					
REREAD	...	...		OS	VS(2)	...	...	...	...	...
RERUN	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RESERVE	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RESERVED			I2							
RESET	...	...	I2	OS	VS(2, 3, 4)	...	MF3	MS2	...	DVS
RESTRICTED							MF11			
RESUME			I2							
RETRY			I2							
RETURN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RETURN-CODE	...	...		OS	VS(2, 3, 4)	XO	MF5	...	...	...
RETURNING	...	...	I2	...	C370	...	MF5	...	...	...
REVERSED	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
REWIND	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

REWRITE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RF	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RH	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RIGHT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RIGHT-JUSTIFY	…	…		…	…	…	MF3	MS2	…	…
ROLLBACK	…	…		…	…	…	MF1	…	…	…
ROLLING	…	…		…	…	…	MF7	…	…	…
ROUNDED	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
RUN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

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
SCREEN	…	…	I2	…	…	XO	MF3	MS2	…	…
SD	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
SEARCH	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
SECTION	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
SECURITY	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
SEEK	…	…		OS	VS(2)	…	…	…	…	DVS
SEGMENT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…
SEGMENT-LIMIT	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
SELECT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
SELECTIVE	…	…		OS	VS(2)	…	…	…	…	…
SELF	…	…	I2	…	C370	…	MF00	…	…	…
SELFCLASS	…	…		…	…	…	MF00	…	…	…
SEMAPHORE-POINTER							MF11			
SEND	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	…
SENTENCE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
SEPARATE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
SEQUENCE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
SEQUENTIAL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
SERVICE	…	…		OS	VS(2, 3, 4)	…	…	…	…	DVS
SET	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
SHARING			I2				MF11			
SHIFT-IN	…	…		…	VS(2, 3, 4)	…	…	…	…	…

SHIFT-OUT	…	…		…	VS(2, 3, 4)	…	…	…	…	…
SIGN	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
SIZE	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
SKIP1	…	…		0S	VS(2, 3, 4)	…	MF7	…	…	DVS
SKIP2	…	…		0S	VS(2, 3, 4)	…	MF7	…	…	DVS
SKIP3	…	…		0S	VS(2, 3, 4)	…	MF7	…	…	DVS
SORT	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
SORT-CONTROL	…	…		…	VS(2, 3, 4)	…	…	…	…	…
SORT-CORE-SIZE	…	…		0S	VS(2, 3, 4)	…	…	…	…	DVS
SORT-FILE-SIZE	…	…		0S	VS(2, 3, 4)	…	…	…	…	DVS
SORT-MERGE	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
SORT-MESSAGE	…	…		0S	VS(2, 3, 4)	…	…	…	…	…
SORT-MODE-SIZE	…	…		0S	VS(2, 3, 4)	…	…	…	…	DVS
SORT-OPTION	…	…		…	…	…	…	…	…	DVS
SORT-RETURN	…	…		0S	VS(2, 3, 4)	…	MF7	…	…	DVS
SOURCE	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
SOURCE-COMPUTER	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
SOURCES			I2							
SPACE	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
SPACE-FILL	…	…		…	…	…	MF3	MS2	…	…
SPACES	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
SPECIAL-NAMES	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
STANDARD	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
STANDARD-1	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
STANDARD-2	…	85	I2	…	VS(3, 4)	X0	MF7	…	…	…
STANDARD-3			I2							
START	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
STARTING	…	…		…	…	…	MF7	…	…	…
STATUS	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
STOP	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
STRING	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
SUB-QUEUE-1	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	…
SUB-QUEUE-2	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	…
SUB-QUEUE-3	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	…
SUBFILE	…	…		…	…	…	MF7	…	…	…
SUBTRACT	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
SUM	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
SUPER	…	…	I2	…	C370	…	MF00	…	…	…
SUPPRESS	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
SYMBOLIC	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	…
SYNC	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS

SYNCHRONIZED	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
SYSIN	…	…		0S	…	…	…	…	…	…
SYSIPT	…	…		0S	…	…	…	…	…	DVS
SYSLST	…	…		0S	…	…	…	…	…	DVS
SYSOUT	…	…		0S	…	…	…	…	…	…
SYSPCH	…	…		…	…	…	…	…	…	DVS
SYSPUNCH	…	…		0S	…	…	…	…	…	DVS
SYSTEM-DEFAULT			I2							

T

TAB									RM	
TABLE	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	
TALLY				0S	VS(2, 3, 4)					DVS
TALLYING	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
TAPE	74	85		0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
TERMINAL	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
TERMINATE	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
TEST	…	85	I2	…	VS(2, 3, 4)	X0	MF7	…	…	…
TEXT	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	…
THAN	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
THEN	…	85	I2	0S	VS(2, 3, 4)	X0	MF1	…	…	DVS
THREAD-LOCAL							MF11			
THREAD-LOCAL-STORAGE							MF10			
THREAD-POINTER							MF11			
THROUGH	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
THRU	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
TIME	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
TIME-OF-DAY	…	…		0S	VS(2)	…	…	…	…	DVS
TIME-OUT	…	…		…	…	…	MF7	…	…	…
TIMEOUT	…	…	I2	…	…	…	MF7	…	…	…
TIMES	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
TITLE	…	…		…	VS(2, 3, 4)	…	MF7	…	…	…
TO	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
TOP	74	85	I2	0S	VS(2, 3, 4)	X0	MF1	MS2	RM	DVS
TOTALED	…	…		0S	VS(2)	…	…	…	…	…
TOTALING	…	…		0S	VS(2)	…	…	…	…	…
TRACE	…	…		0S	VS(2, 3, 4)	…	MF3	MS2	…	DVS
TRACK-AREA	…	…		0S	VS(2)	…	…	…	…	DVS
TRACK-LIMIT	…	…		0S	VS(2)	…	…	…	…	…

TRACKS	…	…		OS	VS(2)	…	…	…	…	DVS
TRAILING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
TRAILING-SIGN	…	…		…	…	…	MF3	MS2	…	…
TRANSACTION	…	…		…	…	…	MF7	…	…	…
TRANSFORM	…	…		OS	VS(2)	…	…	…	…	DVS
TRUE	…	85	I2	…	VS(2, 3, 4)	XO	MF7	…	…	…
TYPE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
TYPEDEF	…	…	I2	…	…	…	MF10		…	…

U

UNEQUAL	…	…		…	…	…	MF7	…	…	…
UNIT	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
UNIVERSAL			I2							
UNLOCK	…	…	I2	…	…	…	MF1	MS2	RM	…
UNSTRING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
UNTIL	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
UP	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
UPDATE	…	…		…	…	…	MF3	MS2	RM	…
UPON	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
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
USE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
USER	…	…		…	…	…	MF3	…	…	…
USER-DEFAULT			I2							
USING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

V

VALID			I2							
VALIDATE			I2							
VALUE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
VALUES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

VARIABLE	...	...		...	...	...	MF3	...	...	...
VARYING	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

W

WAIT	...	...		...	...	...	MF8	MS2	...	...
WHEN	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
WHEN-COMPILED	...	...		OS	VS(2, 3, 4)	...	MF7	...	...	DVS
WITH	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
WORDS	74	85		OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
WORKING-STORAGE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
WRITE	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
WRITE-ONLY	...	...		OS	VS(2, 3, 4)	...	...	...	...	DVS
WRITE-VERIFY	...	...		...	...	...	...	...	...	DVS
WRITING			I2							

Z

ZERO	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ZERO-FILL	...	...		...	...	...	MF3	MS2	...	...
ZEROES	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
ZEROS	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS

!

