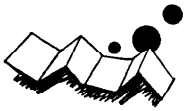
解　説



BASIC の標準化の動向†

木　下　　恂††

1. はじめに

BASIC* の標準化というと「何をいまさら」の感がしなくもない．広く普及しているにもかかわらず，歴史の古いこの言語が今まで標準化されなかったことは不思議ですらある．

一般にプログラム言語の標準化活動の取り組み方を，実際に使われている処理系とできあがった標準規格との比較という観点から見ると，初期のFORTRANのように既存の処理系で広く使われている仕様を厳密に定義する（規格後追い形）という姿勢と COBOL のようにまず規格を定めてそれに沿って処理系を作る（規格先行形）という姿勢の違いがあるように思う．従来，規格化の第一段階では規格後追い形であり，その後規格改訂の第二段階で新しい仕様をとり入れて規格先行に転ずることが多い．最近では最初から規格先行する形がふえてきている．ところで，BASIC では初めての標準化であるにもかかわらず，上記の比較からいえば最初から規格先行形がとられ既存の処理系にはない仕様がとり入れられている．このような形をとった理由およびこれから述べる BASIC の細かい仕様を理解するためには，BASIC のたどってきた歴史を知ることが早道である．また，BASIC の標準化活動がたどってきた，あるいは，これからたどろうとする過程はプログラム言語の標準化をすすめる者にとってきわめて暗示的であり，標準化の時期とその方法について考えさせられる点が多い．

本稿では，第2章で BASIC の標準化の経過を説明し，第3章で基本 BASIC の概要を紹介する．さらに第4章では仕様決定にあたって特に議論された点について代表的なものを挙げる．第5章に拡張 BASIC の仕様について簡単に紹介するが，BASIC の言語仕様に興味のない読者は3章とともにこの章を読みとばしてもよい．最後に第6章で標準化をすすめる上での問題点について言及する．

2. 標準化の経過

BASIC の誕生は 1964 年にさかのぼる．正確にはダートマス大学で 1963 年から 1964 年にかけて GE-225 計算機の上で処理系が開発された．この間の事情については文献1）が詳しい．その後，1973 年*に至り，初めて BASIC 標準化の可能性検討が，生みの親であるカーツ博士を中心として始められた．実に9年ものあいだ言語仕様が野放しになっていた訳で，この間に様々の処理系が作られ，様々の仕様が乱立するに至るのである．このことがその後の標準化活動をきわめて困難なものとした

1974 年1月，ANSI** X3J2 のもとで正式に標準化活動が始められた．続いて9月には ECMA*** TC 21でも取り上げられ，その後 ANSI と ECMA は協力して標準規格案作成に取り組むことになった．

1975 年8月には，カーツ博士が来日され BASIC 標準化について日本の関係者への協力要請があった．その際に Minimal BASIC（基本 BASIC）と呼ぶ核（core）の部分をまず定義し，その後 Enhancement Module（拡張 BASIC）として各種の機能を付け加える形，すなわちいわゆる “core＋module” 形式をとることが紹介された．様々な仕様が乱立している状況から考えて，まず核の部分を定義するというこのアプローチは，きわめて賢明な策であったといえよう．最初から，合意のとりにくい仕様の標準化に取り組んでいたならば共通仕様としてまとめ上げることができず標準化活動は失敗に終ったであろうといわれている．

このような努力が続けられている間，1976 年にオランダが突然 Elementary BASIC というものを ISO**** に提出し標準化を提案した．Elementary

† The Trend of BASIC Standardization by Shun KINOSHITA (Medical Engineering Laboratory, Toshiba).
†† 東京芝浦電気(株)医用機器技術研究所
* Beginner's All-purpose Symbolic Instruction Code.

* 一部の文献に 1972 年とあるがこれは誤りである．
** American National Standard Institute.
*** European Computer Manufacturers Associations.
**** International Organization for Standardization.

BASIC はその適用分野を教育分野に限ったものであったが ANSI や ECMA の活動を知らない国々が多く2月の文書投票の結果6対4で賛成多数となった*. この結果に困った ISO 事務局は，オランダに対し ANSI/ECMA 案をベースにして標準化をはかりたい旨伝えオランダ案を引っ込めるよう勧告する事態になった．オランダは11月に予定されていた SC 5 国際会議でこの問題を取り上げる動きをみせたが結局オランダ案を引っ込めることにした．主要な言語の標準化では，従来 ANSI/ECMA が常に主導権をとってきたが，これを破ろうとするオランダの意欲的な試みも結局失敗した訳である．

このような経過をへて1977年11月オランダのハーグで開催された ISO の SC 5 国際会議で ECMA 版ドキュメントが最初の DP[3] (Draft Proposal) として採用された．ECMA 版ドキュメントは ANSI 案をベースにして一部仕様変更と章の構成を変えただけで内容はほぼ同一と考えてよいものである．しかし主導権は ECMA がとったことを示している．

この DP に対する1978年4月の文書投票の結果，賛成4カ国，コメント付き賛成5カ国，反対2カ国となった．反対の2カ国とは米国と英国であり，米国の反対理由はすでに2月に正式に ANS 標準規格 (X3.6—1978 Minimal BASIC) となっていた米国案と DP すなわち ECMA 案との間の仕様上の微妙な相違点に関連したものであった[2]．一方英国は，配列の下限の既定値を0とすることに対して反対した．

その後各国のコメントを反映した DP 第2版[4]が作られ1979年4月の文書投票の結果賛成9カ国，コメント付き賛成1カ国，反対1カ国となり正式に DIS すなわち国際標準規格案[5] となった．反対1カ国とは英国であり，英国の修正要求に対して明確な回答がないまま却下されたことを理由とした反対投票であった．特に米国が ANS 標準との相違があるにもかかわらず賛成した点が注目された．これで技術的な論争点は一応決着がついたといえる．

