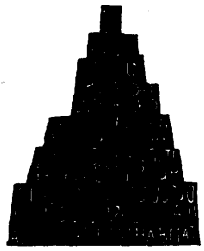

4. C O B O L†

今　城　哲　二††

1. はじめに

COBOL は，世界で最も使われているプログラム言語であり，各種の言語使用状況調査でもほかの言語を大きく引き離して一位の座を占め，今なお減る傾向を示していない．この原因は，事務処理用としてバランスのとれた言語仕様を備えていること，大部分のコンピュータで昔から使えること，CODASYL という公平な言語仕様の保守・改良機構があること，国際規格があり，世界各国のどの機種の COBOL もそれをベースにしていること，書き方が英語に似ておりプログラマにとってなじみやすいことなどが考えられる．

本稿では，第2章で COBOL の歴史について触れ，第3章から第5章でまだ規格化されていない CODASYL COBOL の代表的な3つの機能 ——データベース，整構造プログラミング，ビット操作——を紹介する．第6章では，今後の動向の中で注目すべき事項を指摘する．

2. COBOL の歴史[1),2),10)]

[発端] 1959年5月28日と29日に，コンピュータのユーザとメーカなどから約 40 名が，米国国防総省に集まった．そこで，“共通の事務用プログラム言語”の必要性と実現性が検討され，積極的に推進すべきとの結論に達した．この会議が母体となってデータシステムズ協議会 (CODASYL, the COnference on DAta Systems Language) が作られた．CODASYL の中に既存の事務用言語のコンパイラの評価を任務とした委員会があった．しかし，その委員会は実際には3カ月で言語を開発するという野心的な目標を設定し，9月4日に新言語の報告書を提出した．この言語は更に修正され，事務用共通言語という意味の COBOL (COmmon Business Oriented Language) と名付けられ，1959年12月17日に関係者に配布された．

これらの中で特筆すべきことは，競争関係にある複数のメーカとユーザが利害得失と企業の壁を越えて協同作業によって，ハードウェアに独立な言語を作ったことである．この良き伝統は，その後も CODASYL の中で引き継がれ，20年以上過ぎた今でも，積極的に言語仕様の改良を行っている．

COBOL 言語の設計目標はいくつかあるが，代表的なものは次の2つであり，これらは今でも変わっていない．

（1） 共通性：コンピュータを新しく置き換えたり，ある仕事を一時的にほかのコンピュータで実行したりする場合に，再プログラミングを避ける．

（2） 読みやすさ：プログラムを書く時間よりも，書いた人またはそのほかの人が読む時間の方が長く，回数も多い．このために，きちんと文書化される必要がある．

[発展] CODASYL には COBOL 委員会が常設されている．この委員会は，年に7〜8回，延べ約30日の会合を行い，COBOL について討議し，言語仕様の保守と改訂を行っている．この委員会の正式報告としては，CODASYL COBOL 開発報告 (Journal of Development—JOD) が1〜3年ごとに発行されている．各開発報告で追加された代表的な機能を **表-1** に示す．

[規格化] CODASYL の組織は任意団体であり，また言語仕様は毎日変更・発展を続けている．このため，CODASYL は COBOL の技術的な典拠である

表-1 CODASYL COBOL の発展

JOD 発行年度	代表的な追加機能
1963	整列 (sort)，報告書作成 (report writer)
1965	大記憶ファイル（乱呼出し），表操作
1968	プログラム間連絡
1969	通信，文字列操作
1970	デバッグ，併合 (merge)
1973	大記憶ファイルの大幅改訂（相対，索引）
1976	データベース，ビット操作
1978	整構造プログラミング

† COBOL by Tetsuji IMAJO (Software Works, Hitachi Ltd.).
†† (株)日立製作所ソフトウェア工場

が，別に公式の標準化が必要となった．COBOL の標準化作業は，1963 年から米国で始まり米国規格協会が担当した．

第 1 次規格は，1967 年 1 月 1 日現在の CODASYL COBOL に基づいて作られ，完全な部分集合となっている．これは，米国規格——ANS (1968 年)，国際推薦規格——ISO (1972 年)，日本工業規格 ——JIS (1972 年) などになった．

第 2 次規格は，1971 年 12 月 31 日現在の CODASYL COBOL の完全な部分集合になるように作られ，米国規格 (1974 年)，国際推薦規格 (1978 年)，日本工業規格 (1980 年)[1),9)] などになった．