+	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
–	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
*	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
/	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
**	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
>	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
<	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
=	74	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
>=	...	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
<=	...	85	I2	OS	VS(2, 3, 4)	XO	MF1	MS2	RM	DVS
&			I2							
*>			I2							
::			I2							
>>			I2							

C.4 文脈依存語表

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。

ALIGNED	USAGE 句	I2
ARITHMETIC	OPTIONS 段落	I2
ATTRIBUTE	報告書記述項 及び 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 及び SET 属性 文	I2, XO, MF3, MS2
BLINK	画面記述項 及び 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 句 及び INTERFACE 句	I2
FOREGROUND-COLOR	画面記述項	I2, XO, MF3, MS2
FOREGROUND-COLOUR	画面記述項	I2, MF3
FOREVER	RETRY 句	I2
FULL	画面記述項	I2, XO, MF3, MS2
HIGH	ACCEPT 文 及び DISPLAY 文	RM
HIGHLIGHT	画面記述項 及び 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	LOCALIZE 句 及び SET 文	I2
LC_CURRENCY	LOCALIZE 句	I2
LC_MESSAGES	SET 文	I2
LC_MONETARY	SET 文	I2
LC_NUMERIC	SET 文	I2
LC_TIME	SET 文	I2
LOCALE	ALPHABET 句、PICTURE 句、SET 文 及び 特殊名(SPECIAL-NAMES) 段落	I2
LOCALIZE	OPTIONS 句	I2

LOW	ACCEPT 文 及び DISPLAY 文	RM
LOWLIGHT	画面記述項 及び SET 属性 文	I2, XO, MF7
MANUAL	LOCK MODE 句	I2, MF1, MS2
MULTIPLE	LOCK ON 句	all dialects
NONE	DEFAULT 句	I2
NORMAL	STOP 文	I2
NUMBERS	COLUMN 句 及び LINE 句	I2
ONLY	ALLOW 句、SHARING 句 及び SHARING 句	I2, MF11
PARAGRAPH	EXIT 文	I2, MF7
PREVIOUS	READ 文	I2, MF3
RECURSIVE	プログラム名(PROGRAM-ID)段落	I2, C370
RELATION	ERROR 句	I2
REQUIRED	画面記述項	I2, XO, MF3, MS2
REVERSE	ACCEPT 文 及び DISPLAY 文	RM
REVERSE-VIDEO	画面記述項 及び SET 属性 文	I2, XO, MF3, MS2
SECONDS	RETRY 句	I2
SECURE	画面記述項	I2, XO, MF3, MS2
SIGNED	USAGE 句	I2
STEP	OCCURS 句	I2
STRONG	TYPEDEF 句	I2
SYMBOL	CURRENCY 句	I2
UCS-2	ALPHABET 句	I2
UCS-4	ALPHABET 句	I2
UNDERLINE	画面記述項 及び SET 属性 文	I2, XO, MF3, MS2
UNSIGNED	USAGE 句	I2
UTF-8	ALPHABET 句	I2
UTF-16	ALPHABET 句	I2
YYYYDDD	ACCEPT 文	all dialects
YYYYMMDD	ACCEPT 文	all dialects

用　語　集

概要

この用語集は、このCOBOLシステムで使用される用語の意味を定義したものである。したがって、ここに取り上げた用語の意味は、他の用途で使われた同じ用語とは異なることがある。

以下に説明する用語は、参考資料または入門資料とすることを目的としているので、マニュアルの言語仕様を読む前に目を通しておくとよい。用語の定義は、簡潔な説明にとどめてあり、詳細な構文規則は含んでいない。

この用語集は、COBOLの方言を含まない。

定義

（あ行）

あて先（destination）

待ち行列から伝送する、受信側の記号名。

暗黙区分（implicit segment）

コード区分の大きさを調整するために、COBOLシステムによって作成される区分。

暗黙範囲符（implicit scope terminator）

先行する文の有効範囲の終わりを区切る、分離符の終止符（.）。または、文の中の指定で、それが出て来ることにより、先行する指定が含まれる文の有効範囲を区切るもの。

一意名（identifier）

データ項目に名前を付ける、構文的に正しい文字列と分離符との組合わせ。

関数ではないデータ項目を指すときは、一意名は、データ名とその参照の一意性を保つために必要な修飾語、添字、部分参照から構成される。関数を指すときは、関数一意名を使用する。関数の参照、修飾、添字付け、部分参照は、一般形式の規則の中で禁止されている場合がある。

印字位置（column）

印字行中の文字位置。

受取り側項目（receiving item）

画面節中のPICTURE句において、TO指定またはUSING指定が対象とするデータ項目。

英字（alphabetic character）

下記の文字集合に属する文字。

A，B，C，D，E，F，G，H，I，J，K，L，M，N，O，P，Q，R，S，T，U，V，W，X，Y，Z，a，b，c，d，e，f，g，h，i，j，k，l，m，n，o，p，q，r，s，t，u，v，w，x，y，z、空白。