一方，これと並行して1978年頃から拡張 BASIC の原案作りも始められた．主として配列，チェイニング，組込み関数，中核，テキスト編集，ファイル，書式および実時間処理などに対する機能拡張が検討されている．当初の予定では，1979年4月までに仕様を決定しコメントを求める．7月にコメント受付けしめ切り，11月に ANSI/ECMA/Purdue 合同会議で検討し1980年11月に標準化することになっていたがかなりスケジュールが遅れているようである．まだ仕様が流動的ということもあり，日本での関係資料入手は遅れているが現時点では文献6)が最新の情報となっている．詳しくは第5章で述べる．

3. 標準規格の概要

基本 BASIC の言語仕様をここに詳述することはできないので概略の紹介にとどめる．読者は BASIC についてある程度の予備知識をもっていることを前提に説明する．さらに詳しく知りたい読者は，ANS Minimal BASIC の邦訳については文献7)を参照されたい．ISO の Minimal BASIC については文献8)に紹介されている．ただし，後者は内容が最新のものではないので注意が必要である．用語の日本語訳については，本稿では前者のものを採用した．

(1) 一般事項

基本 BASIC の規格としての目的は，多くのデータ処理システム間でのプログラムの互換性を高めることでありこれまでのプログラム言語の規格と変りはない．ただしこれまでの規格との大きな相違点として次のものがあげられる．まず，今までの規格では処理系に依存する事項とされていた数値表現の精度と範囲について最低保証すべき値を定めている点である．また，検出すべき誤りと例外条件およびその処理方法をも定めている．BASIC 言語は本来対話的使用を前提としたものであるが本規格ではその形にのみ制限せずバッチ処理的環境をも考慮に入れている．したがって，いわゆるコマンドについては言及していない．もう一つの特長は適合性*(conformance) の記述である．適合性には二つの意味があり，記述されたプログラムが規格に適合する条件とは何か，および処理系が規格に適合する条件とは何かが明確に定められている．特に後者の条件として，「規格では定義されない事項」とか「処理系によって規定される事項」について明確にし，その仕様書が存在することを条件としてあげている点が注目される．

(2) 文字と文字列

BASIC 用文字セットは ISO 標準の7ビット入出

* 日本は，情報処理学会 SC 5 専門委員会で検討した結果，BASIC の標準化という方向には賛成であるが，Elementary BASIC をそのベースとすることには問題があるとして反対投票をした．

* 最近のプログラム言語の規格では適合性を規定する傾向にある．たとえば，標準 PL/I が最初であろう．次いで BASIC, FORTRAN 77 でも取り入れられている．

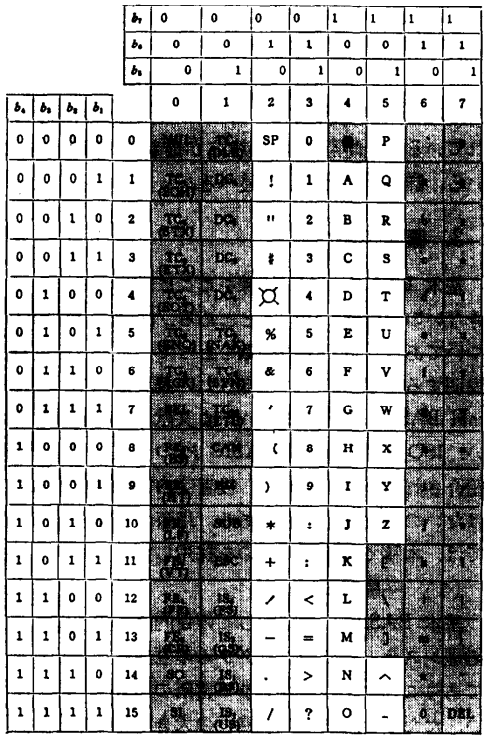


図-1 ISO 標準7ビット入出力コード

力コード（**図-1***）を用いる．文字列を構成する文字列用文字は**表-1** のように分類される．

文字列の例：

（a） 引用符付き文字列 "ABC, DEF"

（b） 引用符無し文字列 A+B C−D.

空白は行の始め，行番号の間，キーワードの間などに使ってはならないが，それ以外ならプログラムを読みやすくするためにどこに使ってもよい．

（3） プログラム

プログラムは一連の文から成り最後は END 文で終る．文としては**表-2** に示す19種**があり1行1文に限る．

各文は，キーワードで識別され1～4個の数字から成る行番号を頭にもつ．

例：999 END

* 図-1 の中で斜線のついていない文字が BASIC で意味のある文字である．文字 "¤" は通貨記号を意味しており通常は "$ "を用いる．

** 規格では END, FOR, NEXT は行の扱いになっており，正確には文は17種類である．本稿では説明の便宜上 END と FOR を文として扱う．

表-1 文字列用文字の分類

文字列用文字	引用符付き文字列用文字	引用符			"
		(特殊記号1)			! # $ % & ' ( ) * , / : ; < = > ? ^ _
		引用符無し文字列用文字	空白		
			素文字列用文字	(特殊記号2)	+−・
				数字	0～9
				英字	A～Z

