プログラムのページ

担当　吉　澤　　正

7002 FORTRAN による再帰的呼出し

牛島　和夫（九州大学工学部）

はじめに　FORTRAN では，再帰的呼出しは許されていないが，われわれにはプログラムのいろいろな場面で，再帰的呼出しが必要となる場合が多い．LISP[1]などのプログラムシステムを自由に使うことができれば，問題はかなり減るのであるが，わが国では残念ながらまだこれらのシステムは，簡単には利用できないのが現状である．さらに，LISP では不十分である FORTRAN の持っている強力な数値計算機能もあわせて十分に利用したいという要求もある．そこで FORTRAN の範囲内で再帰的呼出しのプログラムを記述する方法を工夫してみた（工夫することを余儀なくされたといった方が正しいかもしれない）．FORTRAN を基礎にした言語で再帰的呼出しを許しているものはほかに，SLIP[2][3]があるが，これは，アセンブラー言語で書いた VISIT や TERM といった副プログラムを利用しなければならない．ここでは，このような補助的なルーチンも含めて，すべて FORTRAN 語の範囲内で行なう．

FORTRAN の制限と再帰的呼出しの記述　再帰的呼出しの副プログラムを記述（または実行）する際の問題点は，つぎのようなものである．

A1. FORTRAN では呼び出された副プログラムのもどり先が固定されているので，もし，もどらないうちにもどり先が変更されると復原できない．

A2. 副プログラムの中にその副プログラムを呼び出す文を書くことが許されない（同じ名前を別の目的に使用しているというエラーメッセージが出るであろう）．

A3. A2 の帰結として，パラメタあるいは中間結果の引渡しに従来の形式をとれない．

そこで，ここでは，FORTRAN で再帰的呼出しのプログラムが記述できればよいとして，FORTRAN 語の読みやすさは多少犠牲になることもある．再帰的呼出しの記述のために，つぎのような用意と約束をする．

B1. 再帰的呼出しを可能にするために，push down stack を整数型配列としてソフト的に用意する．

B2. push down stack を処理する2つのサブルチンPDOWN (IPARAM, N) および POPUP (IPARAM, N) を定義する．ここで IPARAM は整数型配列，N は整数を示す．

PDOWN は IPARAM にある N 個のパラメタを push down stack に push down するサブルチン，POPUP は push down stack の上から N 個までの内容を IPARAM にとり出すサブルチン．

B3. 再帰的呼出しを記述するプログラム単位内に push down stack の top(N 個) として，整数型配列を定義しておき，再帰的呼出しに用いるパラメタ，中間結果などを示す変数をそれぞれ EQUIVALENCE 文で結合しておくと，あとのプログラムの記述が容易になる．

B4. 再帰的呼出しをもった副プログラムの記述は（a）再帰的呼出しのための宣言部，（b）再帰的呼出しのための準備，（c）再帰的プログラムの本体，（d）副プログラムからの脱出，の4つの部からなり，その詳

第1表　再帰的呼出しの記述

<副プログラムの入口>	(a)
パラメタ受渡し用配列の宣言	
上記の配列の内容を示す EQUIVALENCE 文	
パラメタのうつしかえ	(b)
<副プログラムの出口>を≪もどり先≫に設定して push down	
GO TO<副プログラムの本体の入口>	
<副プログラムの本体の入口>	(c)
本体の計算	
旧パラメタ，中間結果の push down	再帰的呼出し
<もどり先1>を≪もどり先≫に設定して push down	
新パラメタの設定	
GO TO<副プログラムの本体の入口>	
<もどり先1>	
本体の計算（つづき）	
GO TO<副プログラムの本体の出口>	
<副プログラムの本体の出口>	(d)
旧パラメタ，中間結果の pupup	
≪もどり先≫の popup	
GO TO≪もどり先≫	
<副プログラムの出口>	
RETURN	

注）（a）再帰的呼出しのための宣言部，（b）再帰的呼出しのための準備，（c）再帰的プログラムの本体，（d）副プログラムからの脱出，＜　＞は文の番号を意味する．

```
      FUNCTION NFACT(N)
      DIMENSION L(2)
      EQUIVALENCE(L(1),K),(L(2),IN)
      ASSIGN 3000 TO K
      CALL PDOWN(L,2)
      IN=N
C     NORMAL ENTRY
 1000 CONTINUE
      IF(IN.EQ.1) GO TO 1
      ASSIGN 2 TO K
      CALL PDOWN(L,2)
      IN=IN-1
      GO TO 1000
    2 CONTINUE
      NFACT=NFACT*IN
      GO TO 2000
    1 NFACT=1
      GO TO 2000
C     EXIT OF RECURSIVE FUNCTION
 2000 CONTINUE
      CALL POPUP(L,2)
      GO TO K,(2,3000)
 3000 RETURN
      END
```

