三重大学教育学部研究紀要　第54巻　自然科学（2003）　1－8頁

VBAにおけるCellsの役割に関する歴史的考察

正田　良*

A Historical Note on the Role of the Array <<Cells>> of VBA

Rio SHOWDER

要　旨

VBAとは、Microsoft社の統合製品であるOfficeに含まれる諸ソフトウエアを利用者が統御するためのプログラミング言語で、初心者用プログラム言語Basicの流れをくむものである。

この20年余りの間にパーソナル・コンピュータに関する状況は大きな変化がが見られた。それを歴史的に略述し、Cellsというプロパティーが言語の基礎部分の一部として位置付けられることを指摘し、その利用例を紹介する。

キーワード：VBA, Cells, パソコンの動作スキーマ, 言語の基礎部分

1. Basicに関する歴史的記述

1.1　パソコンの初期でのBasic

Basic（Beginner's All-purpose Symbolic Instruction Code）は、コンピュータに対する命令をデータとして読み込み記憶して、それを機械語に翻訳して実行させるための言語として開発されたものである。当時、そのような類似の言語としてFortranが既にあったが、Basicは次に示すような特色があった。第1に、大型コンピュータをタイム・シェアリング・システムのもとで端末から利用するための特性として、各行に行番号を有し、ラインエディタでの編集が便利であるようになっていた。第2に、データの入出力に関してそのデータ記述の形式（フォーマット）を指定しなくとも、規定様式（デフォルト）で入出力を行なえるようにしたこと。第3に文字列に関するデータの宣言や操作を容易に行なえるようにしたことの3点である。

1977年にコモドール社のPETなどのすべてのアルファベットをキーボードから打てる形のパソコンのROMにBasicのインタプリタが焼き付けられ、1979年には国産のPC-8001（NEC社製）のROMにBasicを扱える機能が書き込まれている。当時のパソコンが置かれた環境としては、アプリケーションは充分にあるとはいえなかった。そこでアプリケーションを使うというよりも、データを入力して、処理をし、その結果を、数値もしくは、文字列をディスプレイなどに表示するBasicによって記述されたプログラムを実行すること。利用者がプログラムを作ることにおもきがおかれた。

1.2　アプリケーションの発展とOS

2年たつとパソコンの性能は2倍になると言われているが、実際、処理速度、記憶容量などの増大は、パソコンの応用範囲を広げるのに役立った。補助記憶装置についても、フロッピーディスクが、付けられるようになり、そして、ハードディスクが付けられるようになると、漢字変換に必要な情報が得られるようになる。また、プリンタは、初めのうちは英数字と若干の記号が打てる程度であったが、漢字が打てるようになると、日本語ワードプロセシング・ソフトが開発されるように

* 三重大学教育学部数学教室

なる。また、それと並行して表計算ソフトが、プログラムレス・ソフトとして、開発利用されるが、これらのアプリケーションソフトの普及は、一般の利用者にとってのプログラミング言語の必要性を薄れさせていった。

ディスクの制御も当初は、Basic によっていたが、ディスクを制御するためのシステム MS-DOS が、採用されるようになる。さらには、初期の版には、MS-DOS の付録として QBasic という Basic のインタプリタが付いていたが、それもなくなる。MS-DOS のもとで動く Basic の処理系（コンパイラ、インタプリタ）が発売されるようになった。それで、Basic は多くあるプログラム言語のうちのひとつに過ぎなくなった。例えば、Logo を標準に付ける FM-Town（富士通製）もあったほどである。

プログラミング言語としても、機械の速度が充分ではないころ、速度の必要なゲームなどは、機械語で開発されていた。また、コンパイラの使用を前提とするようになっても、統語構造が優れた Pascal や、ライブラリが豊富で、MS-DOS を記述するという実績のある C 言語、及びその後継言語を用いて多くの実用的な商用アプリケーションの開発が行なわれている。

1.3　GUI とインタネット

1990年代になると複数のパソコンを電話回線などを用いてサーバを介して接続するパソコン通信やインタネットが急速に普及し、パソコンは電子計算機というよりも通信機器として利用される機会が多くなった。そこで、電子メール、インタネットブラウザ、ワードプロセシングソフト、表計算ソフトなどのアプリケーションの利用が増えて、一般の利用者がプログラミングをするといった割合はかなり減った。

