Ratforの使用について

古城, 久美子
九州大学大型計算機センター研究開発部

https://doi.org/10.15017/1472540

出版情報：九州大学大型計算機センター広報. 11 (3), pp.191-198, 1978-09-01. 九州大学大型計算機センター
バージョン：
権利関係：

Ratforの使用について

古城久美子*

0. はじめに

Ratfor（Rational fortran）は，ベル研究所のKernighanらによって，開発されたプログラム言語である。

本センターにおいて，Ratforが使えるようになったので紹介する。

Fortranは，プログラマの意図をプログラムの構造に，反映しにくい言語であると言われている。Ratforとは，Fortranのこの点を，いくらか手直しした改良版Fortranとでも思っていただいたら良いのではないだろうか。Ratforは，プリプロセッサ方式になっていて，Ratfor語で書かれたプログラムは，プリプロセッサでFortran語のプログラムに翻訳される。処理の流れを示すと，図1のようになる。

Ratforは，Fortranに基づいているので，ここでは，文法的にFortranと異なる部分について簡単に述べた。したがって，詳細は，文献1）を参照していただきたい。

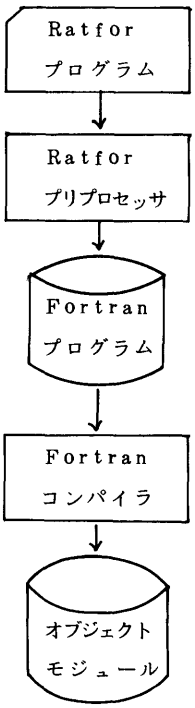

図1　Ratforにおける処理の流れ

1. Ratforの構文

Ratforの構文を，Backus-Naur Formで書くと，下記のようになる。

```
<program>      ::= <statement>
                 | <program><statement>
<statement>    ::= if(<condition>)<statement>
                 | if(<condition>)<statement1>
                     else <statement2>
                 | while(<condition>) <statement>
                 | for (<initialize>;<condition>;<reinitialize>)
                                 <statement>
                 | repeat <statement>
                 | repeat <statement> until(<condition>)
                 | do <limits> <statement>
                 | <digits><statement>
                 | break
                 | next
                 | <other>  （注1）
```

（注1）　otherとは，IF文，DO文以外のFortranのstaementである。

---

* 九州大学大型計算機センター研究開発部

1） if 文

condition が真の時，statement$_1$ を，偽の時，statement$_2$ を実行する。

（変換方法）

Ratfor		Fortran
if (condition)		IF(.NOT.(condition)) GOTO n_1
statement$_1$		statement$_1$
else	⇨	GOTO n_2
statement$_2$		n_1 CONTINUE
		statement$_2$
		n_2 CONTINUE

2） while 文

condition が真の間，statement を実行する。condition が，先に判定されるので，statement が1回も実行されないことがある。

（変換方法）

Ratfor		Fortran
while(condition)		CONTINUE
statement		n_1 IF(.NOT.(condition)) GOTO n_2
	⇨	statement
		GOTO n_1
		n_2 CONTINUE

3） for 文

while 文に，初期設定と更新を加えたものである。

initialize，condition，reinitialize のいずれも省略できるが，区切り文字 ▼;▼ は，必要である。

condition を省略すると，無限ループとなるので注意していただきたい。

（変換方法）

Ratfor		Fortran
		initialize
for(initialize ;		n_2 IF(.NOT.(condition)) GOTO n_2
condition ; reinitialize)		statement
statement	⇨	reinitialize
		GOTO n_1
		CONTINUE

4） repeat 文

1回以上のループを作る。条件が真となるまで実行する。

until (＜condition＞) のない repeat 文では，無限ループとなる。

（変換方法）

Ratfor		Fortran
repeat		n　CONTINUE
statement	⇨	statement
until (condition)		IF(.NOT.(condition))　GOTO　n

5）do文

limitsは，Fortranにおける DO文のループ制御と同じである。ただし，文番号は，使用していない。

（変換方法）

Ratfor		Fortran
		DO　n　limits
do limits	⇨	statement
statements		n　CONTINUE

6）break文

while，for，repeat，do文のループの中から，ループの外へ飛び出す時に，用いる。

7）next文

この文があると，その回のループの残りの処理は，無視され，次の回のループの処理へ移る。

8）define文

次の形式で書く。

define（ character-string$_1$ ，character-string$_2$ ）（注２）

この文以降で，define文の character-string$_1$ に一致する文字列があると，全てcharacter-stting$_2$ に置きかえて，処理する。再定義されない限り，ソースプログラムの最後まで有効である。