英数字（alphanumeric character）

計算機の文字集合中の、任意の文字または数字。

英数字関数（alphanumeric function）

関数のうち、得られる値が、計算機の文字集合の中の1文字以上の文字から構成されるもの。

演算符号（operational sign）

数字データ項目または数字定数に付けて、値が正であるか負であるかを示す算術符号。

送出し側項目（sending item）

画面節中のPICTURE句において、FROM指定またはUSING指定が対象とするデータ項目。

（か行）

カーソル（cursor）

CRT画面上の記号で、現在入出力の対象となっている行とカラムの座標位置の目印となる。

外部データ（external data）

プログラム中で、外部データ項目および外部ファイル結合子として記述したデータ。

外部ファイル結合子（external file connector）

実行単位の中で、1つ以上の実行用プログラムによって呼び出すことのできるファイル結合子。

カウンタ（counter）

他の数値に基づいて、値を増減したり、ゼロまたは任意の正か負の値に変更したり、再設定したりするために使用する、データ項目。

拡張モード（extend mode）

EXTEND指定をしてOPEN文を実行した後の、ファイルの状態。このファイルに対してCLOSE文を実行するまで、拡張モードのままである。

可変反復データ項目（variable-occurrence data item）

反復回数が変えられる、表の中のデータ項目。そのデータ記述項にはOCCURS DEPENDING ON句が含まれるか、またはそのような項目の下位に属する。

画面記述項（screen description entry）

データ部の画面節中の記述項。レベル番号と、それに続く画面名と、一連の画面句から構成される。画面名は指定しなくてもよく、画面句は必要なものを指定する。

この記述項は、形の上ではデータ記述項とよく似ている。ただし、データ記述項は記憶領域を定義するのに対して、画面記述項は画面領域を定義する。

画面項目（screen item）

画面記述項がその属性を定義する、画面上の項目。

画面節（screen section）

データ部中の最後の節。この節で、ACCEPT文の書き方4およびDISPLAY文の書き方2を通じて呼び出す、画面領域のレイアウトを定義する。

カラム（column）

文字位置。左端の文字位置を1として、順次右側に1ずつ繰り上げて番号付けされる。

仮原文（pseudo-text）

仮原文区切り記号（==）によって囲まれた、一連の文字列や分離符。ただし、仮原文区切り記号は含まない。

仮原文区切り記号（psuedo-text delimiter）

等号（=）を2つ並べたもの。仮原文を区切るために使用する。

環境句（environment clause）

環境部の記述の一部として書く句。

完結文（sentence）

1つ以上の文を連ね、その最後の文を終止符（.）に続く空白で止めたもの。

関数（function）

関数は、呼び出されて実行されると作成者が設定した機構に従って、その時点で該当する一時データ項目の値を定める。

関数一意名（function-identifier）

関数を参照する、構文的に正しい文字列と分離符の組合わせ。関数によって表わされるデータ項目は、関数名と、引数がある場合にはその引数とによって、一意に識別される。

関数一意名には、部分参照を適用することができる。

英数字関数を指す関数一意名は、一般形式中の一意名を指定できる箇所であればどこにでも指定できる。ただし、何らかの制限がある場合がある。

整数関数または数字関数を指す関数一意名は、一般形式中の算術式を指定できる箇所であればどこにでも指定できる。

関数名（function-name）

関数の名前。（関数の項を参照）

キー（key）

レコードの所在を識別するためのデータ項目、またはデータの整列順を決定するデータ項目の組。

記号関数（symbol function）

データの種類を表わすために、PICTURE句内で指定された文字を使用すること。

記述項（entry）

COBOL原始プログラムの見出し部、環境部、データ部に書く一連の句。分離符の終止符（.）で止めたもの。

記述項の左辺（subject of entry）

データ部内の記述項において、レベル指示語またはレベル番号の直後に来る、作用対象（オペランド）または予約語。

基本項目（elementary item）

論理的にさらに分割されることのない形で記述されている、データ項目。

逆編集（de-edit）

数字編集データ項目からすべての編集文字を論理的に除去して、編集前のそのデータ項目の値に戻すこと。

行順ファイル編成（line sequential file organization）

順ファイルの一種であって、ホスト・オペレーティングシステムによって生成される、テキスト・ファイル形式の可変長レコードを含む。

共通プログラム（common program）

あるプログラム中に含まれているプログラムであるが、その親プログラム中に直接的または間接的に含まれる任意のプログラムからも呼出せるプログラム。

句（clause）

記述項の属性を指定するために並べた、一連のCOBOLの文字列。

句読文字（punctuation character）

下記の文字がある。

文字	意味
,	コンマ
;	セミコロン
.	終止符
"	引用符
(	左かっこ
)	右かっこ
	空白
=	等号
'	アポストロフィ

区分番号（segment-number）

手続き部内の節を、区分化の目的で分類するための利用者語。区分番号には、0から9までの数字だけが使用できる。区分番号は、1桁または2桁の数字で表わすことができる。

組合せ条件（combined condition）

2つ以上の条件を、論理演算子の"AND"または"OR"で結んだ条件。

計算機名（computer-name）

原始プログラムを翻訳（コンパイル）したり、実行用プログラムを実行したりする計算機を識別するためのシステム名。

形式（format）

データを整える形。

けた位置（digit position）

1桁の数字を収めるのに必要な、物理的記憶単位。その大きさは、該当するデータ項目の用途によって変わる。

言語名（language name）

特定のプログラミング言語を指定する、システム名。

現在レコード（current record）

ファイルのレコード領域中で、利用できる状態にあるレコード。

原始プログラム（source program）

このマニュアルにおいては、原始プログラム（ソースプログラム）とは、見出し部で始まり手続き部の末尾で終わる、文法的に正しい一連のCOBOLの文を指す。混同される危険がない文脈においては、原始プログラムの代わりに、単にプログラムという語を使うことがある。

原文語（text-word）

COBOL登録集、原始プログラム、仮原文の中の、境界Aから境界Rの間の1文字または一連の文字のうち、下記のもの。

1. 分離符。ただし、空白、仮原文区切り記号、文字定数の両端の区切り記号を除く。左かっこと右かっこは、登録集、原始プログラム、仮原文の文中のどこにあっても常に原文語とみなされる。
2. 定数。文字定数の場合、その両端を囲む引用符を含む。
3. 上記以外の、分離符によって区切られた連続するCOBOLの文字。ただし、注記行および語COPYを除く。

原文名（text-name）

登録集原文（ライブラリ・テキスト）を識別する、利用者語。

語（word）

30字以下の文字列。利用者語、システム名、予約語、関数名として使用される。

降順キー（descending key）

データ項目の比較規則に従って、値が最大のものから最小のものの順にデータを整列させるために使用するキー。

更新項目（update field）

画面項目のうち、その記述中にUSINGが指定されているもの。

構成節（configuration section）

環境部内にあって、翻訳用計算機および実行用計算機の全般的な仕様を記述する節。

構文（syntax）

プログラムをコーディングするときに、各要素を組み合わせて並べる順序、またその規則。

固定方式（fixed format mode）

画面上の数字項目および数字編集項目にデータを入力するときの、省略時解釈の方式。

この方式では、入力されたデータは形式を整えられて画面に表示される。また、キーボードからデータを入力するにつれて、該当する項目に指定されているPICTUREの内容に応じて、カーソルが移動される。

"+"と"-"と小数点以外の文字は受け付けられない。編集項目中の挿入文字は、カーソルが前後に移動する際に、飛ばされる。符号を表わすものは正規の仕様に従って変更される。浮動記号は項目中を左右に移動される。数字が挿入されたり削除されたりするのに応じて、挿入記号が表示されたり消去されたりする。

固有の大小順序（native collating sequence）

実行用計算機段落に指定した、計算機に固有の省略時解釈の文字の大小順序。

固有文字集合（native character set）

実行用計算機段落に指定した、計算機に作成者が定義した文字集合。

（さ行）

最右端（low order end）

文字列の右端の文字。

最左端（high order end）

文字列の左端の文字。

作業場所節（working-storage section）

データ部内にあって、作業場所データ項目を記述する節。作業場所節は、独立項目または作業場所レコードのどちらか一方または両方から構成される。

索引ファイル（indexed file）

索引編成のファイル。

索引編成（indexed organization）

ファイル内の各レコードを、レコード中の1つ以上のキーの値によって識別する、永続的な論理ファイル構造。

作用対象（operand）

作用対象（オペランド）の一般的な定義は、「操作の対象となる要素」である。このマニュアルの記述では、一般形式の中に日本語で示した語が、作用対象を表わしている。日本語で示した語は、作用対象によって表わされたデータ項目を暗黙的に参照することも表わす。

