情報処理基礎講座

COBOL基礎編

監修　工学博士　松　尾　三　郎

電子開発学園

謝　　辞

本テキストは，データ組織言語協議会（CODASYL）のCOBOL委員会とヨーロッパ計算機製造者協会（ECMA）との努力によって開発されたCOBOLに基づき，その要請にしたがってつぎの謝辞を掲げます。

「いかなる組織体も自由に教育用説明書その他の目的で，COBOL文法説明書とその仕様の一部または全部を複製し，あるいは報告書中の着想を利用してかまいません。ただし，その文書の序文の一部にこの部分を掲載しなければなりません。書評などで短い文章を引用するときは，出典についての謝辞にCOBOLの名称を挙げれば全文を掲げる必要はありません。

COBOLは，産業界の言語であって，いかなる会社，会社団体，組織または組織団体の所有物でもありません。

COBOL言語作成の寄与者またはCOBOL委員会は，プログラミング組織と言語の正確さ，機能についていかなる保証をもしません。また，これに関する事項について，いかなる責任をも負いません。

COBOL保全の手続きは確立しています。変更を提案する手続きについては，データ組織言語会議（CODASYL）の理事会に照会して下さい。

ここで利用した以下の資料の著書および版権者は，COBOL仕様書にその一部または全部を利用することを認可しています。このことは，プログラミング説明書や類似の出版物にCOBOL仕様書を複製，利用する場合にも適用されます。

FLOW－MATIC（スペリランド社の商標），
Programming for Univac (R) Ⅰ and Ⅱ, Data Automation systems,
スペリランド社，1958年，1959年版権；
IBM Commercial Translator,
図書番号　F 28－8013，IBM社，1959版権；
FACT,
DSI　27A5260－2760，ミネアポリス・ハニウエル社，1960年版権；」

本テキストの作成者は，CODASYL COBOL委員会各寄与者の努力と成果を高く評価し，ここに感謝の意を表します。

はじめに

●本書の特徴

昨今のコンピュータの普及には目を見張るものがあります。装置も次第に小型化し，いわゆるオフィスオートメーション（OA）化が全国的にかなり浸透してきました。これらに伴い，従来コンピュータとは縁のなかった一般の人々でも，コンピュータを扱わなければならない状況になってきています。また，コンピュータの新機種が次々と登場する一方，事務処理用の共通言語である「JIS COBOL」にも改正が重ねられてきました。

このような情勢をふまえ，本書は電子開発学園の長い歴史と豊富な教育経験を基に，COBOLテキストの基礎編として，初心者を対象に編集したのです。

「COBOL基礎機能編」初版を発行し，以来いろいろな情報を取り入れながら版を重ねてきましたが，その間の情勢の急激な変化に伴い，内容的に少々不備な点が目立ってまいりました。そこで，このたび現在の情勢を念頭におき，それらを十分に反映させたテキストとして，内容を全面的に見直し，新版として発行する運びとなりました。

本書は改版前と同様のねらいを持って編集し，COBOLの基礎機能をわかりやすく解説してあります。具体的には次のような特徴を持っています。

⑴ 文法の説明はとかく堅苦しくなりがちなため，例題を中心におき，その例題に必要な文法を優先的に説明するようにしました。

⑵ COBOLを一度にあれこれと見てしまうと，知識の混乱を招くばかりか，理解があやふやなものになってしまいます。COBOLを学習するに当たっては，コンピュータを実際に使いながら，少しずつ身につけていくのが効果的です。このため，段階的に学習できるよう，簡単な例題から複雑な例題に進むようにしました。

具体的には，「基本的なファイル処理」，「編集処理」，「グループ制御」，「表の処理」，「複数ファイルの処理」，「内部整列」，「表探索」などの例題を用意しています。

⑶ 文法の説明では，説明のつど使用例を豊富に挙げ，理解しやすいよう配慮しました。

⑷ 章の最後には練習問題や演習問題を用意し，理解度をチェックできるように配慮しました。

⑸ 最後には総合演習問題を用意し，本書全体を通じての理解を確たるものにするように配慮しました。

●本書を使用するに当たって

本書は，基本的には，「JIS X3002(1992）電子計算機プログラム言語COBOL」に準拠しているため，個々の計算機によるCOBOL文法の相違点については触れていません。ただし，本文中のプログラム例の装置名は，MS−DOSに準拠した形にしています。

