寄　　書

FORTRAN 語の拡張と互換性について*

—— 再帰的呼出しの場合 ——

牛　島　和　夫**

1. はじめに

最近，九州大学大型計算機センターを利用して，FACOM 230-60 に SLIP[1]を implement する作業を行なった．よく知られているように SLIP は，FORTRAN の副プログラム群からなり，FORTRAN 語の拡張という形式をとっているので，implement する際に FORTRAN の互換性について当初予期[2]しなかった問題に二，三遭遇した．SLIP のような特殊目的の言語を開発する際に，今後も基礎となる言語として FORTRAN が選ばれることは，多いであろうと考えられる[2)~4)]．たとい特殊目的といっても，このような言語はできるだけ多くの機種に implement できることが望ましいので，筆者の出会った互換性の問題を紹介して，今後同じような作業を行なう際の参考に供したい．さらに，FORTRAN 処理系製作者にとって，なんらかの作製上のヒントになれば幸いである．

2. 副プログラムの互換性

FORTRAN 語を基礎に言語を開発する場合，大きな理由は2つ考えられる．

（1） FORTRAN の汎用性・安定性．FORTRAN はほとんどの機種で使用可能であり，FORTRAN 語を知っている利用者が圧倒的に多い．さらに，デバッグ用の利便もかなり整っている．

（2） 言語の拡張機能が存在する．すなわち，外部関数の定義を多項演算子の導入とみなすことができるから，これによって，新しい機能を FORTRAN 語の基本言語に付加することができる．次例により自明であろう．

```
FUNCTION ADD (A, B)
FUNCTION SUB (A, B)
```

により

```
ADD (SUB (A, ADD(B, C)), ADD(A, A)).
```

算術 IF や論理 IF も前者は負，0，正の3値を，後者は true, false の2値を関数値とする演算子と考えると，統一的に考えることができる．

互換性について当初から予期できる注意点は，1語の精度と文字数であるが，これについてはここでは触れない．ここでは(2)の関連から副プログラムの引用に関する互換性に限る．

JIS FORTRAN 7000[5]（以下 JIS と略）の関係部分は，以下のような諸点である．

J1 外部関数の引用　外部関数は算術式または論理式中で，1次子としての関数の引用を用いることによって引用する．引数の並びに書かれている実引数は，対応する仮引数と順序，個数および型が一致しなければならない（JIS p. 34）．

J2 サブルーチンの引用　サブルーチンはCALL文により引用する．CALL文において，引数の並びを構成している実引数は，対応する仮引数と順序，個数および型が一致しなければならない．実引数として文字定数が使える（JIS p. 36）．

J3 プログラム単位内の英数字名は，つぎのように分類される．——中略——
第V類　外部関数
第VI類　サブルーチン
——以下略——（JIS p. 38）

J4 式の評価は，その式の構成順序に従って行なわれるものとする．一つの式の構成規則に従って構成していく順序はいろいろありうるが，式の評価は，原則としてどの順序に従ってもよいものとする．——中略——式の中に現われる関数を評価するときに，その式やその式を含む代入文やCALL文のほかの要素の値を変更してはならない（JIS p. 10）

J5 一つの文の中で関数副プログラムが2回以上同じ引数の並びで引用される場合には，その副プロ

*Extension of FORTRAN Capabilities and its Compatibility, by Kazuo USHIJIMA (Faculty of Engineering, Kyushu University)

**九州大学・工学部

表 1

項　　目	JIS	F	H	N	備　　考
1. 引数の型の一致（整と実）	J 1, J 2	検査していない	検査していない	検査していない	
2. 引数の個数の一致	J 1, J 2	実行時に検出し，不一致の場合は，実行を打ち切る	検査していない†	検査していない†	
3. 関数をＣＡＬＬ文で呼び出す	文法の範囲外	実行時に検出し，以下実行を打ち切る	検査していない	検査していない	
4. サブルーチンを関数の１次子とする	同　　上	同　　上	同上，実行時Ａレジスタ８番地の無意味な値を関数値とみなす	同上，実行時ＡＣＣ上の無意味な値を関数値とみなす	付録参照
5. 第Ⅴ類と第Ⅵ類の混用	J 3	コンパイル時にエラー検出	同　　左	同　　左	文献(1)では混用を認めていると思われる††
6. 外部関数引用における関数値を受け渡す場所	J 4, 処理系に任せる	Core 上の番地を指定	Ａレジスタ８番地	ＡＣＣ	付録参照
7. 実引数に式が２つ以上ある場合の引数の評価の順序	J 4, J 5	右から左へ	左から右へ	左から右へ	

