事務用共通言語COBOLの紹介*

関根智明**　吉村鉄太郎***　敷地　望***

まえがき

アメリカにおいて，事務用共通言語を使うことによってプログラミングを楽にし，電子計算機の事務方面への応用をより効果的かつ経済的に行おうという声が高まってきた．そこで産業界における計算機の使用者，国防総省，他の連邦政府諸機関および計算機の製造者たちが集まり，共同作業によって 1960 年 4 月 Common Bussiness Oriented Language の第 1 回仕様（COBOL 60）が作られた．いままで各社で開発されてきた事務用プログラミングシステム（例，IBM Commercial Translator，UNIVAC-I，II FLOW-MATIC 等）も今後 COBOL に統一される見通しである．そして現在すでに UNIVAC-II COBOL と，RCA-501 COBOL Narrator が完成している．また 1962 年中には最近各社から発表されている大形機のための COBOL processor が相当数完成するものとみられている．

1961 年 7 月に COBOL-61 が発表された模様であるが，資料未着のためここでは COBOL-60 を紹介する．以下この仕様書の解説を行うが，紙数の関係で細部にわたる完全な記述は行なわず，重点的な紹介に止めた．ただ COBOL 開発の過程，組織について多少くわしく書いたのは，それらを通じて COBOL の思想やその背景を明らかにするためである．この方面に関心をお持ちの方々に役立てば幸である．なお COBOL のより詳細な手引書が，実際のプログラム例と共に近く日本電子工業振興協会より刊行される予定である．

この報告書では，COBOL システムをつくりあげ，そのマニュアルを書く場合にまえがきの一部に以下の全文をかかげることを要求している．この中で COBOL-60 の開発に参加した各組織体や参考となったプログラミングシステムについて言及している．

この解説は，1959 年アメリカ政府内の電子計算機使用者およびアメリカの電子計算機製造者によって構成された自発的な委員会が開発した COBOL System に関する報告書（Report to Conference on Data Systems Language, 1960）に基づくものである．

最初の開発に参加した組織体は次のとおり．

Air Material Command, U.S. Air Force
Bure au of Standards Department of Commerce
Datamatic Division, Minneapolis-Honeywell Corporation
David Taylor Model Basin, Bureau of Ships, U.S. Navy
Electro-Data Division, Burroughs Corporation
International Bussiness Machines Corporation
Radio Corporation of America
Remington Rand Division of Sperry Rand, Inc.
Sylvania Electric Products, Inc.

COBOL 言語の最初の仕様は，上に記したすべての組織体によってなされた寄与の結果である．プログラミングシステムおよび言語の正確さと機能に関しては各寄与者あるいは委員会とも完全な保証を与えるものではない．またそれに関係する事項に関する責任を負うものでもない．

COBOL に対する多くの改良と追加が行なわれることは当然期待されるところである．プログラミングに対する，現在までに行なわれた投資が無駄にならないように，秩序だった改善や訂正が加えられるよう，万全の努力が払われることであろう．各コンパイラ開発者はこのことを積極的に保証するよう努力されたい．

COBOL の維持のための手続が確立されている．この手続および変更を提案する方法についての照会は Executive Committee of the Conference on Data Systems Languages にあてていただきたい．

使用された版権をもつ資料，すなわち FLOW-MATIC (Trade-mark of Sperry Rand Corporation) Rrogramming for the UNIVAC ® I and II, Data Automation Systems © 1958, 1959, Sperry Rand Corporation および，IBM Commercial Translator,

* An Introduction to COBOL-60, by Tomoharu Sekine (Dept. of Administration Engng., Faculty of Engng., Keio Univ., Tokyo), Tetsutaro Yoshimura and Nozomi Shikichi (Tokyo Shibaura Electric Co,. Ltd.)