算術演算子（arithmetic operator）

下記の1文字または2文字の組合わせ。

文字	意味
+	加算
-	減算
*	乗算
/	除算
**	べき乗

算術式（arithmetic expression）

数字基本項目の一意名、数字定数、それらの一意名や定数を算術演算子でつないだもの。または、算術式を算術演算子でつないだもの。または、かっこで囲んだ算術式。

参照キー（key of reference）

索引ファイル中のレコードを呼び出すために、現在使用されているキー。

システム名（system-name）

操作環境に関する情報を伝えるために使用する、COBOLの語。

実行時（run-time, execution time）

実行用プログラムを実行する時点。

実行単位（run unit）

1つ以上のプログラムの組。実行時に、問題解決の単位として相互に作用する。

実行用計算機（object-computer）

実行用プログラムを実行する計算機。また、これを記述する、環境部内の段落の名前。

実小数点（actual decimal point）

データ項目中の小数点の位置を、終止符（.）またはコンマ（,）を使用して、実際に表現したもの。

指定（phrase）

COBOLの手続き文または句の中で、詳細仕様を指定するために使用する語。

指標（index）

表の中の個々の要素を識別するために使用する、計算機の記憶領域またはレジスタ。

指標付きデータ名（indexed data-name）

データ名とその後ろにかっこで囲んで付けた、1つ以上の指標名からなる一意名。

指標データ項目（index data item）

指標名に関連する値を格納する、データ項目。

指標名（index-name）

表に付けた指標の名前を表わす、利用者語。

修飾語（qualifier）

1. 一意名を指定するときに、同じデータ階層中の下位のレベルのデータ名と共に使用するデータ名。
2. 段落名を指定するときに共に指定する、その段落を含む節名。
3. 原文名を指定するときに共に指定する、その原文を含む登録集（ライブラリ）名。

修飾されたデータ名（qualified data-name）

一意名のうち、連結語のOFまたはINの後にデータ名を書いた修飾語の組を、データ名の後ろにいくつか続けたもの。

集団項目（group item）

一連の基本項目の組に、名前を付けたもの。

自由方式（free format mode）

固定方式（画面上の数字項目および数字編集項目にデータを入力するときの省略時解釈）、とは別の方式。

この方式では、カーソルが入力項目内にある間は、キーボードから入力したデータはそのまま入力項目内に保持される。カーソルが入出力項目の外に出された時点ではじめて、PICTURE仕様に従ってデータが編集される。このとき、数字と符号と小数点以外のすべての文字は無視される。そして、取り上げられた数字が編集されて格納される。この作業は、同じPICTUREのデータ項目を転記するときの、COBOLの規則に従って行われる。

得られた数値は、画面上に表示されるのが普通である。

終了記号（message indicator）

通信管理システムに対して、通信の終わり（切れ目）を知らせるための論理記号。下記のものがある。

EGI　通信群の終わり
EMI　通信文の終わり
ESI　通信行の終わり

EGIとEMIとESIは、階層関係をもつ。したがって、EGIはEMIとESIを含み、EMIはESIを含む。このため、通信行は、EMIによってもEGIによっても終了される。また、通信文は、EGIによっても終了される。

出力項目（output field）

画面項目のうち、FROM指定を含むもの。

出力手続き（output procedure）

SORT文の実行中に整列が完了した後で、またはMERGE文の実行中に併合順の次のレコードが選択された後で制御が移される、一連の手続き。

出力ファイル（output file）

出力モードまたは拡張モードで開かれたファイル。

出力モード（output mode）

OUTPUT指定またはEXTEND指定をしてOPEN文を実行した後の、ファイルの状態。そのファイルにCLOSE文を実行するまで、出力モードのままである。

主レコードキー（prime record key）

索引ファイル中のレコードを、一意に識別するキー。

順ファイル（sequential file）

順編成のファイル。

順編成（sequential organization）

ファイル中にレコードを格納したときに確立される先行・後行関係に基づいてレコードを識別する、永続的な論理ファイル構造。

順呼出し（sequential access）

論理レコードを、ファイル中のレコードの順序によって決まる連続した先行レコードと後行レコードの順に、読み込んだり書き出したりする呼出し法。

条件（condition）

真理値を決定するための、プログラムの実行時の状態を指定する記述。

この言語仕様において、一般形式の「条件」（条件-1、条件-2、など）の中または条件を参照する中に、「条件」（条件-1、条件-2、など）という用語があるとき、これは真理値を定める条件式を表わす。

条件式を構成するものには、単純条件（かっこで囲んでもよい）、組合せ条件（構文的に正しい単純条件の組合わせからなる）、論理演算子、かっこがある。

条件式（conditional expression）

IF文、PERFORM文、EVALUATE文、SEARCH文の中に指定された、単純条件または複合条件。単純条件（Simple Condition）および複合条件（Complex Condition）の各項を参照。

条件文（conditional statement）

条件の真理値を判定して、その値によって、実行用プログラムのその後の処理を分ける文。

条件変数（conditional variable）

1つ以上の値に条件名を割り当てられたデータ項目。

条件名（condition name）

条件変数がとることのできるすべての値の中の、ある特定の値や、値の組や、値の範囲に付けた利用者語。または、作成者が定義したスイッチまたは装置に付けた利用者語。

条件名条件（condition-name condition）

ある条件変数の値が、その条件変数に関連する条件名に割り当てられている値の組に含まれるかどうかを問う命題。この結果によって真理値が定まる。

条件名データ記述項（78 level-description-entry）

条件名を記述するデータ記述項。レベル番号に78を使用することになっている。

昇順キー（ascending key）

データ項目の比較規則に従って、値が最小のものから最大のものの順にデータを整列させるために使用するキー。

字類条件（class condition）

作用対象（オペランド）がすべて英字であるか数字であるかを問う命題。この結果によって真理値が定まる。

さらに、小文字の英字、大文字の英字、環境部の特殊名段落中でCLASS句によって定義されている文字集合中の文字、だけを含むことを検査できる。

真理値（truth value）

ある条件を評価した結果の値。「真」または「偽」のどちらかで表わされる。

スイッチ状態条件（switch-status condition）

作成者が定義した「オン」または「オフ」の状態に設定できるスイッチが、指定された状態になっているかどうかを問う命題。この結果によって真理値が定まる。

数字（numeric character）

下記の文字がある。

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

数字関数（numeric function）

字類および項類が数字に属する関数であるが、関数評価上の何らかの理由によって、整数関数の条件を満たさないもの。

数字項目（numeric item）

0から9の数字によってだけ表わされる値を取る、データ項目。符号付きの場合は、"+"と"-"またはそれ以外の演算符号を含むことができる。

数字定数（numeric literal）

いくつかの数字からなり、小数点と算術符号を含むことができる定数。
小数点は、右端に来てはならない。算術符号がある場合は、左端に来ること。

正書法（reference format）

COBOLの原始プログラムをコーディングする、標準的な書き方。

整数（integer）

1. 小数点の右側に有効数字のない数字定数。
2. データ部中に定義された数字データ項目。小数点の右側に有効数字のないもの。

一般形式の中に「整数」と示されている場合、これは整数である数字定数でなければならない。また、符号が付いていてもゼロであってもならない。ただし、書き方の規則中で許されている場合は、例外である。

