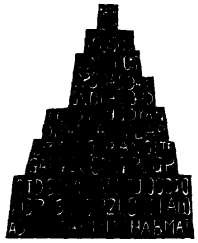

10. B A S I C†

木　下　　恂††

1. はじめに

本稿では，BASIC*の設計当初のねらい，最近の動向，今後の見通しなどについて概括的紹介を行う．BASIC の標準化を中心とした最近の動向については文献[1]で紹介したばかりなので，できるだけ重複を避け BASIC 誕生のいきさつ，設計時の目標に重点を置くことにする．BASIC 言語は今後も発展していくであろうが設計当初のねらいを正しく理解しておくことは，新しい仕様を BASIC に取り入れる際の判断基準の一つともなり有意義なことと考える．

BASIC はよく知られているようにダートマス大学において1963年頃からカーツ，ケメニ両教授の指導のもとに開発された．ダートマス大学は現在でも BASIC 言語発展の中心的役割を果しており，以下ではダートマス大学と計算機との係わりのあたりから述べることにする．

（1） 設計時のねらい

ダートマス大学では，1963 年当時学生の約 75% は文科系であり残りが理工系の学生で占められていた．理工系の学生にとっては計算機を学ぶことは比較的容易であったが 3/4 を占める文科系の学生にも計算機を学ばせる必要があると考えられていた．文科系の学生にアセンブラ言語でプログラムを作らせることなどは到底考えられなかったし，パンチカードの形でプログラムを管理するという当時の標準的な方法も理工系の学生にはともかく文科系の学生を対象にした場合無理があった．そこでカーツ教授らは文科系の学生にも受け入れられるように次の点を重視したシステムを指向した．

（a） 学びやすく使いやすいものであること．何よりも計算機を親しみの持てるものにする．

（b） アセンブラ言語ではなく人間の言葉に近い新しい言語を作る．

（c） 計算機に自由にアクセスでき，ターンアラウンド時間を短くする．

このような要請の中に後の汎用言語としての BASIC とタイムシェアリングシステム (TSS) の出現の芽を見ることができる．

（2） BASIC の誕生とその経過

ダートマス大学の計算機との係わりは，1956 年 MIT にあるニューイングランド地区計算センタの利用を通じて始まった．当初は SAP (Share Assembly Language) と呼ばれるアセンブラ言語を用いていたが一般の学生や研究スタッフに教え込むには問題があった．そこでケメニ教授はよく使う算術演算を SAP の 2～3個の命令から成るテンプレートで表現する方法を用いた DARSIMCO (DARtmouth SIMplified COde) と呼ばれるものを考案した．これがダートマス大学における最初のプログラム言語である．

1957 年，MIT の IBM-704 型計算機で FORTRAN が利用可能となった．当時 FORTRAN はアセンブラ言語に比べて実行効率が悪いという評判であり，アセンブラ言語で直接コーディングすることによって高価な計算機時間を節約すべきであるというのが一般の考え方であった．カーツ教授はアセンブラを用いて数カ月間計算機と格闘した結果，704 の計算機時間を合計1時間とその何百倍ものデバッグ時間を費やした．一方同じプログラムを今度は FORTRAN で実行したところ約5分の計算機時間で完了してしまった．この経験はカーツ教授に強い印象を与えた．高水準言語を用いればプログラミングとデバッグの時間ばかりでなく計算機の時間も節約できるということを身をもって体験することができたのである．

1959 年，ダートマス大学に LGP-30 という小型計算機が導入された．早速 ALGOL コンパイラの開発が始められた．このコンパイラは ALGOL-30 と呼ばれたが学生に広く用いられるには至らなかった．それ

† BASIC by Shun KINOSHITA (TOSHIBA Medical Engineering Laboratory).

†† 東京芝浦電気(株)医用機器技術研究所

* Beginner's All-purpose Symbolic Instruction Code

はこのコンパイラの処理方式が中間言語を一度紙テープに出す2パス方式だったために使いにくかったからである．この経験からカーツ教授は Load & Go*方式のコンパイラの必要性を痛感し，その後 SCALP (Self Contained ALGOL Processor) を開発した．SCALP のプログラムは紙テープから読み込まれると，ほとんど同時にコンパイルが完了する．更に，実行中は原始プログラムの記号を用いてトレースしたりパッチしたりすることができた．この二つの特長，すなわちターンアラウンド時間の短縮と原始プログラムレベルですべてを扱えるということは学生を対象とした教育システムでは本質的な点であることが分かった．SCALP は後に BASIC で置き換えられるまで数百人の学生によって利用された．

1962年，DOPE (Dartmouth Oversimplified Programming Experiment) という言語が設計された．この言語は成功しなかったが変数名として英字1字か英字に数字を一つ付けたものを用いるという規則が使われ現在の BASIC の規則のもとになった．