また，本書は，COBOL文法を初歩から拡張機能にいたるまでを解説したものですが，あまり使用されないと思われる機能については省略しました。

省略した機能は下記のとおりです。

・見出し部

PROGRAM−ID段落のCOMMON指定およびINITIAL指定，
AUTHOR段落，
INSTALLATION段落，
DATE−WRITTEN段落，
DATE−COMPILED段落，
SECURITY段落

・環境部

構成節のSOURCE−COMPUTER段落のDEBUGGING MODE指定，
構成節のOBJECT−COMPUTER段落のMEMORY SIZE指定，SEQUENCE指定およびSEGMENT−LIMIT指定，
構成節のSPECIAL−NAMES段落，
入出力節のI−O−CONTOROL段落，
ファイル管理記述項のOPTIONAL指定，RESERVE指定，PADDING CHARACTER指定およびFILE STATUS指定

・データ部

連絡節，
ファイル記述項のEXTERNAL指定，GLOBAL指定，VALUE OF指定およびCODE−SET指定，
データ記述項のEXTERNAL指定，GLOBAL指定，SIGN指定，SYNCHRONIZED（SYNC）指定およびBLANK ZERO指定，
報告書記述項のGLOBAL指定およびCODE指定，
報告集団記述項のSIGN指定，JUSTIFIED（JUST）指定およびBLANK ZERO指定

・手続き部

宣言部分,
区分番号,
ACCEPT文のFROM指定,
ALTER文,
CANCEL文,
DISPLAY文のUPON指定およびNO ADVANCING指定,
ENABLE文,
ENTER文,
EXIT文,
PURGE文,
RECEIVE文,
SEND文,
USE文,
入れ子の原始プログラム

なお，より深くCOBOLを学習され，実際の業務等に応用される方のために，本書の続編として「COBOL応用編」が発刊されておりますので，本書に引き続いてのご利用をお勧めいたします。

目　　次

序　　論

われわれがコンピュータを使って仕事をするためには，コンピュータ内にプログラムという手順の指示のための情報を記憶させる必要がある。このプログラムはある一定の規則に従って作られたものでなければならない。これはちょうどわれわれが言葉を用いて人に仕事を依頼する場合に似ている。われわれの言葉も文法規則にのっとっている必要がある。そこで，人間の言葉との類推から，プログラムを作る規則を文法，その体系をプログラミング言語と呼ぶようになった。

このプログラミング言語の変遷を見てみよう。コンピュータにはコンピュータの種類ごとに固有な機械語と呼ばれるものが存在する。この機械語は，最初は数字の組み合わせだけでしか表現できず，プログラムを作る人（プログラマという）の労力は多大なものであり，また他の人がプログラムを見て手順を理解することも困難であった。そこで，プログラムの作成や修正に要する労力を軽減するために，数字の組合せの一部を英語の単語に置き換えて表現できるアセンブラ言語と呼ばれるプログラミング言語が開発された。もちろん，アセンブラ言語で記述されたプログラムをあらかじめ機械語に変換することが必要であり，その変換のためのプログラムをアセンブラと呼ぶ。アセンブラ言語の開発により，プログラム作成の労力はある程度軽減されたが，コンピュータの種類によってプログラミング言語が違うという問題は残されたままであった。

そこで，プログラム作成の労力の，より一層の軽減とコンピュータの種類を問わない共通性とを目的として，コンパイラ言語が開発された。コンパイラ言語は，コンピュータの機械語をいかにわかりやすく表現するかというアセンブラ言語の発想とは逆で，人間にわかりやすいプログラミング言語を作り，それをいかにして機械語に変換するかという発想で作られたものである。したがって，コンパイラ言語で記述されたプログラムはアセンブラ言語と比較すると格段にわかりやすくなったし，機械語に変換するプログラム（コンパイラという）の工夫によって，コンピュータの種類を問わないプログラミング言語が可能になったわけである。

当初，コンパイラ言語としては，FORTRAN（Formula Translator）を中心とした科学技術計算用しかなかったが，その後，事務処理用のコンパイラ言語として，COBOL（Common Business Oriented Language）が開発された。