（変換例）

Ratfor		Fortran
define（ SIZE，100）		
dimension　a（ SIZE）	⇨	DIMENSION　A（100）

9）include文

次の形式で書く。

include logical-unit

論理機番に割当てられたデータセットの内容を，Ratforのプログラムの一部として，この文のある所に組み込む。

10）その他

- Ratforは，自由形式であるので，１カラムから７２カラムまでのいずれに書いても良い。７３～８０カラムは，無視される。
- if文や，while文などで，まだ文が終っていない行や，コンマで終っている行は，次に継続しているとみなす。算術式は，アンダーバーを継続の印とする。

（注２）　defineの次とコンマの前にブランクがあってはいけない。

例

1枚め　50 format(▼␣␣▼ ,
2枚め　　f6.2)　⇨　50 format(▼␣␣▼ ,f6.2)

1枚め　i=j+ _ (アンダーバー)　⇨　i=j+3*k
2枚め　3*k

・ 1行に2つ以上のステートメントを書く時には，ステートメントの間に，▼;▼ を必ず書く。ただし，この場合，1行以内にステートメントが，おさまっていなければならない。

・ 行の途中に▼#▼があると，それ以降は，注釈とみなす。（注3）

・ 文字定数は，全て引用符（▼か又は▼▼）で囲まなくてはならず，1行以内におさまっていなければいけない。nH形記述子は，使用できない。（注3）

文字列の中に，▼（クォート）を含みたい時は，その文字列を ▼▼（ダブルクォート）で囲み，▼▼を含みたい時は，▼ で囲む。

例　▼▼don▼t be noisy!▼▼
▼　▼▼what?▼▼ she said.▼

・ 行の最初にでてきた数字の列は，文番号とみなされる。

・ Ratforで用いられる比較演算子，および，論理演算子は，次のようになっている。

（Fortranの方式でも支障はない。）

Ratfor	Fortran
>	.GT.
>=	.GE.
<	.LT.
<=	.LE.
==	.EQ.
¬=	.NE.
&	.AND.
|	.OR.
¬	.NOT.

・ 23000番台の文番号は，Ratforのプリプロセッサで使われているので，ユーザは，23000番台の文番号を使用してはいけない。

・ 穿孔機および，端末に，▼{▼と▼}▼の記号がない時，それぞれ▼¥(▼と▼¥)▼で，代用できる。（注4）

・ Ratforのソースプログラムは，本来シンボリックパラメータを除いて，小文字であるが，大文字でもよい。ただし，大文字を使う場合には，制御文に少し相違がある。

2. 変換例

読み込んだ整数が，素数であるかどうかを調べるプログラムである。11111を読み込むと，処理を終わるようになっている。

（注3）　ここで言っている行とは，80カラム，カード1枚のことである。

（注4）　穿孔機および端末では，¥が$となっているものがある。

Ratfor ソースリスト

```
***  RATFOR SOURCE LIST ***
      1          #PROGRAM TO TEST IF N IS A PRIME
      2      DEFIN(EOF,11111)
      3      INTEGER R
      4          100 FORMAT(I5)
      5      FOR(READ(5,100) N ; N ¬= EOF ; READ(5,100) N ) ¥(
      6          IF(N <= 1) ¥(              #ILLEGAL INPUT ?
      7              WRITE(6,110) N
      8              110  FORMAT(' ','ILLEGAL INPUT N=',I5,' <=1')
      9              ¥)
     10          ELSE IF(N == 2) ¥(                  #N=2   ?
     11              WRITE(6,120) N
     12              120  FORMAT(' ',I5,' IS A PRIME NUMBER')
     13              ¥)
     14          ELSE IF(MOD(N,2) ==0) ¥(   #N= EVEN NUMBER ?
     15              WRITE(6,130) N
     16              130  FORMAT(' ',I5,' IS NOT A PRIME NUMBER')
     17              ¥)
     18          ELSE ¥(                    #N= PRIME NUMBER ?
     19              RN=FLOAT(N)
     20              FOR( R=3 ; R <= SQRT(RN) ; R=R+2)
     21                  IF(MOD(N,R) == 0) BREAK
     22              IF(R <= SQRT(RN)) WRITE(6,130) N
     23              ELSE WRITE(6,120) N
     24             ¥)
     25          ¥)
     26      STOP
     27      END
```

