第1章 COBOL入門

指導目標

本編では、ソフトウェアの作成手順の中の「プログラム開発」に必要な基礎知識であるプログラミング技法を教える。

ＣＯＢＯＬ言語は、コンピュータ処理に必要な考え方が「ＦＯＲＴＲＡＮ」や「Ｃ」などと同じようにコンピュータのアーキテクチャやオペレーティング・システムが異なっても、共通に使用できる。
また、ＣＯＢＯＬ言語にはＦＯＲＴＲＡＮやＣとの違いとして、ファイル処理や事務処理で必要な報告書作成について充実した機能が言語規約に定められている。

本章では、ソフトウェアの作成技法の基本事項として、プログラミング技法を学ぶ上で重要な言語規約（規格）の考え方をＣＯＢＯＬ言語中心に説明し理解させる。

また、ＣＯＢＯＬの発展の経緯からプログラムの構造やプログラムの書き方などの知識を理解させる。

内容のあらまし

内　　容	説　明	議　論	机上実習	計算機実習
はじめに	COBOLﾌﾟﾛｸﾞﾗﾑ例を説明しﾌﾟﾛｸﾞﾗﾑの特徴を解説する。			ＰＣ等で例題ﾌﾟﾛｸﾞﾗﾑを翻訳、実行する
コンパイル リンク	ｺｰﾃﾞｨﾝｸﾞから実行までの各段階の役割について解説する。	言語記述の互換性と実行環境の違いについて議論を行い言語規格に求められる実行環境の差異がどのように反映されているかを説明する。		簡単なﾌﾟﾛｸﾞﾗﾑを用いてｺﾝﾊﾟｲﾙ、ﾘﾝｸの役割について体験させる。
プログラムの構成	「一般形式」の記述でﾌﾟﾛｸﾞﾗﾐﾝｸﾞの構文の理解と書き方を説明する。	COMPUTE文を例に必要語と補助語の役割と文書化について議論する。		
COBOL言語の構造	COBOLでの個々の文字や文字列が組み合わされて原始ﾌﾟﾛｸﾞﾗﾑが構成することを説明する。			
正書法	COBOLｺｰﾃﾞｨﾝｸﾞで重要な正書法について説明する	Ｃ言語とCOBOLとの違いはどこからきているのかを議論する。	ﾌﾟﾛｸﾞﾗﾑ例をＣ言語で記述させ書き方の違いを体験させる。	文の継続についてｺﾝﾊﾟｲﾙと実行で確認を行う。
COBOLﾌﾟﾛｸﾞﾗﾑ構成	COBOLの４個の「部」の意味と言語上の役割を説明する。			

基礎編

第１章　ＣＯＢＯＬ入門

１．１　はじめに

ＣＯＢＯＬ（COmmon Business Oriented Language）は、１９６０年代初期に米国防省の標準言語として開発され、現在世界的に最も普及している。
英語に似た記述であり、データ記述と処理手続きを完全に分離したのが特徴である。
主な適用分野は事務処理である。

COBOLﾌﾟﾛｸﾞﾗﾑの構成

見出し部
環境部
データ部
手続き部

（１）ＣＯＢＯＬプログラムの例

事務処理の一例として、価格を入力して消費税を計算するＣＯＢＯＬプログラムを作成し、コンパイルから実行までを説明する。

COBOLｿｰｽﾌﾟﾛｸﾞﾗﾑ

```
IDENTIFICATION DIVISION.
PROGRAM-ID. ﾌﾟﾛｸﾞﾗﾑ名.
ENVIRONMENT  DIVISION.
CONFIGURATION  SECTION.
SOURCE-COMPUTER.   翻訳用計算記名.
OBJECT-COMPUTER.   実行用計算機名.
DATA  DIVISION.
WORKING-STORAGE  SECTION.
01  KAKAKU  PICTURE  9999999.   ←データ記述
01  ZEIKIN  PICTURE  ZZZ,ZZ9.
                                  処理手続き
PROCEDURE DIVISION.                  ↓
HAJIME.
    DISPLAY "価格を入力て下さい。消費税を計算します。".
    ACCEPT  KAKAKU.
    COMPUTE  ZEIKIN  =  KAKAKU  *  0.03.
    DISPLAY  "価格は" KAKAKU "円、消費税は"  ZEIKIN "円です。".
    STOP RUN.
```

（２）プログラムの概要

ＣＯＢＯＬソースプログラムは、次の順序の四つの部で構成される。

① 見出し部
プログラム名等のプログラムを識別するための情報を記述する。

```
IDENTIFICATION  DIVISION
PROGRAM-ID.  ﾌﾟﾛｸﾞﾗﾑ名.
```

② 環境部
入力ファイルや出力ファイルの構成等、プログラムを実行するための環境の情報を記述する。

```
ENVIRONMENT  DIVISION.
CONFIGURATION  SECTION.
SOURCE-COMPUTER.  翻訳用計算機名.
OBJECT-COMPUTER.  実行用計算機名.
```

③ データ部
ファイルのレコード構成等、プログラムで使用するデータの属性と構造を記述する。

```
DATA  DIVISION.
WORKING-STORAGE  SECTION.
01  KAKAKU  PICTURE  9999999.
01  ZEIKIN  PICTURE  ZZZ,ZZ9.
```

④ 手続き部
データの入力、計算や編集、出力等、プログラムの処理手続きをＣＯＢＯＬの文（命令）で記述する。

```
PROCEDURE DIVISION.
HAJIME.
    DISPLAY  "価格を入力して下さい。消費税を計算します。".
    ACCEPT  KAKAKU.
    COMPUTE  ZEIKIN  =  KAKAKU  *  0.03.
    DISPLAY  "価格は"  価格 "円、消費税は" ZEIKIN "円です。".
    STOP RUN.
```

ＣＯＢＯＬには手続きを記述するために特定の動作を指示する文（命令）がある。

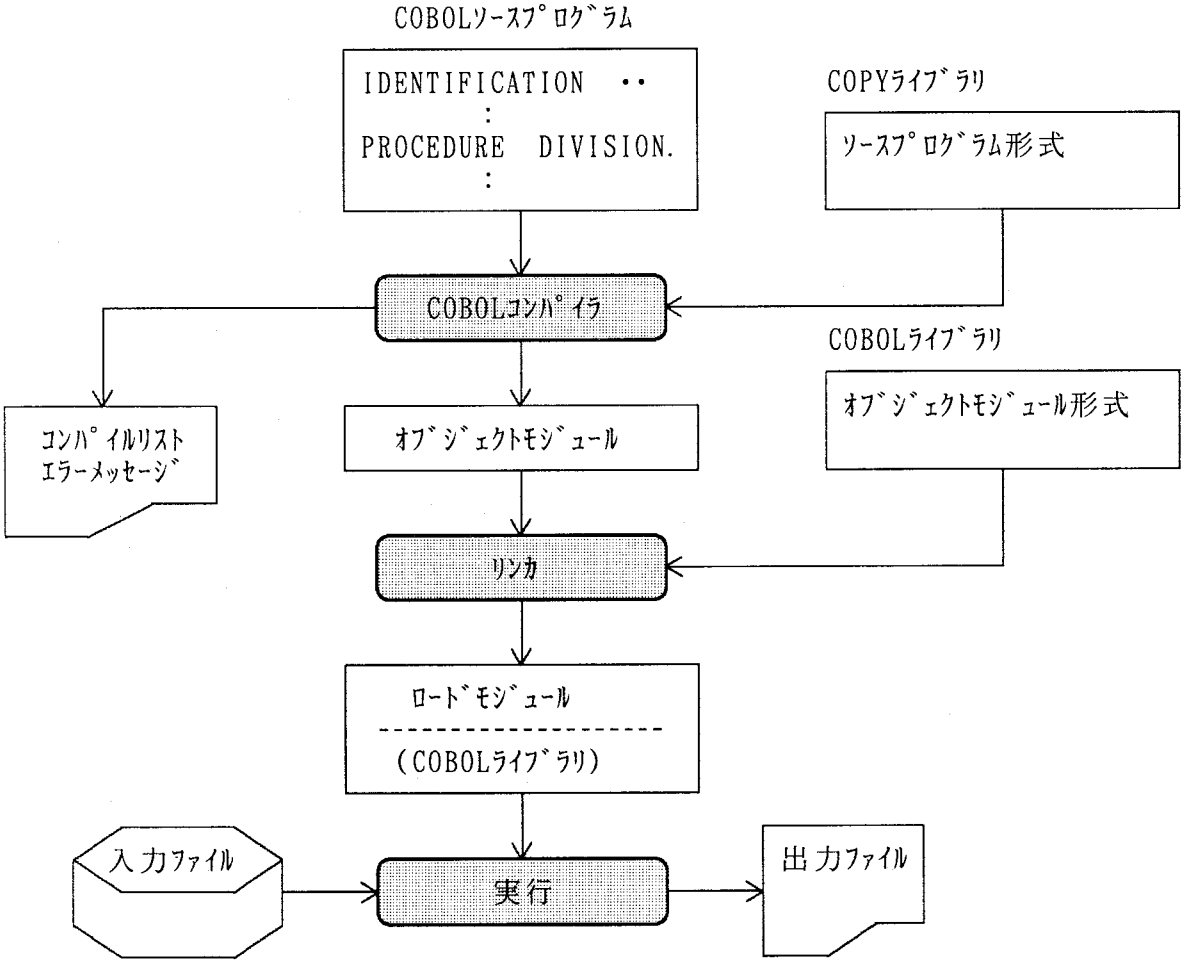
ACCEPT	GENERATE	RELEASE
ADD	GO TO	REWRITE
ALTER	INITIALIZE	SEND
CALL	INITIATE	SET
CANCEL	INSPECT	SORT
CLOSE	MERGE	START
COMPUTE	MOVE	STOP
CONTINUE	MULTIPLY	STRING
DELETE	OPEN	SUBTRACT
DISABLE	PERFORM	SUPPRESS
DISPLAY	PURGE	TERMINATE
DIVIDE	READ	UNSTRING
ENABLE	RECEIVE	WRITE
EXIT		USE