表-2 文の種類と使用例

文	例（行番号省略）
DATA	DATA 3, 4, 1, −6E10 DATA 3, BASIC, ","
DEF	DEF FNF(X)=X↑4−1 DEF FNA(X)=A＊X+B
DIM	DIM A(6), B(10, 10)
END	END
FOR -NEXT	FOR I=1 TO 10 ～ NEXT I　　FOR I=A TO B STEP−1 ～ NEXT I
GOTO	GOTO 120
GOSUB	GOSUB 4000
IF-THEN	IF X>Y+83 THEN 200 IF A$<>B$ THEN 550
INPUT	INPUT X INPUT X, A$, Y(2)
LET	LET P=3.14159 LET A(X, 3)=SIN(X)＊Y+1 LET C$="ABC"
ON -GOTO	ON L+1 GOTO 300, 400, 500
OPTION	OPTION BASE 1
PRINT	PRINT X; (X+Z)/2 PRINT TAB(10); A$; " IS DONE." PRINT ,,,X
RANDO-MIZE	RANDOMIZE
READ	READ X, Y, Z READ X(1), A$, B$
REM	REM CHECK FOR NEGATIVE DISCRIMINANT
RES-TORE	RESTORE
RETURN	RETURN
STOP	STOP

（4） 定　数

定数はスカラの数値定数と文字列定数とがある．

例：

（a） 数値定数　500　−21.　5E−1

（b） 文字列定数　"XYZ"　"X-3B2"

数値の有効桁数は6桁以上，絶対値の範囲は0および 10^{-38}〜10^{+38} である．文字列定数の長さは1行に書ける文字の数によってのみ制限される．

（5） 変　数

数値変数としては，単純変数と1〜2次元の配列要素を参照する添字付き変数とがある．単純変数は英字1字かまたは英字に数字を付けたものである．添字付き変数は配列を表す1文字の英字のうしろに1〜2個の数値式をかっこでかこんで付ける．

例：X　A5　V(3)　W(X, X+Y/2)

文字列変数としては，英字のうしろに通貨記号を付けて表す．

例：*S*$　*C*$

変数の型の明示的宣言は必ずしも必要ではない．添字付き変数に対する宣言がないときは最初に現れたときに0〜10までの大きさで暗黙に宣言される．文字列変数のもつ文字列の長さは，0から18までプログラム実行中に変えられる．単純変数と配列に同じ英字を用いてはならない．

（6） 式

式には数値式と文字列式とがある．数値式は変数，定数および関数参照に加減乗除算（＋−＊/）およびべき乗（^）の演算をほどこしたものであり，かっこも通常の用法に従って用いられる．

例：3＊X−Y^2　A(1)+A(2)+A(3)

SQR (X^2+Y^2)

文字列式は文字列変数または文字列定数である．

数値式が評価されるときの精度は処理系に依存するが，少なくとも10進6桁の精度を保つことが望ましい．

（7） 組込み関数

数値組込み関数として**表-3** にあげるものが用意されている．

（8） 配列宣言

DIM 文で1〜2次元の配列の記憶域を確保する．配列の添字の上下限は，特別に宣言されない限り下限として0，上限として10をとる．DIM 文を使えば11以上の上限をもつように宣言することができる．

例：DIM　A(6), B(10, 10)

表-3 組込み関数

関　数	関　数　の　値
ABS(X)	X の絶対値
ATN(X)	ラジアンで表したXの逆正接．すなわち，正接がXであるような角度．関数の範囲は，$-(\pi/2)<ATN(X)<(\pi/2)$ である．ここで π は，円周率を表す．
COS(X)	Xの余弦．ここで，Xはラジアン．
EXP(X)	Xの指数．すなわち，自然対数の底（$e=2.71828\cdots$）のX乗の値．もし，EXP(X) が機械の最小値よりも小さいならば，値として0を与えなければならない．
INT(X)	Xを越えない最大の整数．たとえば，INT(1.3)=1，INT(−1.3)=−2 である．
LOG(X)	Xの自然対数．Xは0より大きくなければならない．
RND	範囲 $0\leqq RND<1$ で，一様に分布する疑似乱数で，処理系が提供する順序に従った疑似乱数．
SGN(X)	Xの符号． Xが負ならば −1 Xが0ならば 0 Xが正ならば +1
SIN(X)	Xの正弦．ここで，Xはラジアン．
SQR(X)	Xの非負の平方根．Xは負であってはならない．
TAN(X)	Xの正接．ここで，Xはラジアン．

OPTION 文を使えば，すべての配列について下限として1をもつように指定することができる．

例：OPTION　BASE　1

（9） 文

BASIC のその他の文については表-2を参照されたい．特に説明を要しないであろう．

4. 基本 BASIC の仕様決定の背景

前章で基本 BASIC の言語仕様についてごく簡単に紹介したが，この章ではなぜそういう仕様になったか，あるいはどのような議論のすえに仕様が決定されたかについて少し詳しく述べてみたい．

（1） 誤りと例外条件について

この規格では，誤り (error) と例外条件 (exception) とを明確に区別している．誤りは，構文上の違反によって発生しこれがあるとプログラムは規格に適合しないことになる．本来，誤りは利用者に示されるべきであるが，この規格ではそれを強制していない．この結果，拡張された処理系がそれを構文的に正しいとして処理することが可能となる．一方，例外条件は規格に適合するプログラムの実行時に不正なデータの計算や資源の制限などによって発生する．例外条件は必ず利用者に示されなければならない．ただし，これに対しても拡張された処理系で例外条件を何らかの形で制御

できる場合は表示しなくてもよいことになっている．

例外条件は，致命的 (fatal) なものと致命的でない (non-fatal) ものとに分けられるが，致命的でない例外条件が発生したときの回復手続きも規格で定められている．ただし，ANS 規格では必ずそのような手続きをとることと定められているが ISO 規格では推奨案となっているところが大きな違いである．実はこの点が ANSI と ISO との仕様上の一番大きな相違点であり米国が最後まで ISO 案に反対した点である．

