プログラムのページ

担当　吉　澤　　正

7001. 算術式の計算および分解について

藤崎哲之助（東京大学・工学部・計数工学科）

算術式の分解やその値の計算は，compiler や interpreter を作成するときに必要であるが，ここで示すプログラムは後者，すなわち，算術式をインプットとして，その値を計算するものである．プログラムは PL/1 により書かれており，その recursive call と割込み機能を利用している．

1. ここで扱う算術式の定義

〈表現〉⇨ とは，以下コロンまでに〈 〉内の説明が行なわれることを示す．

$\left\{\begin{matrix}\text{説明 1}\\ \text{説明 2}\\ \vdots\\ \text{説明 } n\end{matrix}\right\}$ とは（説明 1〜説明 n）の 1 つを選ぶことを示す．以上の約束を使えば

〈a. e.〉⇨算術式：

〈a. e.〉⇨$\left\{\begin{matrix}\text{a. e.}\left\{\begin{matrix}+\\-\end{matrix}\right\}\text{term}\\ \text{term}\\ \text{nil}\end{matrix}\right\}$：

〈term〉⇨$\left\{\begin{matrix}\text{term}\left\{\begin{matrix}*\\/\end{matrix}\right\}\text{factor}\\ \text{factor}\end{matrix}\right\}$：

〈nil〉⇨何もないこと：

〈factor〉⇨$\left\{\begin{matrix}\text{factor}\ **\ \text{primary}\\ \text{primary}\end{matrix}\right\}$：

〈primary〉⇨$\left\{\begin{matrix}\text{identifier}\\ \text{constant}\\ \text{(a. e.)}\end{matrix}\right\}$：

〈constant〉⇨$\left\{\begin{matrix}\text{基本実定数}\\ \text{整定数}\end{matrix}\right\}$†：

〈identifier〉⇨ すでに値を持っている変数名で，英字と数字の組合せ．ただし，最初は英字：

＋，－，＊，／，＊＊は算術演算子であり，その意味については，JIS FORTRAN 入門(上)森口繁一著 4.5 節を参照のこと．

† JIS FORTRAN 入門(上) 森口繁一著（東大出版会）の 4.2 節 77 ページ参照のこと

2. プログラムの説明

プログラムは 2 つの procedure から成っている．

（a） **INEXP**　準備のための procedure であり，その機能は，算術式に現われる変数の表を計算機内に作ること，および算術式をカードから読み込み，それを引数として proceduer EVALUATE を呼び，返ってきた値を印刷することである．

したがって，実際に interpreter に組み込むときには必要としない．

（b） **EVALUATE**　引数のある function procedure である．その機能は，引数により与えられた算術式を分解し，最終的に値を計算する．