コンパイラー指示（原始文操作機能）として次のものがある。

COPY	REPLACE

（３）　コンパイル、リンク

ＣＯＢＯＬソースプログラムを実行するには、まず、ＣＯＢＯＬコンパイラでオブジェクトモジュールを作成する。次に、リンカでＣＯＢＯＬライブラリを結合して実行可能なロードモジュールを作成する。



（４）　実行

プログラムの実行は、プログラムで定義したファイルとオペレーティング・システムが管理している物理ファイルを関連付けして実行環境を整えてから実行する。

①パソコンの場合
単にプログラム名を指定して実行する。

```
＞プログラム名
```

②汎用コンピュータの場合
ＪＣＬ（Job Control Language）で使用する入力や出力ファイルを関連付けてから、プログラムを呼び出して実行する。

```
//ジョブ名  JOB  ・・・
//GO      EXEC  PGM=プログラム名
//STEPLIB   DD  DSN=ロードモジュールファイル,DISP=SHR
//INFILE    DD  DSN=入力データファイル,DISP=SHR
//OUTFILE   DD  DSN=出力データファイル,DISP=SHR
```

１．２　プログラムの構成

ＣＯＢＯＬ文一般形式は、次のように記述されている。

```
COMPUTE  {一意名１  [ROUNDED]}・・・  =    算術式１
      [ON  SIZE  ERROR  無条件文１]
      [NOT  ON  SIZE  ERROR  無条件文２]
      [END-COMPUTE]
```

文は各種の要素からなる。それらの要素の並べ方を一般的に記述したものを一般形式という。要素は大文字の語、括弧、特殊文字等である。文の要素は一個以上の空白等の分離符で区切る。

「一意名１」または「定数」は利用者語であって自由に構成できるが、予約語と一致してはいけない。
予約語の一覧は付録１にあるが、一般形式に含まれるCOMPUTE、ROUNDED、ON等はその一部である。なお、一般形式に含まれる予約語で下線を引いたもの（COMPUTE、ROUNDED等）を必要語、下線のないもの（ON等）を補助語という。

```
COMPUTE  GOUKEI  =  KAKAKU  *  1.03
```

必要語は、その文を使用するときは必ず書く。補助語は書いても書かなくてもよい。命令文の機能（動作）は変わらない。

```
COMPUTE   GOUKEI   =   KAKAKU   *   1.03   ON   SIZE ERROR ～
COMPUTE   GOUKEI   =   KAKAKU   *   1.03        SIZE ERROR ～
```

角括弧［］の部分は、指定しても省略してもよい。指定するかしないかで命令文の機能（動作）は異なる。

```
COMPUTE  GOUKEI   ROUNDED   =  KAKAKU  *  1.03
COMPUTE  GOUKEI              =  KAKAKU  *  1.03
```

反復記号「・・・」は、繰り返し指定できることを示す。

```
COMPUTE   GOUKEI  GOUKEI2              =  KAKAKU * 1.03
COMPUTE   GOUKEI  GOUKEI2  GOUKEI3    =  KAKAKU * 1.03
```

一般形式で波括弧｛　｝で囲んである部分は、その波括弧の中の一つを指定する。

１．２．１　文字集合

言語の、それ以上分割できない基本単位は文字とする。

ＣＯＢＯＬ文字集合（character　set）は、次の通り。

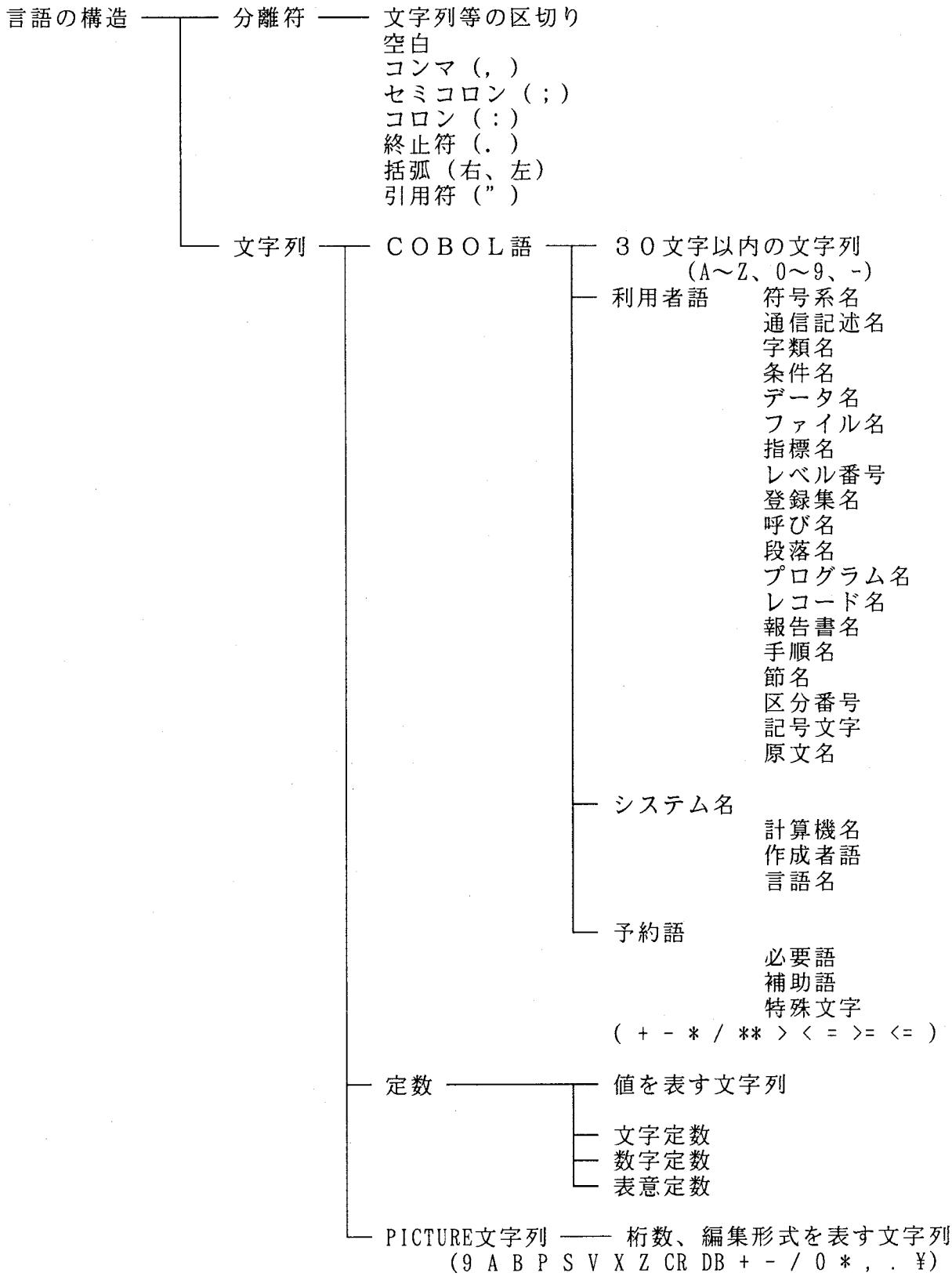
文　字	意　　味	
０～９	数字	
Ａ～Ｚ	英大文字	
ａ～ｚ	英小文字	文字定数や編集用文字を除いて英大文字と等価
	空白	分離符
＋	正号	算術演算子、編集用文字
－	負号	算術演算子、編集用文字、継続文字
＊	星印	算術演算子、編集用文字、注釈行
／	斜線	算術演算子、編集用文字、注釈行
＝	等号	比較文字、算術代入
￥	編集用文字	
,	コンマ	分離符（句読文字）、編集用文字
;	セミコロン	分離符
:	コロン	分離符
.	終止符	分離符　小数点、編集用文字
"	引用符	
（	左括弧	分離符
）	右括弧	分離符
＞	より大きい記号	不等号　比較文字
＜	より小さい記号	不等号　比較文字

文字定数、注記項、注記行には計算機の文字集合の任意の文字が使える。
漢字についての規定は特にないが、ＪＩＳの規格として、備考に次の記述がある。

備考　１．計算機によっては、文字集合にこの文字をすべて含むとは限らないので、他の文字で書き換えてもよい。
ＣＯＢＯＬ文字集合は、ＪＩＳ　Ｘ　０２０１（情報交換用符号）の部分集合である。通貨記号を除いてアメリカ規格ＡＮＳＩ　Ｘ３．４及び、ＩＳＯ６４６の国際標準版（International　Reference　Version）の部分集合である。