（2） 文字と文字列について

ISO 7ビット入出力文字セットのうち図-1に示す60文字が BASIC 文字セットであることはすでに述べた．しかしこの規格では処理系内部での変換の方法を規定している訳ではない．斜線をほどこした部分の文字については，端末や周辺装置の中にはこれらの文字を受け付けないものもあるので処理できることを義務付けるべきではないという考えである．したがって，文字列の中に制御文字 (0/0～1/15) を入れることは，この規格では規定していない．また，文字列定数の中に引用符を入れることも拡張 BASIC で検討することにした．

BASIC プログラムでの空白の使用法は，英語の規則に適合しプログラムが読みやすくなるようにという方針で決められたが将来の拡張を考慮して特定の場所での空白の使用法には制限がつけられる．たとえば，

```
FOR  I=1  TO  N  STEP  2
FOR  I=1  TO  NSTEP 2
```

において，将来長い変数名を許すような仕様が導入されると異なった意味になってしまう可能性がある．したがってキーワードの前後の空白は常に必要である．しかしタイプをうまく打てない初心者のために基本 BASIC の処理系はできるだけ空白に関する制限をなくすことが望ましいとしている．

基本 BASIC では，文字の照合順序も規定しない．標準的な文字順序は拡張 BASIC で定める予定である．

（3） プログラムについて

この規格では，文の後に注釈を置くことを規定しない．基本 BASIC では REM 文で十分であり他の文に注釈を付けることは拡張 BASIC で扱うことにした．また，一行に複数の文を書いたり2行以上にまたがる文を書いたりすることもできない．初心者が不完全なプログラムを実行させるのを防ぐためプログラムの物理的な終りには必ず END 文が必要であると定めた．その他に，基本 BASIC に含めるよう提案された文が種々あったが複雑で専門的すぎるので拡張 BASIC で検討しようということになった．

（4） 定数について

文字列定数で許される最大の長さは特に定めず，プログラムの行の長さによってのみ制限されることにした．

数値表現の有効桁を6桁としたのは，有効桁数の値を格納するために処理系は少なくとも20桁の2進数と符号を用いることを前提としたからである．0でない数の数値表現の範囲を 10^{-38}～10^{+38} としたのは指数部の値を格納するために7桁の2進数と符号を用いることを前提としたからである．

（5） 変数について

変数の初期値については処理系ごとに定めてよいことにした．現在使われている処理系では数値変数に対し0を設定しているものが多いが，すべての処理系に対しこのように初期設定することを要求すべきではない．たとえば，変数に未定義という値を代入しておきこれを検出可能にできるような処理系では，値が代入される前に変数の値が評価されたときに誤りメッセージを出力できるようにしてもよい．

単純変数と添字付き変数に対し同じ名前を用いてはならないことにした．これは将来拡張 BASIC において各々の名前に一意な型を対応させることができるようにするためである．たとえば，DIM A(20) として A を配列であると宣言すると同じプログラム内では A を単純変数として用いることはできない．

この規格では文字列変数はあるが文字列配列はない．文字列演算子がなくても会話型プログラムで利用者が "YES" や "NO" のような単語を入力できるようにするため文字列変数は重要であると考えた．

配列の添字計算は，切り捨てではなく丸めにした．これは添字の値と添字付き変数の値を対で出力するような場合，不都合が起こるからである．たとえば，

```
100  PRINT  X; A(X)
```

において，A(1)=1，A(2)=2，X を2より小さいが2に非常に近い値であるとすると添字式の計算において切り捨てを採用すると，

```
 2.    1
```

という結果になる恐れがある．

文字列変数は可変長の値をもつが，その実現方法として最大長を適当に小さく定めることにより，いわゆる可変長ブロックの割り当てのような方法ではなく，

もっと単純な方法で経済的に行ってもよいように配慮した.

(6) 式について

式の評価の正確な順序は本規格では定めない. これは次のような問題を避けるためである.

(a) 各処理系は同じ構文解析を行わなければならなくなる.

(b) 算術演算の順序を変える等の最適化ができなくなる.

(c) BASIC の精神に反する.

特に(c)については, 評価順序に依存するプログラムが書けるからである. たとえば, 1E20+1−1E20 の値が1でなく0と評価されることを期待するプログラムなどは BASIC が対象とする初心者には理解し難いものとなろう.

べき乗演算子の山形記号 (^) をもたないキーボードがありうるが, 代りの文字は用意しないことにした. **を代りの文字とする提案もあったが, もし規格がべき乗として二つの表現を許すなら, すべての処理系は一つだけではなく二つの表現を使えるようにしなければならなくなる. この結果規格に適合することがいくつかの処理系で容易になるどころか逆にすべての処理系にとって困難なものとなる. 山形記号のない処理系は代りの文字表現 (たとえば**) を選んでもよいが他の処理系では使用できないものとした.

A^B^C は他の算術演算と一致した規則に従って A^(B^C) ではなく (A^B)^C と解釈することにした. また, 0^0は通常不定であるが公式の表現に使われる数学的記法の慣例に従い1とした.

(7) 組込み関数について

数値組込み関数の評価において最低限保証すべき精度は定めなかった. この問題は BASIC 言語外の問題という考えである. したがって引数の値の大きさが, 関数値のすべての有効桁が失われる程大きくなった場合 (たとえば, 有効桁が6桁であるのに SIN(1E20) とするなど) でも例外条件として扱っていない.