整数関数（integer function）

項類が数字の関数であって、どのような場合にも、その関数を評価した結果として返される値の小数部分がゼロであると定義されるもの。

正負条件（sign condition）

データ項目または算術式の代数値が、正であるか負であるかゼロであるかを問う命題。この結果によって真理値が定まる。

整列併合用ファイル記述項（sort-merge file description entry）

データ部のファイル節中の記述項。レベル指示語のSDと、それに続くファイル名と、必要であれば一連のファイル句から構成される。

整列用ファイル（sort file）

SORT文によって整列（ソート）される、レコードの集合。整列用ファイルは、整列機能によってだけ作成され、使用される。

節（section）

節の見出しと、節の本体から構成される。冒頭に節の見出し、その後ろに本体としての一連の段落、または記述項が続く。ただし、段落または記述項がない場合もある。

節の見出し（section header）

環境部、データ部、手続き部の節の始まりを示す語の組。終止符（.）に続く空白で止める。

環境部およびデータ部では、節の見出しに用いる語は予約語である。
下記のものがある。

環境部

```
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
```

データ部

```
FILE SECTION.
WORKING-STORAGE SECTION.
LOCAL-STORAGE SECTION.
LINKAGE SECTION.
COMMUNICATION SECTION.
REPORT SECTION.
SCREEN SECTION.
```

手続き部では、節の見出しに用いる語は、節名に続く予約語のSECTIONと節番号から成る。ただし、節番号はなくてもよい。

節名（section name）

手続き部内の節に付ける名前として使用する、利用者語。

宣言完結文（declarative-sentence）

単一のUSE文を、分離符の終止符（.）で止めた翻訳指示完結文。

宣言部分（declaratives）

手続き部の冒頭に続けて書いた、特殊な目的用のいくつかの節。

最初の宣言節の冒頭には必要語DECLARATIVESを置き、最後の宣言節の末尾には必要語END DECLARATIVESを置く。宣言部分には、まず節の見出しを書き、続けてUSE翻訳指示完結文と必要な段落を書く。段落は、必要ない場合もある。

相対キー（relative key）

相対ファイル中の論理レコードを識別するためのキー。

相対ファイル（relative file）

相対編成のファイル。

相対編成（relative organization）

ファイル中のレコードを1以上の整数値によって一意に識別する、永続的な論理ファイル構造。この整数値は、ファイル中のレコードの論理的な順序を表わす。

想定小数点（assumed decimal point）

データ項目中に実小数点を含まないで指定された、小数点位置。想定小数点は、論理的には意味をもつが、物理的な表現はなされない。

添字（subscript）

表の中の、個々の要素を識別するための出現番号。この番号は、整数またはデータ名または指標名によって表わす。

データ名または指標名を使用する場合は、その後ろに、演算子の正号（+）または負号（-）に続けて整数を記述してもよい。関数の引数に添字付けした一意名を使用するときは、語ALLを添字として使用できる。

添字付きデータ名（subscripted data-name）

データ名の後ろに、いくつかの添字をかっこで囲んで付けたデータ名。

（た行）

単項演算子（unary operator）

変数または算術式の中の、左かっこの前にある正号（+）または負号（-）。それぞれ、式に+1または-1を掛ける効果をもつ。

単純条件（simple condition）

下記の条件の1つ。

- 比較条件
- 字類条件
- スイッチ状態条件
- 条件名条件
- 正負条件
- 上記の単純条件をかっこで囲んだもの

端末（terminal）

対話型の入出力装置。表示画面または印字装置とキーボードから構成される。この装置を通じて、操作員がデータを入力したり視覚データを受け取ったりすることができる。

段落（paragraph）

見出し部および環境部においては、段落の見出しに続く、いくつかの記述項（記述項がない場合もある）。

手続き部においては、段落名（終止符に続く空白で止める）に続く、いくつかの文（文がない場合もある）。

段落の見出し（paragraph header）

見出し部および環境部で、段落の始まりを示す予約語。終止符（.）に続く空白で止める。

下記のものがある。

見出し部

```
PROGRAM-ID.
AUTHOR.
INSTALLATION.
DATE-WRITTEN.
DATE-COMPILED.
SECURITY.
REMARKS.
```

環境部

```
SOURCE-COMPUTER.
OBJECT-COMPUTER.
SPECIAL-NAMES.
FILE-CONTROL.
I-O-CONTROL.
```

段落名（paragraph-name）

手続き部で、段落を識別し、その始まりを示す利用者語。

注記行（comment line）

原始プログラム中の標識領域に、星印（*）を付けた行。その行のA領域およびB領域に、計算機の文字集合中の任意の文字を記述できる。注記行は、プログラム中で説明のためだけに使用する。

もう1つ、特別の注記行を表わすものとして、標識領域に斜線（/）を書くことができる。この場合、その注記行が印字される前に、改ページされる。

注記項（comment entry）

見出し部中の記述項であって、計算機の文字集合中の任意の文字を含めることができる。

通貨編集用文字（currency symbol）

特殊名段落中で、CURRENCY SIGN句によって定義された文字。COBOL原始プログラム中にCURRENCY SIGN句を指定しないと、通貨編集用文字は通貨記号と同じになる。第3章「中核」の「特殊名段落」の節を参照。

次の実行完結文（next executable sentence）

現在の文の実行が完了した後で制御が移される、次の完結文。

次の実行文（next executable statement）

現在の文の実行が完了した後で制御が移される、次の文。

次のレコード（next record）

ファイル中の現在レコードの、論理的に次のレコード。

定数（literal）

記述したとおりの値が、暗黙に設定される文字列。

定数名（constant-name）

固定値（定数）の名前として付ける、利用者語。

定数項目（literal field）

画面上の基本項目であって、その記述項中にPICTURE句が含まれないもの。

データ記述項（data description entry）

データ部中の記述項。レベル番号と、それに続くデータ名と、必要な一連のデータ句とから構成される。

データ句（data clause）

データ部のデータ記述項の中の句。データ項目の性質を表わす情報を含む。

データ項目（data item）

COBOLプログラムまたは関数評価の規則によって定義される、データ（定数を除く）の単位。

データ辞書（data dictionary）

COBOLシステムによって作成され、記憶領域中に保持される表。各利用者語に関する情報を記録している。

データ名（data-name）

データ部中のデータ記述項に記述する、データ項目の名前に付ける利用者語。一般形式において「データ名」と示されているときは、添字も指標も付けていないものを指す。ただし、書き方に関する規則に、添字または指標を使えると明記してある場合は例外である。