１．２．２　ＣＯＢＯＬ言語の構造

ＣＯＢＯＬ言語は、文字列と分離符（separator）をつないでいくことでソースプログラムが構成される。



（１）　用語の意味

ここでの用語はＣＯＢＯＬにおける意味の要約であり、他の言語に対して必ずしも同じ意味をもつものでない。

利用者語(user-defined　word)
　句又は文の書き方を満足するために、利用者が与えるＣＯＢＯＬの語。

符号系名(alphabet-name)
　環境部の特殊名段落において利用者が定義する名前であって、特定の文字集合及び文字の大小順序につける名前。

通信記述名(cd-name)
　データ部の通信節における通信記述項で記述され、通信管理システムとの論理的な連絡領域に命名する利用者語。

字類名(class　name)
　環境部の特殊名段落で定義した利用者語であり、データ項目の内容が字類名の定義で並べた文字だけで構成されているか否かで、真理値が決まる命題に付けた名前。

条件名(condition-name)
　条件変数がとりうる値の一部に付けた利用者語、又は作成者が定めるスイッチ若しくは装置の状態に付けた利用者語。"条件名"を一般形式中で使用する場合は、"条件名"は条件名と一意参照のために付けた修飾語又は添字の構文的に正しい組からなる一意なデータ項目の参照を意味する。

データ名(data-name)
　データ部のデータ記述項で書いたデータ項目を命名する利用者語。一般形式中で使われるデータ名は、規則で特に許されない限り部分参照、添字付け又は修飾してはならない。

ファイル名(file-name)
　データ部のファイル節中のファイル記述項又は整列併合用ファイル記述項において、ファイル結合子に命名する利用者語。

指標名(index-name)
　特定の表に関係付けられた指標を命名する利用者語。

レベル番号(level-number)
　データ項目の階層中の序列を指定したり、データ記述項の特別な性質を指定したりする１個又は２個の数字で表現する利用者語。０１から４９までのレベル番号は、レコード階層構造におけるデータ項目の位置付けを指定する。１から９までのレベル番号は、１けたで書いてもよいし、ゼロの後に有効数字を書いてもよい。レベル番号６６、７７及び８８は、データ記述項の特別な性質を指定する。

登録集名(library-name)
　原始プログラムの翻訳のためにコンパイラが使うＣＯＢＯＬ登録集を命名する利用者語。

呼び名(mnemonic-name)
　環境部で特定の作成者語と関連付けられる利用者語。

段落名(paragraph-name)
　手続き部の段落を識別するために段落の先頭に書く利用者語。

プログラム名(program-name)
　見出し部及びプログラム終わり見出しにおいて、ＣＯＢＯＬ原始プログラムを識別する利用者語。

レコード名(record-name)
　ＣＯＢＯＬプログラムのデータ部のレコード記述項で定義したレコードを命名する利用者語。

報告書名(report-name)
　データ部の報告書節中において、報告書記述項に記述する報告書を命名する利用者語。

手順名(routine-name)
　ＣＯＢＯＬ以外の言語で記述された手続きを識別する利用者語。

節名(section-name)
　手続き部の節を命名する利用者語。

区分番号(segnemt-number)
　区分化のために手続き部の節を分類する利用者語。区分番号は、１けた又は２けたの数字からなる。

記号文字(symbolic-character)
　利用者が定める表意定数を指定する利用者語。

原文名(text-name)
　登録原文を識別する利用者語。

システム名(system-name)
　操作環境との通信に使うＣＯＢＯＬの語。

計算機名(computer-name)
　プログラムを翻訳又は実行する計算機を識別するシステム名。

作成者語(implementor-name)
　作成者の処理系で使える固有の機能を参照するためのシステム名。

言語名(language-name)
　特定のプログラム言語を指定するシステム名。

予約語(reserved　word)
　ＣＯＢＯＬ原始プログラムで使われる決まった語であって、利用者語又はシステム名として使用してはならない。

必要語(key　word)
　原始プログラムを書くときに、ある書き方の中では必ず書かなけらばならない予約語。

補助語(optional　word)
　言語を読みやすくするだけのために、ある書き方の中に書く予約語であって、原始プログラムを書くときに書くかどうかは利用者に任される。

１．２．３　予約語と利用者語

ＣＯＢＯＬでの「語(word)」は次のもので形成する。

COBOL 語	予約語（reserved word） ＣＯＢＯＬ言語のなかで、あらかじめ定義された意味を持つ語。 必要語（key word）、補助語（optional word） 連結語（OF/IN AND/OR/NOT）、算術演算子（+-*/）、比較文字（<>=） 表意定数（ZERO SPACE HIGH-VALUE LOW-VALUE QUOTE） READ ﾌｧｲﾙ名 [NEXT] RECORD [INTO 一意名] key key optional key [AT END 無条件文1] optional key [NOT AT END 無条件文2] key opt. key [END－READ] key
	利用者語（user-defined word） 利用者が自由に定義する語。 英字、数字、－（ﾊｲﾌﾝ）で構成され、最大３０文字。語の最初と最後の文字に「－」は使用できない。（レベル番号は数字のみ） データ名、ファイル名、節名、段落名、レベル番号、条件名、指標名 01 データ名 ＰＩＣＴＵＲＥ Ｘ（１００）． ＲＥＡＤ ﾌｧｲﾙ名 ＩＮＴＯ ﾃﾞｰﾀ名 ＧＯ ＴＯ 段落名．
	システム名（system name） ＣＯＢＯＬ言語と計算機システムの環境を定義する語。 計算機名、作成者語、言語名 ＳＯＵＲＣＥ－ＣＯＭＰＵＴＥＲ． 計算機名． ＳＥＬＥＣＴ ﾌｧｲﾙ名 ＡＳＳＩＧＮ ＴＯ 作成者語．

ＣＯＢＯＬには予約語が約３６０種類ある。（付録－１参照）

ＦＯＲＴＲＡＮには予約語はない。キーワードか英字名（symblic name）かの識別は文脈できまる。

Ｃ言語の予約語は、次の通り。

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

１．２．４　語と分離符

文字列の後に分離符を書かなければならない。文字列と分離符をつないでソースプログラムができる。

```
文字列 分離符 文字列 分離符 文字列 分離符
```

①　空白（半角スペース）
１文字または２文字以上でも意味は変わらない。

```
MOVE ZERO TO KINGAKU.
MOVE     ZERO   TO    KINGAKU.
```

②　「, 」、「; 」
コンマやセミコロンの直後に空白を続けたものは分離符である。プログラムを読み易くするために用いる。（空白と同じ）

```
MOVE ZERO TO KINGAKU, TANKA, GOUKEI.
MOVE ZERO TO KINGAKU; TANKA; GOUKEI.
```

③　「. 」
終止符は完結文の終わりを示すため注意を要する。

```
IF  SUURYOU  >  0  THEN
                   COMPUTE  KINGAKU  =  TANKA  *  SUURYOU.
```

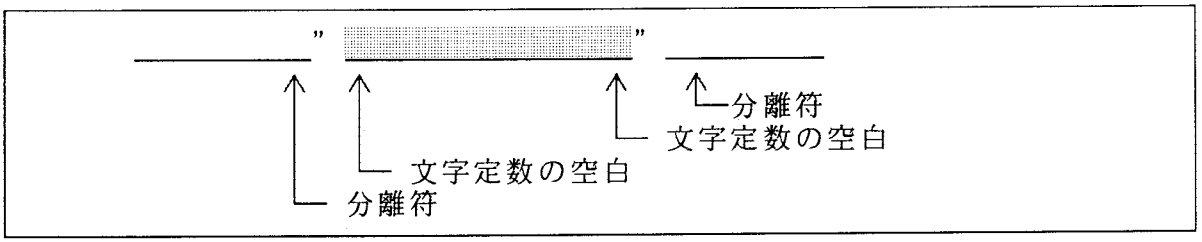
④　「(」、「) 」
添字、部分参照、算術式、条件式で用いる。

```
MOVE ZERO TO KINGAKU-TABLE (10).
MOVE ZERO TO TANKA-TABLE (X, Y).
COMPUTE GOUKEI = (KINGAKU1 + KINGAKU2) * 1.03.
```

⑤　「" 」
左引用の前は、空白か「(　」でなければならない。
右引用符の後は分離符（空白、コンマ、セミコロン、終止符、右括弧）でなければならない。
文字定数は対になっている引用符で囲む。
文字定数に引用符を含める場合は、２個連続した引用符で１文字の引用符を定義できる。

```
MOVE "ABC" TO ROMAJI.
IF ("XYZ" = XXXX) OR (AAAA = "ABC") THEN
```

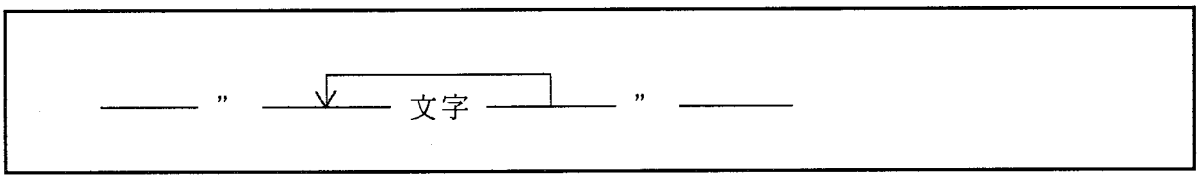
分離符の直前には、空白を置いてもよいし、置かなくてもよい。
分離符の直後には、空白を置いてもよいし、置かなくてもよい。
ただし、引用符の前後では空白の意味が異なる。