例外条件のオーバフローは, 組込み関数の最終値に対してのみ表示するものとし関数の評価の途中でおこるオーバフローは表示する必要はない. 処理系はこのような評価途中の例外条件に対し最終値の精度を保証するよう適切な処理をすることが要求される.

RND 関数は, 引数をもたないことにした. 乱数発生機構の初期設定は RND 関数の引数を使うのではなく RANDOMIZE 文によって行う. 多くの処理系で初期化に引数を用いているが, これは歴史的に見てすべての関数は1つ以上の引数をもたなければならないと決めた後に, RND 関数の引数を用いることを思いついたという事情によるものであって, そのような使用法が初めから協定されていた訳ではない. もし RND(1) が乱数発生機構の初期化を指定するとすれば, 乱数の必要なときはいつでも1でない引数, たとえば 0, を用いて RND (0) と書かなければならないことになる. 単に RND と書ける方が単純であり分かりやすいプログラムを作れると考えた.

(8) LET 文について

基本 BASIC では, 言語を学びやすくするためにキーワード LET を必ず書くことにした. このキーワードがなくても処理系はプログラムを解析することができるが, すべての文がキーワードをもつなら初心者にとって BASIC を学ぶのが容易になるであろう. さらにキーワードは代入文であるという事実を述べているのではなく動作を規定していることが利用者に分かるという訳である.

(9) 制御文について

「等しい」という関係は, ほぼ等しいというよりは厳密に等しいことを意味する. このことは数値に対してばかりでなく文字列にもあてはまる. すなわち, 文字列の前の空白や後の空白の部分だけが違う文字列でも実際は等しくないものと考える*.

ON-GOTO 文の式の値は切り捨てずに丸めることにした. これは配列の添字式と同じ扱いにするためである. ON-GOTO 文の式の値が, 並びから行番号を選ぶのに使えない場合例外条件がおこる. 単に次の行から実行を続けるという案はプログラムの誤りが発見されない危険性があるので採用されなかった. また, 拡張 BASIC で導入されるであろう ON-GOSUB 文の仕様ともつり合いをとる必要があった.

BASIC の文字の照合順序の定義は拡張 BASIC まで保留された. したがって, この規格には照合順序はないので文字列の比較は＝と＜＞以外は定義しない.

(10) FOR-NEXT 文について

NEXT によって FOR ブロックから出た場合の制御変数の値は, 最後に使用された値ではなく次に使用されるべき値である. これは次に使用されるべき値というのは必ずあるが最後に使用された値というのは, FOR ブロックが全く実行されないときは定義するの

* A␣␣ と A␣ とは等しくない (␣ は空白を表す). 他の言語 PL/I や FORTRAN 77 では等しい.

が難しいからである．さらに，このように定めるとNEXTによってブロックから出たのか，FORブロック本体の中の制御文によって出たのかが制御変数の値によって分かるので，FORブロックから出た後の処理が容易になるという利点がある．

ゼロの増分を許して誤りとしないことにした．これは，無限のループのうち特定のものだけを誤りとして抜き出す特段の理由はないし，すべて抜き出すためにはループの実行前に増分の値を検査する必要がおこり実行時間が増えるという欠点が出てくるためである．

(11) PRINT 文について

PRINT 文の出力において，隣り合う数値表現が互いに連続しないよう各出力項目の終りに常に1つの空白を入れるように定めた．しかし文字と数字の混在した出力では空白をはさまないで連続して出力するようにもしたい．そこで色々なやり方が提案された．たとえば，数値式の後の区切りとしてセミコロンが空白を1つ作り出すものと定めるという案が出たが，これは数値表現のすぐ後に文字を出力することが不可能になる．また，二つの数値式がならんだときだけ1つの空白を作るように定めると印刷並びの最後の区切りとしてのセミコロンがきたときの処理が難しくなるので結局良い案を作れなかった．

印刷区分の個数と長さは処理系によって決められる．これは種々の端末が存在するのであまり厳密に仕様を決めるのは現実的でないという判断によるものである．すべての処理系に対し，特定のプログラムが完全に同じ出力をするように期待してはならない．すなわち，BASIC の書式なしの PRINT 文は一定の書式で出力することだけを意図したものと考えるべきである．

TAB 関数について，現在の行がすでに n 文字を越えているとき，TAB(n) を出力すると次の行の第 n 桁になるものとする．これはタブ指定を用いて表などを作っているとき表の中にある項目が次のタブ位置までに収まらないときにできるだけ表の構造をこわさないようにするためである．タブ指定を無視するという方法も考えられたがあまり望ましいとは思えなかった．第 n 桁にもどるという案は，多くの端末がこの機能をもっていないので却下された．また例外条件を発生させるという案も却下された．

(12) INPUT 文について

入力応答の誤りに対する回復手続きは，入力応答の中のすべての誤りを利用者が修正できるよう定められた．通常行われている回復手続きはある種の誤りに対しては修正が困難であるという理由で採用されなかった．たとえば，INPUT X, Y に対し 1.5 を入力した場合「データが足りない．さらにデータを入力せよ」という意味のメッセージが出されたとする．この場合 1,5 を 1.5 と打ち間違えた利用者に対しては X の値を打ち直すことができなくなる．同様に INPUT X に対し 1,5 を入力した場合「データが多過ぎる．多過ぎる部分は無視する」という意味のメッセージが出されたとする．この場合 1.5 を 1,5 と打ち間違えた利用者に対しては手の施しようがない．

この回復手続きを実現するためには，処理系は変数への代入を始める前にすべての入力応答の検査をしなければならないことになる．この処理に要する費用はたいしたことはなく利用者の便宜を考えれば許容できるものであると判断された．