** 慶応義塾大学工学部管理工学科
*** 東京芝浦電気株式会社

Form No. F 28-8013, copyrighted by IBM の著作者ならびに版権所有者は，COBOL の仕様書中にその一部または全部を使用することを認可した．

このことはプログラミング・マニュアルやそれに類似の刊行物において COBOL 仕様書を複製したり，使用したりする場合にも適用されている．

いかなる組織体も，自由に COBOL に関する報告書や第1回仕様書を複製または一部を使用すること，

この報告書からのアイデアを使用すること．この報告書をインストラクション・マニュアル，その他のあらゆる目的に使用することができる．しかしこれらすべての組織体は，その文書の序論の一部として COBOL Initial specification, April 1960 の I-6 Acknowledgement の全文をかかげなくてはならない．書評などにおけるように，短い文章を用いる時は資料の出所に関する謝辞において“COBOL”に言及すれば全文をかかげる必要はない．

なお，この解説書作成のためにいろいろ御便宜を計ってくださった東芝本社総務部機械計算課の宇都宮課長および吉村賢譲氏に厚く御礼申し上げる次第である．

1. 序

1.1 COBOL の目的

電子計算機を用いてデータ処理をしている企業体，政府諸機関，学校等の数は非常に多い．中には何台もの異なった種類の計算機を使っている所もある．こういった使用者達の得た一般的な経験の一つとして，「電子計算機を経済的に広い範囲にわたって応用するには，プログラムの開発と維持に要する時間と費用をできるだけ少なくしなければならない」ということが明らかになった．

どのような特定の計算機とも独立で，自由に追加補足ができ，そして英語で記述できるような一種の共通事務用語がこれらの問題を解決するのに有効である．

一般に次のような事柄が事務用共通言語の開発をうながしたものといえよう．

(a) あるデータ処理システムにおいて，その中で使われている計算機を，将来より強大なものととりかえる時の変換費用を最小にとどめたいという要求，また異なった種類の計算機を何台も使っている場合には，それらの機械に対するプログラムの互換性の要求．

(b) 経営活動の変化，拡張に伴ってデータ処理システムも不断の改訂，拡大を要する．このような変更，追補を容易に行なえ，それに要する費用と時間が最小で済み，しかも現在のシステムの内容を完全な形で記述できるような言語体系が要求されている．

(c) 大量のプログラムを急いで作り上げる必要にせまられた時は，現存のプログラミング・スタッフに過重の負担をかけるか，さもなければ比較的未経験なプログラマーを沢山集めなければならない．

このような不合理を解決する手段が欲しい．

1.2 COBOL 開発の歴史

COBOL 開発のためのアメリカにおける組織は次のようにして構成された．1959年5月28～29日，国防総省で事務用共通言語の必要性および実行可能性を論議する目的の「データシステム用言語会議」が開かれた．この会議に出席したのは，民間および政府諸機関の使用者，計算機製造者およびその他この問題に興味を抱くグループであって，このような言語の開発がのぞましく，かつ実行可能であるという点で意見が一致した．そして実際の開発に当るための3種の委員会，すなわち

短期作業委員会
中期作業委員会
長期作業委員会

を設立した．短期作業委員会は6人の製造者代表および3人の政府代表から構成され，その第1回会合は1959年6月23日に開かれた．そして短期作業委員会の仕事を下記の四つに分け，それぞれの作業グループを設けた．

Data Description
Procedural Statements
Application Survey
Usage and Experience

これらのグループの活動の結果は短期作業委員会で検討され，言語の第1回仕様が Common Bussiness Oriented Language の名を付して，1959年12月にデータシステム用言語会議の理事会に提出された．そして短期作業委員会は1959年12月以降も存続して，実際のコンパイラーの開発状況のモニターとしてその活動をつづけることになった．

理事会は1960年1月にこの報告を受理承認し，汎用電子計算機の使用者たちが事務データ処理の問題のプログラムを組む場合に，できるだけこの COBOL 言語を使用することを推薦している．

1.3 参加者

短期作業委員会に参加したメンバーは次のとおり.

Air Material Command, U.S. Air Force
Bureau of Standards, Department of Commerce
Computer Science Corporation
Datamatic Division, Minneapolis Honeywell Corporation
David Taylor Model Basin, Bureau of ships U.S. Navy
Electro-Data Division, Burroughs Corporation
International Bussiness Machines Corporation
Radio Corporation of America
Remington Rand, Division of Sperry-Rand Inc.
Sylvania Electric Products Inc.