１．２．５　定数（literal）

（１）　文字定数（nonnumeric　literal）

両端を引用符（”）で囲まれた任意の文字列である。
項類（category）は「英数字」である。



最大文字数は、１６０文字（ＪＩＳ）である。
引用符（”）を文字定数に含める場合は、２文字連続して定義すると１文字の引用符として使用できる。

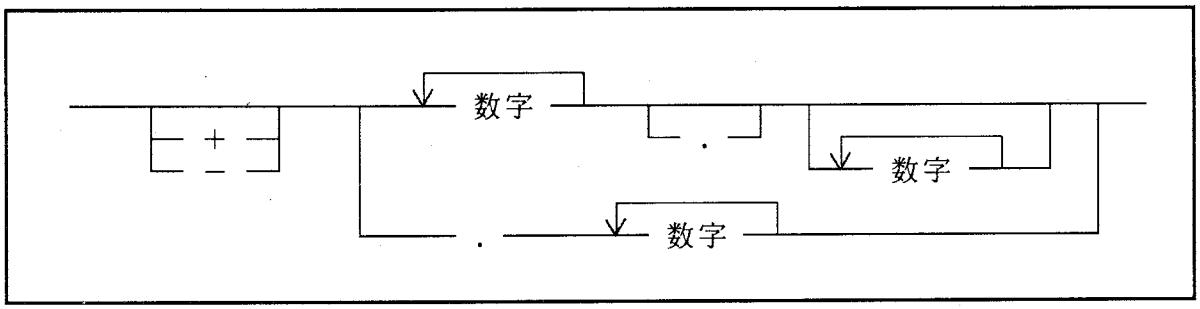
文字列の記述例　　記憶装置の状態

”ＡＢＣ” →	Ａ	Ｂ	Ｃ			
”１２３４５６” →	１	２	３	４	５	６
”ＣＡＴ”” Ｓ” →	Ｃ	Ａ	Ｔ	”	Ｓ	

文字列の不正例
”ＣＡＴ” Ｓ”

（２）　数字定数（numeric　literal）

０から９までの数字と、符号（+、-）と小数点（.）で構成される。
項類（category）は「数字」である。

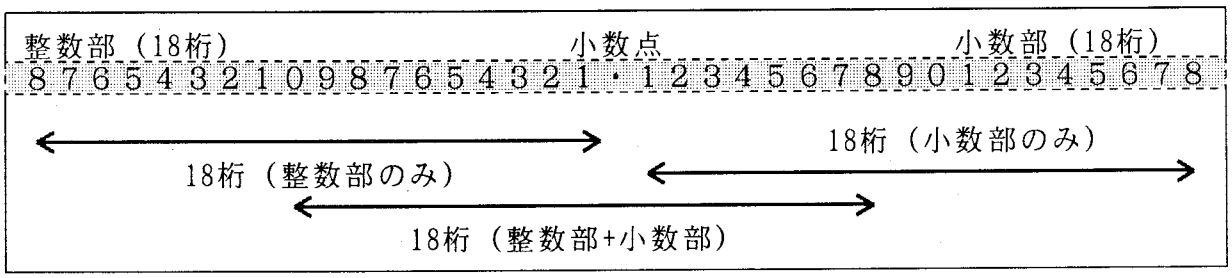


最大桁数は１８桁（符号、小数点は含まない）である。

数字定数の例
１２３０
－１２３４
＋１２３４
－５．６７
＋．０３

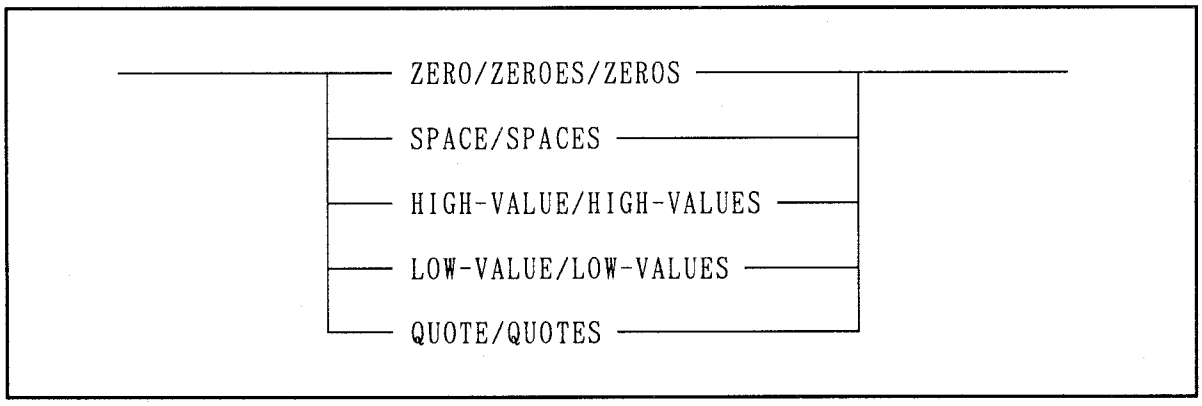
数字定数の不正例
１２３Ａ０
１２３４－
１２３４＋
－１０，０００

数字定数で表現できる範囲は次の通り。



（３）　表意定数（figurative　constant）

定数の値をＣＯＢＯＬの予約語で表現した定数である。
単数形（ZERO）と複数形（ZEROS、ZEROES）は同じ意味であり、どちらを用いてもよい。



①［ＡＬＬ」ＺＥＲＯ、［ＡＬＬ］ＺＥＲＯＳ、［ＡＬＬ］ＺＥＲＯＥＳ
数字ゼロは何桁かのゼロを表す。

```
01  金額        PICTURE  9(5)   VALUE  ZERO.
01  商品コード  PICTURE  X(10)  VALUE  ZEROS.
```

数字と英数字の項目に使える。

```
                                                    記憶装置の内容
01  ﾃﾞｰﾀ名１    PIC  X(4)  VALUE  ZERO.            ０｜０｜０｜０
01  ﾃﾞｰﾀ名２    PIC  X(4)  VALUE  "0".             ０｜　｜　｜
01  ﾃﾞｰﾀ名３    PIC  X(4)  VALUE  ALL "0".         ０｜０｜０｜０
```

②［ＡＬＬ］ＳＰＡＣＥ、［ＡＬＬ］ＳＰＡＣＥＳ
何桁かの空白を表す。

```
        MOVE  SPACE  TO  印刷領域
```

③［ＡＬＬ］ＨＩＧＨ－ＶＡＬＵＥ、［ＡＬＬ］ＨＩＧＨ－ＶＡＬＵＥＳ
計算機の文字の大小順序で最高の位置を占める文字の何桁かを表す。

```
        READ  ﾌｧｲﾙ名
                AT  END  MOVE  HIGH-VALUE  TO  ﾃﾞｰﾀ名
```

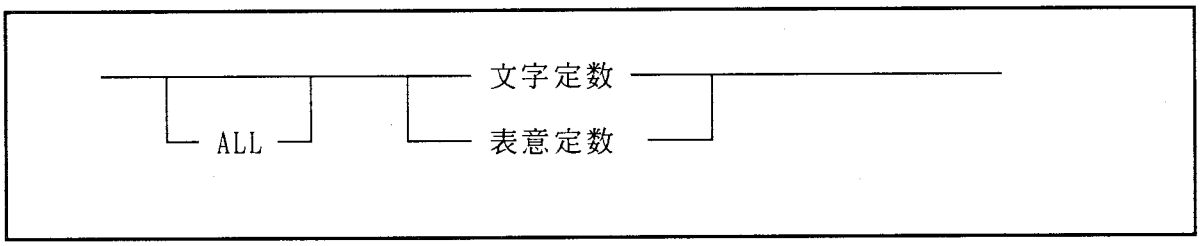
④［ＡＬＬ］ＬＯＷ－ＶＡＬＵＥ、［ＡＬＬ］ＬＯＷ－ＶＡＬＵＥＳ
計算機の文字の大小順序で最低の位置を占める文字の何桁かを表す。
⑤［ＡＬＬ］ＱＵＯＴＥ、［ＡＬＬ］ＱＵＯＴＥＳ
何桁かの引用符（”）を表す。

表意定数は桁数が関係付けられないとき、１文字を表す。
（DISPLAY、STOP、STRING、UNSTRINGで用いるとき）

```
ＤＩＳＰＬＡＹ　ＺＥＲＯＳ.                表示結果　０

ＤＩＳＰＬＡＹ　ＡＬＬ　ＺＥＲＯＳ.        表示結果　０
```

（４）　ＡＬＬ指定

次に続く定数の繰り返しを表す。繰り返し回数はＡＬＬが使用されるところで決まる。表意定数は任意の桁数を表しているので、ＡＬＬは意味を持たない。



ＡＬＬを転記文で使用した場合は、次のとおり。

```
ＭＯＶＥ　”＊”　　ＴＯ　　＊　　PICTURE　X(4)

ＭＯＶＥ　ＡＬＬ　”＊”　　ＴＯ　　＊＊＊＊　　PICTURE　X(4)

                                PICTURE　X(1)

ＭＯＶＥ　ＡＬＬ　”ＡＢＣ”　ＴＯ　→　Ａ

                                PICTURE　X(7)

                                →　ＡＢＣＡＢＣＡ

                                PICTURE X(7)　JUST　RIGHT

                                →　ＣＡＢＣＡＢＣ

                                PICTURE　X(7)

ＭＯＶＥ　ＡＬＬ　”ＺＥＲＯ”　ＴＯ　　ＺＥＲＯＺＥＲ
```