一方、パソコンの性能が向上すると、GUI（Graphic User Interface）での入出力が可能となった。OS も MacOS や Windows が一般的となり、それを用いた仕様がアプリケーションにも求められ、それへの対応するコードの仕様は、複雑なものとなっていった。この理由でさらに一般ユーザがプログラミングをする機会が遠のいたことになる。

1.4　アプリケーションの統合

一方、Windows は、GUI としての仕様を共通にし、それに要するプログラムの素材を利用可能にしたため、それぞれのアプリケーションが共通の仕様を共有することになった。

また、一方、プログラム・レス・ソフトとして、マルチプランなどが売り出されたが、定形業務で毎日同じ操作をする煩雑さを解消するためなどの理由で、「マクロ」という機能が考えられた。これは一連のキーボードなどの操作を記憶。登録して、より簡単な操作でそれを実行させるものである。

MS-Windows の開発・販売会社である Microsoft 社は、Windows での統合アプリケーション MS-Office でのマクロを記述する言語を、Visual Basic for Applications（VBA）と定めた。ワープロ・ソフトの中に表計算ソフトのオブジェクトを挿入することなどのでき、それを編集することもできるようになったが、これらの操作を Basic の流れをくむ言語で記述することが可能になった。ただし、オブジェクトを作ったりする記述は、C 言語の流れをくむもの（Visual C など）で行ない、VBA にはオブジェクトを開発する機能は持たされてはいない。

2.　パソコンのスキーマの変遷

2.1　ハードウエアに関して

コンピュータの動作を説明する図式（スキーマ）として、図1が良く用いられている。

CPU が、入力装置からデータを取得し、必要に

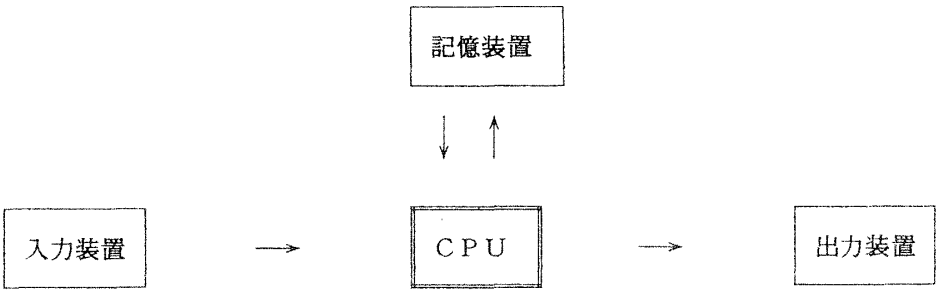

図1　コンピュータの動作を説明する図式

応じて記憶装置に記録保存し、これらのデータを加工し、出力装置を用いて表示するといった動きが記述されている。

ここで、想定されている装置は、

1) 入力装置：キーボード、マウス

2) 出力装置：ディスプレイ、プリンタ

3) 記憶装置：コンピュータの内部のメモリ

であった。

Basic では、入力装置、特にキーボードからの入力を行なうコマンドとして、input。出力装置、特にディスプレイへの出力させるコマンド print、プリンタへの出力のためのコマンド lprint が用意されている。また、前節に略述したコンピュータの発展によって、記憶装置として、ハードディスクなどの補助記憶装置が想定され、これに対応するコマンドも付け加えられている。

しかし、このスキーマはパソコンのソフト・ハードの発達によって書き換えられていく。補助記憶装置の容量がかなり大きなものとなった。また、パソコンのメモリも大きなものとなり、動作をさせるアプリケーションや、データファイルがかなり大きな物であってもそのまま半導体のメモリへ読み込ませることができる。ハードディスクとのアクセスに要する時間が半導体メモリのそれに比べて長いので、メモリの容量が充分である場合には、初めの読み込みと最後の保存だけしか、ハードディスクへの入出力を行なわない。