ここでの引数は長さ 1 の character string の array であり，そこに算術式の一字一字が空白をつめてはい

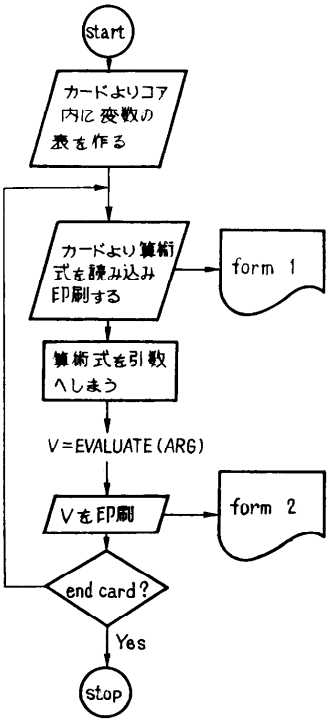

図 1　INEXP の流れ図

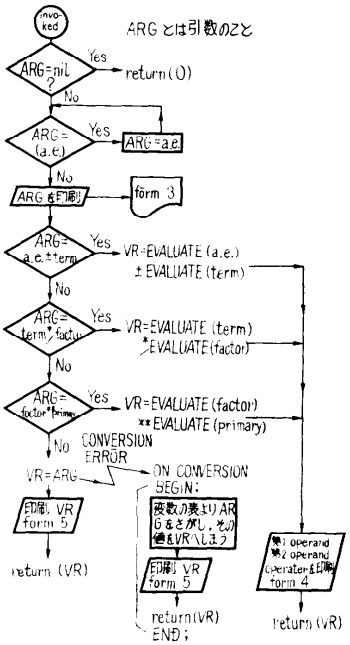

図 2 EVALUATE の流れ図

```
      T_PARA.FWD_CHAIN=NULL;
      T_PARA.NAME=NAME8;
      T_PARA.VALUE=NUM;
/*                                        */
/*    TABLE IS MADE FOR 1 STEP
                                          */
      IF C1=';' THEN FLOW=2;
    GOTO GET;
/*                                        */
LAB(2):
/*  PRINT FORM-1  */
    PUT EDIT('*** EXPRESSION TO BE EVALUATE IS ',WS)
              (SKIP(3),A,SKIP,A);
IN:
    BEGIN;
  /* THIS BLOCK IS MADE IN ORDER TO ADJUST THE LENGTH OF  THE
     ARGUMENT                             */
       DECLARE ARG(LENGTH(WS)) CHARACTER(1);
       DO INS=1 TO LENGTH(WS);
         ARG(INS)=SUBSTR(WS,INS,1);
       END;
       WS=NIL;
       ANSWER=EVALUATE(ARG);
    /*  THIS FUNCTION IS INVOKED RECURSIVELY  */
/*  PRINT FORM-2  */
       PUT EDIT('RETURNED VALUE IS ',ANSWER)
                (SKIP(2),COLUMN(10),A,SKIP,COLUMN(10),F(20,5));
    END IN;
    IF C1¬=';' THEN GOTO GET;
               ELSE STOP;
EVALUATE:
/*
  THIS ROUTINE EVALUATES AN EXPRESSION AND RETURNS A VALUE OF IT.

                PROCEDURE(TERM)     RECURSIVE  BIN FLOAT(53);

/*  *ARGUMENT*        IS AN ARRAY OF CHARACTER(1)
                      ITS BOUND IS DETERMINED BY THE PROCEDURE
                      WHICH CALLED THIS PROCEDURE.

                DECLARE  (V1,V2)    BINARY    FLOAT(53),
                         TERM(*)    CHARACTER(1), /*ARGUMENT*/
                         OPER(5) CHARACTER(1) STATIC
                                    INITIAL('+','-','*','/','**'),
                         AC         CHARACTER(1)  STATIC,
                         XLAB(1:5) LABEL,
                         (I,J,NL)  BINARY FIXED(15,0);
```

プログラムリスト(2)

```
INEXP:
    PROCEDURE OPTIONS(MAIN);
    DECLARE
          NIL       CHARACTER(0),
          WS        CHARACTER(500)  VARYING,
          NAME8     CHARACTER(8),
          C1        CHARACTER(1),
          LAB(2)    LABEL,
          NUM       BINARY FLOAT(53),
          FLOW      BINARY FIXED(7,0),
          (PTRPARA, TOP@PARA, VOTUM@PARA)  POINTER,
          1 T_PARA   BASED(PTRPARA),
            2 NAME    CHARACTER(8),
            2 VALUE   BINARY FLOAT(53),
            2 FWD_CHAIN          POINTER,
          ANSWER    BINARY FLOAT(53);
/*   STARTING POINT                    */
    TOP@PARA,VOTUM@PARA=NULL;
    FLOW=1;
    WS=NIL;
 GET:
    GET EDIT(C1)(A(1));
    IF C1=' ' THEN GO TO GET;
    IF C1='*' THEN
/* VARIABLE TABLE MAKING  */
      DO;
         NAME8=WS;
         WS=NIL;
         GOTO GET;
      END;
    IF C1¬=';' & C1¬='.' THEN
      DO;
         WS=WS||C1;
         GOTO GET;
      END;
    GOTO LAB(FLOW);
LAB(1):
    NUM=WS;
    WS=NIL;
/*    */
/*   STORE THE NAME AND VALUE INTO T_PARA TABLE      */
    ALLOCATE T_PARA SET(PTRPARA);  /* PUSH DOWN  */
    IF TOP@PARA=NULL  THEN
                      TOP@PARA=PTRPARA;
                 ELSE
                      VOTUM@PARA->T_PARA.FWD_CHAIN=PTRPARA;
    VOTUM@PARA=PTRPARA;
```