比較は、ＡＬＬ定数の長さの整数倍と一致しないと比較条件は成立（真）しない。条件文で使用する場合は、次のとおり。

```
ＩＦ　ＡＬＬ　”ＸＹＺ”　ＥＱＵＡＬ　Ｘ　　条件は偽

ＩＦ　ＡＬＬ　”ＸＹＺ”　ＥＱＵＡＬ　ＸＹＺＸ　　条件は偽

ＩＦ　ＡＬＬ　”ＸＹＺ”　ＥＱＵＡＬ　ＸＹＺＸＹＺ　　条件は真
```

１．２．６　データ記述の概念

ＣＯＢＯＬでは、データをできるだけ機種に依存しないものにするため、データの属性を装置の形式ではなく標準のデータ形式に合わせて記述する。

標準データ形式では、数値を表現するのに１０進数（０～９）を用いる。文字データの表現は、計算機の文字集合の任意の文字を用いる。

プログラミング言語に於ける数値表現は、次のとおり。

ＣＯＢＯＬ	ＦＯＲＴＲＡＮ	Ｃ	ＰＬ／Ｉ
１０進数	１０進数	８進数 １０進数 １６進数	２進数 １０進数 １６進数
１．２３	１．２３	０７７ ７７ ０ｘ７７	’１０１１’Ｂ １０１１ ’１０１１’Ｘ

（１）データの字類の概念

ＣＯＢＯＬでは、データ項目を五つの項類（category）に分類する。五つの項類は、「英字」、「数字」、「英数字」の三つの字類（class）にまとめられる。

項目のレベル	字類(class)	項類(category)
基本項目	英字	英字
	数字	数字
	英数字	数字編集 英数字編集 英数字
集団項目	英数字	英字 数字 数字編集 英数字編集 英数字

ＣＯＢＯＬでは、データの記述をＰＩＣＴＵＲＥ句で指定する。

ﾚﾍﾞﾙ番号　ﾃﾞｰﾀ名　PICTURE　項類（大きさ）

コンピュータでデータを表現するのに幾つもの形式がある場合は、その選択をＵＳＡＧＥ句で指定できる。

ﾚﾍﾞﾙ番号　ﾃﾞｰﾀ名　PICTURE　項類（大きさ）　USAGE　ﾃﾞｰﾀ表現

データ項目の大きさとは、項類（英字、数字、数字編集等）を表現したときの桁数や文字数である。ただし、コンピュータの内部表現に必要な大きさと異なる場合がある。

１．２．７　データ参照

ＣＯＢＯＬプログラムのすべての「利用者語」は、利用者（プログラマー）が言語の規定に従って自由に命名できる。
手続き部では、その利用者語を一意に識別するために「修飾」、「添字付け」、「部分参照」ができる。

```
01  TABLES.
    03  配列  OCCURS  100  TIMES.
        05  要素１   PICTURE  S9(5).
        05  要素２   PICTURE  X(10).


    MOVE  ZERO  TO  要素１

                    添字が指定されていないので一意でない。

    MOVE  ZERO  TO  要素１(56)
```

一意名とは、データ名を表すのに用いる用語である。データ名がプログラム中で一意でなければ修飾、添字、部分参照を組み合わせて一意にする。

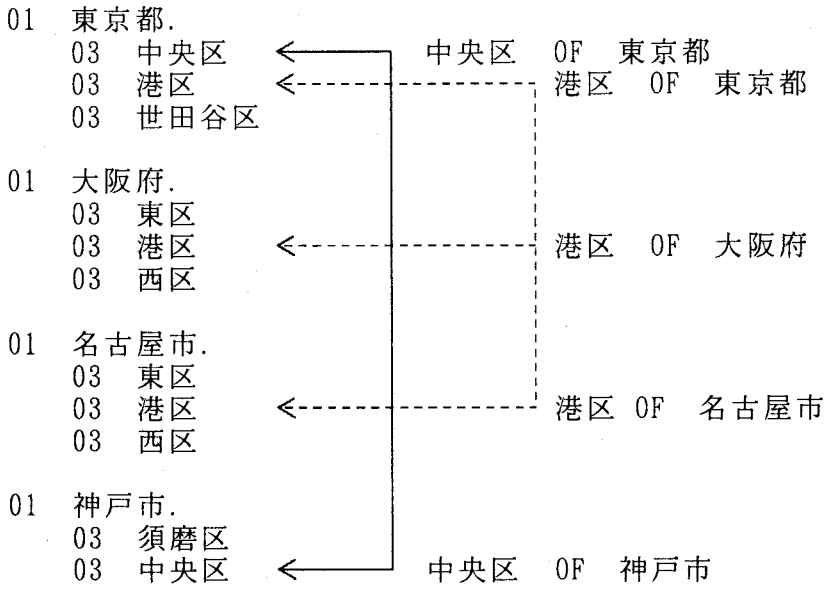
```
ﾃﾞｰﾀ名１ [ { IN } ﾃﾞｰﾀ名２ ] ･･･ [ { IN } { ﾌｧｲﾙ名     } ]
           { OF }               { OF } { 報告書名   }
                                       { 通信記述名 }

        [ ( {添字} ･･･) ] [ (最左端文字位置 : [長さ]) ]
```

データ参照の例

```
FD  ﾌｧｲﾙ名
    DATA  RECORD  IS  ﾃﾞｰﾀA.
01  ﾃﾞｰﾀA.
    02  ﾃﾞｰﾀB.
        03  ﾃﾞｰﾀC.
            04  ﾃﾞｰﾀD  OCCURS  100  TIMES.
                05  ﾃﾞｰﾀ1  PICTURE  X(20).

    MOVE  SPACE  TO  ﾃﾞｰﾀ1 (3)
    MOVE  SPACE  TO  ﾃﾞｰﾀ1  OF  ﾃﾞｰﾀD  IN  ﾃﾞｰﾀC  OF  ﾃﾞｰﾀB (3)
    MOVE  SPACE  TO  ﾃﾞｰﾀ1  OF  ﾌｧｲﾙ名  (3)

    MOVE  "***"  TO  ﾃﾞｰﾀ1  (3)  (3:)          ←添字付けと部分参照
    MOVE  "***"  TO  ﾃﾞｰﾀ1  IN  ﾃﾞｰﾀA  (3)  (3:5)
```

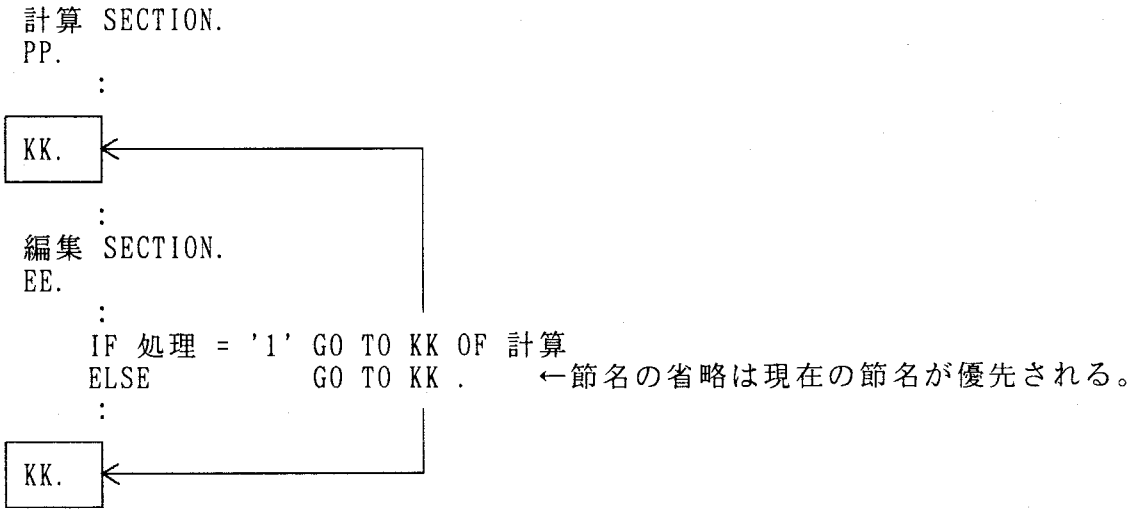
（１）　データ部での修飾

```
データ名1　［OF／IN　データ名2］・・・
```