また、アプリケーションの GUI に伴う変化も特筆するべきである。ワード・プロセシング・ソフトに見られるような、WYGIWYS（What you see is what you get）の状態は、紙に出された結果や、ディスプレイに表示された結果などと、アプリケーションの内部でのデータの様子とが利用者の意識の上では、同じものとみなされるようになった。つまり、利用者のデータを加工する際の努力は、ほとんどディスプレイ上に表示されるものを最善のものとすることに払われ、紙に打出すとか、ディスクに保存するといった作業は動作時間はかかるものの、知的な負担とはみなされなくなった。

2.2 時間的変化による計算と空間的変化

表計算ソフトは、それぞれのセル（「セル」を以降「欄」と略記する）と呼ばれる表の中の場所にメモリを割り当てて、集計表の比喩を用いてメモリを利用するためのソフトである。この実現にあたっては、充分なメモリの量がパソコンに存在し、処理する速度も必要程度に得られる必要がある。

その潤沢な資源を用いての計算は、そうではなかった時の計算に比べて、計算に対する発想が大きく異なるものとなった。資源が少ないときの計算は、その変化のたびごとにメモリのデータを書き換えた。しかし、表計算の場合は計算のためのメモリ資源を潤沢に使えるので、古いデータを消してそこに新しいデータを書き込むという「書き換え」を要しない。初期値を上の行に書き入れ、計算のたびごとに 1 行ずつ下にデータを書き込むように、途中経過も含めて表の中に記せばよい。

上に述べた計算に対する発想の違いは、表計算ソフトの当初のキャッチフレーズである「プログラムレス・ソフト」としての特徴ともなった。以前の計算は、計算手順を時間系列を追って記述する手続き型言語の特色を持つプログラムによって記述したが、計算の進行を時間ではなくデータの空間での位置をパラメータとする表計算ソフトでの計算は、それぞれの欄の値を、他の欄の値との関係で記述すれば、条件を満たすように適宜再計算が為されて行くので、計算の手順を問題としない。つまり、静的な関係を必要なすべての欄に関して記述することが、動作を指示することに代わって計算の指示となったのである。

プログラム言語では、入力の指示、出力の指示の他に、計算の指示が必要である。計算の指示には、関数や演算子の仕様、そして、代入文の他に、「制御文」と呼ばれる構文が必要となる。条件分岐の指示と繰り返しの指示がそれである。しかし、表計算ソフトでは、各行に計算の 1 回ずつを表せば、行の作業内容のコピーをすることによって実現する。そして、そのコピーは「オートフィル」などのマウスの操作によって比較的容易に行なえる。これで、繰り返しの指示の代わりとすることができる。では条件分岐についてはどうだろうか。図 2 に見られるような二分法による 5 の平方根の計算でさえ、手順を指示するプログラムを必要とはしない。その条件分岐を、条件によって、値が変わるワークシート関数

if（条件，条件が真のときの値，偽のときの値）

によって実現できるからである。例えば、図 2 での、欄 B4 は、「数式」と呼ばれる欄の値がとるべき指示（以降、「欄の運命」と略記する）を、

=if(D3>5, B3, C3)

B4　　fx　=IF(D3>5,B3,C3)

Book1

	A	B	C	D	E
1	回数	下限	平均	その自乗	上限
2					
3	0	1.0000	3.0000	9.0000	5.0000
4	1	1.0000	2.0000	4.0000	3.0000
5	2	2.0000	2.5000	6.2500	3.0000
6	3	2.0000	2.2500	5.0625	2.5000
7	4	2.0000	2.1250	4.5156	2.2500
8	5	2.1250	2.1875	4.7852	2.2500
9	6	2.1875	2.2188	4.9229	2.2500
10	7	2.2188	2.2344	4.9924	2.2500
11	8	2.2344	2.2422	5.0274	2.2500
12	9	2.2344	2.2383	5.0099	2.2422
13	10	2.2344	2.2363	5.0012	2.2383
14	11	2.2344	2.2354	4.9968	2.2363
15	12	2.2354	2.2358	4.9990	2.2363
16	13	2.2358	2.2361	5.0001	2.2363
17	14	2.2358	2.2360	4.9995	2.2361
18	15	2.2360	2.2360	4.9998	2.2361
19	16	2.2360	2.2361	4.9999	2.2361
20	17	2.2361	2.2361	5.0000	2.2361
21					