(注)　† コンパイル時に同じ名前で，引数の個数の異なる呼出しを行っても，エラーも警告も発せられない．
†† 文献(1)には CALL STRDIR (STRDIR (STRDIR (0, X(I)), W(J)), W(J+1)) という文がある．

グラムの実行の結果は，その文の評価順序に関係なく同一でなければならない (JIS p. 43)．

筆者は，上記の諸点について，九州大学大型計算機センター FACOM 230-60 FORTRAN-C (F と略)[6] 東京大学大型計算機センター HITAC 5020 E/F FORTARN-HARP (H と略)[7]，および大阪大学大型計算機センター NEAC 2200-500 FORTRAN-L (N と略)[8] の各機種について，簡単なテストプログラムを作って調べた．その結果を**表 1** にまとめた．これからつぎのような点が指摘できる．

(1) JIS 文法違反であるが，プロセッサは，チェックも警告もしない．利用者側の道義的責任にゆだねられているもの．

(2) 処理系製作者の裁量に任されているため，実行上に違いをもたらすもの．

次節以下で，これらの具体例を示す．

3. 関数の引用

外部関数の定義によって導入された多項演算子について，被演算子に作用するだけで，return value に関心がない場合がある（実際には COMMON 変数などでこの目的は達している）．このときは，関数として定義されているにもかかわらず，あたかもサブルーチンのように CALL 文で用いても文法違反を問われないものが多い．

例　CALL　ADD (A, B)

この場合 H と N では，return value は A レジスタにおきざりになる．一方，F では同じ名前の副プログラムを同一プログラム単位内で CALL 文および式中の１次子として両様に使わないかぎり，第Ⅴ類または第Ⅵ類として確定するので，実行時にはパラメタの個数の不一致として検出され，以後計算の実行が打ち切られる．したがって，作用するだけで return value に関心がない場合でもダミー変数をもうけて

DUMMY＝ADD (A, B)

のような文を構成しなければならない．また，INTEGER FUNCTION の場合には，このままでは，自動的に換算関数がそう入されるので，整数のダミー変数を左辺におくようにする．

4. 再帰的呼出し

FORTRAN では，再帰的呼出しを認めていないから，とくに記号処理などを含む言語を FORTRAN を基礎に開発しようとすると，再帰的呼出しを実現する工夫が必要である[1),9)]．後者はすべて FORTRAN 語の範囲内で，再帰的呼出しを実現しているので，記述方法はかなりめんどうである．一方前者は，アセンブラ語の助けを借りているので，後者に比べればかなりスマートな記述ができる，前者の方法をとるならば，$n!$ の計算プログラムは**図 1** のようになる．ここで

W(100) は 100 個の push down stack の引出し口

PARMT $n(a_1, a_2, \cdots, a_n)$ は入力パラメタ $a_1, \cdots, a_n$ を push down stack W(1), …, W(n) に push down し，a_1 を関数値とする．

TOP(L) は push down stack L の最上段の要素の値を関数値とする．

RESTOR(n) は push down stack W(1), …, W(n) の最上段の要素をすべて pop up する．

VISIT(K, A)　この関数を評価したつぎにもどるべきもどり先番地を push down して，K に付された文番号へジャンプする．関数値はつぎの TERM 参照（アセンブラ語で書かれている）．

TERM(X, A)　VISIT で push down されたもどり先番地を pup up して，その番地へジャンプする．そのとき X をVISIT の関数値とする（アセンブラ語で書かれている）．

なお，VISIT および TERM の第2引数 A はともにダミーである．すなわち，図1の例にみるように，VISIT または TERM の実行に先だって実行しておきたい手続き（関数）を代入する用法が便利である．

さて，ここで**図1**文番号5の文は，数学的には

```
7 FACT=VISIT(NFACT,PARMT1(TOP(W(1))-1.0))*TOP(W(1))
```