これらの DARSIMCO, ALGOL-30, SCALP, DOPE といった言語の開発と使用の経験は後の BASIC の誕生に直接間接に影響を及ぼしたと思われるが，特に利用者と計算機のインタフェースをできるだけ簡略化するというカーツ教授の一貫した目標が具体化されている点に注目すべきであろう．BASIC に影響を及ぼしたその他の言語としては FORTRAN, ALGOL, JOSS, CORC などがある．FORTRAN の影響としては，ループ制御パラメタの書き順や増分1の省略などがその例である．ALGOL からは FOR ループや STEP の指定，ループの実行前の終了判定などがある．JOSS からは会話型言語としての影響を受けている．コーネル大学の CORC という言語からは整数と実数を区別しない学生の教育用言語としての影響を受けた．

1964年，GE 社の 225 型計算機**が導入された．これに先立って 1963 年夏よりケメニ教授は GE 社の協力を得て BASIC の開発にとりかかった．同時にオペレーティングシステムの開発も並行して行われた．そして 1964 年5月1日の午前4時頃最初の BASIC プログラムが GE-235 TSS のもとで動いたのである．

（3） 言語仕様と処理系

BASIC プログラムの例を図-1 に示す．BASIC の

```
10   REM ***SAMPLE PROGRAM***
20   DIM A(2)
30   DEF FNX(K)=A(1)*K+A(2)*N
40   READ A(1), A(2), S, N
50   IF S*N=0 THEN 140
60   FOR I=1 TO 10 STEP 1
70   GOSUB 110
80   PRINT S*FNX(I+1)
90   NEXT I
100  GOTO 40
110  LET S=S+FNX(I)
120  RETURN
130  DATA 3, 4, 1, 13, 1, 3, 2, 10, 1, 1, 0, 0
140  END
```

図-1 BASIC プログラムの例

表-1 ダートマス BASIC の歴史

版	年	主な機能追加
初 版	1964	ANS Minimal BASIC 相等
2 版	1964	添字の下限0から
3 版	1966	MAT, INPUT
4 版	1968	RANDOMIZE, ON, TAB
5 版	1970	FILE, CHAIN, CHANGE
6 版	1971	CALL, SUB, SUBEND

初版ではこの 14 種の文が用意された．データの型は，数値と文字列であり数値はすべて浮動小数点の扱いである．配列は1～2次元で添字の値は0から始まる．DIM 文で配列の宣言をするが，これをしないで配列を用いた場合は暗黙の宣言として0～10の領域がとられる．TSS などの会話型環境のもとで動作するためこのほかに利用者コマンドとして NEW, LIST, DELETE, RUN, SAVE などが用意されているのが普通である．BASIC の初版は，1964年に作られその後表-1に示すように版を重ねて改良が行われてきた．世にダートマス BASIC と呼ばれているものである．

BASIC は TSS と密接な関係をもって発展してきたので言語仕様にもそれが反映されている．たとえば，端末から入力する際に BASIC の文は常に行番号を頭に付ける．この行番号は端末から文を追加したり修正したり削除したりする際の指標として用いられる．モニタにとっては端末から入力されるテキストの頭に行番号があるとこれをプログラムの一部であるとして保存し，行番号がないとモニタに対する指令（コマンド）であると解釈して即実行するという仕組みになっている．またプログラムは文を入力した順に構成されるのではなく，行番号にもとづいて自動的に並べ換えられ小さい行番号順に入力されたかのごとく解釈される．このように行番号は会話型で BASIC 言語を用いる際にきわめて重要な役割をはたしていることが分かる．

* 目的プログラムを中間媒体に出さず翻訳と同時にすぐ実行する方式．

** General Electric 社の計算機は後に 235 型に置き換えられた．

BASIC のコマンドについては一応言語外の問題と考えてよいが，できるだけ利用者が親しみを持てるようにという考えから LOGON, CATALOG, EXECUTE, LOGOFF, ……といったものではなく HELLO, NEW, OLD, RUN, BYE など初心者になじみやすく作られているのが普通である．

BASIC といえば会話型言語の代名詞のようにいわれているが設計の初期の段階では端末からデータを入れるための INPUT 文はなく本当は会話型言語ではなかった．事実 BASIC はバッチ処理環境下でも利用可能なように言語仕様が定められている．

BASIC の処理系は RUN コマンドを受けとるとプログラムの翻訳と実行を連続して自動的にやるようになっているものが多い．この結果利用者の立場から見ると BASIC プログラムの目的コードの存在を意識する必要がなく，即実行結果が得られるように見えるのである．もっとも，このようにできるためにはコンパイルが速いことが要求される．BASIC 言語の仕様はそのために1パスでコンパイルできるように設計されている．

