構造化FORTRAN STRFORの前処理プログラムについて

吉村　卓

On a Preprocessor for Structured Fortran Strfor

Takashi YOSHIMURA

（昭和56年10月31日受理）

1．はしがき

読み易いプログラムのもつべき条件の一つとして制御構造が明確に組み立てられていることがあげられる。Fortranというプログラミング言語は残念ながら構造化されたプログラムを書くための言語としては必ずしも十分な能力を備えていない。

たとえば，Fortranでループ構造を直接表現できる文はdo文しかないが，do文はあらかじめ定められた回数だけループ本体を繰り返すために用いられる（回数判定型ループと呼ばれる）ものであって，ある条件が満足されたときにループを抜ける条件判定型ループを表現するには，Fortranではif文と組み合わせたgo to文を用いなければならない。

また，do文も制御変数を増す方向（前進型ループ）にしか回せないので，制御変数の値を減ずる方向にループを回す後退型ループを直接には書けない。さらに3つのパラメータには一般の式が書けない。

また，if文に関しては，条件の論理式が真の場合に実行すべき文を一つだけしか書けないので，図1aに示す分岐構造を表現するためにはgo to文を使わなければならない。

```
        IF(条件) GOTO L1
        V=e1
        GOTO L2
    L1  V=e2
    L2  CONTINUE
```

もっとも，e_1がe_2に比べて簡単な式であれば図1-bに示すように

```
        V=e1
        IF(条件) V=e2
```

と書けば，go to文は不要となりプログラムは簡単になる。

しかし図1-cに対しては

```
        IF(条件) GOTO L1
        P1
        P2
        .
        .
        .
        Pf
        GOTO L2
    L1  Q1
        Q2
        .
        .
        .
        Qt
    L2  CONTINUE
```

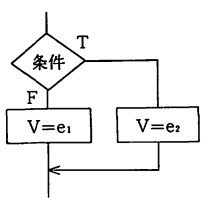

図1－a

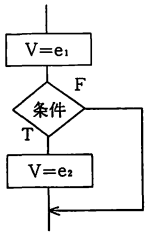

図1－b

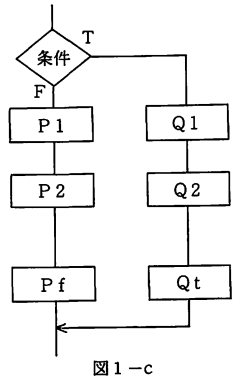

図1－c

図1

とgo to文の使用を避けることはできない。

一般にgo to文の乱用はプログラムの構造を理解しにくくする原因となっている。

このように，プログラミング言語としてのFortranには多くの欠点をあげることができるのであるが，Fortranは今までに広く使われてきており，処理系もよく整備され，言語の規格も十分よく定まっていて，多くのシステムに変更なしで通用するプログラムを書くことができることを考えると簡単には捨て難い言語である。

しかし，上述のように，はだかのFortranはプログラミング言語としては，はなはだ貧弱なので，構造化されたプログラムを書き易くするために，いくつかの繰返し構文と条件構文とを新しく導入し，拡張された言語（これをStrforと呼ぶことにする）で書かれたプログラムを前処理プログラムにより標準的Fortranのプログラムに変換するという手段を介して，標準的Fortranに不足している機能を補うことを考えよう。

2．構造的Fortran言語Strforの構文

図2−aに示す分岐構造を表現するためにif ～else構文

```
IF (条件)
    文1
ELSE
    文2
```

を用いる。図2−bのようにELSE以下はなくてもよい。なお，標準Fortranにおける算術if文はStrforでは使えない。

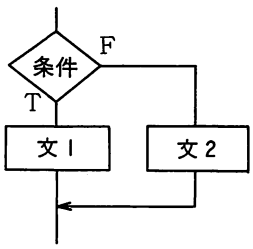

図2−a

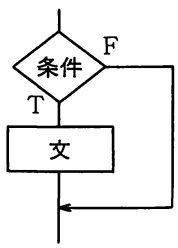

図2−b

図2

図3−aに示す判定先行型ループ構造を表現するためにwhile文

```
WHILE (条件)
    文
```

を用いる。

また，図3−bに示す判定後行型ループ構造を表現するにはrepeat ～until文

```
REPEAT
    文
    UNTIL (条件)
```

を用いる。

さらに，図3−cのループ構造を表わすためには，Fortranにおけるdo型構文に類似のfor文

```
FOR (初期設定; 条件; 再設定)
    文
```