(13) DATA 文と READ 文について

READ 文での数値データの変換の際に生ずるアンダフロー，オーバフローなどの例外条件は，プログラムの実行をできるだけ続行させるため，INPUT 文の実行の場合とは違い，式の評価のときと同じように取り扱うことにした．

(14) 配列宣言について

省略時の配列の添字の下限を0にすべきかそれとも1にすべきかという問題は容易に結論が得られず大論争が展開された．議論の要点を整理して示すと次のようになる．

(a) 利用者の使いやすさからの議論

0に賛成する人々は，多項式の評価のような数学的応用面ではベクトルの0番目の要素を使うのが普通であり基本 BASIC でこのような使い方を禁止したりあるいは省略時の値を変えるような文をプログラム中に置いた場合だけ許したりするのは，BASIC のもつ教育面における利点を損なうものであると主張した．

一方1に賛成する人々は，0という添字の値が必要な利用者は BASIC 利用者全体から見れば少数であり，利用者の使いやすさという利害関係だけで0または1のどちらかに決定すべきではないと主張した．

(b) 必要記憶域の観点からの議論

1に賛成する人々は，ほとんどのプログラムで0行や0列を使わないのにそのための記憶域を確保してしまうのは無駄であり通常の大きさのプログラムをミニコンピュータの処理系で処理することが困難になってくると主張した．

0に賛成する人々は，小さな計算機で大きなプログラムを実行させたいと思うのは多分訓練を積んだプログラマであろうし，それ以前に添字の値として0を使いたいという初心者としての要求が発生すると反論した．

（c） 規格との適合性からの議論

0に賛成する人々は，現在あるプログラムはそのプログラムが作成された処理系が省略時の値として0をとっていようと1をとっていようと省略時の値として0をとる規格にすべて適合するが，1を採用する規格にはすべて適合するとは限らないと主張した．

1に賛成する人々は，ほとんどのプログラムでは0の値の添字は使っていないので適合性の問題はないと反論した．

結局 BASIC の利用者の大多数のプログラムは，0の値をとる添字は使っていないし記憶域のきわどい使い方をしている訳でもないので，省略時の値としてどちらを選んでもこの大多数の利用者にとっては違いはないことを両者とも認めるに至った．しかしどちらの立場にしても自分の立場を否定する規格を認めることができないのは明らかなので両者に受け入れられる妥協案を見い出すことにした．この妥協案とは，規格に省略時の下限を明記し同時に利用者にこの省略時の値を変えることができるようにするというものである．現在，省略時の値を変えるような機能は一般的にはまだ実現されていない*ので，そのようなものを基本 BASIC に入れるべきではないが，この問題の解決策はほかにはないので採用することにした．結局，省略時の下限は0としオプションで省略時の値を変えられるようにした．最初は DIM 文の変形により各々の配列ごとに添字の下限を指定できるようにするという案があったがその後すべての配列の下限について0か1を指定することにした．ここでは，一般に一つのプログラム言語の美しさや一貫性や無矛盾性よりも利用者の便宜がより重要であるという考え方をとっている．すなわち，配列要素の指標（0から始まる）と印刷行の欄指標や ON-GOTO 文の文番号指標（ともに1から始まる）の間には一貫性がない．しかし印刷行や ON-GOTO 文では利用者は指標位置を最初から順次使用していかなければならないから指標の最小値を意識した上でその値を使うのに対し，配列においては必ず使用しなければならない特定の指標位置というものはなく最小の指標値を知る必要もない．もし下限として1を選べば添字として0を使いたい人は省略時の下限を常に意識しなければならず，「必要なことだけを学べばよい」という BASIC の基本的考えに反することになる．

下限を変更する方法として BASE 文を設けて BASE 1 とする案もあったが，OPTION 文というものを導入し，OPTION BASE 1 と書くことにした．これは OPTION 文の将来の拡張を考えた上での配慮である．

この問題の論争の経過を考えればこのような仕様になったのもやむをえないと思われる．また，既存の処理系の製作者にとっても妥当な結論であったかに見える．しかしこの妥協案は BASIC の将来の拡張仕様との関係から考えると，実現上大きな重荷になってくることが予想される．これについては紙数の関係もあり詳しくは述べない．

(15) 定義関数について

定義関数のパラメータの個数は0個または1個に限った．これは，仮りに2個を許すと2つの数の最大値や最小値を求める関数を定義できるなど利用上のプラス面も大きいが，処理系作成上の負担というマイナス面の方が勝っていると判断した結果である．

(16) REM 文について

この規格では，REM 文がプログラム中に注釈を入れる唯一の方法である．REMARK というようなキーワード REM の別の書き方は，空白の扱いとの関係で許されない点に注意されたい．文の後に注釈を入れたり，空文（すべて空白の行）の使用は拡張 BASIC で考えることにした．

5. 拡張 BASIC の動向

拡張 BASIC の仕様は，表-4に示すように15種の機能単位に分類される．各機能単位はさらに1～3の水準に分けられる．処理系作成にあたっては，基本 BASIC の仕様に加えて各機能単位ごとに水準を積みあげることによって適当な言語仕様を設定することができるようになっている．

仕様検討段階ではこのような形ですすめられているが最終的にどのようにまとめられるかは現段階ではまだ明らかではない．表-4に，各水準ごとに導入される予定の機能を示した．詳しくは文献7) または9) を参照されたい．

