構造化プログラミング

正員　東洋大　情報工学科　中村慶一

はじめに

オランダのDijkstra によって口火を切られたGOTO論争[1]は GO TO文というプログラム構造を混乱に導くおそれのある要素は有害であるという議論に端を発し、現在では プログラムは"良い構造"の集積でなければならぬという点については広く認められるところとなっている。ここで 良い構造 とは 一つの入口と一つの出口を持つ次の3種の構造

1. 直列構造（複合文）　CON [F1 ; F2]
2. 並列構造（選択文）　IF[P ; F1 ; F2]
3. 反復構造　WHI [P ; F]

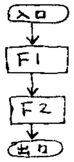
のことであって、すべてのプログラムは この3種の構造の集積として表現可能であることは Böhm 及び Jacopini が示した[2]。

しかしアルゴリズムをこの3種の構造記号あるいは流れ図で表現しただけでは、それをメモとして人間同志のコミュニケーションに使用するには読みやすさ及び保守性という良いプログラムとしての要件には欠けるところがあるので、本文では日常頻出するパタンを要領よくまとめ、さらに構造脱出の機能を付け加えたアルゴリズム記法[3)〜6)]によって"構造的に"構造的プログラムを記述する方法について触れる。

1° 良い構造

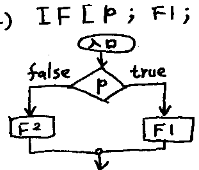
A. 基本構造

(1) CON [F1 ; F2]



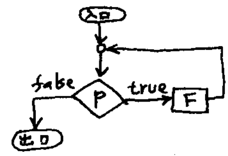
流れ図で示せば左のようになり作業F1 終了後に F2 を実行することならびにこれらをまとめて一つの構造とみなそうとするものである。

(2) IF [P ; F1 ; F2]



論理式Pが真(true)であればF1を、偽(false)であればF2を選択するものである。

(3) WHI [P ; F]



論理式Pが真である間は永久に作業Fを反復実行することを命じており、F中にはPの値を変更する作業が（または反復構造を脱出する）含まれている筈である。

B. 論理式

IF および WHI の第1要素Pは論理式であって、その定数は true , false (t, f と略記) の2種である。算術値または文字値 (その定数は 'A', 'P1=' 等文字記号を引用符でつつんだもの) を > ≧ = ≦ < ≠ でつないだ比較式や論理値の前に ¬(not) をつけたもの等を ∧(and) や ∨(or) でつないだものが論理式である。

C. アクション（作業）

CON , IF , WHI にあらわれた F, F1 等は算術値、論理値、文字値について次の作業を行うものである。

a. 代入

たとえば A ← B+C のように ← を用いて、その右辺の式の結果を登録用紙上の左辺の名前に対応する場所に記入することを命ずる。

b. 読込み

たとえば read (A,B,C) とか read (M,N, $A_{M \times N}$) のように read をもって示し、データ用紙上にカンマまたは空白を置いて並んでいる値を、逐次登録用紙上のかっこ内の名前に対応する場所に記入することを命ずる。ただし後の例は配列宣言が行なわれている場合に限り添字は 11, 12, … と行なりに変化するものとする。

c. 書出し

たとえば printl (A,B,C, 'P=', $\sqrt{P}$) とか printl ($\underset{N \times N}{PQ}$) のように printl によって示し、報告用紙上に対応する値を1行に（配列の場合は行ごとに）印刷実行することを命じている。

後の例で P= の前に5文字スペースを入れたければ
print(A,B,C); space(5); print('P=',P)
あるいは最後も print として印刷を保留し crlf(1)
をつけ加えても良い。

d. 宣言

プログラム全体を通じて使用する制御値は便宜出現のたびに登録するが、あるCONの範囲内のみに使用する名前や、全体に対するものであっても、配列、文字、論理変数等はCONの第1作業としてこれらの名前の宣言を行う。この場合そのCONをブロックと云い、そのブロックを出たときはそれに対応する登録用紙は破棄される。また内側のCONで外と同じ名前が宣言されたときは内側が優先するがそうでないときは外の意味は内側でも適用する。

この他に手続や関数の宣言も考えられる。何れもある名前が一群の構造の概集様を代表するものであることを宣言するもので、実行時に実パラメタが与えられる予定のもとに仮パラメタを用いて説明してあってもよい。手続や関数の宣言は、いわばこの仮パラメタを仮に宣言した準ブロックを作っているものと解される。

準ブロックはWHI構造についても考えられる。WHIでは脱出時の構造識別子として制御変数を一個設定するが、いわばこの制御変数を仮に宣言した準ブロックを作っているものと解される。(しかしこの制御変数は外で使われた名前を使用する)

e. 手続を呼び出し

すでに宣言されて登録用紙にのっているか、または将来共有用なものとして併書にのせられている手続きに実パラメタが与えられたときはそれを与えて実行を命ずる。(関数は式中で呼び出し使用する)

f. 空作業

何もしないで次に移る作業は WHI[false;F] によっても表現されるが null あるいは空白でこれを表現してもよい。

g. 構造脱出

exit 構造識別子 によってその構造を脱出(その構造の出口に到達)することを命ずる。

2° 複合構造

a. CON[F1;F2]