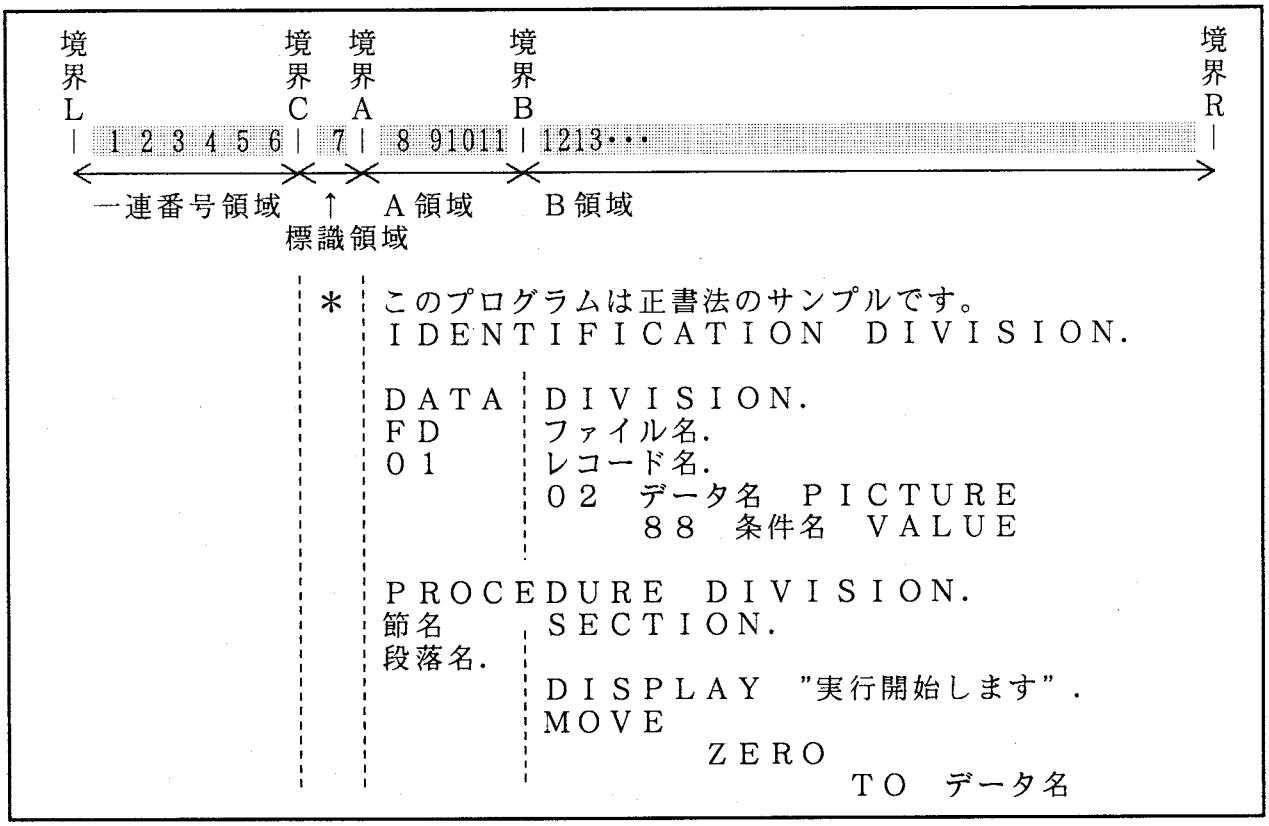
（２）　手続き部での修飾

```
段落名　［OF／IN　節名］
```



１．２．８　正書法（reference　format）

正書法とはＣＯＢＯＬソースプログラムを書くときの標準体裁である。
ＪＩＳの規定は次のとおり。
Ｂ領域の文字数の規定はないが、通常７２カラムまででその後にプログラム識別の領域がある



・一連番号領域は、ソースプログラムの行の識別に利用者が自由に使える。

・標識領域には、空白以外に次の文字を指定できる。
　＊　注記行（注釈行）となりその行は構文解析されない。任意の文字列が書ける。
　／　＊と同じ。ただし、ソースプログラムが印刷される場合は改ページが行われてから、その注記行が印字される。
　－　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に続くことを示す。

・Ａ領域から書き始めるものは、次のとおり。
　部の見出し（DIVISION）
　節名　　（SECTION）
　段落名
　レベル指示語（ＦＤ、ＳＤ、ＣＤ、ＲＤ）
　レベル番号（０１、７７）

・Ｂ領域から書き始めるものは、次のとおり。
　レベル番号（０２～４９、６６、８８）
　文　　（無条件文、条件文、翻訳指示文、範囲明示文）

１．２．９　継続について

（１）　文などの継続

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に、間の空白に関係なく続くことを示す。間に注記や空白行があってもよい。

```
0             1 A領域                                   7
12345678901234                                          234567890
                              MOVE ZERO TO ABCDE
          -       FGHI.
                    ↓
                  MOVE ZERO TO ABCDEFGHI.
```

（２）　文字定数の継続

前の行に引用符で閉じていない文字定数があるときは、後の行の空白でないＢ領域の文字は引用符でなければならない。そして、文字定数の続きはこの引用符のすぐ後から書く。前の行の終わりまでにある空白は文字定数の部分であるとみなされる。

```
0             1 A領域                                   7
12345678901234                                          234567890
                              MOVE "ｱｲｳｴｵ
          -       "ｶｷｸｹｺ" TO ABCDEFGHI.
                    ↓
                  MOVE "ｱｲｳｴｵ      ｶｷｸｹｺ" TO ABCDEFGHI.
```

ＦＯＲＴＲＡＮの書法の概要

```
 ｶﾗﾑ位置
| 1 | 2 3 4 5 | 6 | 7 8 ・・・                                   72 |
<-----------X-----X------------------------------------------------->
   ↑    ↑        ↑    ↑
   |    |        |    開始行　（７～７２）
   |    |        |    Ｒ＝ＲＥＡＬ（Ｉ）＊＊Ｒ２
   |    |        |
   |    |        継続行　空白、０以外の文字を書くと前の行の続きとみなす
   |    |                最高１９行まで継続できる）
   |    |              ＩＦ　（Ｉ．ＥＱ．１０．ＡＮＤ．
   |    |        ＋          Ｊ．ＧＴ．２０）　ＧＯＴＯ　１０００
   |    |
   |    文番号　個々の文を参照するための手段に用いる。
   |    １００　　　　ＦＯＲＭＡＴ（Ｉ４，３Ｘ，Ａ１０）
   |
 注釈行　「Ｃ」か「＊」を書くとその行が注釈行になる
 Ｃ　　ｆｏｒｔｒａｎのコメント行です。
```

１．２．１０　ＣＯＢＯＬプログラム構成

ＣＯＢＯＬソースプログラムの表記順序は次の通り。

見出し部
[環境部]
[データ部]
[手続き部]
[プログラムの終わり見出し]

見出し部　－　ＩＤＥＮＴＩＦＩＣＡＴＩＯＮ　ＤＩＶＩＳＩＯＮ.
プログラムを識別するための記述

```
IDENTIFICATION DIVISION.
PROGRAM-ID. ﾌﾟﾛｸﾞﾗﾑ名.
```

環境部　－　ＥＮＶＩＲＯＮＭＥＮＴ　ＤＩＶＩＳＩＯＮ.
プログラムの環境とファイルに関する記述

```
ENVIRONMENT  DIVISION.
CONFIGURATION  SECTION.

INPUT-OUTPUT  SECTION.
```

データ部　－　ＤＡＴＡ　ＤＩＶＩＳＩＯＮ.
プログラムで使用するデータに関する記述

```
DATA  DIVISION.
FILE  SECTION.

WORKING-STORAGE  SECTION.

LINKAGE  SECTION.

COMMUNICATION  SECTION.

REPORT  SECTION.
```

手続き部　－　ＰＲＯＣＥＤＵＲＥ　ＤＩＶＩＳＩＯＮ.
プログラムの処理手続きを記述

```
PROCEDURE  DIVISION.
DECLARATIVES.

END  DECLARATIVES.
xxxx  SECTION.



yyyy  SECTION.


END  PROGRAM.
```

（1）見出し部（ＰＲＯＣＥＤＵＲＥ　ＤＩＶＩＳＩＯＮ）

見出し部は、プログラムを識別するために最初の段落としてPROGRAM-ID.を置きプログラム名を書く。

```
IDENTIFICATION  DIVISION.

PROGRAM-ID.  ﾌﾟﾛｸﾞﾗﾑ名  [IS  COMMON/INITIAL  PROGRAM].

 [AUTHOR.   [注記項] ･･･]
 [INSTALLATION.   [注記項」･･･]
 [DATE-WRITTEN.   [注記項] ･･･]
 [DATE-COMPILED.  [注記項] ･･･]
 [SECURITY.       [注記項」･･･]
```