COBOL システムの作成にあたって，多くの資料を参考にした．特に Sperry-Rand で開発された FLOW-MATIC* System, IBM によって設計された Commercial Translator System および Air Material Command と Sperry-Rand の共同開発になる AIMACO System から多くの情報とアイデアを借用した．また製作者および発行者の許可を得て，下記の版権をもった刊行物から若干の資料を得た．それらは Sperry-Rand 社の FLOW-MATIC Programming System, © 1958 および IBM 社の General Information Mannual: IBM Commercial Translator © 1959 の二つである．

1.4 COBOL システム開発の3段階

短期作業委員会は COBOL システムの開発の段階を次の三つに区分した．

第Ⅰ段階　最小限度の機能を持った COBOL (Basic COBOL)

第Ⅱ段階　短期作業委員会より提出された第1回仕様を満足するような COBOL (すなわち現在の COBOL)

第Ⅲ段階　より理想的な内容を持った COBOL (Extended COBOL)

短期作業委員会の定義によれば，Basic COBOL とは次の三つの部分からなるシステムをいう．

(1) ENVIRONMENT DIVISION　compiler 設計者が要求した性能を持つ入出力システムを，COBOL compiler の中で実際に利用できるようになっていればよい．

(2) DATA DIVISION　COBOL 本文中に定めるすべての機能をそなえていること．

(3) PROCEDURE DIVISION　下記の例外を除いたそのすべての特長と動詞が利用できるようになっていること．下記の機能および動詞が除外される．

a. 数式と動詞 COMPUTE

b. プログラムの分割機能（プログラム全体が計算機の主記憶装置に全部入り切らない時に使われるテクニック）

c. 二つ以上の IF を含む条件文

d. 動詞 OPEN において REVERSE の機能の指定ができること

e. 動詞 PERFORM の中で VARY を選択した時その後に BY, FROM, TO "field-name" と続けること

f. 動詞 PERFORM の中で UNTIL を選択して使用すること

g. 動詞 INCLUDE

h. 動詞 USE

i. 動詞 DEFINE

j. 動詞 ENTER

k. 動詞 MOVE の中で CORRESPONDING を選択して使うこと．

(もちろん，これらの除外例はすべて現 COBOL システムでは使用可能になっている)．

また短期作業委員会では次のような機能ができるだけ早く COBOL に追加されるべきであるとしている．

数表処理機能

データを一つの外部媒体から別の媒体に写したりすること（例．カード→磁気テープ変換）およびその時のデータの形式の変換機能

報告書作成機能

SORT-MERGE 機能

1.5 COBOL の維持

COBOL に関する各方面からの質問に回答し，また COBOL の仕様を改良，変更するため，COBOL の維持組織が設立されている．それらは下記の2委員会である．

技術委員会

短期作業委員会に参加した計算機製造者6社の代表1人ずつと，その他の製造者で特にこの委員会に参加を希望する会社の代表とで構成され，委員は各社における自動プログラミング技術の最高権威で，しかも自社を代表して発言できるような地位にある人

* Trademark of Sperry-Rand Corporation.

でなくてはならない．この委員会は製造者の立場より COBOL を追補するための提案をとり上げ，それらの必要性，技術的可能性および実際的な価値の有無を検討し，すべてのこのような提案に対して委員会の action として賛成または反対をする．またこの委員会自体からの提案は理事会において提案番号がつけられ，維持委員会で同時に検討される．

維持委員会

この委員会のメンバーは空軍，海軍，U.S. スチール，Esso Standard Oil および特に参加を希望する使用者によって構成される．委員の資格および委員会の内外から提出される提案についての取扱いなどは，技術委員会の項で述べた内容に準ずる．