このCOBOLの開発によって，それまで科学技術計算などの限られた分野でしか利用されていなかったコンピュータが，経済活動全般にわたって利用されるようになった。

第1章　概　　説

1.1　プログラマに求められること

1.1.1　システム開発の工程

一般に，システム開発は図1－1のような工程に分けて行われる。

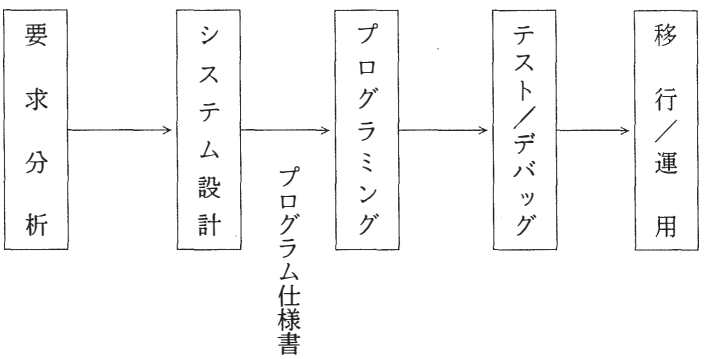


図1－1　システム開発の工程

1.1.2　プログラマの役割

プログラマは，プログラム仕様書を受け取ると，その内容をよく理解した上で，プログラム作成にとりかかる。

プログラマは仕様書に忠実でかつわかりやすいプログラムを書かなければならない。このためには，プログラマには，COBOLの文法に精通していることが求められる。

それではプログラマの第一歩として，これからCOBOLの文法を学んでいこう。

1.2　COBOLの歴史

1950年代の後半になると，コンピュータの利用が拡大し，利用分野もそれまでの科学技術計算主体から事務計算へと広がってきた。

このような背景のもとに，1959年にアメリカで，事務応用向きのプログラミングのための共通言語を定めることが必要であるか，また可能であるかを検討する会議がアメリカ国防総省の主催により開かれた。この会議には，アメリカのユーザ，メーカ，その他の組織の代表者が出席した。この会議は，共通言語は，「拡張的であり変更や改正が可能であること」，「問題向きであり特定の計算機に依存しないこと」，「簡単な英語または英語的な表現とし，できる限り記号表現を避けるべきであること」を結論した。

この会議の結論を推進し，共通言語を制定するために，この会議を母体として，データシステムズ言語協議会(CODASYL:the COnference on DAta SYstem Languages)が作られた。

COBOLの最初の仕様は，CODASYLの1つの委員会が開発したものである。これに基づき，一般事務用の効果的な言語が作成された。事務処理用共通言語という意味のCOBOL（COmmon Business Oriented Language）の名称もここで採用された。この委員会の報告は，1960年4月に発行されたので，のちにこの言語のことを「COBOL−60」と呼ぶようになった。

CODASYLの理事会は，COBOLを定める仕事が継続的なものであり，COBOLに保守と改訂が必要であることを認め，COBOL保守委員会を組織した。COBOL保守委員会は，のちに組織変更が行われ，3つの小委員会からなるCOBOL委員会となった。この委員会は，COBOLの標準化に関して，アメリカ規格協会（USASI，後にANSIと改称）および国際標準化機構（ISO）と連絡を取り合った。

このような標準化の流れの中で，次のような規格が制定，改正された。

表1−1　規格制定の歴史

規格品	アメリカ規格	ISO推薦規格	日本工業規格（JIS）
作成者	アメリカ規格協会（ANSI）	国際標準化機構（ISO）	日本規格協会
制定年	1968年制定	→1972年制定	→1972年制定（第一次規格）
	1974年制定	→1978年制定	→1980年制定（第二次規格）
	1985年制定（COBOL−85）	→1985年制定	→1988年制定（第三次規格）

1.3　COBOLの特徴

1.3.1　事務処理用言語

COBOLの歴史でも述べたように，COBOLは最初の段階から事務応用向きであることを念頭において作成されたものである。したがって，事務処理に便利な機能が種々取り入れられている。

1.3.2　共通言語

COBOLの歴史でも述べたように，COBOLは最初の段階から特定の計算機に依存しないことを念頭において作成されたきた。また，アメリカ規格とISO推薦規格を基に日本工業規格が作成されてきた。したがって，機種の異なる計算機であっても，プログラムをほとんど変更せずに使用できるようになっている。

1.3.3　文書化に適した言語

当初のプログラムは，プログラマ個人個人の個性を色濃く反映したものであったため，本人でなければ理解できず，他人がプログラムの修正を行うのは容易ではなかった。COBOLはだれが見てもわかるよう，プログラムの文書化を意識した最初の言語である。

