談話室

FORTRAN の追加機能に関する一提言†

磯　道　義　典††

新しい FORTRAN の規格が FORTRAN 77 として 1978 年4月に発表された．本誌上にも西村恕彦氏によるその紹介がなされている[1]．この内容を見ると構造的プログラミングの手法である段落（ブロック）IF 文が採用されている反面，繰返し型構文 while do や repeat until は採用されていない．また段落 IF 文そのものも ELSEIF 文や ENDIF 文を伴うため構造的プログラミングにおけるものよりも見苦しい格好となっている．

さらに言うならばそもそもこのブロック構造的なあり様は FORTRAN の考え方とはマッチしない何ものかを含んでいるように思われる．そこで本稿では，FORTRAN の思想にマッチしたこれらにかわる新しい文を提案してみたい．

（1） 条件付き構文 (EXECUTE WHEN 文)

EXECUTE WHEN 文は算術 IF 文，論理 IF 文，段落 IF 文のかわりとして導入するもので次の形を持つ．

EXECUTE [文番号] WHEN (論理式) {,[文番号] WHEN (論理式)} [,文番号]

ただしここで {・} はこの形のものを0回以上任意回書けることを示しており，[・] はこの部分は無くとも良いことを示している．

例をあげて具体的にその働きを説明しよう．図-1 のような例の場合には次のような仕事を行う．A>2.0 では EXECUTE の次の文から文番号 10 の文までを実行し，A>2.0 でなくて A^2<7.0 の時には文番号 10 の文の次から文番号 20 の文までを実行する．その他の場合には文番号20の文の次から文番号30の文まで実行する．これらの実行が終れば文番号30の文の次へ制御が移される．この例での文番号30は EXECUTE の終端となっているのでこの間を EXECUTE の範囲と言う．またもしもこの例で第2の WHEN の次の [,30] がなければ EXECUTE WHEN の実行において2条件を満足しない場合には何も実行せずにこの，EXECUTE の範囲を出てしまう．この時 EXECUTE の範囲は 20 までである．WHEN の前の文番号が無ければこの条件の時には何もしないで EXECUTE の範囲を出てしまうのである．この具体的説明で一般の場合も理解出来ることと思われる．（ただし論理式の評価は左のほうから行われる．）

```
      EXECUTE 10 WHEN (A. GT. 2. 0), 20
     *WHEN (A**2. LT. 7. 0), 30
10   CONTINUE   } 第一条件の実行文部
20   CONTINUE   } 第二条件の実行文部（第一条件をみたさない状況で）
30   CONTINUE   } その他の場合の実行文部
```

図-1　条件付構文

この条件付き構文は if then else を何重にも使う形と同じであるが文番号で範囲が明示されている点で理解しやすい面を持つとともに FORTRAN の考え方，思想をまげずに構文が作られている点で優れている．さらに論理 IF 文のような複文になっていないこと，およびある条件のとき複数の文がきれいに書ける点で優れていると思われる．またこの構文が使えると計算型 GOTO 文や割当て型 GOTO 文も本質的に不用となるはずである．

（2） 繰返し構文 (DO WHILE 文 DO UNTIL 文)

PASCAL 等で使われている繰返し構文 while do や repeat until は非常に便利なものであるが中途でぬけ出すことが許されていない．この点の改良を含めて2つの新しい繰返し構文を導入する．これらは DO WHILE 文と DO UNTIL 文といい次の形を持つ．

DO 文番号 WHILE (論理式)

DO 文番号 UNTIL (論理式)

両者は論理式を否定型にとればまったく同じ働きを持つ．以下では DO WHILE を例によって説明しよう．

IOLD=1

† On New FORTRAN Statements by Yoshinori ISOMICHI (Faculty of Integrated Arts and Sciences, Hiroshima University).