手順名（routine-name）

COBOL以外の言語で書かれた手続き（ルーチン）を識別するための利用者語。

手続き（procedure）

手続き部内の、1つまたは論理的に連続したいくつかの段落。または、1つまたは論理的に連続したいくつかの節。

手続き部の終わり（end of procedure division）

COBOL原始プログラム中の、物理的な終端。これ以降には、プログラム手続きは現れない。

手続き名（procedure-name）

手続き部内の段落または節に、名前を付けるために使用する利用者語。段落名、または節名から成る。

デバッグ行（debugging line）

標識領域に、"D"または"d"を書いた行。

デバッグ節（debugging section）

USE FOR DEBUGGING文を含む節のこと。

動詞（verb）

翻訳（コンパイル）時、固有コード（機械語）の生成時、実行時に、COBOLプログラムが取る動作を記述する語。

動的呼出し（dynamic access）

呼出し方式で、OPEN文を実行してから、論理レコードを順序に従わずにディスク・ファイルから取り出したり格納したりできる（乱呼出し（Random Access）の項を参照）。また、論理レコードを順序に従ってディスク・ファイルから取り出すこともできる（順呼出しの項を参照）。

登録集原文（library-text）

COBOL登録集（ライブラリ）中の、一連の文字列および区切り文字。

登録集名（library-name）

目的コードを生成中に、COBOLシステムによって使用されるCOBOL登録集原始ファイルに名前を付ける利用者語。

特殊名（special-names）

作成者語に利用者が付ける呼び名。また、これを記述する環境部内の段落。

特殊文字（special character）

下記の文字がある。

文字	意味
+	正号
-	負号
*	星印（アスタリスク）
/	斜線（スラッシュ）
=	等号
\	通貨記号
,	コンマまたは小数点
;	セミコロン
.	終止符（ピリオド）または小数点
"	引用符
(	左かっこ
)	右かっこ
>	より大きい記号
<	より小さい記号
'	アポストロフィ
&	アンパサンド

特殊文字語（special-character word）

算術演算子または比較演算子として使用する、予約語。

特殊レジスタ（special registers）

COBOLシステムによって用意される、記憶領域。主として、特定のCOBOL機能を使う利用者用に作成される情報を記録するために使用される。

独立項目（noncontiguous item）

作業場所節、局所記憶節、連絡節中の基本データ項目で、他のデータ項目と階層関係をもたないもの。

独立データ記述項（77 level-description-entry）

独立データ項目を記述する、データ記述項。レベル番号に77を使用することになっている。

独立のプログラム（separate program）

中に含んでいるプログラム以外の他プログラムとは、すべて別々に処理されるプログラム。

（な行）

内部データ（internal data）

プログラム中に記述されているデータで、外部データ項目と外部ファイル結合子をすべて除いたもの。連絡節中に記述したデータは、内部データとして扱われる。

内部ファイル結合子（internal file connector）

実行単位中の1つの実行用プログラムからだけ呼び出すことのできる、ファイル結合子。

入出力管理（i-o control）

環境部内にあって、入出力に関する実行用プログラムの要件を記述する段落の名前。この段落中に指定する事項には、入出力技法、再開始点、複数のデータ・ファイル間での同じ領域の共有、単一の入出力装置上への複数のファイルの格納、がある。

入出力節（input-output section）

環境部中にあって、ファイルの入出力に関する記述を書く節。この節で、プログラムで使用するファイルおよび外部媒体に名前を付け、プログラムの実行中にデータを伝送したり操作したりするために必要な情報を記述する。

入出力両用ファイル（input-output file）

入出力両用モードで開いたファイル。

入出力両用モード（i-o mode）

I-O指定をしてOPEN文を実行した後の、ファイルの状態。このファイルに対してCLOSE文を実行するまで、入出力両用モードのままである。

入力項目（input field）

記述中にTO指定が含まれる、画面項目。

入力手続き（input procedure）

整列用ファイルにレコードを引き渡すたびに実行される、一連の文。

入力ファイル（input file）

入力モードで開かれるファイル。

入力モード（input mode）

INPUT指定をしてOPEN文を実行した後の、ファイルの状態。このファイルに対してCLOSE文を実行するまで、入力モードのままである。

（は行）

発信源（source）

待ち行列へ通信文を送信する、発信元の記号名。

範囲明示文（delimited scope statement）

範囲符を明示した（記述した）文。

比較（relation）

比較演算子の項を参照。

比較演算子（relational operator）

比較条件を記述するために使用する、予約語や比較文字の組。下記のものがある。

比較演算子	意味
IS [NOT] GREATER THAN IS [NOT] >	より大きい、または、[より大きくない]
IS [NOT] LESS THAN IS [NOT] <	より小さい、または、[より小さくない]
IS [NOT] EQUAL TO IS [NOT] =	等しい、または、[等しくない]
IS GREATER THAN OR EQUAL TO IS >=	以上
IS LESS THAN OR EQUAL TO IS <=	以下
IS UNEQUAL TO IS <>	等しくない
EQUALS	等しい
EXCEEDS	より大きい

比較条件（relation condition）

ある算術式またはデータ項目の値が、他の算術式またはデータ項目の値と、指定した関係にあるかどうかを問う命題。この結果によって、真理値が定まる。比較演算子の項を参照。

比較文字（relation character）

下記の文字がある。

文字	意味
>	より大きい
<	より小さい
=	等しい
>=	以上
<=	以下
<>	等しくない

引数（argument）

関数の評価に使用される値を指定する、一意名または定数または算術式。

必要語（key word）

書き方の中に示されている語の中で、原始プログラム中でその書き方を使用するときに、必ず指定しなければならない予約語または関数名。

否定組合せ条件（negated combined condition）

かっこで囲んだ組合せ条件の直前に、論理演算子のNOTを付けたもの。

否定単純条件（negated simple condition）

単純条件の直前に、論理演算子のNOTを付けたもの。

表（table）

データ部でOCCURS句を使用して定義する、論理的に連続したデータ項目の集合。（テーブル）。

表意定数（figurative constant）

意味を表わす語を用いて指定する定数。このための予約語が用意されている。その予約語の記述から、COBOLシステムによって該当する定数値が生成される。

標識領域（indicator area）

COBOL原始コードの、左端の文字位置。ここに記入した値によって、その用途が示される。

標準データ形式（standard data format）

COBOLプログラムのデータ部内でデータの特性を記述するために用いる概念である。

この概念では、無限の長さと幅をもつページにデータを印字することを想定して、データの性質や特性を表現する。計算機の内部や何らかの外部媒体にデータを格納することには目を向けない。