COBOLプログラムは，簡単な英文に似ているので，書きやすく，読みやすい。したがって，プログラムの保守が容易である。

1.3.4　ポピュラーな言語

現在，プログラム言語にはFORTRAN, BASIC, C, PASCAL, PROLOG, LISP等いろいろな種類がある。

その中で使用率が最も高いのがCOBOLである。

1.4　COBOLプログラムの枠組み

COBOLプログラムは，次の4つの部（DIVISION）からなっている。

COBOLプログラム

見出し部（IDENTIFICATION　DIVISION）	プログラムの見出しとなる部分で，プログラムの名前などを記述する。
環 境 部（ENVIRONMENT　DIVISION）	プログラムが投入されるハードウェア環境，つまり，プログラムが使用する電子計算機名や入出力装置名などを記述する。
データ部（DATA　DIVISION）	プログラムで使用する入出力領域や作業用領域などを定義する。
手続き部（PROCEDURE　DIVISION）	プログラムで行う仕事の具体的な処理手順を記述する部分であり，流れ図をCOBOLの文に当てはめて記述する。

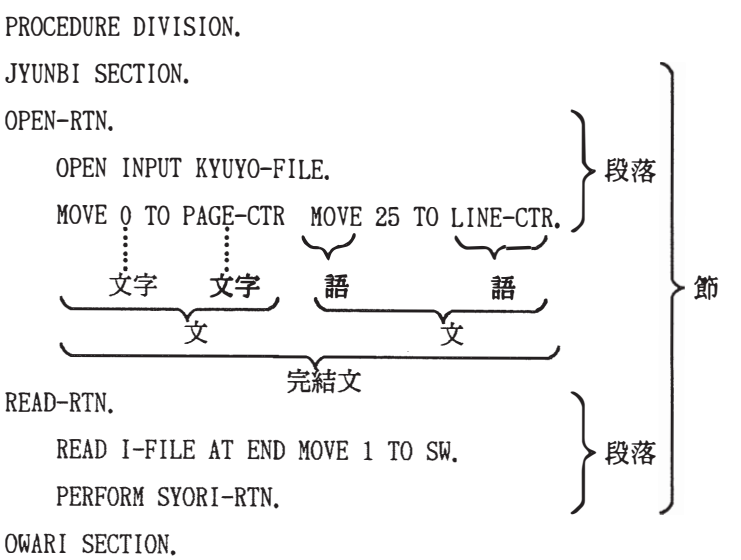
1.5　プログラムの構成

1.5.1　プログラムの構成要素

プログラムは次のような要素で構成される。

部　　（DIVISION）
節　　（SECTION）
段　落（PARAGRAPH）
完結文（SENTENCE）
文　　（STATEMENT）
語　　（WORD）
文　字（CHARACTER）

＜例＞　手続き部（PROCEDURE　DIVISION）の場合



① 語とはいくつかの文字を並べたものであり，何らかの意味を表す最小単位である。

② 文とは動作の単位であり，一定の約束ごとに従って語を分離符（空白など）でつないで並べたものである。

③ 完結文とは1個以上の文を並べた後に終止符をつけたものである。
完結文の中では，文は記述されている順序で実行される。

④ 段落とは段落名（後述）の後に終止符をつけて，その後にいくつかの完結文を書いたものである。

⑤ 節とはいくつかの段落の集まりであって，節の見出し（～　SECTION.）で始まる。

1.5.2　COBOL文字集合

0，1，2，…，9	数　字
A，B，C，…，Z	英大文字
a，b，c，…，z	英小文字
空　白	空　白（blank,space）
＋	正　号（plus）
－	負号またはハイフン（minus,hyphen）
＊	星　印（asterisk）
／	斜　線（slash）
＝	等　号（equal）
￥	通貨記号（currency sign）
,	コンマ（comma）
;	セミコロン（semicolon）
.	終止符または小数点（period）
”	引用符（quotation）
（	左かっこ（left parentheses）
）	右かっこ（right parentheses）
＞	より大きい記号（greater）
＜	より小さい記号（less）
:	コロン（colon）