プログラムリスト(1)

```
                      NAME8     CHAR(8)  STATIC VARYING,
                      OPERATER  CHARACTER(2)   VARYING  STATIC;
                NL=HBOUND(TERM,1);
                IF NL=0   THEN
/* ARITHMETIC EXPRESSION IS NIL.
                   DO;
                   V1=0;
                   RETURN(V1);
                   END;
/*   PRINT FORM-3    */
                PUT EDIT('CALLED','ARGUMENT IS  ',(TERM(K) DO K=1
                         TO NL))
                        (SKIP(1),A,SKIP,X(3),A,(NL)(A(1)));
                K=0;
                IS=1;
PUT_OFF:
                IF TERM(IS)¬='('  THEN GOTO STEP&1;
                IF TERM(NL)¬=')'  THEN GOTO STEP&1;
                DO I=IS+1 TO NL-1;
                   AC=TERM(I);
                   IF AC='(' THEN K=K+1;
                ELSE IF AC=')' THEN
                   IF K =0   THEN GO TO STEP&1;
                ELSE K=K-1;
                END;
/*   TERM IS FOUND TO BE INCLUDED IN PARENTHESIS PAIR.
     LET'S   CANCEL  THEM.
                IF  K ¬=0   THEN GO TO ERRP;
                IS=IS+1;
                NL=NL-1;
                GOTO PUT_OFF;
STEP&1:
          LOOP1:  DO  I= 1 TO 5;
                     DO J=NL TO IS BY -1;
                        AC=TERM(J);
                        IF AC=')'    THEN K=K+1;
                     ELSE  IF AC='('   THEN K=K-1;
                     ELSE  IF K=0    THEN ;
                     ELSE  IF AC¬=OPER(I) THEN ;
                     ELSE  IF I ¬= 3 THEN GOTO DECOMPOSITION;
                     ELSE  IF TERM(J+1)='*' THEN ;
                     ELSE  IF J  = 1 THEN ;
                     ELSE  IF TERM(J-1)='*' THEN ;
                     ELSE  GOTO  DECOMPOSITION;
                  END  LOOP1;
 ON CONVERSION
                  BEGIN;
```

プログラムリスト(3)

```
                    FIND=TOP#PARA;
          AGAIN:    IF FIND = NULL THEN GOTO ERRP1;
                    IF FIND -> T_PARA.NAME ¬= NAME8 THEN
                         DO;
                         FIND=FIND->T_PARA.FWD_CHAIN;
                         GO TO AGAIN;
                         END;
                    V1=FIND -> T_PARA.VALUE;
/*   PRINT FORM-4    */
                    PUT EDIT(V1)(SKIP,COLUMN(15),F(20,5));
                    GO TO LOGOUT;
     END;
/*   TERM IS FOUND TO BE AN ATOM.
     NOW LET'S FIND THE VALUE AND RETURN

                  NAME8=NIL;
                  DO I=IS TO NL;
                  NAME8=NAME8||TERM(I);
                  END;
                  V1=NAME8;
/*   CONVERSION ERROR MUST BE SAVED BY ON UNIT
/*   PRINT FORM-4    */
                  PUT EDIT(V1)(COLUMN(15),F(20,5)) SKIP;
LOGOUT:
                  RETURN(V1);
DECOMPOSITION:
                 OPERATER=OPER(I);
                  IF I=5  THEN OPERATER=OPERATER||'*';
/*   PRINT FORM-5    */
                  PUT EDIT((TERM(K)  DO K=IS TO J-1-FLOOR(0.22*I)),
                           (TERM(K)  DO K=J+1 TO NL),
                                OPERATER)
                          (SKIP,
                           COLUMN(15),(J-IS-FLOOR(0.22*I))(A(1)),SKI
                          ,COLUMN(15),
                          (NL-J) (A(1)), SKIP,
                          COLUMN(15),A);
```

プログラムリスト(4)

```
                  BEGIN;
                       DECLARE ARG(J-IS-FLOOR(0.22*I)) CHAR(1);
                            DO KK=1 TO J-IS-FLOOR(0.22*I)
                            WHILE(J-IS-FLOOR(0.22*I)>0);
                            ARG(KK)=TERM(IS+KK-1);
                            END;
                       V1=EVALUATE(ARG);
                  END;
                  BEGIN;
                       DECLARE ARG(NL-J) CHAR(1);
                            DO KK =1 TO NL-J;
                            ARG(KK)=TERM(J+KK);
                            END;
                       V2=EVALUATE(ARG);
                  END;
/*
                  GOTO XLAB(I);
XLAB(1):          RETURN(V1+V2);
XLAB(2):          RETURN(V1-V2);
XLAB(3):          RETURN(V1*V2);
XLAB(4):          RETURN(V1/V2);
XLAB(5):          RETURN(V1**V2);
/*
/*
ERRP1:            PUT EDIT('UNRECOGNIZABLE TERM',NAME8)
                        ( COLUMN(15),A,  X(10),A);
                         V1=0;
                         GOTO LOGOUT;
ERRP:             PUT EDIT('THE TOTAL OF OPEN AND CLOSE PARENTHESIS',
                         ' DOES NOT COINSIDE.')
                        ( COLUMN(15),A,A);
                  V1=0;
                  GOTO LOGOUT;
END EVALUATE;
 END INEXP;
```