COBOL に対するすべての追加，明確化および変更は，理事会より発表される番号付きの“追補”の形で行なわれる．理事会は外部からこのような追補のための提案を受取ると，提案番号をつけて技術および維持委員会に送付して検討させる．また上記2委員会からの提案は，提案番号をつけるために理事会に送られ，番号がつけられてもう一方の委員会に送られる．二つの委員会の両方で承認された提案は直ちに受理され，COBOL に対する公式な番号のついた追補として配布される．二つの委員会の賛否が対立したは時は理事会における票決が行なわれる．これらの追補は定期的に集められ，COBOL の第1回仕様と同じ方法で公刊される．一方の委員会で承認された提案が他方の委員会に付託された場合，その委員会は2週間以内にその態度を決定しなければならない．さもないと承認とみなされることになっている．

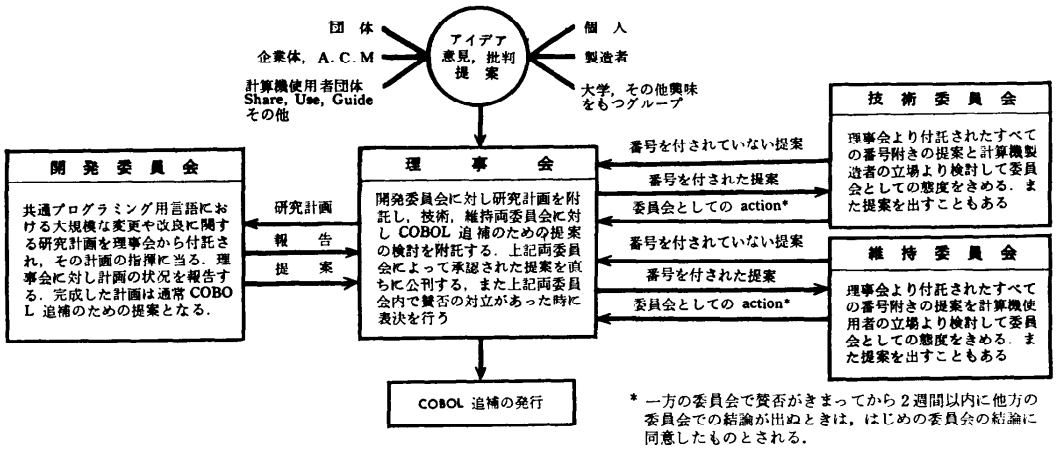
COBOL の維持に関連して別に開発委員会が設けられた．この委員会はプログラミング用共通言語の大規模な変更，改良に関する研究計画の指揮に当るもので，これらの研究計画は理事会から付託されたものか，理事会によって承認されたものである．開発委員会でとり上げられる研究計画のための提案は，この委員会や理事会の中から出たものでもよいし，外の組織からのものでもよい．それらは理事会によって計画番号がつけられる．開発委員会は研究計画の進展状況について定期的な報告を行なう．開発委員会はまた COBOL の追補のための提案を行なうこともできる．

理事会および上述の各委員会との関係は，下の図によって示すことができる．

2. COBOL の一般的な説明

2.1 COBOL 開発の一般的な思想

事務データ処理と呼ばれている種類の問題の解法を表現するための標準的な方法を確立しようとする努力が，共通事務用言語をつくり出すことになった，ここで共通（common）という意味は source program で使う言語が異なった種類の計算機の間で通用するということである．もちろん計算機によって記憶容量，付属装置の形，命令構造などは大きく異なるから，完全な共通性を持たせることはできない．そこで現存する計算機の間で最大の両立性を持たせるようにすることを目標におき，これを現実的な骨組としてすべての仕事をすすめることになった．



データシステム用言語会議において COBOL 維持のためにつくられた組織

データ処理の問題を記述する時には次にのべるような事柄を明確に表現しなくてはならない．一つはデータがどのように処理されるかを指定する手続の集合であり，もう一つは扱われるデータそれ自身についての記述である．さらに問題を記述するための必要な部分として，計算機そのものに関する情報も欠くことができない．もちろんこの部分は計算機を変える時には書き直さなくてはならないが，このような計算機ごとにはっきり変ってくるような項目の記述のために，標準的な書き方を開発することは非常に有益なことである．