なお本書では，空白であることを明確に示す場合には，△や␣を用いることがある。

これらの文字を含む例を次に挙げる。

＜例1＞　02　SYOKEI　PIC　￥￥￥，￥￥9．

＜例2＞　COMPUTE　A　＝（B　＊　C　＋　D　＊＊　2）／　E．

＜例3＞　IF　P　＞　Q　THEN　MOVE　1　TO　SW．

1.5.3　分離符

個々の文字をつないで，文字列または分離符を作る。分離符の後には，他の分離符または文字列をつなぐことができる。文字列の後には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いくことによって，原始プログラムの本文ができる。

分離符は，次のような1文字または連続した2文字である。

△

，△　（分離符の空白が使用できるところなら，どこでも使用できる。）

；△　（分離符の空白が使用できるところなら，どこでも使用できる。）

．△　（完結文の終わりを示す。）

（　）　（左右を一対として，添字，算術式，条件式を囲むときだけに使用する。）

：　（ある特定の役割がある。詳細は後述する。）

（注）① 分離符の直前には，分離符の空白を置いてもよいし，置かなくてもよい。

② 分離符の直後には，分離符の空白を置いてもよいし，置かなくてもよい。

＜例1＞　MOVE　ZERO　TO　A，B，C．

＜例2＞　COMPUTE　A　＝（B　＋　C）／　D．

1.5.4　COBOLの語

⑴　COBOLの語の定義

COBOLの語は，30文字以内の文字列であって，利用者語，システム名，予約語からなる。

⑵　語を構成する文字

COBOLの語の各文字は，英文字，数字およびハイフンの組み合わせであり，ハイフンを語の最初や最後の文字として使用することはできない。

＜正しい例＞	＜誤った例＞
LINE−CTR	−WORK
PRINT−REC	DISKFILE−
TAPE−F	

1.5.5　利用者語

利用者語は，COBOLの語であって，利用者が定義するものである。

利用者語は予約語と同じつづりであってはならない。

利用者語には，字類名，条件名，データ名，ファイル名，指標名，レベル番号，登録集名，段落名，プログラム名，レコード名，報告書名，節名，原文名などがある。

節名，段落名，レベル番号以外のすべての利用者語は，少なくとも1文字の英字を含まなけ

ればならない。（英字を1文字も含まないと，後述する数字定数などと混同される）

(1) 手続き名

手続き部中の段落または節を命名する利用者語であって，段落名と節名がある。

節は0または1個以上の段落からなり，段落は0または1個以上の完結文からなる。

段落名の後には分離符の終止符が必要である。

＜例＞

JYUNBI　SECTION.
節名

OPEN−RTN.
段落名

(2) データ名

手続き部で使用するデータ項目をデータ部で命名するための利用者語である。

(3) ファイル名

手続き部で使用する論理的なファイル（内部ファイルともいう）をデータ部で命名するための利用者語である。

(4) レコード名

手続き部で使用する論理的なファイルに含まれるレコードをデータ部で命名するための利用者語である。

＜例＞

FD　PRINT−F〜
ファイル名

01　PRINT−R.
レコード名

02　SYAIN−CODE　PIC X(5).
データ名

(5) プログラム名

COBOLプログラムの識別のために見出し部で命名する利用者語である。

＜例＞

PROGRAM−ID.　KYUYO−KEISAN.
プログラム名

(6) 指標名

指標とは，表中の特定の要素を識別するために用いられる記憶場所またはレジスタであり，指標名は指標の識別のために命名する利用者語である。

1.5.6 システム名

システム名は，COBOLの語であって，計算機名，作成者語，言語名の3種類であり，操作環境との連絡（プログラムを実行する計算機名やプログラム実行時の論理ファイルと物理ファイルとの対応づけなど）に用いる。

システム名は，コンパイラによってあらかじめ決められており，コンパイラによってその名前が異なるため，ここでは紹介だけにとどめる。

1.5.7 予約語

予約語は，COBOLの語であって，そのつづりと意味があらかじめ定められているものである。したがって，予約語と同じつづりの語を利用者語やシステム名として利用することはできない。

例えば，PROCEDUREという語はPROCEDURE　DIVISION. という部の見出しとしてしか使えないし，READという語はファイルを読み込むときにしか使うことはできない。