Fortran に変換されたソースリスト

```
000001          1NTEGERR
000002  100     FORMAT(I5)
000003          CONTINUE
000004          READ(5,100)N
000005  23000   IF(.NOT.(N.NE.11111))GOTO 23002
000006          IF(.NOT.(N.LE.1))GOTO 23003
000007          WRITE(6,110)N
```

```
000008   110    FORMAT(IH ,16HILLEGAL INPUT N=,I5,4H <=1)
000009          GOTO 23004
000010   23003  CONTINUE
000011          IF(.NOT.(N.EQ.2))GOTO 23005
000012          WRITE(6,120)N
000013   120    FORMAT(1H ,I5,18H IS A PRIME NUMBER)
000014          GOTO 23006
000015   23005  CONTINUE
000016          IF(.NOT.(MOD(N,2).EQ.0))GOTO 23007
000017          WRITE(6,130)N
000018   130    FORMAT(1H, I5,22H IS NOT A PRIME NUMBER)
000019          GOTO 23008
000020   23007  CONTINUE
000021          RN=FLOAT(N)
000022          CONTINUE
000023          R=3
000024   23009  IF(.NOT.(R.LE.SQRT(RN)))GOTO 23011
000025          IF(.NOT.(MOD(N,R).EQ.0))GOTO 23012
000026          GOTO 23011
000027   23012  CONTINUE
000028   23010  R=R+2
000029          GOTO 23009
000030   23011  CONTINUE
000031          IF(.NOT.(R.LE.SQRT(RN)))GOTO 23014
000032          WRITE(6,130)N
000033          GOTO 23015
000034   23014  CONTINUE
000035          WRITE(6,120)N
000036   23015  CONTINUE
000037   23008  CONTINUE
000038   23006  CONTINUE
000039   23004  CONTINUE
000040   23001  READ(5,100)N
000041          GOTO 23000
000042   23002  CONTINUE
000043          STOP
000044          END
```

3. Ratfor の制御文

Ratforを使用するための制御文を以下に示す。カタログドプロシジャとしては，RATFORとRATFORCの2つがある。

1）ソースプログラムが，大文字で書かれており，変換されたFortranプログラムを，カタログドプロシジャFORTCGで，実行する。

```
//  EXEC RATFORC  （注5）
// RAT.SYSIN DD *
                .
                .
                .
          ソースプログラム
                .
                .
                .
//  EXEC FORTCG ,FORT.PARM=NOSOURCE （注6）
// FORT.SYSIN DD DSN=&&OUT,
                    DISP=(OLD) （注7）
//GO.SYSIN DD *
                .
                .
                .
             データ
                .
                .
                .
//
```

2）変換されたソースプログラムを，すぐ実行せずに，データセットF9999.SOURCE.FORTに，保存しておく。

```
//  EXEC RATFORC
// RAT.SYSIN DD *
                .
                .
                .
             データ
                .
                .
                .
```

（注5） 小文字でプログラムを書いた時には，RATFORというカタログドプロシジャ名を使う。他の点では，相違はない。

（注6） 使用したいFortranのカタログドプロシジャ名。この場合，変換されたFortranプログラムのリストはでないように指定してある。

（注7） 変換ルーチンによって，1時データセット&&OUTに変換されたプログラムを出力するので，Fortran処理系への入力として必ずこれを指定する。

```
//RAT.FT99F001 DD DSN=F9999.SOURCE.FORT,
//                DISP=(NEW,CATLG),UNIT=PUB,
//                SPACE=(TRK,(20,20),RLSE),
//                DCB=(LRECL=80,BLKSIZE=1200,RECFM=FB)
//
```

4. おわりに

Ratforは，プリプロセッサで処理する時間だけ，Fortranのプログラムより時間がかかる。しかし，プログラムの見やすさ，変更のしやすさ，プログラマの意図をプログラムへ反映しやすいことなどでは，Ratforに，軍配があがるであろう。

一度，使ってみていただきたいものである。

文　献

1) Kernighan and Plauger : Software tools , Addison-Wesley,1976