このような考え方が発展して，COBOL は三つの部分からなっている．それらを計算機相互間の共通性の大きさの順に書くと下のようになる．

手続きを表現する文章
データの記述
機械に関する記述

2.2 COBOL System の構成

以上の3部分に対応して事務用データ処理の問題の解法を記述指定するために次の三つの機能を遂行する記字が必要となる．

（1） データがどのようにして処理されるかを決定するための手続の集合
（2） 処理されるべきデータについての記述
（3） 処理に際して使われる機器の記述

COBOL ではこれらの三つの division を次のような名前で呼んでいる．

PROCEDURE DIVISION
DATA DIVISION
ENVIRONMENT DIVISION

そしてこれら三つの division においては，すべて英語を使用することになっている．そして COBOL では単語を構成する文字は数字，アルファベット，負号にかぎられ，その他句読，数式表現，編集（editing）のために若干の記号を使う．

まず PROCEDURE DIVISION では計算機にやらせようとしていることを英語の文章で書き表わす．これらは単語(名詞，動詞，接続詞)および諸記号からなる命令文や条件文である．特に何個かの文（sentence）で一つのまとまった操作を表現しているようなときは，このあつまりを paragraph といい，これに paragraph name をつけることができる．さらに paragraph のあつまりを section と呼び，section name をつけることができる．PROCEDURE DIVISION がふつう“プログラム”と呼ばれているものに相当するように見えるが，COBOL System においてはこの部分は問題解法のための完全な記述の一部分に過ぎない．たとえばこの部分ではいろんなデータをとり扱う手順を明示するけれども，データそれ自体の構造についての明確な定義は DATA DIVISION の中でしかなされていないからである．

PROCEDURE DIVISION では，特定の動詞を含む命令文と条件文とによってデータ処理の手続を書き下すので，他の二つの division に比べて，通常の英語に一番よく似ている．またこの部分は本質的に特定の計算機に依存しない．

COBOL PROCEDURE の例：

IF X EQUALS Z THEN MOVE A TO B; OTHERWISE ADD A TO C AND ALSO IF NOT POSITIVE, GO TO ERROR-ROUTINE; OTHERWISE ADD A TO B.

DATA DIVISION では，object program によって操作され，あるいはつくり出されるすべての logical record およびそれらが集まってできた file についての記述が行なわれる．

これらは file および record description と呼ばれる．File の持つ物理的な特性についての記述は大部分扱われない．というのはそれらはある程度個々の計算機に適応したものになるからである．一般的にいえば DATA DIVISION は計算機を変えた場合には書き直されなくてはならないが，データの配列を注意深く計画すればそうしなくてすむ場合もありうる．

FILE DESCRIPTION ではある FILE がどのような RECORD から構成されているか，FILE の大きさはどのくらいかなどを指定するが，その形式は次のようである．

FD *file-name* RECORDING MODE IS *mode*, BLOCK CONTANS *integer*-1 TO *integer*-2 RECORDS, LABEL RECORDS ARE STANDARD, VALUE OF *field-name* IS *literal*, DATA RECORD IS *data-name*-1,…

RECORD DESCRIPTION は FILE 中の名 RECORD の形式についての詳細な情報を含んでいる．この中には，何進法表示であるか，小数点はどこにあるか，何桁くらいか，有効数字の最左桁のすぐ左にドル記号をつけておくか，……といった指定がある．RECORD DESCRIPTION の書き方の例は次のとおり，

level-number data-name SIZE IS *integer* CHARACTERS, BASE IS BASIC TYPE, POINT LOCATION IS LEFT *integer* PLACES, CLASS IS NUMERIC, FLOAT DOLLER SIGNBLANK WHEN ZERO.