* PL/I の DEFAULT 文のような実現例はある．

表-4 拡張 BASIC の仕様案一覧

機能単位	水準	導 入 さ れ る 機 能
中 核	1	行の継続，文字列の中の2連引用符，英小文字，タブ文字の扱い，文末の注釈，空文，ON-GOSUB 文，IF-THEN-ELSE 文，数値配列名の拡張，文字列変数名と配列名の拡張，複数の引数並び，組込み関数 (TIME，DATE 等11種)，多重代入文と LET の省略，複数行にわたる定義関数
	2	複数文/行，組込み関数 (MAX, MIN)，関係演算子 (AND, OR, NOT)，IF-THEN-ELSE 文の入れ子，FOR-NEXT 文での WHILE/UNTIL，コロンによる上下限指定，RANDOMIZE 文の引数による初期値制御
配 列	1	配列文 (MAT A=B, MAT INPUT, MAT READ, MAT PRINT)，配列関数 (TRN, INV, DET, DOT)，配列定数 (CON, IDN, ZER)，配列境界の再定義，UDIM, LDIM
	2	配列要素ごとの乗除算，べき乗，配列式の一般化
組込み関数	1	ACOS, ASIN, ATN2, COSH, COT, CSC, DEG, LOG10, LOG2, PI, RAD, SEC, SINH, TANH
型 宣 言	1	実数型，整数型，10進数型 (スケール指定，精度)，複素数型の宣言，型変換関数 (REAL, CMPLX, CONJ, IMAG, DEC)
	2	精度の指定
ファイル	1	OPEN 文，CLOSE 文，編成 (SEQUENTIAL)，ファイル形式 (DISPLAY)，処理様式 (INPUT, OUTPUT, APPEND, UPDATE)，レコード形式 (VARIABLE)，LINPUT 文，PRINT USING 文，INPUT USING 文，RESTORE 文の拡張，SCRATCH 文，MARGIN 文，組込み関数 FILE，例外条件 AT-THEN-
	2	編成 (RELATIVE)，データ表現形式 (INTERNAL)，OPEN/CLOSE 文の拡張，INPUT/PRINT 文の拡張，POSITION 文，DIRECT ファイル，ファイル関数 REC
	3	編成 (KEYED)，レコードの削除挿入，CREATE/DELETE 文，RESTORE 行番号
文 字 列	1	文字列式，文字列配列，文字列配列文 (入力，行入力，読込み，印刷，代入，初期化)，連結演算子，文字列関数と組込み関数 (STR$, CHR$, LEN, ORD, POS, VAL)，OPTION 文のオプション (COLLATE (NATIVE/STANDARD))，文字列関係式，文字列長限定子，文字列の中の引用符
	2	文字列組込み関数 (SREP, LPAD, RPAD, LTRM, RTRM, RPT, UPRC, LWRC)
チェイニング	1	CHAIN 文
	2	パラメタ付き CHAIN 文
副プログラム	1	主プログラムと副プログラム，SUB 文，SUBEXIT 文，SUBEND 文，CALL 文
	2	LIBRARY 文
書 式 制 御	1	PRINT USING 文，IMAGE 文
	2	FORM 文，制御指定 (X, SKIP)，データ表現 (C, PIC)
グラフィック	1	PLOT 文，SET 文 (BOUNDS, VIEWPORT, WINDOW, POSITION, CLIP, BRUSH)，INQUIRE 文，CLEAR 文，GPRINT 文，GINPUT 文，PICTURE 文，END PICTURE 文
	2	多角形グラフィック出力，3次元の2次元への投影
例 外 処 理	1	CAUSE 文，ERROR 文，RETRY 文，RESUME 文，例外処理の設定，組込み関数 (EXCODE$, EXLINE, EXPAR$, EXTIME$)，CANCEL 文
デバッギング	1	DEBUG 文，BREAK 文，TRACE 文，MONITOR 文，GO ON 文
リアルタイム	1	並列処理用の文 (PARACT, PAREND, PARSTOP, PROCESS) プロセス入出力文 (IN, OUT, CONNECT, DISCONNECT) スケジュール文 (START, WAIT, SIGNAL) ビット表現と操作 (2進，8進，16進の文字列)
	2	TASK 文，MESSAGE 文，PROCESS DIM 文，USE 文 CONNECT/DISCONNECT 文の拡張，CONTROL 文，MAT IN 文，MAT OUT 文 CANCEL 文，START/SIGNAL 文の拡張
編 集	1	編集用コマンド (DELETE, LIST, RENUMBER)
	2	編集用コマンド (EXTRACT, MERGE, CHANGE, SEARCH, RELOCATE)，DELETE/LIST の拡張
コ モ ン	1	COMMON 文

表-5 X3J2 小委員会

小委員会名	委員数
Enhancements	15
Nucleus	5
Strings	3
Files	13
Formatting	11
MAT/h	3
IPL	6
Real-Time	17
Exception Handling	19
Editing	6
Data Types	12
Graphics	9
Debugging	4

6. 標準化をすすめる上での問題点

ここでは BASIC の標準化にまつわるいくつかの問題点について，私見を混じえながら考えてみたい．

（1） 仕様決定機関の規模

BASIC の仕様決定機関の中心は ANSI X3J2委員会であり，その下に表-5 に示す通り 13 の小委員会が存在し，それぞれの責任で各機能単位の仕様を検討してきた．前章の拡張 BASIC のめざしている仕様をみて，余りにも仕様が多岐にわたっているのに驚き「BASIC よどこへゆく？」という感をいだいた方もあろう．仕様がこのように無制限に拡がってしまった背景には，このような委員会構成でそれぞれの機能単位ごとに分担して仕様を決めてきた体制にも原因があるのではないかと思われる．すなわち，各小委員会のメンバはそれぞれの活動成果を競う形になりやすく，おのずとあれもこれもと仕様を大きくしてしまいがちである．多数の人の意見を取り入れて民主的に仕様を決定してゆくことがはたして良い結果を生むものなのかきわめて疑問である．プログラム言語の仕様決定にあたっては，小人数で独断専行的に決めてしまった方が良い結果を生むのではあるまいか．