第 3 次規格は，米国で 1980 年制定を目標に作業が始められたが，1981 年 4 月現在制定されていない．この遅延原因は，第 2 次規格との互換性がなくなることに対しユーザを中心に批判が多いこと，データベースの取り扱い方針がなかなか決まらないこと，CODASYL の進歩が激しいため規格の文書化自体の作業が大変なことなどである．

3. データベース機能[2)~6)]

[経緯] COBOL データベース機能の検討は，1967 年から CODASYL で始まった．実際の作業に従事したのはデータベース作業班であり，いくつかの有名な報告書を 1971 年までに発表した．これらの提案の中には，データベースの定義と操作の分離，スキーマや副スキーマの概念の導入などがあり，データベースの研究や発展に大きな影響を与えた．

CODASYL でデータベース作業班の提案を審議した結果，「スキーマのためのデータ記述言語」の部分を COBOL と独立した言語として扱うことにし，その言語仕様作成のためにデータ記述言語委員会を 1971 年秋に発足させた．データ記述言語委員会は，1973 年に最初の言語仕様書「CODASYL データ記述言語開発報告」を作成した．

一方，「COBOL 副スキーマのためのデータ記述言語」と「COBOL データ操作言語」の部分は COBOL の拡張として扱うことにし，その言語仕様は 1976 年版の CODASYL COBOL から含まれるようになった．

[スキーマと副スキーマ] データベースの設計，操作，保全などの責任者をデータベース管理者と呼ぶ．データベース全体の論理的な構造と内容との記述をスキーマと呼び，物理的な編成法や内部表現の記述を記憶スキーマと呼ぶ．データベース管理者は，スキーマと記憶スキーマをデータ記述言語と記憶データ記述言語を用いて書く．なお，当初はスキーマと記憶スキーマの区別はなくスキーマだけであったが，1978 年版のデータ記述言語から区別されるようになり，スキーマはデータベースの物理的構造には極力触れず，論理構造だけを取り扱うという方向になった．

データベースの利用者は，それぞれの応用プログラムを用いてデータベースを操作する．このためには，個々のプログラムで必要なデータベースの一部の記述 (副スキーマ) とデータベースを操作する命令があればよい．COBOL では，データベース管理者が副スキーマを記述し，利用者がそれを参照することを前提にしているので，副スキーマ自体はプログラム中には書かない．図-1 のように，副スキーマは副スキーマ記述言語を用いて書き，COBOL 副スキーマ登録集に入れておく．COBOL プログラム側は，図-2 のように，データ部の副スキーマ節で使用する副スキーマの名前を宣言するだけでよい．

[データベース操作命令] データベースを取り扱う命令として，ファイルの入出力命令に相応し開始，終了，読み込み，書き出し命令などがあるが，書き方や機能は大幅に異なり種類も多い (表-2)．

```
TITLE DIVISION.                          表題部
  SS 副スキーマ名 WITHIN スキーマ名 (スキーマと副スキーマを対応づける.)
  ⋮
MAPPING DIVISION.                        写像部
  ALIAS SECTION.                           別名節
  ⋮                                      (スキーマと副スキーマの名前を対応づける.)
STRUCTURE DIVISION.                      構造部
  REALM SECTION.                           領域節
    RD 領域名 …                          (領域の記述を行う.)
    ⋮
  SET SECTION.                             親子集合節
    SD 親子集合名 …                      (親子集合の記述を行う.)
    ⋮
  RECORD SECTION.                          レコード節
    01 レコード名 …                      (レコードの記述を行う.)
    ⋮
```

図-1 COBOL 副スキーマの構成

```
IDENTIFICATION DIVISION                  見出し部
  ⋮
DATA DIVISION.                           データ部
  SUB-SCHEMA SECTION.                      副スキーマ節
    DB 副スキーマ名 WITHIN スキーマ名 (COBOL 副スキーマ登録集から副スキーマの定義を呼び出す.)
    ⋮
  FILE SECTION.
    ⋮
```