ENVIRONMENT DIVISION は二つの部分にわかれ，まず CONFIGURATION SECTION では，実際の処理作業およびプログラムの編集の両方に使われる機器の指定を行なう．ここで指定される項目としては次のようなものがある．すなわち記憶容量，磁気テープ操置の数，などを指定し，金物として付いているスイッチ類，プリンターその他の特定の機器に対して，問題の記述に都合のよいような名前をつけることが行なわれる．ENVIRONMENT DIVISION のもう一つの部分は I/O CONTROL SECTION といって object program が使う入出力制御方式の指定や，一度中断した作業を再開する時の手続の指定をする．

ENVIRONMENT DIVISION は次のような形式で書かれる．

```
ENVIRONMENT DIVISION
    CONFIGURATION SECTION
        SOURCE-COMPUTER. computer-name
          WITH SUPERVISOR CONTROL,
            MEMORY SIZE integer-1
        WORDS ADDRESS integer-2 THRU
          integer-3
        OBJECT-COMPUTER. computer-name
          ……
    INPUT-OUTPUT SECTION
        FILE CONTROL. SELECT……
        I-O-CONTROL. APPLY input-output
          techniques. RERUN ON file-name-1
            EVERY integer
        -2 CLOCK-UNITS……
```

3. COBOL で使用する文字と単語に関する

3.1 文字

1. 単語を構成する文字

単語とは以下に記す文字37種から構成されるものをいう．

0, 1, ……, 9
A, B, ……, Z
—　　（ハイフンまたは負号）

ある単語の中に空白やスペースがあってはならない．それらは単語や文章を区切る時に使われる．

2. 句読のために用いられる文字

"	引用符
(	左括弧
)	右括弧
	空白
.	終止符
,	句点，コンマ
;	セミコロン

3. 数式中に用いられる文字

+	加算
−	減算（ハイフン）
*	乗算（** 巾乗）
／	除算
=	等号

4. 関係を示す記号

＞	…より大きい
＜	…より小さい
=	に等しい

5. 編集（editing）の時に使われる文字

$	ドル記号
*	小切手改竄防止用記号
,	コンマ
.	実際の小数点

6. 文字に関する事項のまとめ

COBOL System が認識することのできる文字は，アルファベット，十進法の数字，句読点，数式や大小関係を表わすための諸記号などである．

しかし計算機によっては上記の文字を完全に取扱うことができない場合もある．このような場合には別の記号で代用してもかまわない．また扱える字の種類が51種より少ない時には2文字を一組としてある文字を表現する方法を採用することもできる．このような代用記号の標準的な一覧表は後日発表されることになっている．

3.2 COBOL で使われる単語

1. 定義

単語とは 0〜9，A〜Z，ハイフンの組合わせからなり，30字を越えないものをいう．単語の終りを示すものは空白かもしくは終止符，右括弧，コンマ，セミコロンである．これらの句読点のあとには空白がくる．ある単語の左端または右端にハイフンがくることは許されない．ただし literal（後述）についてはこれら

の規則の適用が除外される．また，literal の場合をのぞいて，単語とそれにとなり合う句読点との間には空白が入ることは許されない．

2. 単語の種類

（1） 名詞（noun）

下記のいずれかに該当する単語を名詞という．

Data Name
Condition Name
Procedure Name
Literal
Figurative Constant
Special Register Name
Special Names

Noun の中に，読みやすくするためのハイフンが含まれていてもよい．たとえば

quantity-on-hand
stock-number

といった使い方は正しい．ただし，

stocknumber-

のような記法をすると名詞とはみとめられない．

一般に使われている label, tag, field name, operation number などの語は，COBOL の名詞に相当するものである．

a） Data Names

少なくとも一つ以上の英字を持ち，data description で指定された任意のデータにつけられる名前である．そしてデータのレベルの区別をするために，file name と field name の2種類が用いられる．

b） Condition-Names

条件によって値の変わる field が持つことを仮定されている値に対してつけられる名前で，少なくとも一つの英字を含まねばならない．この field それ自身を conditional variable という．Condition-name の実際の値は DATA DIVISION の中の Record Description で定義されている．

たとえば TITLE という conditional variable を考えよう．いま ANALYST, PROGRAMMER, CODER の三つの condition-name にそれぞれ 1, 2, 3 という値がわり当てられているとき

