アルゴリズム記法によるプログラミング教育

東洋大学　正会員　中村慶一

はじめに

何事によらず入門に当つて使用した道具の印象は強烈で、上達した後でも、その道具によつて身に付けた習慣はなかなか直らず、その人のものの考え方にまで影響を及ぼすことが多い。

プログラム言語で最も重要な概念は制御構造であつて、これらの大部分を欠く言語による入門は、危険な後遺症を残すおそれがある。そこで入門用として、アルゴリズム記述上通常必要とされる機能は完備しており、しかも簡略化可能なシステムを苦心して、アジア工科大学院や東洋大学でのプログラミング入門に使用してみて相当の効果が認められるので、その概要を報告する。

1° プログラム言語としての必要条件

N. ワースの本に「アルゴリズム＋データ構造＝プログラム」(文献1)という表題のものがある。彼の云う通り、プログラムとは、配列、木構造、リスト構造のデータに働きかけてコンピュータの了解可能なアルゴリズムを記述したものであつて、ここに云うアルゴリズムについては岩波国語辞典の算法の項に、「演算手続きを指示する規則特に同類の問題一般に対して有限回の基本的操作を指示の順を追つて実行すれば、解がある場合にはその解が、解がない場合にはそのことが確かめられるようにはつきりと仕組んである手順」と明快に定義してある。

従つてプログラミング教育は、これらのデータ構造に対し頭の中に画かれたアルゴリズムを、如何に簡単、厳密且つ後の修正に便利なように書き下すかという訓練に始まることになる。(文献2,3)

そこでプログラム言語として必要な条件を列記してみると次のようなものになる：

1. 制御構造の完備

制御構造はすべて1入口1出口で、次の4種と構造脱出機能があること。

a. 直結構造

文の括弧により文をつなぎ、まとめ、さらに宣言により名前の有効範囲を局所的にするブロック構造をつくる。

b. 選択構造

条件により一群を選択する。

c. 反復構造

制御変数に値を与えて指定条件を満す間ループを反復する。

d. モジュール構造

くり返し現れる部分を手続きまたは関数として宣言する。その実行は呼び出しによる。

e. 構造脱出

指定された構造の直後に脱出する。指定名のない時はプログラム終了。

2. データ構造の完備

3. 配列算

配列の全要素またはある断面について、対応要素間の演算を実行、一方がスカラの場合は他方の配列と同じ形に拡張する線形演算の一般化。

4. モジュールとの情報授受しが簡明なこと。

2° プログラム言語

プログラム言語には文が主体の命令型(手続型)、式が主体の作用型(関数型)、事実とその推論規則を主体とする関係型(論理型)の3種がある。現用計算機はいわゆるノイマン型の計算機であり、これに適しているのは命令型言語で、右辺の式の結果を左辺のアドレスへ格納する代入文、手続の呼び出し文、入出力文、プログラムの制御文とすべて文の形をとつている。

これに対して作用型言語では、値を評価する式が主体となり、関数の合成あるいは関数に働きかけて新らしい関数を作り出す作用を特長とし、リスト処理を主眼とした LISP、対象を配列とし組み込み関数の合成を認めたAPL、ユーザ定義関数の合成を許す本質的な関数型言語(FP)などがある。

論理型言語はPROLOGに始まるが、いわゆる第5世代計算機用言語として取り上げられて脚光を浴びるに至つた。その特長及び流れを図1,表1に示す。

		1962 FORTRAN	1962 ALGOL	1965 BASIC	1965 PL/I	1967 APL	1971 Pascal	1972 C	1979 Ada	1982 JIS FORTRAN
制御構造	連結	X	○	X	○	X	○	○	○	X
	選択	X	○	X	○	X	○	○	○	○
	反復	① 正整数	○	○	○	X	① スカラ型	○	○	○
	モジュール	○	○	X	○	○	○	○	○	○
	構造脱出	D STOP RETURN	X	X	D STOP RETURN	X	X	① break	① exit	X
配列	配列算	X	X	D (MAT)	○	○	X	X	○	X
	配列断面	X	X	X	○	○	X	X	X	X
	配列関数	X	X	X	D 組込関数	○	X	X	○	X
モジュール	address	○	○ name	X	○	○	○ var	○	○ out	○
	value	X	○ value	X	X	X	○	○	○ in	X
	配列サイズ	同じ型	自由	X	自由	自由	同じ型名	同じ型	同じ型名	同じ型
データ構造	配列	○	○	○	○	○	○	○	○	○
	木構造	X	X	X	○	X	○ record	○	○	X
	リスト	X	X	X	○	X	○	○	○ .	X

