Mortran 3 (2)

平山　英夫
KEK, 高エネルギー加速器研究機構

マクロの定義

REPLACE {pattern} WITH {replacement}

- “pattern” と同じ表現がプログラム中で使用されると、その部分は、“replacement”に置き換えた後に FORTRANに変換される。
- “replacement”としては、数値でもステートメントでも使用できる。

REPLACE {ARRAYSIZE} WITH {50}

DIMENSION X(ARRAYSIZE);…….DO J=1,ARRAYSIZE […..;]

は、次の文と同じ

DIMENSION X(50);……DO J=1,50 [……;]

プログラム中でのブランク

- プログラム中のブランクは、“pattern”と比較する場合に、無視される。

REPLACE {SIGMA(1);} WITH {SIGMA1;}

というマクロが定義されているとする。.

SIGMA (1);	→	**SIGMA1;**
SIGMA(1);	→	**SIGMA1 ;**

“pattern”部のブランク

- “pattern” 部のブランクには意味がある。.

  REPLACE {DUMP X;} WITH {OUTPUT X; (F10.2);}

  というマクロが定義されたとする。

DUMP X;	→	**OUTPUT X; (F10.2);**
Y=DUMPX;	→	**Y=DUMPX;**

引数

- “pattern”部の引数－－ #
  “replace”部の引数－－ {P1}, {P2}, etc. .
  - {P1} は、最初の引数、{P2}は、2番目の引数

REPLACE {PLUS #;} WITH {{P1}={P1}+1;}

というマクロが定義されているとする。

PLUS A(I,J,K);　　→　　A(I,J,K)=A(I,J,K)+1;

マクロの関する注意

- Mortran3のマクロでは、**大文字** と**小文字** は、*別な文字として扱われる。*
- EGS4で使われているほとんど全てのマクロは、**大文字*で書かれている*。**
- FORTRAN のリスト中にMortran3のマクロが残っている場合には、マクロが、**大文字で書かれているかどうかを調べる。**
- Mortranで書かれたプログラム中で、同じ“pattern”が、多重に定義された場合には、一番最後の定義が有効になる。従って、EGS4MAC で定義されているマクロをユーザーコード中で再定義すると、その再定義されたマクロがEGS4では有効になる。
- この性質を利用することにより、EGS4そのものを変えずにEGS4を変更する事ができる。
- EGS4システムで使用されているマクロは、マクロである事を示すために**先頭に$**が書かれている。

PARAMETERマクロ

REPLACE {PARAMETER #=#;} WITH { REPLACE {{P1}} WITH {{P2}}

- この例の様に、“replacement”中にマクロを使用する事ができる。
- A PARAMETERマクロは、DIMENSION等の引数を定義するのに便利である。
- PARAMETERマクロ中の値を変更する事により、全ての引数を変更する事ができる。

PARAMETER **$NBATCH=5;**

PARAMETER **$NDET=11;**

……

……

COMMON/TOTALS/PHEI(**$NDET**,150),TRLG(**$NDET**,150),DELE;

……

COMINマクロ

REPLACE {;COMIN/#,#/;} WITH {;COMIN/{P1}/;COMIN/{P2}/;}

- 次のような文字列が見つかると

  ;COMIN/BOUNDS,EPCONT,STACK/;

  以下の文字列に変換される。

  ;COMIN/BOUNDS/; COMIN/EPCONT,STACK/;

  これは、更に以下の文字列に展開される。

  ;COMIN/BOUNDS/; COMIN/EPCONT/; COMIN/STACK/;

COMINマクロ

- 個々のCOMINは、個々のCOMMONを定義したマクロにより、COMMON文に置き換えられる。

REPLACE {;COMIN/BOUNDS/;} WITH

{;COMMON/BOUNDS/PCUT($MXREG),PCUT($MXREG),VACDST;}

- ユーザーコードでCOMINマクロを使用する事により、ユーザーコードとEGS4の間で情報の交換を行う事が可能になる。
- COMINマクロは、メインプログラムとサブプログラムで使用される名前付きCOMMONを定義するのに便利である。

```
PARAMETER $NDET=10;
REPLACE {;COMIN/TOTALS/;} WITH
    {;COMMON/TOTALS/PHEI($NDET,150),TRLG($NDET,150),DELE;}
```

;COMIN/BOUNDS,EPCONT,TOTALS,USEFUL/;

→

COMMON/BOUNDS/ECUT,……..