F1,F2 が他の構造だとそれがCONである場合を考えると CON[F1;F2;F3;F4] と何個あっても良い。

b. IF[P;F1;F2]

F1,F2 が他の構造だとそれがIFである場合を考え、
IF[P1;IF[P2;F1;F2];IF[P3;F3;F4]]
は CASE[P1∧P2;F1; P1∧¬P2;F2;
¬P1∧P2;F3; ¬P1∧¬P2;F4]
と考えてもよい。

c. WHI[P;F]

脱出の都合上制御変数を考えそれについて

c1. 無条件くり返し

CON[W1←A; WHI[true; F]]

Aは何であってもよく、Fの中に exit ^{W1} があり、と永久にくり返されることになる。

c2. 公差Bでのくり返し

初期値A きざみB 終期判定値C でのくり返し

CON[W1←A; WHI[W1≦C; CON[F;
W1←W1+B]]]

c3. 公比Cでのくり返し

CON[W1←A; WHI[W1≦C; CON[F; W1←W1*B]]]

c4. 任意反復 (A,B,C,D と変更)

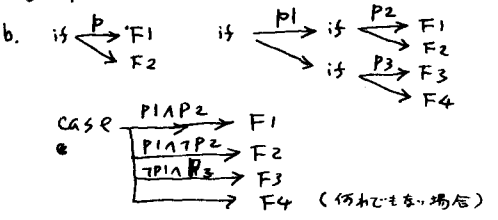
CON[$A_{E_{n+1}}$←(A,B,C,D); I←1; WHI[I≦4;
CON[E←A_{E_I}; I←I+1]]]

3° アルゴリズム記法

対応する機械構造を簡単で読みやすいアルゴリズム記法によるものに書き換えれば次のようになる。

a.
F1 ; F2 ; F3
F4

アクションの切れ目が分りにくい時だけ;を入れれば良い。

b.


c. WHI

c1. W1 ← A while true
　　F

c2. W1 ← A, W1+B while W1 ≦ C
　　F

　または

　　W1 ← A(B)C
　　F

c3. W1 ← A, W1*B while W1 ≦ C
　　F

c4. AE(4×1) ← (A,B,C,D) または
　　I ← 1(1)4
　　E ← AE_i
　　F

　　E ← A,B,C,D
　　F

d. 配列演算

d1. 行列和

　　I ← 1(1)M
　　J ← 1(1)N
　　A_ij ← B_ij + C_ij

　または A(M×N) ← B(M×N) + C(M×N)

d2. 宣言された範囲すべてにわたる演算(添字の利用)

　A(M×N) の第3行をベクトル V(N×1) に代入は

　　I ← 1(1)N
　　V_i ← A_3i

であるが　V_* ← V_3* でもよい。従って

A(M×N) の全要素に0を代入するのは

　A(M×N) ← 0　または　A_** ← 0　でよい。

d3. ベクトルへの定数代入

　　I ← 1
　　E ← 4.5, 2.8, 3.6
　　A_i ← E ; I ← I+1

は

　A(3×1) ← (4.5, 2.8, 3.6) または

　A_* ← (4.5, 2.8, 3.6)

d4. 行列への定数代入

　B(2×3) ← (2.1, 3.6, 3.9), (1.1, 1.5, 2.6)

または

　B_** ← (2.1, 3.6, 3.9), (1.1, 1.5, 2.6)

d5. 単位行列作成

　　I ← 1(1)N
　　J ← 1(1)N
　　if I=J → A_ij ← 1
　　　　 → A_ij ← 0

または条件式を用い

　　I ← 1(1)N
　　J ← 1(1)N
　　A_ij ← if I=J → 1
　　　　　　 → 0

あるいは

　　A(N×N) ← 0
　　I ← 1(1)N
　　A_ii ← 1

論理式の結果 true は1, false は0に対応し

　　I ← 1(1)N
　　J ← 1(1)N
　　A_ij ← I=J

でもよい。

e. 入れ換え

代入すればもとの値は失なわれるので A と B の値を入れかえは　W ← A ; A ← B ; B ← W

となるが、メモとして左5

　A ↔ B

配列の場合は

　P(N×M) ↔ Q(N×M)　または　P_** ↔ Q_**

f. 構造脱出

　IF, CON, WHI 脱出　exit 名札

　(ただし名札は構造の前に：を付けてつける)