表要素（table element）

表を構成する反復項目の中の、個々のデータ項目。

部（division）

COBOLプログラムを構成する最大の単位。見出し部、環境部、データ部、手続き部がある。各部は、部の見出しと、一定の規則に従って構成されるいくつかの節または段落から構成される。

ファイル（file）

レコードの集まり。

ファイル位置指示子（file position indicator）

ファイルの入出力を行う一連の操作において、次にどのレコードを呼び出すかを論理的に示すもの。この概念は、出力モードまたは拡張モードで開いたファイルには適用されない。ファイル位置指示子の設定に影響を及ぼすのは、OPEN文とREAD文とSTART文だけである。

ファイル管理（file-control）

環境部の中にあって、プログラムで使用するデータ・ファイルについて宣言する段落の名前。

ファイル記述項（file description entry）

データ部のファイル節中にあって、ファイルに関する情報を記述する部分。レベル指示語のFDから始まって、それに続くファイル名と、必要な一連のファイル句から構成される。

ファイル句（file clause）

データ部の、下記の記述項の中に記述する句。

ファイル記述（FD）
整列併合ファイル記述（SD）
通信記述（CD）

ファイル結合子（file connector）

ファイルに関する情報を収録した記憶領域。ファイル名と物理ファイルの間、およびファイル名とそのレコード領域の間をつなぐために使用される。

ファイル固有属性（fixed file attribute）

ファイルを作成したときに定まり、以降そのファイルが存在している間は変更できない、ファイルの属性情報。

具体的には、ファイルの編成（順編成、相対編成、索引編成）、主レコードキー、副レコードキー、符号系、レコードの最大の大きさと最小の大きさ、レコードの型（固定長か可変長か）、索引ファイル用のキーの文字の大小順序、物理レコードの最大の大きさと最小の大きさ、充てん文字、レコードの区切り文字、がある。

ファイル終了条件（at end condition）

下記のどれかの場合に発生する条件。

1. 順呼出しファイルに対するREAD文を実行中に、次のレコードがなくなったとき。
2. RETURN文を実行中に、整列併合用ファイルの次のレコードがなくなったとき。
3. SEARCH文を実行中に、関連するWHEN指定の条件を1つも満たすことなく表引き操作が終了したとき。

ファイル節（file section）

データ部内にあって、ファイル記述項とレコード記述項を含む節。

ファイル編成（file organization）

ファイルを作成するときに永続的に定まる、ファイルの論理構造。

ファイル名（file name）

ファイルの名前として付けた利用者語。データ部のファイル節のファイル記述項、または整列併合ファイル記述項中に指定する。

複合条件（complex condition）

1つ以上の論理演算子が、1つ以上の条件に作用する条件。否定単純条件、組合せ条件、否定組合せ条件の各項を参照。

副プログラム（subprogram）

サブプログラム。呼ばれるプログラムの項を参照。

副待ち行列（subqueue）

待ち行列を、論理的に階層分割したもの。

副レコードキー（alternate record key）

索引ファイル中のレコードを識別するための、主レコードキー以外のキー。

符号系名（alphabet-name）

使用する文字集合や、文字の大小順序に付けた名前。環境部の特殊名段落中で指定する、利用者語。

物理レコード（physical record）

ブロックの項を参照。

浮動小数点数定数（floating-point literal）

浮動小数点形式で表わした数量。仮数部と指数部に分けて表現される。

仮数部は、10を基数にした固定小数点定数であり、小数点を含まなければならない。指数部は、文字"E"の後ろに符号付きで小数点を含まない固定小数点定数を続けたものである。

浮動小数点数データ項目（floating-point data item）

下記の形で表現した数字。

1. 各数字を、仮数部と指数部の2つに分けて表現する。
2. 数字の値は、仮数部に指数部の値の10のべき乗を掛けて算出する。

部の見出し（division header）

部の始まりを示すいくつかの語を、終止符（.）と続く空白で止めたもの。下記のものがある。

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.
                  [{USING   }                              ]
PROCEDURE DIVISION [{        } データ名-1 [データ名-2] ...]
                  [{CHAINING}                              ]
```

部分参照（reference modification）

データ項目の一部を、データ項目として定義すること。このためには、元のデータ項目の文字位置の中で、新しいデータ項目の左端とする位置（オフセット）と、取り出す文字の数（長さ）を指定する。

プログラム終了見出し（end program header）

COBOL原始プログラムの終わりを示す見出し。下記の形式をしている。

```
END PROGRAM プログラム名 .
```

プログラム名（program-name）

COBOLの原始プログラムを識別する、利用者語。

ブロック（block）

データの物理的な単位。通常は1つ以上の論理レコードから成る。

大記憶ファイルの場合は、複数のブロックにまたがって1論理レコードを構成することもできる。

ファイル中のブロックの大きさとそのファイルの大きさとの間には、直接的な関係はない。また、1ブロックに複数の論理レコードが含まれる場合でも、1論理レコードが複数のブロックにまたがる場合でも、ブロックの大きさとファイルの大きさとの間には、直接的な関係はない。

ブロックと物理レコードとは、同義語である。

プロンプト文字（prompt character）

画面上で、空の文字位置を示して入力を要求する文字。

文（statement）

手続き部内で、動詞に続けて語や記号を文法的に正しく組み合わせて書いたもの。命令文。

分割キー（split key）

ファイル中のレコード内の隣接していないデータ項目を、論理的にいくつかつなげて1つのキーとすること。この形のキーは、START文とREAD文においてだけ使用できる。

例：

プログラム中に下記のように定義されているデータ項目があるとする。

```
01 REC.
   03 FORENAME     PIC X (10).
   03 PERSONNEL-NO PIC X (4).
   03 SURNAME      PIC X (15).
```

ここで、下記のように指定すると、

```
RECORD KEY IS FULLNAME = SURNAME FORENAME
```

FULLNAMEは、下記のように明示的に定義した集団項目と同様に扱われる。