(a) (b) (c) (d)

第1図 $n!$ のプログラム例

細は**第1表**のとうりである．ここで< >は文の番号を意味する．

B5. B4 の約束からわかるように，再帰的呼出し

```
      FUNCTION IDIF (IFX, IATOMX)
      COMMON/CONST/IZERO,IONE,ITWO
      DIMENSION IPARAM(3)
      EQUIVALENCE(IPARAM(1),IADDRS),(IPARAM(2),IFUNC),(IPARAM(3),IFDASH)
C     INITIALIZATION TO CALL THE RECURSIVE FUNCTION
      ASSIGN 3000 TO IADDRS
      CALL PDOWN( IPARAM, 3)
      IFUNC = IFX
C     NORMAL ENTRY
 1000 CONTINUE
      K = LHEAD(IFUNC)
      IF(K) 1, 2, 1
    1 IF(IFCONS(IFUNC,IATOMX)) 5, 3, 5
C     ATOM OR CONSTANT
    2 IF(IFUNC - IATOMX) 3,4,  3
C     CONSTANT
    3 IDIF = IZERO
      GO TO 2000
C     ATOM
    4 IDIF = IONE
      GO TO 2000
    5 ASSIGN 6 TO IADDRS
      CALL PDOWN(IPARAM ,3)
      IFUNC = LFIRST(IFUNC)
      GO TO 1000
    6 IFDASH = IDIF
      K  = LHEAD(IFUNC)
      GO TO (10,15,15,40,10,15,15,10), K
   10 CALL ERRORN(1)
   15 ASSIGN 80 TO IADDRS
      CALL PDOWN(IPARAM, 3)
      IFUNC = LSECND(IFUNC)
      GO TO 1000
   80 JF = LFIRST(IFUNC)
      JG = LSECND(IFUNC)
      K  = LHEAD(IFUNC)
      GO TO(90,20,30,90,90,60,70,90), K
   90 CALL ERRORN(2)
C     DIF(F*G) = F*DIF(G)+DIF(F)*G
   20 IDIF = ITREE(6,ITREE(2,JF,IDIF),ITREE(2,IFDASH,JG))
      GO TO 2000
C     DIF(F/G) = (DIF(F)*G-F*DIF(G))/(G*G)
   30 IDIF = ITREE(3,ITREE(7,ITREE(2,IFDASH,JG),ITREE(2,JF,IDIF)),
     1                ITREE(2,JG,JG))
      GO TO 2000
C     DIF(-F) = -DIF(F)
   40 IDIF = ITREE(4, IDIF,0)
      GO TO 2000
C     DIF(F+-G) = DIF(F)+-DIF(G)
   60 CONTINUE
   70 IDIF = ITREE(K,IFDASH,IDIF)
      GO TO 2000
C     EXIT OF THE RECURSIVE FUNCTION
 2000 CONTINUE
      CALL POPUP(IPARAM, 3)
      GO TO IADDRS, (3000, 6, 80 )
 3000 RETURN
      END
```

第2図 数式微分のプログラム例

第2表 NFACT(4)の実行による変遷

文の番号	NFACT	IN	push down stack
1000		4	(¥3000,)
1000		3	(¥ 2, 4)(¥3000,)
1000		2	(¥ 2, 3)(¥ 2, 4)(¥3000,)
1000		1	(¥ 2, 2)(¥ 2, 3)(¥ 2, 4)(¥3000,)
2	1	2	(¥ 2, 3)(¥ 2, 4)(¥3000,)
2	2	3	(¥ 2, 4)(¥3000,)
2	6	4	(¥3000,)
3000	24		空

¥××で stack された文の番号を示す．

の副プログラム（Pとする）の記述の中で，呼び出された別の副プログラムの記述の中にPを呼び出す結果となる記述が含まれてはならない．これは，A1 の制限によって，Pのはじめのもどり先の復原が不可能になるからである．なお，この制約は，アセンブラの副プログラムを用いている SLIP でも同様である．

第1図に factorial[n]::=[n=1→1, T→n×factorial[$n-1$] を B の約束に従って書いた FUNCTION NFACT(N) を示す．**第2表**に NFACT(4) の実行の際の変数，push down stack の内容の変遷を示した．

数式微分のプログラム **第2図**に B1～5 の約束に従って書いた数式微分のプログラムを示す．ここで IFX は被微分<式>(expression)へのポインタ IATOMX は，微分<変数>へのポインタ，return value は，微分結果の<式>へのポインタである．簡単のため，<式>は<変数>，または<定数>，または<式>を，2項演算子*／+－と単項演算子－で結合したものに限定している．記憶装置内における<式>の構造は，つぎのような3つ組方式である．すなわち．2項演算子については（2項演算子，第1被演算子へのポインタ，第2被演算子へのポインタ），単項演算子については(単項演算子，被演算子へのポインタ，空）としている．