図-2 COBOL プログラムからの副スキーマの呼出し

表-2 データベース操作命令

分類	命令	説明
準備と終了	READY (準備)	データベース中の1つ以上の領域を使用可能とする．また，領域の専有や共用を指定する．
	FINISH (終了)	データベース中の1つ以上の領域をその実行単位から解放する．
呼出し	FIND (位置ぎめ)	データベース中の見付け出したいレコードの条件を指定し，レコードの位置を定め，それを後続の命令で処理可能とする．データベース操作命令の中で最も重要な命令である．
	GET (読込み)	FIND命令により見付かったレコードを読み込む．
更新	STORE (格納)	データベースにレコードを格納し，必要なら親子関係も設定する．
	CONNECT (接続)	データベース中に格納されているレコードを，親子集合に格納する．
	DISCONNECT (切離し)	データベース中のあるレコードを親子集合から切り離す．
	MODIFY (変更)	データベース中のレコードの内容を変更したり，親子集合の接続を切り替える．
	ERASE (消去)	データベースからレコードを論理的に取り除く．
	ORDER (順序付け)	親子集合中のレコードを論理的に整列する．
保管表	KEEP (保管)	データベースキーの値を登録する表を保管表と呼び，COBOLプログラムの副スキーマ節で定義する．KEEP命令は，必要なデータベースキーの値を特定の保管表に登録する．
	FREE (解放)	保管表にあるデータベースキーの一部または全部の登録を抹消する．
障害回復	COMMIT (付託)	実行単位の論理的な区切り目(休止点)を示す．実行単位が保持していた更新権や選択権はすべて放棄される．
	ROLL-BACK (後退復帰)	一番最近の休止点以後に実行したデータベースの更新系の命令をすべて無効とし，データベースを一番最近の休止点の状態にもどす．通常，データベース例外を処理するUSE節で用いられる．
その他	USE (使用)	(1) データベース操作命令実行中に例外条件が発生したときに，実行すべき手続きを記述する． (2) READY命令実行時点に実行され，データベース中の領域に対してその実行単位が呼出し資格があることを証明する．

4. 整構造プログラミング機能[2),7),8)]

[COBOL の問題点] ダイクストラの「go to 文有害説」に代表されるように，1970年前後には，今までのプログラミングに対し各種の問題点が指摘され，プログラム方法論が議論された．この中から整構造プログラミングの考え方が生まれ，新しい言語が多数あらわれた．また，既存の COBOL などの言語の問題点も明確になってきた．すなわち，COBOL には整構造プログラムを書く上で必要とされる3つの制御要素(連続，選択，繰返し)は，一応備えているが，次のような欠陥がある．

(1) IF 命令を入れ子にすることができるが，制御の合流点は一つの終止符のみである．

(2) IF 命令以外の条件命令を入れ子にできない．

(3) PERFORM 命令では，繰返しの本体を外に置かなければならない．

(4) case に相当する多岐分岐命令がない．

このような批判に対して，CODASYL では，まずシンポジウムを開催し (1975年)，そのあと実質的な審議を行い，1977年から1978年にかけて，言語仕様に整構造プログラミング機能を取り入れた．

[言語仕様] COBOL の命令と文は，次のように改訂された．

(1) END-IF など条件命令の範囲，すなわち制御の流れの終りを明示する予約語を導入する．この予約語は「END-動詞」の形をしており，範囲明示符という．「IF…END-IF」のように範囲明示符で終る命令を範囲明示命令といい，これは無条件命令である．

```
例：IF A=B
      THEN IF A > C THEN…END-IF
                COMPUTE A = A + D
      ELSE  MOVE A TO B
    END-IF
```

(2) READ 命令の AT END 句に対応する NOT AT END 句などを追加し，すべての条件命令で対称的な分岐を可能とする．

(3) PERFORM 命令の繰返し本体の命令群を，PERFORM 命令の中に直接書けるようにする．

```
例：PERFORM UNTIL 条件
      無条件命令…
    END-PERFORM
```

(4) PERFORM 命令の繰返し条件の判定を，繰返しの後 (WITH TEST AFTER) でも行うことができる．

(5) case に相当し多岐分岐を可能とする EVALUATE (評価) 命令を追加する．

```
例：EVALUATE BU-CODE
      WHEN 1  無条件命令—1
      WHEN 2  無条件命令—2
              ⋮
      WHEN OTHER  無条件命令—2
    END-EVALUATE
```

[プログラム構造の大幅改訂] 整構造プログラミング機能はプログラムの制御構造の改善であったが，CODASYL ではほぼ同時期にプログラム構造全体の大改訂も行った．