を用いよう。ここに初期設定および再設定は単一のFortran文なら一応何でもよい。

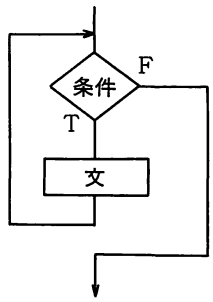

図3−a

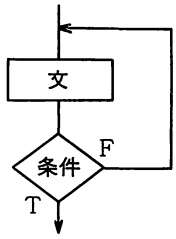

図3−b

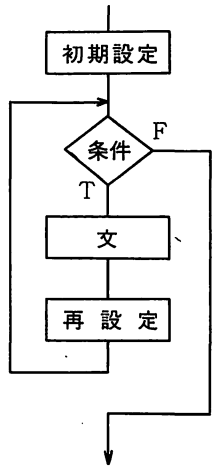

図3−c

図3

なお，while, repeat～until, forで表わされるループ構造の中からそのループを直ちに終らせたいときはbreak文を用いることにする。BREAKが現われるとプログラムの実行はそれを含むループの次の文から再開される。

また，いくつかの文の集まりをLBRACE＜とRBRACE＞とでくくって一つの複合文にまとめることを可能にしよう。このブロック化という手段によってできた複合文は単一の文と同等であって，単一の文が現われ得る任意の場所で使用してよい。