演算子はそれぞれつぎのようにコード化している．

*	2
／	3
－(単項)	4

```
-(1X/(X+A)))                                    ・・・与　式

-(1((X+A)-X)/((X+A)*(X+A))))                    ・・・微分の結果
```

第3図　数式微分の結果

```
+        6
-(2項)   7
```

また，FUNCTION IDIF 中に使われている FORTRAN 関数は，つぎのような機能を持っている．

LHEAD(IFUNC)　<式>IFUNCの演算子をとり出す．

LFIRST(IFUNC)　<式>IFUNCの第1被演算子へのポインタをとり出す．

LSECND(IFUNC)　<式>IFUNC の第2被演算子へのポインタをとり出す．

IFCONS(IFUNC, IATOMX)　<式>IFUNC の中に<変数>IATOMX が含まれているか否かをしらべる関数．含まれていれば IATOMX，含まれていなければ0をそれぞれ値として返す．

ITREE(K, IFIRST, ISECND)　K は演算子，IFIRST は第1被演算子へのポインタ，ISECND は第2被演算子へのポインタを合わせて3つ組を構成し，3つ組へのポインタを返す．

ERRORN(・)　エラーメッセージの出力ルーチン．

そのほか，ラベル付共通領域の IZERO, IONE, ITWO は，それぞれ<定数>0, 1, 2 へのポインタである．IDIF の結果を**第3図**に示す．

考察と注意

1. パラメタの受け渡しに用いている配列（第1図の L，第2図のIPARAM）を整数にしたのは，push down stack を整数配列にしているためである．型が異なれば，SUBROUTINE PDOWN または POPUP 中の代入文において，暗黙の型変換が行なわれるおそれがある．パラメタ，中間結果などは，実数，倍精度などでもかまわない．これらは，EQUIVALENCE 文で解消される．

2. 本来もどり先番地(第1図K, 第2図IADDRS)とその他のパラメタとは同じ push down stack を用いず，別にする方がすっきりするが（たとえば，（b）の部分で，undefined value を形式的に push down するようなことをさけることができる），push down stack を処理するサブルチンを呼び出す回数が増し，したがって，パラメタの受け渡しの回数が増し，計算時間の増大になることをさけた．

3. B1～5 のような約束は，再帰的呼出しのプログラム単位を，利用だけする人には必要ないが，push down stack の用意は初期設定しなければならないので，初期設定用のサブルチンもあらかじめ用意しておいてやるとよい．

4. push down stack を用意する際にその深さが問題になる．第3図の場合には<式>の深さだけ用意があればよいが，第2図では引数 N の大きさだけ必要である．筆者はこれを DYSTAL[4] (Dynamic Storage Allocation Langnage in FORTRAN) によった．DYSTAL は，すべて FORTRAN 語によって書かれた副プログラムからなり，共通領域に大きくとった配列 LOT を動的に制御するプログラムパッケージである．したがって，ここでは LOT の自由領域がつきないかぎり，いくらでも深くとることができる．どのような push down stack を用意するにしろ，B 1～5 の約束はそのままで，PDOWN, POPUP の処理内容が多少変わってくる．

5. 再帰的呼出しのパラメータは，push down stack に push down できるものに限られる．すなわち，ALGOL でいう値とりのパラメタでなければならない．したがって，配列の名，副プログラムの名などは，再帰的呼出しに直接関係あるパラメタとしては使えない．これらのパラメタは FORTRAN 語のレベルからは手がとどかないプロセッサが設定するものである．この制限は実用上問題にならない（SLIP でも同様な制限になる）．

おわりに　PL/1 には再帰的呼出しが規定してあるが，使用経験がないので言及しなかった．プログラムの作製は九州大学大型計算機センターの FACOM 230-60 FORTRAN による．なお，DYSTAL の implementation も同様に終わっている．

参考文献

1) McArthy, J.: LISP 1.5 Programmer's Manual, MIT Press (1962), pp. 106.
2) Weizenbaum, J.: Symmetric List Processor, CACM **6**, 9 (1962), p. 524.
3) Smith, D. K.: An Introduction to the List-Processing Language SLIP, Programming Systems and Languages (ed. Rosen,S.) (1967), p. 363.
4) Sakoda, J. M.: DYSTAL Manual Dynamic Storage Allocation Lauguage in FORTRAN Brown Univ. (1965), pp. 308.

（昭和44年10月3日受付）