COMMON/EPCONT/EDEP,……

COMMON/TOTALS/PHEI(10,150),TRLG(10,150),DELE

COMMON/USEFUL/MEDIUM,…….

$RANDOMSETマクロ

- $RANDOMSETマクロは、COMIN/RANDOM/;と共に使用される。
- SLAC RAN6

```
REPLACE {;COMIN¥RANDOM;} WITH {;COMMON/RANDOM/IXX;}
REPLACE {$RANDOMSET #;} WITH {IXX=IXX*663608941;
        {P1}=0.5+IXX*0.23283064E-9;}
```

$RANDOMSET RN;　”Sample RN uniformly on (0,1)”

PHI=TWOPI*RN;　　”Obtain azimuthal angle”

IXX=IXX*663608491

RN=0.5*IXX*0.23283064E-9

PHI=TWOPI*RN

Marsaglia-Zaman乱数

- Marsaglia-Zaman乱数(Ranmar乱数)も使えるようになっている。
- この乱数は、周期が非常長く、全ての32ビット計算機で使用できる。
- Marsaglia-Zaman乱数用の$RANDOMSETマクロは、もっと複雑である。
- 詳細は、KEKから配布しているサンプルユーザーコードを参照。

デバッグのためのマクロ

- エラーの理由を見つけるために、その値が計算される都度出力するマクロとして以下のマクロがある。
  - **$TRACE;** 引数を持たない実数
  - **$S1TRACE;** 1つの引数を持つ実数
  - **$S2TRACE;** 2つの引数を持つ実数
  - **$ITRACE;** 引数を持たない整数
  - **$IS1TRACE;** 1つの引数を持つ整数
  - **$IS2TRACE;** 2つの引数を持つ整数
  - **$CALL TRACE;** SUBROUTINE名をそれがCALLEされるたびに出力する。. “label”直後のCALLには適用されない。
- メインプログラムの**COMIN** に**DEBUG**が含まれていない場合には追加し、**加え、**実行文中に**QDEBUG=.TRUE.;** を挿入すれば、これらのマクロは有効になる。

MortranあるいはEGS4実行中のエラー

- Mortran中のエラーの原因が判らない場合
  - !TRACE 2; と!TRACE1 ; をエラーを起こしている文の数行前及び後に挿入する。
  - !TRACE2 と!TRACE1の間にマクロがあると、どの様に変換されたかという事の詳細が出力される。(mortjob.mlt[PC], mortjob.list[Unix])

!TRACE1;

…….

…….

$TRACE 2;

$SET INTERVAL GLE, GE;

$EVALUATE GMFPR1 USING GMFP(GLE);

!TRACE1;

0 !TRACE 2;

REDUCED ;

0 $SET INTERVAL GLE,GE;

MATCHED $SETINTERVALp,p;

ARG 1 GLE

ARG 2 GE

REDUCED LGLE=GE1(MEDIUM)*GLE+GE0(MEDIUM);

0 $EVALUATE GMFPR1 USING GMFP(GLE);

MATCHED $EVALUATEpUSINGp(p);

ARG 1 GMFPR1

ARG 2 GMFP

ARG 3 GLE

REDUCED
GMFPR1=GMFP1(LGLE,MEDIUM)*GLE+GMFP0(LGLE,MEDIUM);

0 !TRACE 1;

無限LOOP の原因を見つける(SLAC RAN6の場合)

- 何かの理由で、プログラムがいつまでも終了しない場合
  - OUTPUT IXX;(‘ IXXST=‘,I12); をCALL SHOWERの前に挿入
  - OPEN(6, file=’……..)をコメントにする。
  - プログラムを走らせ、表示された最後の IXXを見つける。
  - IXXSTを上記の値に変え、ヒストリー数を1に変える。
  - $TRACE, $S1TRACE *etc.* のマクロを使い、位置、エネルギー、方向、リージョン数を出力するように定義する。
  - メインのCOMINにDEBUGが含まれてない場合には追加する。
  - 実行文中に、 QDEBUG=.TRUE.; を挿入する。
  - プログラムを実行する。
  - 結果を調べる。