IF CODER THEN……

という conditional expression に対応する object program の中には TITLE という名の field の内容と “3” という値とを比較する操作が含まれる．

ENVIRONMENT DIVISION の中の SPECIAL-NAMES paragraph の中でも condition-name を定義することができる．この場合には金物の装置の点滅状態に condition name をつける．たとえば SENSE FLIP-FLOP を conditional variable として呼ぶ時は，この場合の conditional name に PRESENT と ABSENT という名前を使うことができる．

こうすれば

IF PRESENT THEN…… とか，
IF ABSENT THEN……

というような conditional expression を使うことができる．上のように金物類に対して condition name を使う時には，これらの condition name のとる実際の値の定義がコンパイラで自動的に扱われる点が特長である．

つまり一つの field がいくつかの異なった値を持ちうるとき（一時に一つの値をとるものとして考える），それらの値に一つずつつけられる名前が condition name であり，また field それ自身は conditional variable と呼ばれるのである．一つの variable に関して同一の condition-name を二つ以上の異なる値にわりあててはならないし，condition-name の使用が許されるのは conditional expression の中に限られる．

c） Procedure-Names

COBOL では一つの文章または数個の文章のあつまり，すなわち section で一つの procedure（手続）を表わす．他の procedure から呼ばれることのできるように一つの procedure につける名前を procedure-name という．

他から呼ばれない procedure には procedure-name をつける必要はない．

Paragraph につける procedure-name を paragraph name といい，section につけるそれを section name という．

d） Literal

データに名前をつけずその値そのものを書くとき，それを literal と呼ぶ．Literal は数字，英字または英数字のいずれであってもよいが，英字か英数字の場合はその前後に引用符（quotation mark）をつけなくてはならない．だから literal 自身の中に引用符があってはならない．単一の引用符 ' を literal として書く時は ''' と書く．

数字の literal は0から9までの数字，正号＋，負号－および小数点・を使って書き表わす．一般に数字の literal は引用符で囲んでも囲まなくてもよいが，小数点が最後にくる数字の literal は引用符で囲まなくてはならない．Literal の計算機内部における表現形式はコンパイラが前後関係を判断して自動的にきめる．Literal を使う時のくわしい規則は PROCEDURE DIVISION の各動詞の説明の所にあらわれる．

e) Figurative Constant

Figurative Constant と呼ばれているいくつかの literal がある．それらはきまった名前をもっている．これらの名前は下に記すとおりだが，figurative constant として使うときには引用符で囲んではならない．

ZERO	HIGH-VALUE
ZEROES	HIGH-VALUES
ZEROES	LOW-VALUE
SPACE	LOW-VALUES
SPACES	ALL any literal

Figurative constant は機械語では一連の同質の情報の列として表現され，その長さはその時の前後関係できまってくる．たとえば FIELD という cell があってその長さは前もって六文字分であると決められていたとしよう．いま MOVE ALL 4 FILLING FIELD と書けば FIELD の内容は 444444 となるし，MOVE ALL "FOUR" FILLING FIELD と書けば FIELD の内容は FOURFO となる．

f) Special Register

10進法5桁の特別な register を TALLY という名で呼ぶ．これの主な用途は動詞 EXAMINE を使った時に出てくる繰返し回数の記録である．プログラムの他の部分で使られた情報の記録に使うこともある．コンパイラは TALLY という名詞が初めて出てきたときに，それに対応する記憶装置をわりあてる．

g) Special Names

Special name は金物に対して問題記述に都合のよい名前をつけたり，金物のスイッチの状態に condition-name をつけたりするときに使う．ENVIRONMENT DIVISION でくわしい解説がある．