処理系の実現方法は本来言語仕様の一部ではないが利用者の使い勝手，および実行効率とも密接に関係するのですこしふれてみたい．処理系の実現方法としては，各文を原始プログラムのまま（または多少変更して）保持していて，実行段階で逐一解釈実行していく（インタプリタ）方式と，一度原始プログラムを目的プログラムに変換*し実行する（コンパイラ）方式とがある．ダートマス BASIC では一貫してコンパイラ方式がとられてきたことはあまり知られていない．インタプリタ方式ではコンパイラ方式に比べ実行効率が悪くなるが，直接実行モードといって行番号を付けないで入力するとコマンドのように即実行できるなどの利点がある．しかし BASIC ではこの利点はさほど重要ではなく，コンパイラ方式と端末からの会話型プログラム入力とでバッチ処理よりも大幅なターンアラウンド時間の短縮をはかることをめざしたのである．このような設計時の方針にもかかわらず，その後会話型環境下でインタプリタ型の BASIC が多く開発されてきたのは，おそらく当時のほかのコンパイラ型の言語が多重パス方式のためあまりにも処理時間が長くかかるという背景もあったであろう．また BASIC 普及の原動力となったミニコンがほとんどインタプリタ方式を採用したためもある．この結果 BASIC は広くインタプリタ方式の言語と信じられてきたが必ずしもインタプリタ方式である必要はなく，むしろ多くの点でコンパイラ方式の言語である FORTRAN に似ているといえるであろう．カーツ教授らはもともと1パスで実用的な Load & Go のコンパイラが作れると考えたのである．

実現方法に関してもう一つふれておかなければならないのは会話型*の意味についてである．ある種の BASIC 処理系では1行の文を入力するごとに誤り検出を行い誤りメッセージを利用者にしらせるものがある．この方式では複数の文にまたがる誤りは検出できないし追加修正の処理も複雑になることが多い．また，何よりも利用者の立場からは使いにくい**ことおびただしい．ダートマス BASIC では行ごとの誤り検出は行わず，プログラム全体ができあがり RUN コマンドでコンパイルが始まってから誤り検出が行われる．TSS による会話型使用法がまだあまり普及していない頃のことであるが，コンパイラ型の BASIC を利用者のところへ持っていったところ行ごとの誤り検出を行っていないということで「この BASIC は会話型ではない」とクレームをつけられ閉口した経験を筆者はもっている．今ではこのように会話型の意味を取り違える人はいなくなったが，会話型であってもプログラムを作る段階と誤りを検出する段階とをはっきり分けることが使いやすさの点で重要であることを強調しておきたい．

2. 最近の動向

最近の BASIC は，マイクロコンピュータの発展と切り離しては論じられなくなってきているが，マイクロコンピュータと BASIC の関係については本特集号14 にゆずるとして，ここでは標準化活動について簡単にふれておく．

BASIC の標準化活動は 1973 年頃からカーツ教授を中心として始められた．1978 年 ANS 標準規格として基本 BASIC (Minimal BASIC) と呼ばれる核の部分が定められ，続いて ECMA の協力を得て 1979 年国際標準規格案が制定された．この基本 BASIC は初版

* 最近では原始プログラムを中間言語に変換しそれを解釈実行する方式も増えてきた．これは本来インタプリタ方式に分類されるべきであろうがこの方式でコンパイラ型と呼んでいる処理系もありいささか分類があいまいになっている．

* "会話型" の定義は，それぞれの立場と目的によって異なるであろう．ここでは BASIC のプログラム作成環境に限定している．

** なぜ使いにくいかは体験した人にとっては明らかであろう．書くそばからいちいち誤りを指摘されていてはかなわない．あとでまとめて誤りを修正したい場合もあろう．

表-2 基本 BASIC の文の種類

文	例	
DATA	10	DATA 4, 5
DEF	40	DEF FNX(A)=A^3-5
DIM	30	DIM A(50), B(6,10)
END	999	END
FOR	65	FOR I=1 TO 100
GOSUB	20	GOSUB 4000
GOTO	100	GOTO 60
IF	30	IF X>Y+10 THEN 50
INPUT	70	INPUT A$
LET	50	LET A(I)=SIN(X)+B
NEXT	80	NEXT I
ON	70	ON N+1 GOTO 10,20,30
OPTION	20	OPTION BASE 1
PRINT	25	PRINT "OK", C$
RANDOMIZE	10	RANDOMIZE
READ	300	READ A, B(1), C
REM	115	REM COMMENT
RESTORE	110	RESTORE
RETURN	400	RETURN
STOP	810	STOP

表-3 拡張 BASIC の機能単位の種類

機能単位	水準の数
中 核	2
配 列	2
組み込み関数	1
型宣言	2
ファイル	3
文字列	2
チェイニング	2
副プログラム	2
書式制御	2
グラフィック	2
例外処理	1
デバッギング	1
リアルタイム	2
編 集	2
コモン	1