（2）環境部（ＥＮＶＩＲＯＮＭＥＮＴ　ＤＩＶＩＳＩＯＮ）

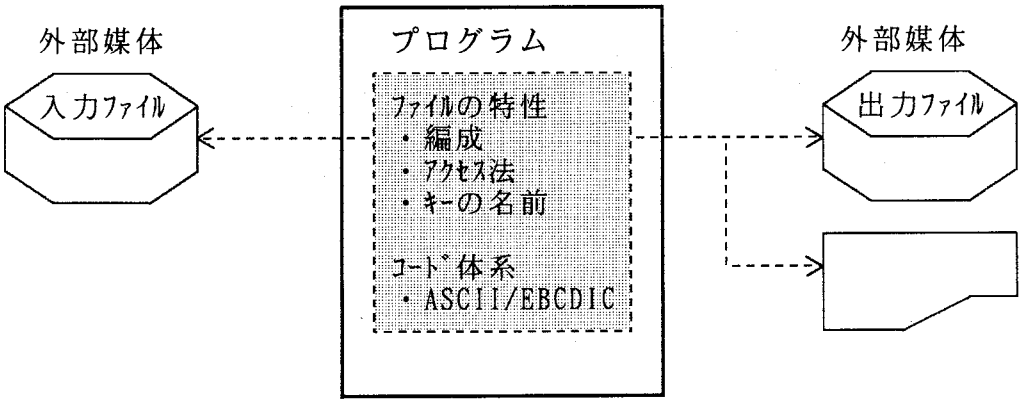
使用する計算機の機械的な特性を記述する。
環境部は構成節（CONFIGURATION）と入出力節（INPUT-OUTOUT）のSECTIONがある。

```
ENVIRONMENT  DIVISION.

CONFIGURATION  SECTION.
SOURCE-COMPUTER.    [計算機名   [WITH  DEBBUGING  MODE] . ]
OBJECT-COMPUTER.    [計算機名
                  [PROGRAM  COLLATING SEQUENCE IS 符号系名] .]
SPECIAL-NAMES.
      作成者語1  IS  呼び名1   [ON  STATUS  IS  条件名1
                               OFF  STATUS  IS  条件名2]
                              { STANDARD-1          }
                              { STANDARD-2          }
      ALPHABET  符号系名1  IS { NATIVE              }
                              { 作成者語2           }
                              { 定数1  THRU  定数2  }
      SYMBOLIC  CHARACTER  記号文字1  IS  整数1  IN  符号系名2  ･･･
      CLASS  字類名1  IS  定数4   [THRU  定数5]   ･･･
      CURRENCY  SIGN  IS  定数6
      DECIMAL-POINT  IS  COMMA.

INPUT-OUTPUT  SECTION.
FILE-CONTROL.

      SELECT   [OPTIONAL] ﾌｧｲﾙ名  ASSIGN  TO  作成者語1/定数1  ･･･  .

I-O-CONTROL.
      SAME  RECORD/SORT/SORT-MERGE  AREA  FOR  ﾌｧｲﾙ名3  ･･･  .
```

CONFIGURATION　SECTION　には、実行用計算機の特性を記述する。
変更可能な特性としては、次のものがある。

- 文字の大小順序（ＡＳＣＩＩ、ＥＢＣＤＩＣ等）
- 通貨記号の指定（￥、＄）
- 小数点の指定　（．、，）
- 作成者語と利用者語の関連付け

INPUT-OUTPUT　SECTION　には、プログラムと外部媒体の間で、データの転送に必要なファイルの特性情報を記述して、ファイルの名前と外部媒体を関連付ける。



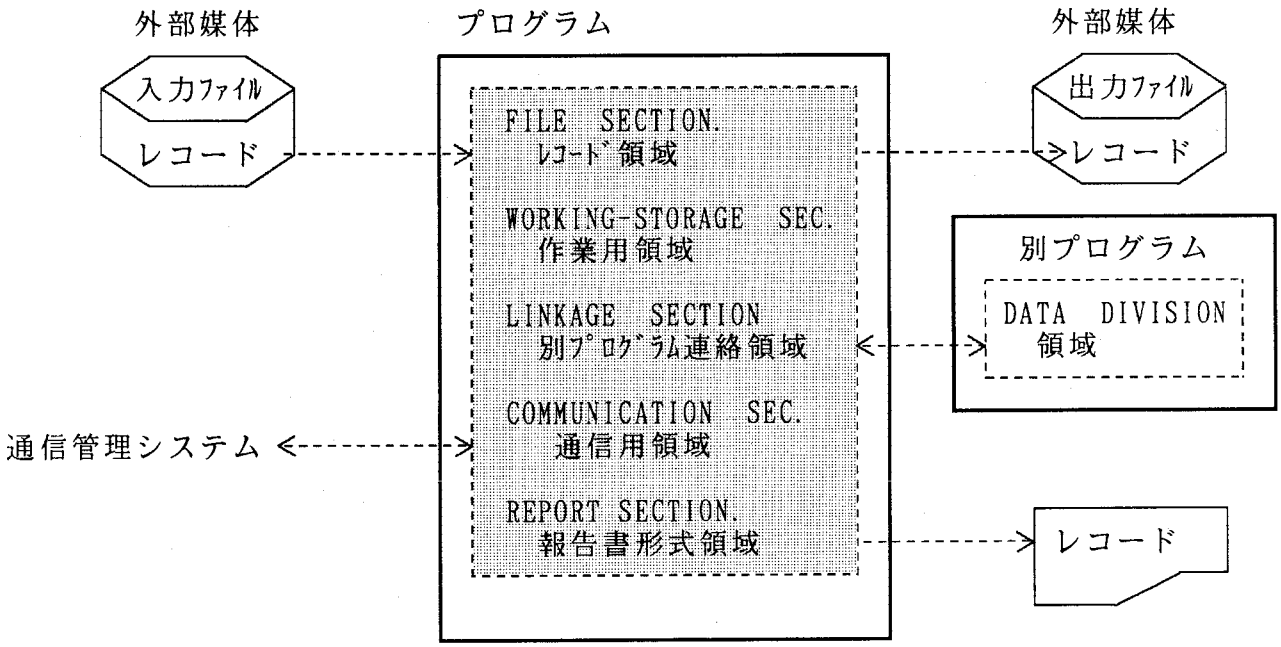
詳細は、第５章を参照のこと。

（3）データ部（ＤＡＴＡ　ＤＩＶＩＳＩＯＮ）

データ部は、ファイル節、作業場所節、連絡節、通信節、報告書節で構成される。
プログラムが入力として読み取り、計算や編集し、出力として書き出すレコードの構造や属性を記述する。

```
DATA DIVISION.
 [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      {レコード記述項} …

 [WORKING-STORAGE SECTION.
   { 独立項目記述項 }
     レコード記述項 } …]

 [LINKAGE SECTION.
   { 独立項目記述項 }
     レコード記述項 } …]

 [COMMUNICATION SECTION.
   [ 通信記述項  [レコード記述項] …] …]

 [REPORT SECTION.
   [ 報告書記述項  {報告書集団記述項} …] …]
```

詳細は、第２章を参照のこと。



（4）手続き部（ＰＲＯＣＥＤＵＲＥ　ＤＩＶＩＳＩＯＮ）

手続き部は、宣言部分と手続き部分からなる。
宣言部分は、任意であり、省略するとシステムの標準処置が行われる。
プログラムの実行は、手続き部分の最初の文から始まる。

```
 PROCEDURE DIVISION  [USING  {ﾃﾞｰﾀ名1} ･･･] .
[DECLARATIVES.
  {節名  SECTION  [区分番号] .   USE  文.
  [段落名.   [完結文] ･･･] ･･･} ･･･

 END DECLARATIVES.]

{節名  SECTION  [区分番号] .
  [段落名.   [完結文] ･･･] ･･･} ･･･
```

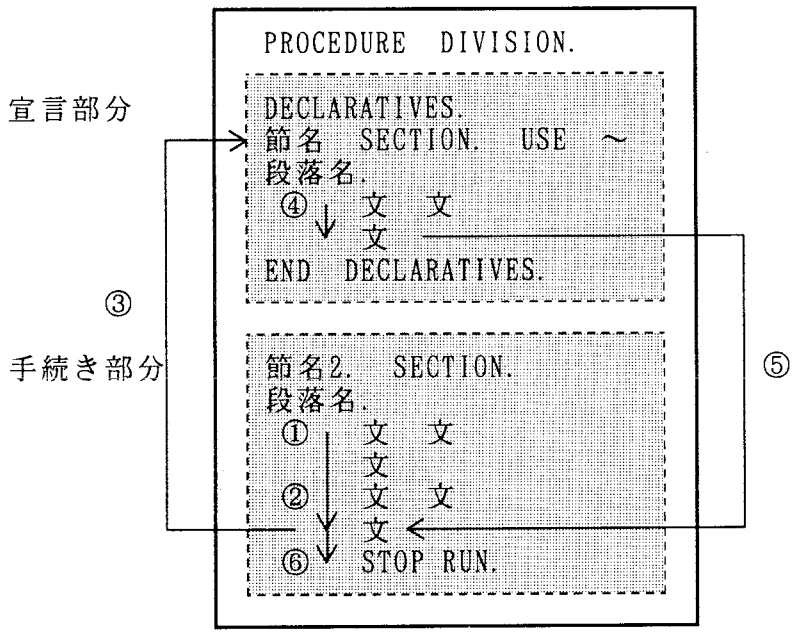
完結文は、いくつかの文からなり、分離符の終止符（．）で終わる。

```
    文  文  文.     文.
```

文は、ＣＯＢＯＬの動詞（命令）の後に、語、定数や予約語、分離符を正しく組み合わせて記述したものである。

```
    MOVE  ZERO  TO  単価,  合計
```

最初の実行は、手続き部分から開始される。手続き部分の文で入出力誤り等が発生すると宣言部分のＵＳＥで宣言した節の文を実行する。
宣言部分の節の最後まで実行すると、制御は手続き部分に戻り次の文から実行を継続する。