さらに，Strforでは行は自由欄形式で書くことにする。したがって，文は入力行のどこから始めても

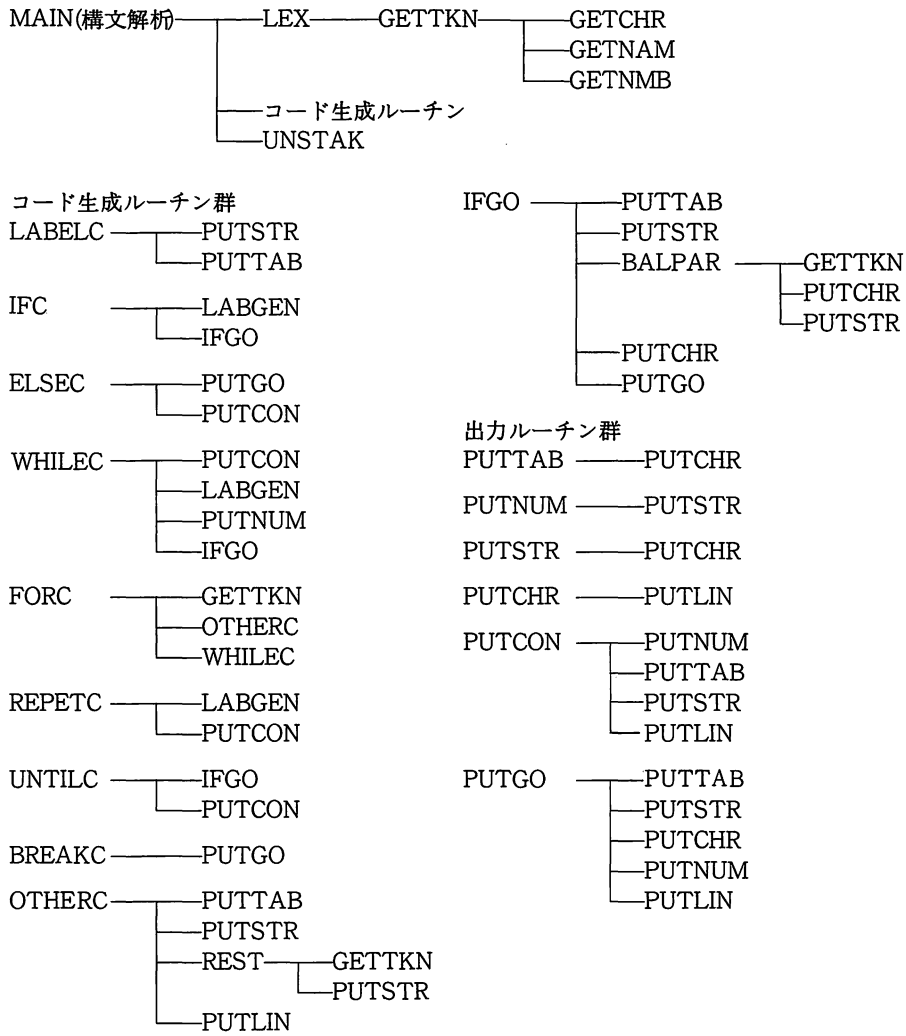

図4　前処理プログラムの諸ルーチン

よく，標準Fortranにおける継続行の表わし方や注釈行の記法は使えない。注釈文は行のどこから始めてもよく，＃記号がその始まりを示しその行の終りまでを注釈文とする。

3．前処理プログラム

Strfor語で書かれたプログラムを標準Fortran語のプログラムに翻訳する前処理プログラムは標準的Fortranで書かれているが，それは図4に示すルーチン群から構成されている。

3・1　構文解析部と字句解析部

前処理プログラムの中枢である構文解析部の概要を図5に示す。

各文の先頭で構文解析部は字句解析部LEXを呼んで文が文法上どの項目に属するかを決める。

```
integer function LEX によりtoken を読む
while(token≠EOF) <
    if((スタックの先頭＝LBRACE)&(token＝RBRACE)) スタックポインタを1つ戻す
    各文(DIGIT～OTHER)に対応するコード生成ルーチンを実行
    if(tokenはDIGIT,IF,ELSE,WHILE,FOR,REPEAT またはLBRACE)
        tokenと名札をスタックに積む
    else(UNTIL,BREAK,OTHER,RBRACE または SEMCOL のはず)
        次のtoken を先読みし,UNSTAK を呼ぶ
    字句解析部 LEX によりtoken を読む
    >
stop


while(スタックは空でない) <
    if(スタックの先頭＝LBRACE) return
    if((スタックの先頭＝IF)&(次のtoken＝ELSE)) return
    if((スタックの先頭＝REPEAT)&(次のtoken＝UNTIL)) return
    スタックの先頭に積まれている文の終了処理(FORの場合再設定部の出力を含む)
    スタックポインタを1つ戻す(ELSEの場合2つ戻す)
    >
return
```

図5　前処理プログラム主部（構文解析部）ならびに副プログラムUNSTAKの概要

字句解析部はtoken切り出しルーチンGETTKNを呼んで文の最初のtokenを取り出す。それによって文の種類が決まる。

GETTKNは入力行を名札，英字名，区切り文字に切り分ける。まず，GETCHRを呼ぶ。GETCHRは入力バッファINBUFから1文字取り出してきてそれが英字か数字かあるいは区切り文字かを識別する。英字ならLETTER(＝1)を，数字ならDIGIT(＝2)を，それ以外のときは0をそれぞれICに入れて戻る。

GETTKNはGETCHRから戻ってきたとき，IC＝LETTERならGETNAMを呼んで英字名を読みとる。また，IC＝DIGITならGETNMBを呼んで数字列を読み出す。英字名のときNAME(＝1)を，整定数のときSEISU(＝2)を，区切り記号のとき0をそれぞれIWに入れ，さらに読みとった英字名，名札あるいは区切り文字は配列STRに格納して呼び出したLEXに戻って行く。

なお，GETCHRは1文字読み出すごとに入力バッファのポインタINPTを1つずつ進め，81より大となったときソースカードよりINBUFに1行分の情報を読み込む。

また，GETNMBは読みとった整定数を内部表現に変換してNMBに格納する。

字句解析部LEXはGETTKNから戻ったとき，IW＝SEISUならLEXにDIGITを，また，STR(1)＝EOF，SEMICOL，LBRACEあるいはRBRACEならその記号をLEXに入れて構文解析部に戻って行く。それ以外のときは見出し語テーブルを検索しなければならない。もし見つかればLEXに対応する型名を入れて戻る。見つからなかったときはLEXにOTHERを入れて戻る。すなわち，if～else文，while文，repeat～until文，for文，break文のいずれでもないものはOTHERとする。標準Fortranの文はほとんどOTHERと考えてよい。