のダートマス BASIC に言語規模の点で近いものである．表-2 に文の種類を示す．更に，より大きな仕様を含んだ拡張 BASIC (Enhancement Module) の原案作りが 1978 年頃から始められ 1980 年頃標準化することをめざしていた．しかしこの作業は現在かなり遅れている．拡張 BASIC は表-3 に示す通り 15 種の機能単位からなり，更にそれぞれ 1～3 水準に分けられている．基本 BASIC を核としてこれらの機能を必要に応じて選択することにより適当な大きさの BASIC 処理系を構成することができるようになっている．

3. 今後の見通し

拡張 BASIC の 15 種の機能単位のうち中核，配列，組み込み関数，型宣言，ファイル，文字列，チェイニング，書式制御の機能単位で計画されている拡張仕様はすでによく知られているものがほとんどである．一方，副プログラム，グラフィック，例外処理，リアルタイム等の機能単位で計画されている拡張仕様についてはあまり知られていないものがあるのですこしふれてみたい．まず副プログラム機能単位であるが，従来の BASIC ではスコープとかモジュラリティの概念*はなかったが CALL 文，SUB 文，SUBEND 文を導入することにより FORTRAN における副プログラムと同様の記述ができるようになる．OPTION 文の効果もこの副プログラムごとに局所的に定められる**．

グラフィック機能単位については PLOT 文，SET 文，INQUIRE 文，GPRINT 文，GINPUT 文などを用いてグラフィックディスプレイ装置に 2 次元グラフや線画像を作り出すことができる．またピクチャの定義を用いて副プログラムのようにグラフィック手続きを記述しグラフ出力を変換したり回転させたりすることもできる．

例外処理機能単位では，例外を発生させたプログラムの環境下で例外を処理する機能と，その処理が成功した場合に制御を元に戻す機能がある．

リアルタイム機能単位では，並列処理のための PARACT……PAREND 文，プロセス宣言のための PROCESS 文，プロセス入出力のための IN 文，OUT 文などが計画されている．

編集機能単位では，従来言語外の問題として扱われていた編集用コマンドが言語の一部として取り入れられようとしている点が注目される．

このように BASIC 言語は，設計初期に比べ驚くほど仕様が拡張されてきている．この傾向は，他のプログラム言語の影響と最近の技術進歩を反映して今後ますます助長されることが予想される．この結果標準化活動はますます困難になり，独自の仕様をもつ BASIC 処理系の乱立は当分続くものと思われる．

* BASIC における変数名の局所性は定義関数の引数に認められるだけでありスコープという概念はないに等しい．すなわち主プログラムもサブルーチンもすべて一緒に一つの広域的環境を構成している．その結果データはすべてプログラム中のどの場所からでも引用可能になっている．また，モジュラリティという概念もないのでサブルーチンの入口点は明確には存在しない．GOSUB 文によってサブルーチンにどこから入り込んでもよく，GOSUB 文によらずにサブルーチンに迷い込んだ場合 RETURN 文でとんでもないところへ行ってしまうなど欠陥があることが知られている．

** この結果，配列の添字の下限を 0 にするか 1 にするかの問題[1]で妥協案として基本 BASIC に導入された OPTION BASE の機能は，一つの処理系で両方をサポートしなければならなくなるので将来 BASIC 処理系作成者にとって大きな負担となることが予想される．

4. 文 献

BASIC 関係の文献については，初心者向きのものとしては市販のものが多数あるので特に挙げない．言語設計者，処理系作成者向けの専門的文献としては以下のものを挙げておく．

BASIC 全般

1) 木下 恂：BASIC の標準化の動向，情報処理，Vol. 22, No. 5 (1981).
2) Kurtz, T. E.: ACM Sigplan History of Programming Language, BASIC ACM Sigplan Notices, Vol. 13, No. 8, pp. 103-118 (Aug. 1978).

基本 BASIC

3) First ISO Draft Standard Minimal BASIC, ISO/TC 97/SC 5 N 391.
4) Draft Proposal ISO/DP 6373 Minimal BASIC, ISO/TC 97/SC 5 N 456.
5) DIS Minimal BASIC, ISO/TC 97/SC 5 N 497.
6) BASIC の標準化に関する調査（言語標準化専門委員会報告書）54-C-374，日本電子工業振興協会.
7) マイクロコンピュータのプログラミング，付録 最小 BASIC 規格案 bit 臨時増刊2月号 (1978).

拡張 BASIC

8) Preliminary Draft, American National Standard for BASIC, X 3 J 2/79-51 (Sep. 1979).
9) BASIC の標準化に関する調査（言語標準化専門委員会報告書）55-C-393，日本電子工業振興協会.

(昭和56年3月2日受付)