予約語一覧表は末尾に掲げてある。

予約語には，必須語，補助語，特殊目的語がある。

(1) 必須語

必須語は，原始プログラム内でその語が現れる書き方を用いるとき，省略できない語であって，次の2種類に分類される。

必要語 …… 書き方で，下線の引いてある大文字の語である。

特殊文字語 … 算術演算子（＋，－，＊，＊＊，／）および比較文字（＝，＞，＜）である。それぞれの意味は次のとおりである。

＋ … 正号または加算

－ … 負号または減算

＊ … 乗算

＊＊ … べき乗

／ … 除算

＝ … 等しい

＞ … より大きい

＜ … より小さい

(2) 補助語

補助語は，それぞれの一般形式で，下線の引いてない大文字の語であり，書いても書かなくてもよく，書き方の意味に影響を与えない。つまり，文意を補助してわかりやすくするために適宜挿入される語であって，記述するかしないかはプログラマにまかされている。

この語を省略しても，コンパイル結果は省略しない場合となんら変わることはない。

ただし，書く場合には，つづりを間違えないようにしなければならない。

＜例1＞　CONTAINS

～ BLOCK CONTAINS 5 RECORDS

＜例2＞　ADVANCINGとLINES

WRITE PRINT-R AFTER ADVANCING 2 LINES

なお，下線のある語は必須語である。

(3) 特殊目的語

特殊レジスタおよび表意定数である。

(a) 特殊レジスタ

特殊レジスタはコンパイラが作り出す記憶場所であって，特別の役割が与えられているものである。したがって，データ部で定義する必要はない。

詳細は後述するが，特殊レジスタには下記のものがある。

LINAGE-COUNTER

LINE-COUNTER

PAGE-COUNTER

(b) 表意定数

表意定数とは，データ処理でよく使用される特定の定数値に名前をつけて，そのつどデータ名を付与する不便を避けたものであり，データ部で定義する必要はない。

表意定数の長さは，表意定数が用いられる文脈によって定まる（例えば，空白が何桁の空白を意味するかなど）。なお，表意定数の前にALLをつけることができるが，ALLをつけないものと同じ意味である。また，単数形と複数形は同じ意味である。

ただし，表意定数を引用符（ ”）で囲むと，後述する文字定数になってしまうので注意すること。

表1.2　表意定数

表意定数	表意定数の値
ZERO ZEROS ZEROES	数値ゼロまたは何桁かの文字ゼロを表す。
SPACE SPACES	何桁かの空白を表す。
HIGH−VALUE HIGH−VALUES	その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。（通常，最大値という。）
LOW−VALUE LOW−VALUES	その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。（通常，最小値という。）
QUOTE QUOTES	引用符（”）を表す。ただし，文字定数を囲む引用符としては使用できない。
ALL定数	この定義を構成する文字列の何回かの反復によって生成される列の全部または一部を表す。定数は文字定数でなければならない。

＜例1＞　MOVE　ZERO　TO　AREA−1.

ある領域の内容をすべてゼロにしたいときに使用する。

この操作をゼロクリアという。

＜例2＞　MOVE　SPACE　TO　AREA−2.

ある領域の内容をすべて空白にしたいときに使用する。

この操作をスペースクリアという。

＜例3＞　MOVE　ALL　”ABC”　TO　PRINT−R.

この結果，PRINT−Rの内容は，左からABCABCABC…で埋められる。

＜例4＞　MOVE　ALL　SPACE　TO　PRINT−R.

この場合，ALLがなくても同じ意味になる。したがって，ALLは単に読みやすさだけのためのものである。

＜例5＞　MOVE　ALL　”−”　TO　PRINT−R.

記憶領域PRINT−Rは，”−”で埋められる。

印字用紙に長い横線を引くときなどに用いられる。

1.5.8　語のまとめ

- 語
  - 予約語
    - 必須語
      - 必要語 … 書き方で下線の引いてある大文字の語
      - 特殊文字語
        - 算術演算子 … ＋　－　＊　＊＊　／
        - 比較文字 …… ＝　＞　＜
    - 補助語 …… 書き方で下線の引いてない大文字の語
    - 特殊目的語
      - 特殊レジスタ
        - LINAGE−COUNTER
        - LINE−COUNTER
        - PAGE−COUNTER
      - 表意定数
        - ZERO
        - SPACE
        - HIGH−VALUE
        - LOW−VALUE
        - QUOTE
        - ALL定数
  - 利用者語
    - 段落名
    - 節名
    - ファイル名
    - レコード名
    - プログラム名
    - 指標名
  - システム名
    - 計算機名
    - 作成者語
    - 言語名