図2　表計算ソフト MS － Excel を使った二分法による5の平方根の計算

として、前回の近似値の自乗が5よりも大きいか小さいかで、調べている範囲（図2では、下限と上限の間）の上半分か、下半分かの条件分岐をしている。

なお、図2では、A列は、A3を0、A4を1としたあと、「オートフィル」によって増分が1の等差数列となるようにし、B列は、B3に下限の初期値1を入れたあと、B4を定義し、これを下へオートフィルを行なった。C3の運命は、

＝0.5＊(B3＋E3)

D3の運命は、

＝C3＊C3

で、それぞれ、下へオートフィルを行なっている。E3は、上限の初期値5を入れ、E4の運命として

＝if(D3＞5, C3, E3)

を入れたあと、これを下へオートフィルを行なった。

2.3　空間的変化計算環境での手続き型言語

以上に述べたように、時間的に変化をさせずとも、空間的な変化によって計算の手順を指示し、それを実行させれば、作業の途中経過も含めて記録されることになるので、計算の妥当性をみてとれるという長所がある。

しかし、既にみたように、「マクロ」などという形で、手続き型言語はなお必要となっている。その理由としては、

(1)　定形業務では、何回も同様な手順を踏むので、それに名称をつけて、記憶させ、自動的に実行させた方が能率が良い。

(2)　操作の誤りによるデータの消失などの事故発生を防ぐ。

(3)　やや複雑となったり、何回も同様なことが多くの欄に関して行なわれる欄の運命の記述を簡明にするために、「利用者定義関数」を用いて関数を定義する。

(4)　ゲームやプレゼンテーション、シミュレーションなどの場面で、時間を追っての変化が必要となる。の4つが考えられる。

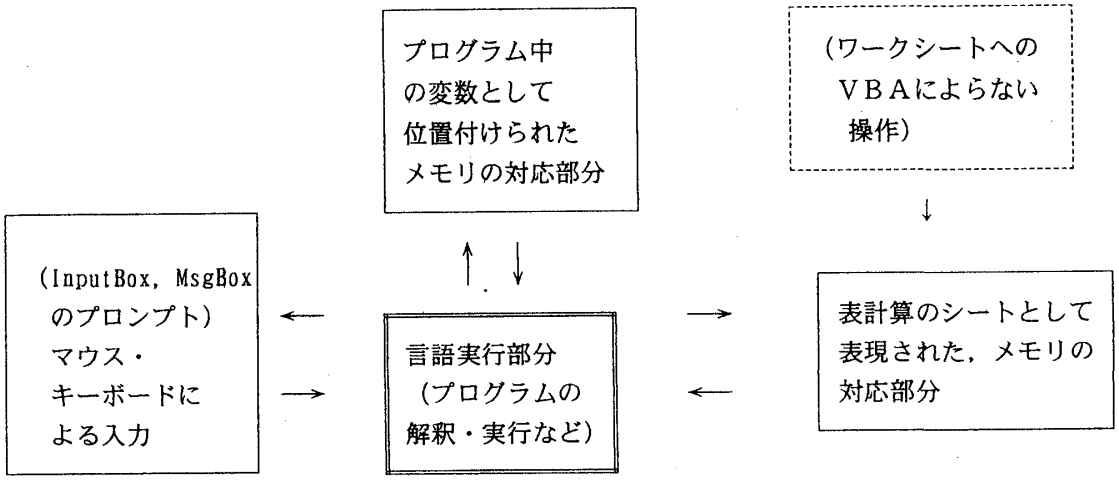

図3　表計算ソフトでの手続きのスキーマ

なお、利用者定義関数は、VBA が持つ機能であるが、この機能を使って定義された関数は、VBA の関数として他の利用者定義関数や、VBA での手順の記述に利用できるのみならず、表計算での欄の運命の記述に用いるワークシート関数としても利用することができる。