（2） Verb（動詞）

動詞はただ一つの単語からなり PROCEDURE DIVISION の中である操作を指令するときに使われる．

（3） 名詞または動詞として使えない単語

三つの種類がある：

a) Connectives（接続詞）

Qualifier や subscript といっしょに使われる logical connective は独立な clause の存在を示すために使われ，また compound conditional の評価のための正確な規則を定義するときにも使う．

b) Optional Words

言語のよみやすさを増すために使う．コンパイラはこれを無視する．

c) Key Words

動詞と一緒に使われて動詞の意味をはっきりさせたり，文章をはっきりさせたりするのに使われる．

3. 単語の特殊な用法

（1） Qualifiers

名前の同じな2人の人間を区別する時には姓字をつけて呼ぶ．この姓字にあたるものが COBOL でいう qualifier で，name を呼ぶ時 qualifier をつけることによってその意味を一義的に決定することができる．当然 qualifier はそれが qualify する name よりレベルが高い．しかしある name を呼ぶとき，その name が属している序列（hierarchy）の最高のレベルから順に下方に qualify を行なう必要はない．その name の意味を一意に定めるに必要かつ十分な範囲の qualification さえ行なえばよい．

COBOL では data-name に対する最高位のレベルは file-name であり，また section-name に対する最高位のレベルは section-name である．

Qualification のやり方には prefixing と suffixing の二とおりがある．前者では最も高いレベルの名詞を先頭として序列の下る順に qualifier をかきならべ，最後に最終的に qualify される名詞がくる．後者ではその順序がちょうど反対になり，各名詞の間に OF または IN を入れて意味をはっきりさせる．この場合 OF と IN とは全く同意語として扱われるから，それらの選択は文のよみやすさのみによってきまる．

Qualification の例として MASTER および NEWMASTER なる二つの record を考えよう．それぞれ CURRENT-DATE および LAST-TRANSACTION-DATE という二つの field をもち，それぞれの field は三つの subfield（MONTH, DAY および YEAR）を持つものとする．いま NEW-MASTER record 中の MONTH を引きたければ次の二とおりの書き方のどちらかになる．

MONTH IN CURRENT DATE OF NEW MASTER

NEW MASTER CURRENT DATE MONTH

前者は suffixing で後者が prefixing である.

Qualification の規則:

1) Qualify される name の外側に qualifier が存在していなくてはならない.

2) 一つの序列中の二つのレベルで同じ名前を使ってはならない. さもないと混乱する.

3) あるプログラムの DATA DIVISION の中で一つの data-name または condition-name が2度以上現われる時は, PROCEDURE DIVISION においてはそれらは一つ残らず qualification が行なわれていなくてはならない.

4) 同一 section の中では paragraph-name は全部ちがったものを使うこと (第10章 LIBRARIES でこの規則の例外が出てくる.後述). Paragraph-name を section-name で qualify してもよいがその時は SECTION という語は使わぬこと. また同一の Section 内の他の paragraph からある paragraph を refer する時は qualification は要らない.

(2) Subscripts (添字)

表の中の一つの要素を選ぶときに subscript を用いる. Subscript の値は整数で, literal または data-name によって表現することができる. そして表の最初の要素に対する subscript は "1" である. また subscript なしに表を呼んだ時はその表全体に対して与えられた名前を呼んだことになる.

Subscript の例は次のとおり:

```
MOVE rate FOR age TO listing.
IF height (10) IS GREATER THAN…
MULTIPLY price FOR 5 By inventory…
EXAMINE class (region) REPLACING…
MOVE rate-table TO storage-area.
```

COBOL では高々3次元の subscript がゆるされている. このとき subscript の順序はレベルの高いものから先に書く. そして subscript の個数はその表の次元の数と一致していなくてはならない. さらに多次元の subscript は括弧でかこまなくてはならない. たとえば

```
MULTIPLY policy-value BY rate
                (age, weight, state).
SUBTRACT rate (10,5,7) FROM…
```

のように書く.

(3) 名詞の列

何個かの名詞を続けて書く時は各名詞の間に次のもののうち一つを入れなくてはならない.

```
,
, AND
, OR
AND
OR
```

コンマを使う時は先行する名詞のすぐ後に書き, そのあとに空白を一字分取る. AND と OR は key word として扱われている.

例:

```
OPEN INPUT MASTER RECEIPTS AND
  ORDERS
ADD ABLE AND BAKER AND CHARLIE
```

(昭和36年6月12日受付)