１．２．１１　節、段落、完結文、文

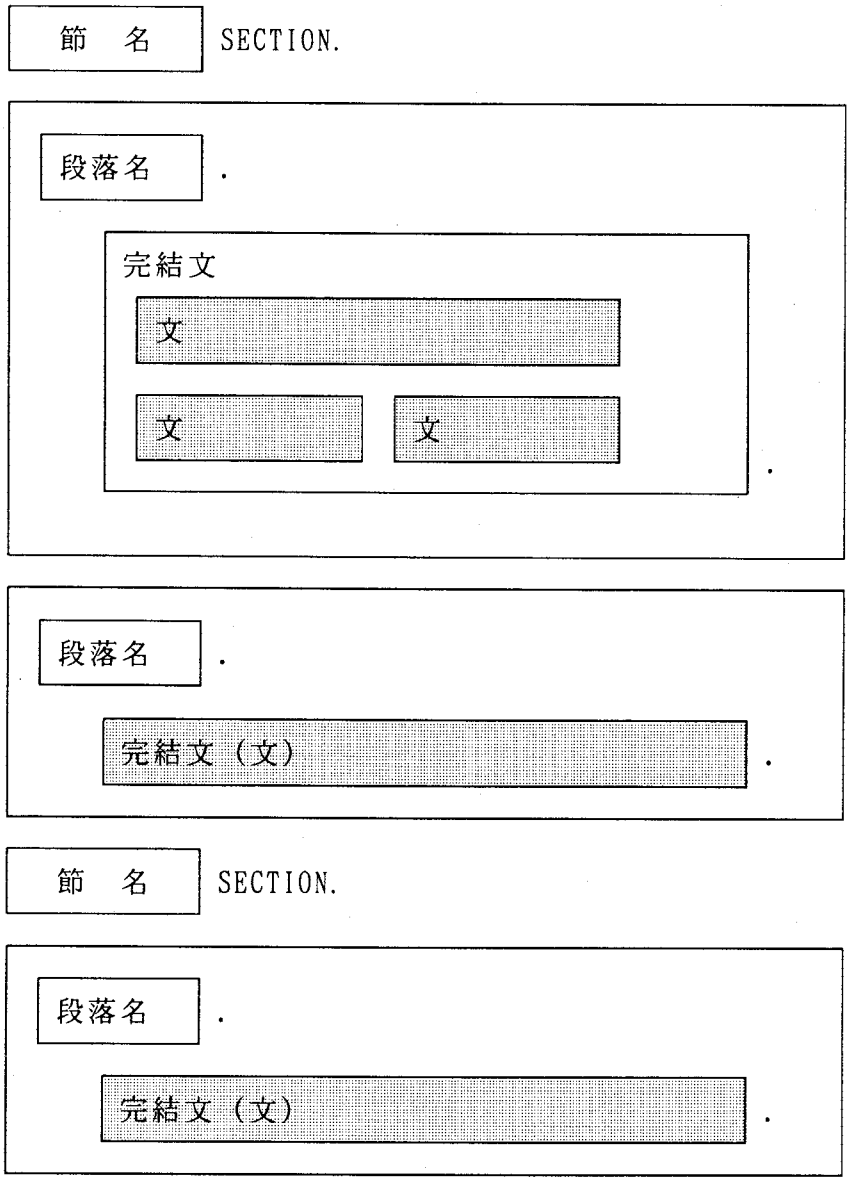
節は、節の見出しとその後の連続した段落からなる。

段落は、終止符（．）と空白でとめた段落名とその後に続くいくつかの完結文なる。

完結文は、いくつかの文からなり、分離符の終止符（．）で終わる。

文は、ＣＯＢＯＬの動詞（命令）の後に、語、定数や予約語、分離符を正しく組み合わせて記述したものである。

全体構成は次の通り。



１．２．１２　文の種類

文（命令）を機能別に分類すると、次のとおり。

【算術文】

ＡＤＤ（加算）
ＣＯＭＰＵＴＥ（計算）
ＤＩＶＩＤＥ（除算）
ＭＵＬＴＩＰＬＹ（乗算）
ＳＵＢＴＲＡＣＴ（減算）

【データ移動】

ＩＮＩＴＩＡＬＩＺＥ（初期化）
ＩＮＳＰＥＣＴ（文字列検査）
ＭＯＶＥ（転記）
ＳＴＲＩＮＧ（連結）
ＵＮＳＴＲＩＮＧ（分解）

【順序付け】

ＭＥＲＧＥ（併合）
ＲＥＬＥＡＳＥ（引渡し）
ＲＥＴＵＲＮ（引取り）
ＳＯＲＴ（整列）

【手続き分岐】

ＡＬＴＥＲ（変更）
ＣＯＮＴＩＮＵＥ（継続）
ＥＸＩＴ（出口）
ＧＯ　ＴＯ（飛越し）
ＰＥＲＦＯＲＭ（実行）

【判断】

ＥＶＡＬＵＡＴＥ（評価）
ＩＦ（判断）

【入出力】

ＡＣＣＥＰＴ（小入力）
ＣＬＯＳＥ（閉じる）
ＤＥＬＥＴＥ（削除）
ＤＩＳＰＬＡＹ（表示）
ＯＰＥＮ（開く）
ＲＥＡＤ（読込み）
ＲＥＷＲＩＴＥ（書換え）
ＳＴＡＲＴ（位置決め）
ＷＲＩＴＥ（書出し）

【表操作】

ＳＥＡＲＣＨ（表引き）
ＳＥＴ（設定）

【プログラム間連絡】

ＣＡＬＬ（呼ぶ）
ＣＡＮＣＥＬ（取消）
ＦＵＮＣＴＩＯＮ（組込み関数）

【停止】

ＳＴＯＰ（停止）

注：報告書作成機能、通信機能は除く。

指導上の留意点

◎ＣＯＢＯＬの歴史
ＣＯＢＯＬの説明書のはじめには「謝辞」が記載されている。
この謝辞の内容でＣＯＢＯＬの歴史がある程度想像できる。

◎ＣＯＢＯＬ言語規格の構造
ＣＯＢＯＬは日本工業規格（ＪＩＳ）で国家規格として制定されている。

「JIS　X　3002　電子計算機ﾌﾟﾛｸﾞﾗﾑ言語ＣＯＢＯＬ」

規格は、ＣＯＢＯＬで表現されるﾌﾟﾛｸﾞﾗﾑの書き方と解釈の両者を規定することで、ハードウェアからの独立性を促進することを目的としている。

ＣＯＢＯＬ言語は、次の１１個の機能単位の概念に基づいて構成されている。

機能単位	定義	ＣＯＢＯＬ集合 上位集合	中位集合	下位集合	本指導書での取扱い
1 中核 nucleus	必須	水準２	水準１	水準１	
2 順ﾌｧｲﾙ sequential i-o	必須	水準２	水準１	水準１	第５章
3 相対ﾌｧｲﾙ relative i-o	必須	水準２	水準１	空	第７章
4 索引ﾌｧｲﾙ indexed i-o	必須	水準２	水準１	空	第７章
5 ﾌﾟﾛｸﾞﾗﾑ間連絡 inter-program communication	必須	水準２	水準１	水準１	第１０章
6 整列併合 sort-merge	必須	水準１	水準１	空	第８章
7 原始文操作 source-text manipulation	必須	水準２	水準１	空	省略
8 報告書作成 report writer	選択	水準１	－	水準１	省略
9 通信 communication	選択	水準２	－	水準１	省略
10 ﾃﾞﾊﾞｯｸﾞ debug	選択	水準２	－	水準１	省略
11 区分化 segmentation	選択				省略
12 組込み関数 intrinsic function	選択	水準１		水準１	第10章

９個の機能単位は、水準１と水準２に分けられる。水準２の部分集合を水準１とする。
整列併合と報告書作成機能、組込み関数は水準１だけで構成される。
規格ＣＯＢＯＬは、上位集合、中位集合、下位集合の集合から構成される。

組込み関数は、１９９２年のＪＩＳ規格から採用された機能単位である。

◎水準の具体例は次の通り。（ 水準２ 部分）

```
SELECT [OPTIONAL] ﾌｧｲﾙ名  ASSIGN  TO  作成者語

03  ﾃﾞｰﾀ名  OCCURS  整数  TIMES
    [ASCENDING/DESCENDING  KEY  ﾃﾞｰﾀ名]
      INDEXED  BY  指標名.

PERFORM  手続き名  [WITH  TEST  AFTER/BEFORE]  UNTIL  条件1
```

◎水準１では規定されていない命令文は、次のとおり。（中核機能単位）

・ＣＯＭＰＵＴＥ
・ＥＶＡＬＵＡＴＥ
・ＩＮＩＴＩＡＬＩＺＥ
・ＳＥＡＲＣＨ
・ＳＴＲＩＮＧ
・ＵＮＳＴＲＩＮＧ

その他の規定は、次のとおり。

・ＡＬＬ定数
・７レベルの添字
・部分参照

◎廃要素

廃要素は、国際規格ＣＯＢＯＬの次の版から削除される予定の言語要素である。
規格ＣＯＢＯＬから削除される言語要素は、その削除に先立って、「廃要素」として明記される。
１９９２年の規格から削除の言語要素は次のとおり。

・見出し部
  AUTHOR段落
  INSTALATION段落
  DATE-WRITTEN段落
  DATE-COMPLIED段落
  SECURITY段落
・環境部
  MEMORY SIZE句
  MULTIPLE FILE TAPE句
  RERUN句
・データ部
  DATA RECORDS句
  LABEL RECORD句
  VALUE OF句
・手続き部
  ALTER
  ENTER
  GO TO手続き名は書いても書かなくてもよい
  STOP 定数
  OPEN REVERSED指定
・デバッグ機能
  区分化機能