2.4　表計算ソフトの手続きのスキーマ

図1に時間的変化によって計算を行なう場合の計算に関する動作のスキーマをしめしたが、図3に表計算ソフト、例えば MS-Excel での VBA のスキーマを示す。

利用者にとって、中央下の制御する部分から上に伸びる両方向の矢線は、計算の便宜上データをコンピュータには利用しやすいが、利用者には状態を読み取りにくい形で記憶する部分へ向っている。ここでのハードウエアのメモリは、この部分と、右の部分とに分割されて示されている。右の部分は表計算の各欄に対応するメモリであり、ディスプレイ上に示されるその表現と同一視されている。

このようにプリンタによって紙に打ち出されたものであったところに、ディスプレイ上に実現されたバーチャルな紙が位置付けられる。そして、その「紙」がバーチャルなものであるがために、比較的容易に書かれた情報を制御部分が直接読み取って、計算に利用することができるので、中央と右との矢線が双方向となっている。また、短い文字列などの場合、MsgBox や InputBox のプロンプトの形で出力することもできるので、図の左の入力装置からの矢線の図3では、これも双方向のものとしている。

3.　言語の基礎部分とは何か

3.1　Basic English

言語の解説本の記述方法として、なるべく全部のコマンドを記述しようとして、コマンドを分類し、その分類に従って網羅的な記述を行なおうとするものがある。これを「辞書的記述」と呼ぶことにしよう。

第2には、特定の目的に限って、その作り方を記述することを軸に、関連する知識を補って行く方法がある。これを「料理本的記述」と呼ぶことにしよう。言語の総体が多様化肥大化するにつれて、辞書的な記述が普通の本のサイズでは難しくなり、また、利用者がその必要とする部分を探しにくくなったので、この記述の方法が要求されるようになったものである。この場合は読者が自分の目的に適合する本を探す必要が出てくるが、一方、辞書的記述では、読破することはできないので、このような料理本的な記述のものを補うコマンドの辞書として補助的な利用をする程度に限られることになろう。

第3の記述の方法として、その言語のあらわそうとする総体を記述することを目標におきながら少ない語で記述しようとする方法である。これはちょうど、C. K. Ogden が1930年に、事実上の国際補助語となっている英語を合理的に整理して、より一層世界語としての機能を発揮させようと開発した英語のサブセット “Basic English” [1] のように、一応それだけでも、言語が想定する事象のスキーマでの動作を記述できる、その言語のサブ

セットを想定することになる。この方法を「カタコト的記述」と呼ぶことにしよう。

Ogden は、主語となっているものの動作・状態を抽象し、そのスキーマ[2] のそれぞれに対応した下記の 16語のみを動詞として用いることにした。

come, get, give, go,
keep, let, make, put,
seem, take, be, do,
have, say, see, send

がそれである。これも、上記の分類で英語の「カタコト的記述」と言える。

但し、言語を解説書などの形で記述説明するときの大まかな分類であるから、例えば、Logo でタートルというオブジェクトを、θ 度左に向きを変えるコマンド left θ は、right -θ と書けるなどの冗長性は許容して「カタコト的記述」と分類することにする。

3.2 「カタコト的記述」の差異

図 1 のスキーマの「カタコト的記述」の例としては、正田[3] を挙げることができる。画面のピクセル単位の制御などは、その対象外として、文字列もしくは数の処理に限っている。その大要を下に記す。

(1) 入出力　input, print, lprint
(2) 代入処理
(3) 制御文
　if (条件) then 行番号 else 行番号
　for next, goto, gosub.
(4) 論理関数・演算子
　and, or, not
(5) 数値関数
　+, −, *, /, ^, sqr(), rnd(), exp(),
　log(), sin(), cos()
(6) 文字列関数
　len(), val(), format(,), left$(,),
　right$(,), mid$(,,)
(7) 型宣言
　DIM, integer, string