と同じはずであるが，J4 から実行上は問題が生ずる．この評価順序は F と H(N は H と似ているので略)では図2のようになる．したがって，Fでは意図どうりの結果を得ることができるが（Fの実行の例については**表2**参照），Hでは，図2 (H)①で評価された結果は，いったん作業用番地に退避されるが，⑤の評価の際それをおいてきぼりにして，NFACT へジャンプしてしまうので，TERM でもどってきたときは，作業用番地の内容は保証されない．したがって，意図した結果を得ることができない．7では，FとHの立場は逆になる．これらは J4 で道義的に禁止されている副作用の一種である．ところで文番号5の文の中の TOP(W(1))の評価を，H も F も別々に行なっているのは，J5 違反である．これを考慮して，プログラム実行の最適化のために補助的に

```
A = TOP(W(1))
```

の文をそう入してやれば

```
FACT = A*VISIT(NFACT,PARMT1(A-1.0))
```

となって，F も H も評価順序は一意的になるが，VISIT でもどってきたときに A の値は保証されない．結局

```
B = VISIT(NFACT,PARMT1(TOP(W(1))-1.0))
FACT = TOP(W(1))*B
```

と分解してやれば，F, H ともに意図した結果を得る．同様に図1 5, 6 をみれば，6の FACT に5を形式的に代入することができる．しかし，実行上は VISIT と RESTOR の評価順序が交絡して，F も H も正しい結果が得られない．5の代わりに7を代入すれば，Hでは正しい結果が得られる．

引数の評価順序（表1の7）に関連して同様な例がある．

```
CALL SUBR(VISIT( . ),TOP( . ))
```

ここでは，Hは正しい結果を得，Fでは不

表 2　FACT(2.0) 実行の例 (F の場合)

実行直前の番地（文番号）	W(1) の stack	もどり先番地の stack	FACT
2	2	空	
4	2	2*	
5	1 2	2*	
4	1 2	5* 2*	
5	0 1 2	5* 2*	
4	0 1 2	5* 5* 2*	
4*	1 2	5* 5* 2*	
5*	1 2	5* 2*	1
6	2	5* 2*	1
5*	2	2*	2
6	空	2*	2
2*	空	空	2
3			2

(注)　2*, 4*, 5* は，それぞれ文番号 2, 4, 5 の文が，機械語の複数個のステップに分解された中間の番地，たとえば 4* は CALL TERM (・) を実行する直前の番地，5* は VISIT (・) に対応する TERM からもどってきて，TOP(W(1)) を乗ずる直前の番地となる．

```
      FUNCTION FACT(X)
      COMMON AVSL,W(100)
    1 ASSIGN 4 TO NFACT
    2 FACT=VISIT(NFACT,PARMT1(X))
    3 RETURN
    4 IF(TOP(W(1)).EQ.0.0) CALL TERM(1.0,RESTOR(1))
    5 FACT=TOP(W(1))*VISIT(NFACT,PARMT1(TOP(W(1))-1.0))
    6 CALL TERM(FACT,RESTOR(1))
      END
```

図 1　$n!$ のプログラム例

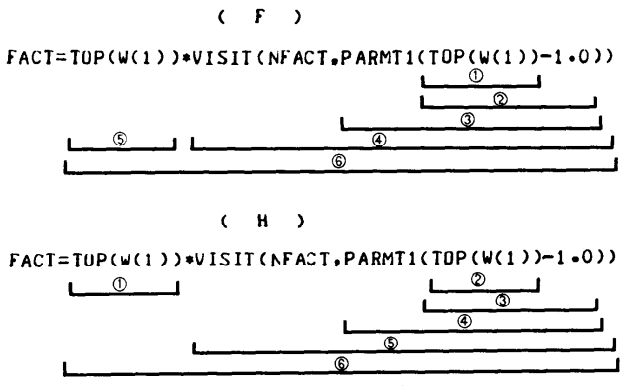

図 2　式の評価順序

可となる．したがって，互換性あらしむるには，J4 に忠実に

```
A = VISIT( . )
CALL SUBR(A,TOP( . ))
```

として副作用の影響を回避するように記述すべきであろう．そのほか

```
A = VISIT( . )+VISIT( . )*VISIT( . )
```

では，演算子の優先順位までからんで，さらに事情は複雑で，文を分解した上に左辺の変数を push down したり pop up したりしなければならず，文献 9) の方法と差がなくなってしまう．結局，文献 1) の方法による再帰的呼出しの限界といってよいだろう．FORTRAN は手続き記述言語 (procedural language) といっても式の評価順序は，処理系にゆだねられているので，このようなあいまいさは不可避である．文献 9) の方法では，おきざりになる作業用変数の値もあらかじめ考慮して push down, pop up が行なわれており非常に手続き的なのでコーディングにかかる努力は大きいが，文献 1) の方法にあるような心配をさけることができる．