プログラムリスト(5)

るようになっている．また，array の長さは算術式の長さに一致するようになっている．

(a)と(b)の流れ図をそれぞれ**図 1**，**図 2** に示す．

3. 結果の見方

計算の結果は，わかりやすいように，分解の様子なども印刷させた．形式としては，つぎの 4 とおりがある．

(a) form-1.
(b) form-2.
(c) form-3 と form-4 との組合せ．
(d) form-3 と form-5 との組合せ．

・form-1 は**図 3** のようなもので，算術式 1 が計算させるべき式であることを示す．

```
*** EXPRESSION TO BE EVALUATED IS
(((Y+XX)/B**(-49+C))+4.3*(1.1+B))/C+5
CALLED
   ARGUMENT IS  (((Y+XX)/B**(-49+C))+4.3*(1.1+B))/C+5
                (((Y+XX)/B**(-49+C))+4.3*(1.1+B))/C
                5
                +
CALLED
   ARGUMENT IS  (((Y+XX)/B**(-49+C))+4.3*(1.1+B))/C
                (((Y+XX)/B**(-49+C))+4.3*(1.1+B))
                C
                /
CALLED
   ARGUMENT IS  (((Y+XX)/B**(-49+C))+4.3*(1.1+B))
                (((Y+XX)/B**(-49+C))
                4.3*(1.1+B)
                +
CALLED
   ARGUMENT IS  (((Y+XX)/B**(-49+C)))
                ((Y+XX)
                B**(-49+C)
                /
CALLED
   ARGUMENT IS  (Y+XX)
                Y
                XX
                +
CALLED
   ARGUMENT IS  Y
                                    41.30000
CALLED
   ARGUMENT IS  XX
                                   -42.50000
CALLED
   ARGUMENT IS  B**(-49+C)
                B
                (-49+C)
                **
CALLED
   ARGUMENT IS  B
                                     2.30000
CALLED
   ARGUMENT IS  (-49+C)
                -49
                C
                +
CALLED
   ARGUMENT IS  -49

                49
                -
CALLED
   ARGUMENT IS  49
                                    49.00000
CALLED
   ARGUMENT IS  C
                                    50.00000
CALLED
   ARGUMENT IS  4.3*(1.1+B)
                4.3

                (1.1+B)
                *
CALLED
   ARGUMENT IS  4.3
                                     4.30000
CALLED
   ARGUMENT IS  (1.1+B)
                1.1
                B
                +
CALLED
   ARGUMENT IS  1.1
                                     1.10000
CALLED
   ARGUMENT IS  B
                                     2.30000
CALLED
   ARGUMENT IS  C
                                    50.00000
CALLED
   ARGUMENT IS  5
                                     5.00000

         RETURNED VALUE IS
                   5.28196
```

結　　果

*** EXPRESSION TO BE EVALUATED IS 算術式 1

図 3

・form-2 は**図 4** のようなもので，算術式 1 の最終的な値を示す．

RETURNED VALUE IS 値

図 4

・form-3 は**図 5** のようなもので，procedure EVALUATE が算術式 2 を引数として呼ばれたことを

示す.

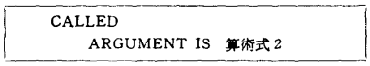

図 5

・form-4 は図**6**のようなもので，算術式2が算術式3，算術式4および演算子に分解されたことを示す.

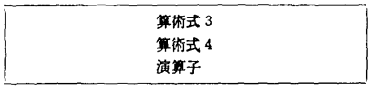

図 6

・form-5 は図**7**のようなもので，算術式2が identifier か constant であったとき，その値を示す.

図 7

ここに示したプログラムは，interpreter 型で算術式の値を計算するが，分解するときにグローバルな stack に積み上げれば，compiler 型のプログラムにもすることができる.

このプログラムのテストは，電力中央研究所の協力を得て，IBM/360 Model 75 と PL/1 (F)Compiler Version 4 Release 16 を使用して行なった.

(昭和44年10月27日受付)