これに比べて、図 3 のスキーマによる「カタコト的記述」は、次のような差異をもつことになる。

[1] (1)の入出力は、Cells (,) への代入、もしくは、これからの出力による。つまり左辺に、

```
Cells(m, n) = x
```

とかくと、変数 x の値を、m 行目の n 番目の欄へ書き込む出力命令となり、逆に、右辺に、

```
x = Cells(m, n)
```

とかくと、変数 x の値に、当該の欄の値を書き込むことになる。

利用者の注意を特に引きたいエラーメッセージや入力の促進などに関して、InputBox()、MsgBox() による。

[2] 制御文は、構造的プログラミングに即して改善が為されている。

```
if (条件) then  else endif
for (exit for) next,
do while (exit do) loopgoto,
call
```

利用者定義関数やサブルーチン（副譜）の定義についても記法上の改善の他に再帰的定義も可能となった。

[3] 型宣言はおよそ不要になった。その第 1 には、変数はバリアント型として柔軟な変数の利用ができる。例えば、

```
x = InputBox("いくらもっていますか")
Dummy = MsgBox("じゃあ、" & x / 2 &"円下さい")
```

と、x は、x/2 として数値型としての計算の対象となりながら、文字列型として、“じゃあ、” などと結合されている。

第 2 に、手続きや利用者定義関数では、その外には引数以外の変数は原則として隠蔽される。他で利用したい場合は、シート上のデータとして書き出しておいて、それを読み込んで利用すればよい。

第 3 に、配列が必要となっても、ワークシート自体が、2 次元の配列であるので、適宜書き出して読み込みを行なえばよい。

[4] 入出力のためのオブジェクトは、いくつかのワークシートが対象となるので、後の3.4に述べるような指定をすることが必要となる。

3.3 図 3 のスキーマでの譜例

3.2で述べた、図 3 のスキーマに対応した譜例を、図 4 へ記す。指定された数よりも小さい素数を B 列に並べる算譜である。

15、16行などでは、Cells を出力に使っているが、11、21、30行では欄の値を読み込んで以降の計算に利用している。

```
 1 :Sub PrimeNumbers()
 2 :Cells(1, 2) = 2
 3 :Cells(2, 2) = 3
 4 :Cells(1, 1) = 2
 5 :If Cells(3, 1) > 0 Then
 6 :    J = Cells(3, 1)
 7 :  Else
 8 :    J = InputBox("どこまで求めればいいの？")
 9 :    Cells(3, 1) = J
10 :End If
11 :  n = Cells(1, 1)
12 :  For i = 5 To J Step 2
13 :    If IsSosuu(i) = 1 Then
14 :          n = n + 1
15 :      Cells(n, 2) = i
16 :      Cells(1, 1) = n
17 :    End If
18 :  Next i
19 :End Sub
20 :Function IsSosuu(i)
21 :  IsSosuu = 1
22 :  k = 1: P = 2
23 :  Do While i >= P * P
24 :    If Int(i / P) = i / P Then
25 :        IsSosuu = 0
26 :        Exit Do
27 :    End If
28 :    k = k + 1
29 :    P = Cells(k, 2)
30 :  Loop
31 :End Function
```

図4　新しい動作スキーマに対応した譜例

3.4　オブジェクトの指定に関する注意

ワークシート関数で、他のシートの値を参照する際は、ちょうどその欄をクリックして表示されるように、

＝(シート名)！(欄の名称)

でとなるが、マクロでは、prog!B3 を参照したり、そこへ出力したりする場合、

Range("prog!B3") とか、
Sheets("prog").Cells(3,2) とかと、

書く必要がある。

3.5　いくつかの解説書

VBA に関する解説書では、当然 VBA の大要がその中に示されるが、膨大なプロパティを「新しいマクロの記録」を利用することで探したりして、記述が膨大となることを避けるなどの工夫がみられる。また、上述のスキーマに関しても工夫されているものもある。

(1)　村田吉徳、1997

記録機能によるマクロを第2部までに扱い、第3部で条件分岐などを使ったプログラミングを扱っている。シートの書式をかえたりダイアログボックスを作ったりの扱いは見られるが、Cells の利用はみられないが、オブジェクトのプロパティの記述として range を使った代入参照が扱われている（p. 38)。シートの書式を変えたり、ダイログボックスを用いたりが操作の対象となっている。

(2)　森口繁一、2000

マクロによって、Basic を使うことに重きが置かれており、「Excel のワークシートを使うことによって、データの入出力が比較的容易になる」(p. 33) と、cells の利用がみられる。