5. おわりに

JIS の規定，とくに J4, J5 を厳密に検査するような FORTRAN プロセッサを作ることはコンパイル速度，目的プログラムの実行速度などから疑問である．むしろ，FORTRAN はもともと数値計算用の言語であり，言語システムプログラムのようなプログラムを FORTRAN を基礎に作るときに，implementer が，互換性に留意して，JIS の範囲内で書くように努めたほうが現実的であるように思う．また，FORTRAN を中間言語とするようなプロセッサを作る際も同様な配慮がほしい．FORTRAN 語のデックとアセンブラ語のデックの結合で，FORTRAN プロセッサが引数の個数の一致を検査しないことを積極的に利用したりすることをやめるように心がけたい．

なお，記号処理では整数型・実数型・論理型変数のほかに，string 型といった変数を想定する．その際，J1, J2 に従って型の一致の検査が厳密に行なわれると互換性に問題がでるが，筆者の調べた F, H, N の範囲では問題が生じなかったが，ここでもプログラムを書く際に，JIS を尊重しておけば問題はない．

参考文献

1) J. Weizenbaum: Symmetric List Processor. Comm. ACM, Vol. 6, No. 9, p. 524 (1963).
2) D. B. Russel: On the Implementation of SLIP. Comm. ACM, Vol. 8, No. 5, p. 263 (1965).
3) 木村 泉：HITAC 5020 用 SLIP システム，第7回プログラミングシンポジウム報告集 (1966).
4) 中村康弘：ページメモリーをもったリスト処理言語 FLIP III. 第10回プログラミングシンポジウム報告集 (1969).
5) 電子計算機プログラム用言語 FORTRAN (水準 7000) JIS C 6201—1967. 日本規格協会, pp. 62 (1967).
6) FACOM 230-60 FORTRAN 解説編(II). 富士通, pp. 171 (1968).
7) HITAC 5020, 5020 E/F FORTRAN (HARP) 日立製作所, pp. 99 (1969).
8) NEAC シリーズ 2200 オペレーティングシステム MOD III/MOD IV FORTRAN コンパイラ説明書. 日本電気, pp. 218. (1969)
9) 牛島和夫：FORTRAN による再帰的呼出し．情報処理, Vol. 11, No. 2, p. 101 (1970).

(昭和45年4月24日受付)

付録 副プログラムの呼出し系列

機種名	CALL文 CALL SUB($a_1, a_2, \cdots, a_n$)	関数の引用 FUNC($a_1, a_2, \cdots, a_n$)	備考
F	LAI n SXJ, 7 SUB NOP a_1,,b NOP a_2,,b ⋮ NOP a_n,,b	LAI n+1 SXJ, 7 FUNC NOP return,,b NOP a_1,,b NOP a_2,,b ⋮ NOP a_n,,b	文献 6) p. 97 n は引数の個数 SUB, FUNCは副プログラムの先頭番地 $a_1, a_2, \cdots, a_n$ は実引数の番地 b はベースレジスタ指定 return は，関数値を受け渡す番地
H	ICA 08, a_1 ICAR 08, a_2 ICA 09, a_3 ⋮ JS 00, SUB	左に同じ JS 00, FUNC	文献 7) p. 95 08, 09 はAレジスタの番地 a_1 | a_2 | a_3 | a_4 ……として実引数の番地が副プログラムに引き渡される 関数副プログラムの中で計算された関数値はAレジスタ8番地にはいってもどる
N	DCSW SUB DCW a_1 DCW a_2 ⋮ DCW a_n DCP @N@	DCSW FUNC 左に同じ	文献 8) p. 154 関数値の引き渡しについて文献 8)には記述がないが，テストプログラムの結果から，ACCによるものと推定される …実引数の終わりを示すレコードマーク