文の分類が終ると構文解析部は対応するコード生成ルーチンを呼ぶ。IFC，WHILEC，REPETCならびに（WHILECを介して）FORCの各ルーチンはLABGENを呼んで一意的な名札を造り出して返してくる。その名札は文の型名とともにスタックに積まれる。

構文解析部は文の終り（UNTIL，BREAK，OTHER，RBRACEまたはSEMCOL）に出会うとUNSTAKを呼んで型名と名札をスタックから取りおろす。

なお，構文解析部から呼ばれたコード生成ルーチンの中，IFC，WHILEC，UNTILC，FORC，OTHERCからも文の残部を取り出すために再びGETTKNが呼ばれる。

3・2　コード生成部と出力ルーチン

Strforの制御構造を表わす文はそれぞれ次のよう

にFortran化される。

if～else構文を標準Fortranに翻訳した結果は

```
        IF(.NOT.(条件)) GOTO L
            文1
            GOTO L+1
    L   CONTINUE
            文2
    L+1 CONTINUE
```

となる。

while文は

```
        CONTINUE
    L   IF(.NOT.(条件)) GOTO L+1
            文
            GOTO L
    L+1 CONTINUE
```

と翻訳される。

repeat～until文の翻訳結果は

```
    L   CONTINUE
            文
        IF(.NOT.(条件)) GOTO L
    L+1 CONTINUE
```

である。

for文は

```
            初期設定
    L       IF(.NOT.(条件)) GOTO L+1
                文
                再設定
                GOTO L
    L+1     CONTINUE
```

となる。

なお，上記3種のループ構文の翻訳結果に現われる名札L＋1はBREAKの行先の名札としての役も果している。

また，forループが何重にもネストしている場合，内側のループに対応するコード生成が終了するまで外側のループの再設定部を記憶させておくために，型名と名札のスタックとは別にfor文用のスタックが必要である。

LABELCはソースプログラムに名札（型名DIGIT）が現われたとき呼び出されて，まず欄1以降にその名札を出力し，そのあとFortranの標準的開始位置すなわち欄7までの間を空白でうめる。

BREAKCはスタックの中を型名WHILE，REPEAT，FORが見つかるまで探してゆき，それが見つかったところで正しい名札をつけて

```
        GOTO L+1
```

を出力する。

OTHERCは入力を単に出力行の欄7から欄72までの間に転写するだけである。

なお，IFGOはIFC，WHILEC，UNTILCで用いられ，

```
        IF(.NOT.(条件)) GOTO L
```

という形の文を作り出す。

また，BALPARはIFGOから呼び出されてif文の条件部（かっこに関して均衡した文字列）をとり集めて出力する。

出力行は配列OUTBUFの中に一度に1文字ずつ送り込まれてゆく。OUTBUFに最後に入った文字はポインタCOLMによって指される。実際にOUTBUFを出力してCOLMを0に戻す仕事はPUTLINによってなされる。それは各種の文の終了時にPUTGO，PUTCON，OTHERCから呼び出されるほか，継続行を開始する際，すなわちCOLMの値が72となったときそれまでにたまっていたものを追い出すためにPUTCHRからも呼ばれる。

PUTCHRはOUTBUFに文字を送り込む。これはまた，上でも述べたように継続行の処理もする。

なお，処理を簡単にするために，入力はGETCHRにおいて配列INBUFへ1語に1文字ずつ読み込むが，出力はPUTCHRにおいてOUTBUFへ1語4バイトに4文字ずつ詰めて格納する。