表 1　各種言語の比較

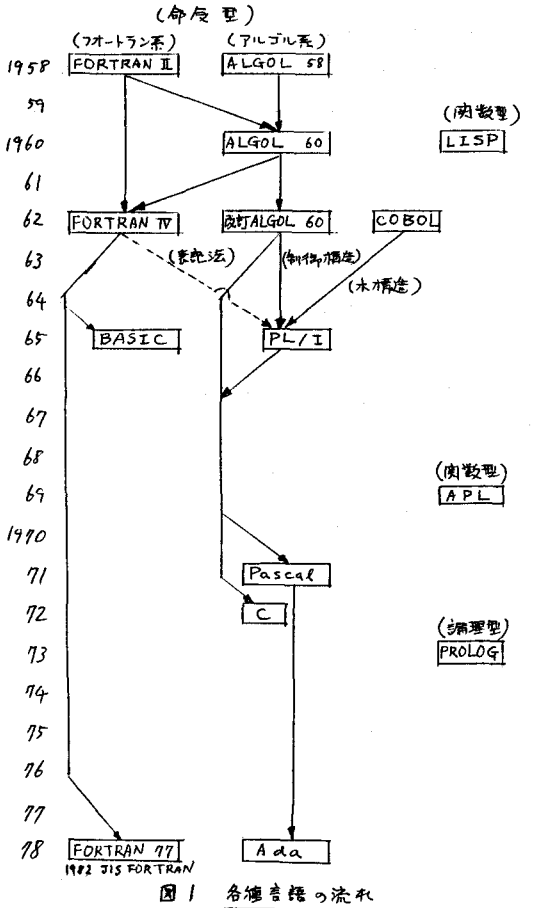


図 1　各種言語の流れ

3° 入門用プログラム言語の要件

入門用プログラム言語として望ましい機能は次のような点ものと考えられる。

a. 4構造とできれば構造脱出機能の完備

b. データ構造としては　配列とできれば木構造

c. 整数と実数の混合演算が可能なこと。常識的な演算結果であることが望ましく　整数割算（たとえば 5/2 の結果は 2 と与える）といった特殊な演算や 5.0 の代りに 5. を許すといった一言語としか通用しない機能は危険でさえある。

d. 暗黙の宣言

ただし整数型の名前に代入すると自動的に切りすてを行なわれるといった機能はむしろ有害で、整数型の宣言には注意を要する。むしろ制御変数は正整数に限るといった制限を外して積極的に切りすてを指示しない時は4捨5入としたほうがよい。

e. できれば配列算が可能なこと。

配列の断面も望ましい。

f. パラメタに配列名を用いた場合、同じ型あるいは同じ型名に限るという制限を外し、ダイナミックな配列の使用が望ましい。

g. 絶対、累乗関数

4° アルゴリズム記法 (文献 2, 10)

今までにのべたプログラム言語としての要件をすべて満すものはまだない。配列算や構造脱出を除けばPascal, 配列算以外ならC, 配列算を主体としたものにAPL, 大部分を満たすものにはPL/I, Adaがあるが、PL/I, APLを除いては厳密な宣言が必要で、またPL/I, APLといえども、やはりその表記から入門した場合の後遺症らしきものが多少は残る。そこで前にのべた望ましい機能を常識的な記号と結びつけた表記法を作り、これを入門用言語として自由な発想でアルゴリズムを表記することを試みた。

アルゴリズム記法の原則

a. 基本文としては 左辺←右辺 という形の代入文、read(A,B,…) という形の入力文、printl('ANS=', P*Q) という形の出力文(ただし最後のlはlineを構成することを示しlがなければバッファに入り出力待ちとなる)、多続文の4種である。

b. 文をくくるのに角括弧を長く伸して用いる。同行に文を並べる時はで区切る。

c. 式表現は誤解のない限り通常の数学記号を用いる。

d. 大小文字は区別しないが、read等のシステム内の名前と添字には小文字を用いる。

e. 変数宣言は array $A_{10\times20}$; int P,Q; alpha P1; 等とし無宣言の名前は real 変数とする。

f. ベクトル式を認め、行ぎりに

$A_{**}\leftarrow((1,3.5,4),(2,3.6,7))$;

などと使用する。

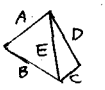
g. 配列算を認め、対応要素間の演算とする。また一方がスカラの時は同じ型の配列に拡張される。また配列の断面を添字*によって示し、Aの第1行は A_{1*} と表現する。

h. 総和(Σ)累積(Π)関数の引数中の添字はその全領域を示す。

i. 値の交換は P↔Q によって示す。

j. proc, func による多続、関数宣言を認める。(結果を配列として返す配列関数も認める。)

5° 四辺形の面積



問題 長さA,B,C,D,Eを読みこの4辺形の面積を、AREA= という字の右に1行に印刷せよ。ただし三角形の面積はHERON(Heroという説もある)の公式によってもとめ、一方または他方が三角形を形成しない時は、-10^{10} より小さい値を印字するものとする。また与の場合各データを 4, 5.5, 6, 3.2, 7.5 としてチェックせよ。