†† 広島大学総合科学部

```
    LAST=1
    PRINT 100, IOLD, LAST
100 FORMAT (1X, I10/1X, I10)
    DO 10 WHILE (NEW. LE. 1000)
    NEW=IOLD+LAST
    PRINT 110, NEW
110 FORMAT (1X, I10)
    IOLD=LAST
10  LAST=NEW
```

この例において DO の出口は変数 NEW が変化した直後すなわち第6行目と7行目の中間である．いわゆる EXIT などをおかないで WHILE 文中の条件に変化のおこりうるところで条件が満足されているかどうかしらべ，条件が満足されなくなった時点でこの DO の範囲を飛び出すのである．この例の DO 文を DO 10 UNTIL (NEW. GT. 1000) でおきかえてもまったく同じ仕事をする．

この形式の繰返し構文では条件の論理式しだいで複数の出口を作ることができるし，出口をただ1つにしたいのであれば論理変数を1つ導入する等の処置によってそのようにすることもできる．もちろんPASCAL の while do や repeat until もここにあげた構文の一種の制限形なのですぐに実現できるわけである．よってまともなプログラムは単純 GOTO 文をほとんど必要としないこととなる．

（3） DO 型関数 (SUM, PROD, MAX, MIN)

FORTRAN 等を数値計算の場面で使っていると，$\Sigma_i a_i$, $\Pi_i a_i$, $\mathrm{Max}_i a_i$, $\mathrm{Min}_i a_i$ の形の演算がしばしば必要となる．こうしたものをわざわざ DO 文を使って記述するのはめんどうであるし，人間の思考をそのまま書き下すという精神からも望ましくない．そこで以下のような形式の組込み関数を許すべきだと考える．

```
B=SUM(A(I); I=1, 20, 2)
C=PROD(A(I); I=2, 20)
D=MAX(A(I, J); I=1, 20; J=1, 8)
E=MIN(A(I); I=20, 1, -1)
```

この形は DO 型並びの形式で複数個の加算等を実現しようとするものである．SUM を例に正式の関数引用形を書けば次のようになる．

```
SUM (算術式; 算術変数名
        =算術式, 算術式[,算術式]
        {;算術変数名=算術式, 算術式[,算術式]})
```

この意味は上述の例から明らかであろう．

（4） 条 件 式

条件付構文の考え方を式表現に使ったものを条件式という．(McCarthy の条件式の制限形とみなすことが出来る)．その形は次の形の式をいう．

```
(式; WHEN(論理式){,式; WHEN(論理式)}
                式)
```

この具体例をあげると次のようなものである．

```
A=(0.; WHEN(X. LT. 0.0), 1.; WHEN
        (X. GT. 1.), X)
```

この意味は以下のようなものである．

$$A=\begin{cases}0 & (X<0)\\ 1 & (X>1)\\ X & (\text{otherwise})\end{cases}$$

以上4種の新しい FORTRAN 文の提案を行ったが最後に構造的プログラム技法の批判をしておきたい．まず第一に単純 GOTO 文を FORTRAN からすててしまうことはよろしくないという点について一言述べる．複数のループが入れ子にならないような状況では条件付き構文や繰返し構文をうまく設計しても多少とも無理を生じ，わかりにくいプログラムとなる[2]．この意味で単純 GOTO 文は残して置くべきである．もちろんこの文の多さはプログラミングの不自然さの尺度であることはまちがいがないから，できるだけ少なくするように努力する必要はあるがそれ以上に禁止するといった態度はまずいと思われる．次にプログラミングのブロック構造化として begin…end を採用する方向について一言．begin—end はいわばカッコのようなものであり，これが何重にも重なると人間には読みづらいものとなる．この意味で begin—end などというものを推奨する立場には反対である．反論として行の上げ下げがあるから問題無いといわれるであろうが，実は逆で begin—end のやり方がまずいからどうしてもこれにたよらざるをえないのである．行の上げ下げなら本稿で提案した文を使ってでもまともなプログラムでは可能であり，多くの人はそのようにしてプログラムを書くのである．行のあげさげをしなかったらとたんに目のちらちらしてくるような言語こそ改良しなければならないのである．

参 考 文 献

1) 西村：FORTRAN 77 の特徴，情報処理 Vol. 20, No. 5, pp. 432-437 (1979).
2) 木村 泉：GO TO 論争，bit Vol. 7, No. 5, pp. 346-379 (1975).

(昭和55年4月28日受付)