（2） 各種の BASIC 仕様

BASIC の仕様に言及する場合，ビジネス BASIC と呼ばれる事務処理を目的としたものやマイクロコンピュータの BASIC にもふれておかなければならないが，紙数の関係で後者についてのみふれる．

マイクロコンピュータの分野では，“BASIC の第二期黄金時代” と呼べるほど BASIC の花ざかりである．タイムシェアリングシステムの誕生と同時に脚光をあびた BASIC は，マイクロコンピュータの出現によって再び生まれ変り，Tiny BASIC と呼ばれる 2 kB 程度のきわめて小さなものから大型機並みの豊富な仕様をもったものまで様々のものが使われるようになってきた．このようなマイクロコンピュータの BASIC の仕様の特長は，従来のダートマス大学や GE 社の流れをくむ “正統派” の BASIC にはない仕様を豊富に取り入れた独自のものである点である．特定の個人，あるいは特定のソフトウェア会社により開発されたこれらの BASIC 処理系は，その名前を冠して呼ばれ高い信頼を勝ち得ている．その結果，マイクロコンピュータのメーカはその会社の BASIC を搭載するために競って開発を依頼するようになる．一方そのソフトウェア会社では各ハードウェアの特徴を生かすためおよび他メーカのために作った BASIC とは違った特色を出すために新しい仕様を導入することになる．その結果，同じソフトウェア会社の開発した BASIC でありながら互換性のない処理系が乱立する事態が，再びマイクロコンピュータの分野で繰り広げられているのである．

前章まで紹介してきた BASIC とは全く異なる BASIC が疑似標準 (de facto standard) ともみなされてまかり通っており，マイクロコンピュータを通じて初めて計算機というものを学んだ人々にとっては，標準 BASIC とはこういうものと誤って理解される可能性もある．

従来，プログラム言語の標準化の中心的役割を演じてきたのは大学，研究所，大中型機メーカおよびその利用者であったが今後はマイクロコンピュータの BASIC の影響も考慮に入れて半導体メーカや上記のようなソフトウェア会社の意見も取り入れて BASIC の仕様を決定してゆく努力が必要となろう．表-5 に示した小委員会のメンバは出席率が悪いと資格を失う規則になっており幽霊会員の存在は許されない．そのかわりに関心をもつ人なら誰でも参加できることになっている．その結果最近ではソフトウェア会社，半導体メーカ，マイクロコンピュータメーカからの参加もみられ拡張 BASIC の仕様決定も明らかにその影響と思われるものが見られるようになってきた．

（3） 処理系の作りやすさ

BASIC の文法は簡潔なのでソフトウェアのちょっとした知識があれば一般利用者でも容易に処理系を改造したり独自に開発したりすることができる．BASIC の仕様が乱れ，方言が乱立したのは長く標準化されなかったからだけではなく BASIC がこのようないわば “草の根” 言語としての性格をもっていたからであろ

う．言語仕様を守るという観点から見ると処理系が作りやすいということは，はたして良いことなのか疑問になってくる．PL/I のようにどでかい仕様にして素人による改造や開発を不可能にするか Ada のようにサブセットの存在を許さず Ada と呼ぶ資格があるか判定する機関を設けるなどのゆき方も注目に値する．Pascal の処理系も作りやすいといわれている．事実色々なところでサブセット版が開発されているが BASIC の轍を踏まないよう何らかの対策が必要であろう．

7. おわりに

本稿では基本 BASIC と拡張 BASIC の仕様について紹介した．原稿執筆の依頼を受けてから脱稿するまでにかなりの日時を要してしまったが，この間に言語仕様決定作業は遅々として進まず，一方処理系は雨後の筍のように次々と新しいものが開発され新しい仕様が導入されているのが現状である．BASIC という言語は，方言をもつ単一の言語と考えるよりも共通の核をもった言語の集りであるというカーツ博士の指摘は的を射ているように思われる．最近開発された BASIC 処理系の中には，「ANS 標準 BASIC 準拠」ということをうたい文句にしたものも散見されるが筆者の知る限りでは，本稿で紹介した「適合性」の観点から見て真に準拠していると言えるものはまだ存在していないように思われる．

なお，日本における標準化活動としては工技院の委託を受けて情報処理学会の BASIC 言語専門委員会（西村恕彦委員長）で JIS 規格の原案作成作業が進められている．成果に期待したい．

最後に本稿の執筆に当って日本電子工業振興協会の言語標準化専門委員会の委員の方々，特に調重俊氏の協力を得たことを記して感謝の意を表したい．

参考文献

1) Kurtz, Thomas E.: ACM Sigplan History of Programming Language Conference, BASIC, ACM Sigplan Notices, Vol. 13, No. 8, pp. 103〜118 (Aug. 1978).
2) Summary of Voting on DP 6373, Minimal BASIC, Attachment 5 to ISO/TC 97/SC 5 N 443.
3) First ISO Draft Standard Minimal BASIC, ISO/TC 97/SC 5 N 391.
4) Draft Proposal ISO/DP 6373 Minimal BASIC, ISO/TC 97/5 N 456.
5) DIS Minimal BASIC, ISO/TC 97/SC 5 N 497.
6) Draft Proposed American National Standard for BASIC, X3 J 2/80-64 Sept. (15 1980).
7) 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.
8) 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).
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 55 年 12 月 17 日受付)