まず入力設計と出力設計を用意し、これに対応するアルゴリズム記法による表現を記述する。

```
I: A,B,C,D,E
O: AREA = <4辺形の面積>
[read(A,B,C,D,E)
 S ← (A+B+E)/2
 D ← S*(S-A)*(S-B)*(S-C)
 A1 ← if D<0 → -10^15
          √D
 S ← (E+C+D)/2
 D ← S*(S-E)*(S-C)*(S-D)
 A2 ← if D<0 → -10^15
          √D
 printl('AREA=', A1+A2)]
```

4, 5.5, 6, 3.2, 7.5

答は

AREA = 19.980232172

ということになろう。

ところで、この2~4行と5~7行は3つの値を除いては全く同じ形をしており、当然仮の名前X,Y,Zに対する関数HERON1に直すことが考えられる。

```
func HERON1(X,Y,Z)
  [S ← (X+Y+Z)/2
   D ← S*(S-X)*(S-Y)*(S-Z)
   HERON1 ← if D<0 → -10^15
                 √D]
read(A,B,C,D,E)
printl('AREA=', HERON1(A,B,E)+HERON1(E,C,D))
```

ここでΣ, Πを利用するために配列 A_{*1} に代入することを考えれば関数は次のように直してもよい。

```
func HERON2(X,Y,Z)
  array A 3×1
  A*←(X,Y,Z);   S←Σ(A*)/2
  D←S*TT(S−A*)
  HERON2 ← if D<0 → −10 15
           √D
```

PL/I と APL にはそのまま変換できる。

PL/I (文献6)では

```
HERON2: PROCEDURE(X,Y,Z)
        RETURNS(FLOAT);
  DECLARE A(3) INITIAL(X,Y,Z);
  S=SUM(A)/2;  D=S*PROD(S-A);
  IF D<0 THEN RETURN(-1E15);
    ELSE RETURN(SQRT(ABS(D)));
END;
```

APL (文献8)では Aに3辺を代入し

```
    ∇Z←HERON2 A
[1] S←(+/A)÷2
[2] D←S××/S−A
[3] →(D<0)/0,Z←−1E15
[4] Z←(|D)*0.5
    ∇
```

で APLでは ∇……∇ によって関数定義をし演算はすべて右優先で行なう。 +/A はAの各要素間に+を挿入することを示し ×/S−A はスカラSがAと同じ配列に拡張されて後対応要素間の引算を行い、出来たベクトルの各要素の間に×を挿入することを命じている。 また*は累中を、|D はDの絶対値をとることを知れば大体理解できよう。

6° ベクトルの正規化

もう一つの例として、ベクトルの各要素をそのノルム(全要素の2乗和の平方根)で割った正規化ベクトルを求める手続きを考えてみよう。 サイズN ベクトルAを知って正規化ベクトルRを求める手続 NORMZ は次のようなものである:

```
proc NORMZ(N, A n×1, R n×1)
  Z←√Σ(A*²)
  R*←A*/Z
```

これは JIS FORTRAN では次のようなものに当る:

```
      SUBROUTINE NORMZ(N,A,R)
      REAL A(*),R(*)
      S=0
      DO 10 I=1,N
      S=S+A(I)**2
   10 CONTINUE
      Z=SQRT(S)
      DO 20 I=1,N
      R(I)=A(I)/Z
   20 CONTINUE
      RETURN
      END
```

これを原理そのままに移せるのは PL/I と APL で、略字 PROC, DCL を使用すれば PL/Iでは

```
NORMZ: PROC(N,A,R);
  DCL (A,R)(*);
  Z=SQRT(SUM(A**2));
  R=A/Z;
END;
```

APLでは次のようになる:

```
    ∇Z←NORMZ A
[1] Z←A÷(+/A*2)*0.5
    ∇
```

おわりに

以上2つの例によってもアルゴリズム記法が入門用として適当であることが、ある程度御了解いただけたことと思う。 今後さらに完全な表記法が作り出されることを期待して本文を終ることとしたい。

参考文献

(1) Wirth, N: Algorithms+Data Structures =Programs, Prentice-Hall (1976) 片山訳 日本コンピュータ協会 (1979)
(2) 中村慶一: コンピュータによる統計解析, 数学ライブラリ 45 森北出版 (1977)
(3) 森口繁一・沼田一道: コンピュータ 共立(1984)
(4) FORTRAN研究会: 制御構造によるFORTRAN77 日本理工出版会 (1984)
(5) 岡本正行: Pascalプログラミング入門 工学図書 (1982)
(6) スコット, ソンダク, 中村訳: プログラマのためのPL/I 森北 (1974)
(7) 竹下亨: プログラミング言語 APL 共立(1982)
(8) 石田晴久: パソコン言語 アスキー出版 (1984)
(9) 有沢誠: プログラム言語への招待 岩波 (1984)
(10) 中村慶一: アルゴリズム記法によるプログラミング 土木技術資料 22巻 6号 (1980)