また、「新しいマクロの記録」を使って、ワークシートを操作して直線や四角などをかくコマンドを取り出して、それを使いやすい名前や形式のサブルーチンとして利用し、図形も扱っている。

(3)　縄田和満、2000

8章までは、もっぱらワークシート関数などを扱い、マクロが9章から「新しいマクロの記録」を利用して導入される。Cells の利用はみられない。

(4)　土屋和人、2001

辞書的記述を目指された本であるが、「新しいマクロの記録」を使ってマクロを導入し、欄への代入や参照には、Cells ではなく、range を用いている。連続した欄などへの系統的な扱いではなく固有の欄を指示するには、この方法でも支障はないだろう。

4.　まとめと応用への示唆

VBA は、インタプリタやコンパイラを別途にインストールしなくとも、表計算ソフト MS-Excel がインストールされていれば使える言語である。

新規にインストールする必要がない点では、かつて、パソコンの ROM に焼付けられていた Basic と共通の特徴があり、言語仕様も共通な部分が見られる。

その機能は充実しているので、用意されているコマンドなどは膨大な量になる。ただ、旧来の Basic での紙への出力を、ワークシートへの出力に対応させれば、以前のものでも行なわれた「カタコト的利用」が可能である。そして、プログラム記述に関して旧来のものよりも、

・利用者定義関数の命名や、再帰的定義が可能になったこと。
・値のトレースは、ワークシートの空間的変化として行なえること。

について便宜がある。このような VBA のサブセットは、旧来の Basic に劣らず、初等的な数値文字列の処理に有用な言語として位置付けられる。

斎藤孝[5]は、いわゆる「生きる力」をより具体的、明確にしたものとして、

・まねる力　　・段取り力　　・コメント力

を挙げているが、宣言文などの負担が軽減されたプログラミングはまさに、この第2の「段取り力」が関わる分野である。また、既存のプログラムを「まねる力」や、既存のプログラム要約し読み取る能力もまねる前提となるだろう。現在では、英語の長文解釈や、数学のステップの多い解法などが、「難問奇問」として敬遠されている教育風土があるが、逆に、プログラミングが特に「段取り力」として注目される気運は充分あると言えよう。

また、1970年改訂の学習指導要領から高等学校での教材としてアルゴリズムなどが、選択科目の中に位置付けられており、大学入試の問題としてもいくつかの出題が見られる。学習指導要領では、どの言語でアルゴリズムを記述するかの指定はないが、大学側としては、用いている教科書によるハンディや受験生の心理的負担を避けることもあってか、多くの教科書が用いている旧来の Basic を出題に用いている。しかし、教科書に用いられている言語は、

・その仕様が古いスキーマによるもので、
・用いる組み込み関数などの範囲が不明確[6]

であるという欠点が指摘される。

図3に示したスキーマによる言語、即ち VBA のサブセットは、このような欠点を補い得るものとして応用が期待されるだろう。

文　　献

1)　室　勝／小高一夫、1982. 英語を書く本 Basic English の理論と応用. 洋販出版.
2)　前掲書. p. 2.
3)　正田　良、1987. カタコト Basic のすすめ — パソコン会話練習帳. 啓学出版.
4)・村田吉徳、1997. 実用例題でわかる Excel 97VBA マクロの使い方. 技術評論社
・森口繁一、2000. Excel/Basic 基礎指南　知らないことを知りたい人へ. 日本規格協会
・縄田和満、2000. Excel による統計入門 第2版. 朝倉書店
・土屋和人、2001. Excel VBA パーフェクトマスター. 秀和システム.
5)　斎藤　孝、2001. 子どもに伝えたい＜三つの力＞　生きる力を鍛える. 日本放送出版協会（NHK ブックス）. p. 4.
6)　2002年度全国算数・数学教育研究（兵庫）大会の分科会での発表である、奥村佳則「教育課程大学入試（コンピュータ関連）を振り返って」と、その発表資料として配られた、奥村「情報科による授業内容の研究」『平成13年度兵庫県私学研究論文集』pp. 17–52. に多くを負う.