PUTTABはポインタCOLMを強制的に欄6の次へ進めるためのものである。

PUTSTRはPUTCHRを繰返し呼んで文字列をOUTBUFに出力する。

PUTNUMは数を内部表現から文字表現に変換し，それをPUTSTRによってOUTBUFに出力する。

PUTCONおよびPUTGOはそれぞれ

```
    L CONTINUE
```

および

```
    GOTO L
```

を出力する。

4．Strforによるプログラム例

以下にいくつかのStrfor語によるプログラム例を示す。

図6ははさみ打ち法による方程式f(x)＝0の解法プログラムの一部ならびにその翻訳結果である。根に近い数で

$$f(a)f(b)<0$$

となる2数a，bから出発して，曲線y＝f(x)上の2

```
      FA=F(A)
      FB=F(B)
      FC=FA
      WHILE(ABS(FC).GE.EPS) <
         C=(A*FB-B*FA)/(FB-FA)
         FC=F(C)
         IF(FA*FC.LT.0) <
               FB=FC
               B=C
               >
            ELSE <
               FA=FC
               A=C
               >
         >
      WRITE(6,600)C
```

```
      FA=F(A)
      FB=F(B)
      FC=FA
      CONTINUE
90000 IF(.NOT.(ABS(FC).GE.EPS))GOTO 90001
      C=(A*FB-B*FA)/(FB-FA)
      FC=F(C)
      IF(.NOT.(FA*FC.LT.0))GOTO 90002
      FB=FC
      B=C
      GOTO 90003
90002 CONTINUE
      FA=FC
      A=C
90003 CONTINUE
      GOTO 90000
90001 CONTINUE
      WRITE(6,600)C
```

図6

点(a, f(a)), (b, f(b))を結ぶ直線がx軸と交わる点cを求め，このcをaあるいはbに替えて再びcの計算に戻る。

$$|f(c)| < \varepsilon$$

となるまでこの操作を繰り返す。プログラムではwhile文とif～else構文が用いられている。

```
      Y=X0
      B=A(1)
      C=B
      REPEAT <
         X=Y
         FOR(J=2;J.LE.N;J=J+1) <
            B=X*B+A(J)
            C=X*C+B
            >
         B=X*B+A(N+1)
         Y=X-B/C
         >
      UNTIL(ABS(Y-X).LT.EPS)
      RETURN
```

```
      Y=X0
      B=A(1)
      C=B
      CONTINUE
90000 X=Y
      J=2
90002 IF(.NOT.(J.LE.N))GOTO 90003
      B=X*B+A(J)
      C=X*C+B
      J=J+1
      GOTO 90002
90003 CONTINUE
      B=X*B+A(N+1)
      Y=X-B/C
      IF(.NOT.(ABS(Y-X).LT.EPS))GOTO 90000
90001 CONTINUE
      RETURN
```

図7

図7はニュートン法によるn次方程式f(x)＝0の解法サブルーチン副プログラムの一部ならびにその翻訳結果である。n次の多項式におけるn＋1個の係数を入力データとして，近似値xにおける多項式f(x)の値およびその微係数f′(x)の値を求めるためにホルナー法が用いられる。引続く2つの近似値xとyが

$$|y - x| < \varepsilon$$

```
      READ(5,500)A,B,EPS
      N=1
      W=B-A
      T=(F(A)+F(B))*0.5*W
      C=F(A+W*0.5)*W
      S=(2.0*C+T)/3.0
      ERR=ABS(T-C)*0.5
      WRITE(6,600)N,T,C,S,ERR
      WHILE(ERR.GE.EPS) <
         N=2*N
         W=W*0.5
         SUM=0.0
         FOR(X=A+0.5*W;X.LT.B;X=X+W)
            SUM=SUM+F(X)
         T=(T+C)*0.5
         C=W*SUM
         S=(2.0*C+T)/3.0
         ERR=ABS(T-C)*0.5
         WRITE(6,600)N,T,C,S,ERR
         >
      STOP
```

```
      READ(5,500)A,B,EPS
      N=1
      W=B-A
      T=(F(A)+F(B))*0.5*W
      C=F(A+W*0.5)*W
      S=(2.0*C+T)/3.0
      ERR=ABS(T-C)*0.5
      WRITE(6,600)N,T,C,S,ERR
      CONTINUE
90000 IF(.NOT.(ERR.GE.EPS))GOTO 90001
      N=2*N
      W=W*0.5
      SUM=0.0
      X=A+0.5*W
90002 IF(.NOT.(X.LT.B))GOTO 90003
      SUM=SUM+F(X)
      X=X+W
      GOTO 90002
90003 CONTINUE
      T=(T+C)*0.5
      C=W*SUM
      S=(2.0*C+T)/3.0
      ERR=ABS(T-C)*0.5
      WRITE(6,600)N,T,C,S,ERR
      GOTO 90000
90001 CONTINUE
      STOP
```