　WHI 脱出　exit 制御変数名

　多続・関数 脱出　exit 多続名

　プログラム脱出　exit

g. メモリとの直接演算

電卓のメモリプラスを拡張し

　I ← I+1　を　I+ ← 1

　B ← B−C　を　B− ← C

　P ← P*Q　を　P* ← Q

　R ← R/N　を　R/ ← N

と表現し書てもよい。

4° 奇数魔法陣の例

N×N の魔法陣を $A_{N \times N}$ に作るには、まず N行中央を1として右下に移りながら Kを $2, 3, \cdots, N^2$ と変化させ Kを代入してゆけばよい。ただし右下対角線上あるいはすでに埋められているときはもとの列の1行上、右の欄外に出たときは新らしい行の左端、下欄外に出たときは新らしい列の上端に移ればよい。

たとえば N=3 とすれば左図のようになり、

4	9	2
3	5	7
8	1	6

・各行各列および対角線の和はすべて15となっている。この手順を手続にまとめれば

手続 MAG1 $(N, A_{N \times N})$

```
[ A(N×N) ← 0
  I ← N ; J ← (N+1)/2 ; A(ij) ← 1
  [K] ← 2(1)N²
   └[ I+ ← 1 ; J+ ← 1
      if A(ij) ≠ 0 ∨ I=N+1 ∧ J=N+1 ──→ [ I− ← 2
                                         [ J− ← 1
        └→ if I=N+1 ──→ I ← 1
             └→ if J=N+1 ──→ J ← 1
      A(ij) ← K
```

K, I, J の記録をとってみると Kを3で割った余り つまり MOD(K,3) が1のときに限り Iは1減じJはそのまま、それ以外の場合はすべて もとのI,Jに対し MOD(I,N)+1　MOD(J,N)+1 をそれぞれ新らしいI,Jとしている。そこで次のような手続 MAG2 に書き換えてみる。(÷を整数割算 5÷2=2 とする)

K	I	J
1	3	2
2	1	3
③	2	1
4	1	1
5	2	2
⑥	3	3
7	2	3
8	3	1
⑨	1	2

手続 MAG2 $(N, A_{N \times N})$

```
[ 整関数 MOD(M,N)
    [ MOD ← M − M÷N*N
  I ← N ; J ← (N+1)/2 ; A(ij) ← 1
  [K] ← 2(1)N²
   └[ if MOD(K,N) ≠ 0 ──→ [ I ← MOD(I,N)+1
                            [ J ← MOD(J,N)+1
         └──────────────→ [ J− ← 1
      A(ij) ← K
```

5° コンピュータプログラムへの変換

アルゴリズム記法による記述から ALGOL 60 (A), PL/I (PL), PASCAL (PA), FORTRAN (F), Structured FORTRAN (SF)[7], BASIC (B) への変換の要点をまとめる。(Iは整数)

(1) 複合文　(A),(PA) では begin と end, (PL) では BEGIN; と END; で他にはない。

(2) ブロック　(A),(PL) だけにある。

(3) 選択文　(A),(PA) では if p then F1 else F2; (PL) では IF p THEN F1; ELSE F2; (SF) では IF (P) THEN ⌋ F1 ⌊ELSE⌋ F2 ⌊ENDIF

(4) 無条件反復　(A) では for WI:=A while true do F; (PL) では DO WHILE (P); F; END; (PA) では while p do F; end; (SF) では WHILE (P) DO ⌋ F ⌊ ENDWHILE

(5) 作変反復　(A) では for E:=A,B do F; (PL) では DO E=A,B; F; END;

(6) exit　(PL),(F),(SF) ではプログラム脱出を STOP, 手続脱出を RETURN

(7) 内部手続宣言　(A),(PL) にはある。(PA) ではプログラム全体についてのみ、(SF) ではパラメタなしのもののみある。

(8) 配列演算　(B),(PL) にのみある。

おわりに

良いプログラムは構造が明らかで読みやすく、修正等保守の容易なものでなければならない。そのためのフローを示すメモとしてはここに紹介したアルゴリズム記法によるものが最も容易であると考えられるので、是非日常の道具として利用されることをおすすめして本文を終ることとしたい。

(参考文献)

1) Dijkstra : GoTo Statement Considered Harmful. Comm. ACM Vol. 11 pp 147-148 (1968)

2) Böhm, Jacopini : Flow Diagrams, Turing Machines and Languages With Only Two Formation Rules. Comm. ACM Vol. 9 No. 5 pp. 366-371 (1966)

3) 中村慶一 : アルゴリズム記法 第1回電算機利用に関するシンポジウム予稿 土木学会 (1976)

4) 中村慶一 : コンピュータによる回帰分析 現代数学 9巻 12号 pp. 63-73 (1976)

5) 中村慶一 : コンピュータによる統計解析 森北出版 (1977)

6) 中村慶一 : プログラミング入門 — アルゴリズム記法による 森北出版 (1977予)

7) Friedman, Koffman : Problem Solving and Structured Programming in FORTRAN. Addison-Wesley (1977)

8) Aho, Hopcroft, Ullman : The Design and Analysis of Computer Algorithms. Addison-Wesley (1976)