```
   03 FORENAME     PIC X (10).
   03 SURNAME      PIC X (15).
```

分離符（delimiter, separator）

1つの文字列の終わりを区切って後続の文字列との間を分離する、1つの文字または連続するいくつかの文字。

併合用ファイル（merge file）

MERGE文によって併合（マージ）されるレコードの集合。併合用ファイルは、併合機能によってだけ作成され、使用される。

編集用文字（editing character）

下記の文字がある。

文字	意味
B	空白
0	ゼロ
+	正号
-	負号
CR	貸方（credit）
DB	借方（debit）
Z	ゼロ抑制
*	小切手改変防止
\	通貨記号
,	コンマまたは小数点
.	終止符または小数点
/	斜線（スラッシュ）

変数（variable）

実行用プログラムを実行することによって、値が変わり得るデータ項目。算術式の中で使用する変数は、数字基本項目でなければならない。

ポインタ項目（pointer item）

USAGE IS POINTER句またはUSAGE IS PROCEDURE-POINTER句を適用する、基本データ項目。

補助語（optional word）

記述を読みやすくするためだけに使用する、予約語。補助語が含まれている書き方を原始プログラム中で用いる場合、補助語は書いても書かなくてもよい。

翻訳時（object time, compile time）

COBOL原始プログラムが、COBOLシステムによって中間コードのプログラムに翻訳（コンパイル）される時点。

翻訳指示文（compiler directing statement）

中間コードを生成する間（翻訳時）に、指定した動作をコンパイラに行わせるための、動詞から始まる文。

翻訳用計算機（source-computer）

原始プログラムを翻訳（コンパイル）する計算機。また、これを記述する、環境部内の段落の名前。

（ま行）

待ち行列（queue）

伝送待ちまたは処理待ち状態にある、通信文や処理や印字ジョブなどの論理的な集合。

待ち行列名（queue name）

待ち行列中にある通信文または通信文の一部を呼び出すために、通信管理システム（MCS）に対して指定する、論理経路の記号名。

無効キー条件（invalid key condition）

索引ファイルまたは相対ファイルのキーに指定された値が、実行時に無効であると判定されたことによって引き起こされる条件。

無条件文（imperative statement）

無条件動詞から始まって、無条件に動作を行うことを指定する文。無条件文は、一連の無条件文で構成できる。

文字（character）

言語の、それ以上分割できない基本単位。

文字位置（character position）

用途がDISPLAYとして記述された標準データ形式の1文字を収めるために必要な記憶単位。

文字項目（nonnumeric item）

計算機の文字集合に属する、任意の文字の組合わせを値に取ることのできる、データ項目。文字項目の項類によっては、使用できる文字が制限される。

文字集合（character set）

COBOL言語の文字集合は、下記のすべての文字から成る。

文字	意味
0, 1, ..., 9	数字
A, B, ..., Z	大文字の英字
a, b, ..., z	小文字の英字
	空白（ブランク）
+	正号
-	負号
*	星印（アスタリスク）
/	斜線（スラッシュ）
=	等号
\	通貨記号
,	コンマまたは小数点
;	セミコロン
.	終止符または小数点
"	引用符
(	左かっこ
)	右かっこ
>	より大きい記号
<	より小さい記号
:	コロン
'	アポストロフィ
&	アンパサンド

注：

計算機の文字集合に小文字が含まれるときは、これらを文字列や原文語に使用できる。

各小文字は、対応する大文字と等しいものとする。ただし、文字定数として使用した場合は例外である。

文字定数（nonnumeric literal）

引用符で両端を囲んだ文字列。この文字列の中には、計算機の文字集合に属する任意の文字を入れることができる。文字定数の中に、1つの引用符を入れる場合は、引用符を2つ並べて記述する。

文字の大小順序（collating sequence）

計算機で処理できる文字を並べる順序。整列、併合、比較のときに使用される。

文字列（character-string）

文字をいくつか並べたもの。COBOLの語、定数、PICTURE文字列、注記項を形成する。

（や行）

呼ばれるプログラム（called program）

CALL文の右辺に指定されているプログラム。実行時に呼ぶプログラムと組合わされて、実行単位を形成する。

呼出し法（access mode）

ファイル中のレコードを操作する方式。（アクセス法）。

呼び名（mnemonic-name）

環境部において、指定された作成者語に付けた利用者語。（ニモニック）。

呼ぶプログラム（calling program）

他のプログラムに対してCALL文を実行し、そのプログラムを呼出すプログラム。

予約語（reserved word）

COBOL言語の一環として、あらかじめ決めてある語。予約語は、COBOLの原始プログラム中で、利用者語またはシステム名として使用してはならない。

（ら行）

ランタイムシステム（run-time system）

COBOLシステムによって生成されたコード（実行用プログラム）を解釈して、オペレーティングシステムとCRTへのインタフェースをとりながら、そのプログラムを実行できる状態にするソフトウエア。（RTS）。

乱呼出し（random access）

相対ファイルまたは索引ファイルに対して、論理レコードを読込み、削除、挿入するときに、対象とするレコードを、プログラム中のキー・データ項目の値によって指定できる呼出し法。

略記組合せ比較条件（abbreviated combined relation condition）

一連の比較条件を続けて記述するとき、共通する左辺、右辺または比較演算子を記述上省略した組合わせ条件。

利用者語（user-defined word）

句または文の書き方を満たすために、利用者が指定しなければならないCOBOLの語。

レコード（record）

論理レコードの項を参照。

レコードキー（record key）

索引ファイル中のレコードを識別するために使用するキー。主レコードキーと副レコードキーがある。

レコード記述項（record description entry）

1つのレコードに関する、データ記述項の全体。

レコード名（record name）

データ部中のレコード記述項に記述した、レコードに付ける名前として使用する利用者語。

レコード領域（record area）

ファイル節中のレコード記述項に記述したレコードを処理するために割り当てられた、記憶領域。

レベル指示語（level indicator）

ファイルの型またはデータ階層中の位置を示す、2文字の英字。

レベル番号（level-number）

論理レコードの階層構造中のデータ項目の位置を示すか、またはデータ記述項の特殊な性質を示すための、利用者語。レベル番号は、1桁または2桁の数字で表わす。1から49の範囲内のレベル番号は、論理レコードの階層構造中のデータ項目の位置を示す。

1から9までのレベル番号は、1桁の数字として書いてもよいし、前に0を付けて書いてもよい。レベル番号66，77，78，88は、特殊な性質のデータ記述項を示す。

連鎖されるプログラム（chained program）

CHAIN文の右辺に指定されたプログラム。

連鎖するプログラム（chaining program）

他のプログラムに対してCHAIN文を実行し、そのプログラムへ連鎖（チェイン）するプログラム。

連続項目（contiguous item）

データ部内で、連続的に記述されている項目。これらの項目間には、階層関係が作られる。

連絡節（linkage section）

呼ばれるプログラムのデータ部中にあって、呼ぶプログラム側で使用するデータ項目を記述する節。この節に記述するデータ項目は、呼ぶプログラムからも呼ばれるプログラムからも参照できる。

論理演算子（logical operator）

予約語のANDとORとNOT。

条件を記述するとき、ANDとORは、論理連結子として使用できる。NOTは、論理否定子として使用できる。

論理レコード（logical record）

最も包括的なデータ項目。レコード用のレベル番号には、01を当てる。

（英字）

COBOLの語（COBOL word）

語の項を参照。

CRT

操作員が視覚的にデータを受け取ることのできる出力装置。

OPEN文のモード（open mode）

あるファイルに対してOPEN文を実行してからCLOSE文を実行するまでの間の、ファイルの状態。どのモードを使用するかは、OPEN文中に指定する。INPUT（入力），OUTPUT（出力），I-O（入出力両用），EXTEND（拡張）がある。