(1) COBOL プログラムの手続き部の最後の部分

に，ほか COBOL プログラムを入れ子にできる．

（2） プログラム属性として COMMON（共通），INITIAL（初期状態）を導入する．

（3） CALL 命令を拡張し，引数として BY REFERENCE（記憶場所共有）と BY CONTENT（値引渡し）を区別可能とする．

（4） データ及びファイル属性として，EXTERNAL（外部），GLOBAL（全域），LOCAL（局所）を導入する．

（5） プログラム間でファイルを共有可能とし，新しくファイル結合子という概念を導入した．

5. ビット操作機能[2)]

標準の COBOL では文字データと数値データしか扱えないので，ビットデータ操作にはアセンブラや PL/I などを使うほかなかったが，CODASYL でビット操作を可能としたので，COBOL の応用範囲は拡大された．CODASYL ではビット操作のことをブール演算という．

（1） ブール定数は B "ブール文字列" と書く．ブール文字は 0 または 1 である．

（2） PICTURE 句の文字列に記号「1」を書くと，ブールデータ項目になる．ビット列データと文字列データの 2 つのデータ形式があり，USAGE 句で BIT または DISPLAY を書くことにより区別する．

例：01 A PICTURE 1(8) USAGE IS BIT
VALUE B "00110001".

（3） ブール演算式は，ブールデータ項目，ブール定数，ブール演算子およびかっこを組み合わせて作る．ブール演算子は，NOT（ブール否定），AND（ブール積），OR（ブール和）および EXOR（排他ブール和）の 4 つである．

（4） ブールデータ項目やブール演算式は比較条件，MOVE 命令，COMPUTE 命令などに書くことができる．

例：COMPUTE A = B OR C.

6. おわりに

本稿では，COBOL の歴史と，現在規格に入っていない 3 つの新機能（データベース，整構造プログラミング，ビット操作）について紹介した．これら 3 つとも今までの COBOL とは異質なものを含んでおり，短期間に普及するには至らないと思われる．いずれにしろ，次の規格がどこで線引きするか注目したい．

1978 年以降の CODASYL COBOL の主要改訂点として，浮動小数点データと標準関数（三角関数など）の追加，連絡節の削除などがある．これらは，1981 年発行予定の JOD 開発報告に収収される．また，今後の検討項目として，利用者定義関数，誤り報告の一般化，複数の副スキーマ，非同期処理，ディスプレイ端末を用いた画面管理機能などが予定に入っている．この中で，画面管理機能は，対話処理（ワークステーション処理）やオンライン処理で必須のものであり，各メーカが独自に機能追加している状況である．CODASYL がどのような考え方で言語仕様に取り入れるか興味深い．

最後に，日本国内では各メーカが COBOL 仕様の中に日本語データ処理を組み込んできている．これら仕様は，一部類似性はあるがそれぞれ独自のものである．このまま推移すると，プログラム互換性の重大な欠陥となるので，国内独自の標準化が望まれる．

参 考 文 献

1) 日本工業規格（JIS）電子計算機プログラム言語 COBOL, JIS C 6205-1980, p. 383, 日本規格協会 (1980).
2) CODASYL COBOL Commitee Journal of Development 1978, the Secretariant of the Canadian Government EDP Standards Committee (1978).
3) CODASYL Data Description Language Committee Journal of Development 1978, 同上 (1978).
4) 情報処理学会データベース言語研究委員会訳：CODASYL データベース用データ記述言語 1973 年 6 月版, p. 208, 情報処理学会 (1977).
5) 植村俊亮：データベースシステムの基礎, p. 237, オーム社 (1979).
6) Olle, T. W.（西村恕彦・植村俊亮監訳）：bit 別冊 CODASYL のデータベース, p. 190, 共立出版 (1979).
7) 日本経営協会訳：CODASYL プログラミング言語委員会研究報告書 COBOL によるストラクチャード・プログラミングその現状と将来, p. 190, 日本経営出版会 (1977).
8) 植村俊亮，真野芳久：整構造 COBOL（1）～（3），bit, Vol. 10, No. 11, pp. 68-75, No. 12, pp. 51-58, No. 14, pp. 68-76, 共立出版 (1978).
9) 植村俊亮：新しい JIS 規格 COBOL, bit, Vol. 13, No. 2, pp. 60-66, 共立出版 (1981).
10) Sammet, J. E.（竹下亨訳）：プログラミング言語ハンドブック, p. 858, 日本経営出版会 (1971).

（昭和 56 年 4 月 15 日受付）