図8

となるまで繰返される。repeat～until構文が用いられている。なお，ホルナー法のための繰返しはforで実現されている。

図8は台形公式，中点公式，シンプソン公式を併用してf(x)のaからbまでの定積分の近似値を計算するプログラムの一部ならびにその翻訳結果である。下に（あるいは上に）凸なる関数に対して，台形公式による近似値Tと中点公式による近似値Cとの相加平均$\frac{1}{2}(C+T)$に含まれる誤差の上限は$\frac{1}{2}|T-C|$で推定できる。なお，この$\frac{1}{2}(C+T)$は分割を2倍にしたとき得られる台形公式による近似値と一致する。与えられた積分区間〔a, b〕を次々と2等分しながら，したがって標本点の数を2倍，2倍と増やしていって，誤差の推定値が許容限界εより

```
      NP1=N+1
      FOR(K=1;K.LT.N;K=K+1) <
         FOR(I=K+1;I.LE.N;I=I+1)  <
            AIK=A(I,K)/A(K,K)
            FOR(J=K+1;J.LE.NP1;J=J+1)
               A(I,J)=A(I,J)-AIK*A(K,J)
            >
         >
      X(N)=A(N,NP1)/A(N,N)
      FOR(I=N-1;I.GT.0;I=I-1) <
         T=A(I,NP1)
         FOR(J=I+1;J.LE.N;J=J+1)
            T=T-A(I,J)*X(J)
         X(I)=T/A(I,I)
         >
```

```
      NP1=N+1
      K=1
90000 IF(.NOT.(K.LT.N))GOTO 90001
      I=K+1
90002 IF(.NOT.(I.LE.N))GOTO 90003
      AIK=A(I,K)/A(K,K)
      J=K+1
90004 IF(.NOT.(J.LE.NP1))GOTO 90005
      A(I,J)=A(I,J)-AIK*A(K,J)
      J=J+1
      GOTO 90004
90005 CONTINUE
      I=I+1
      GOTO 90002
90003 CONTINUE
      K=K+1
      GOTO 90000
90001 CONTINUE
      X(N)=A(N,NP1)/A(N,N)
      I=N-1
90006 IF(.NOT.(I.GT.0))GOTO 90007
      T=A(I,NP1)
      J=I+1
90008 IF(.NOT.(J.LE.N))GOTO 90009
      T=T-A(I,J)*X(J)
      J=J+1
      GOTO 90008
90009 CONTINUE
      X(I)=T/A(I,I)
      I=I-1
      GOTO 90006
90007 CONTINUE
```

図9

も小さくなるまで繰返す。この繰返しはwhile文を用いて実現している。また，中点公式における標本点での関数値の和を計算する繰返し部分にはfor文を用いている。なお，forの設定部と再設定部には式が用いられている。

図9はガウスの消去法による連立1次方程式の解法プログラムの中心部分である。ガウスの消去法は，与えられた方程式の係数行列が上三角行列の形になるように次々と未知数を消去する前進消去過程と，その結果，n番目の未知数から始めて逐次n－1番目，n－2番目，…，1番目の未知数の値を求める後退代入過程の2段から成る。前進消去過程は3重ループのfor構造である。後退代入過程は2重ループ構造であるが，その外側のループは名前の如く後退型ループであって，再設定部で制御変数の値が1ずつ減ぜられている。

参考文献

1）岸田孝一，三田守久　プログラム・フローチャート　日本生産性本部
2）有譯誠　FORTRANソフトウェア工学　共立出版
3）B.W.Kernighan & P.J.Plauger　ソフトウェア作法　共立出版