OUKA
Osaka University Knowledge Archive

Title	（第7回）フォートラン・プログラミングにおけるバッグ（誤り）とデバッグ（修正）：｛手続き副プログラム（SUBROUTINE,FUNCTION）CALL文 RETUR文｝
Author(s)	磯本, 征雄
Citation	大阪大学大型計算機センターニュース. 1976, 21, p. 21-43
Version Type	VoR
URL	https://hdl.handle.net/11094/65304
rights	
Note	

Osaka University Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

Osaka University

大阪大学大型計算機センター・ニュース　(No.21)　1976－6

（第７回）

解　　説

フォートラン・プログラミングにおける バッグ（誤り）とデバッグ（修正）

手続き副プログラム
　　（SUBROUTINE, FUNCTION）
CALL 文
RETUR 文

研究開発部　　磯　本　征　雄

1. はじめに

フォートラン文法を横目で見ながらプログラムのデバッヂングを進めてゆく時，たいていの人が思うことは，デバッグは文法の知識だけではどうしようもないということです。つまり文法はあくまでも守るべき最少限度の範囲を定めたにすぎません。逆に文法規則に違反しなければ，プログラマーはどのようなプログラムを作成しても良いのです。現実に，様々の種類のプログラムが作成されていることからも，この事実は理解されます。デバッキングのための苦闘は，まさにこの自由さに対する報いとでもいえましょう。

さて，プログラム作成手法が何如に自由であるとは言え，やはりある程度の構造を規定する必要があるでしょう。つまり，ひと纏まりの手続きを各々１つの副プログラムとして整理し，これらの組合せとしてプログラムを組立てる手段を講じることです。このためにフォートランで設けられているのがCALL文，SUBROUTINE 文及び FUNCTION文等です。個々の説明は本文でおこなうことにして，これら副プログラムの結合とバッグの関係について一般的事項に簡単に触れておきます。

第１回でも述べましたように，プログラムが処理されるまでには翻訳処理，結合処理，実行の３段階を経由しなければなりません。これらの処理は時間的な順序関係以外には相互に独立になされます。フォートラン文法上の違反は翻訳処理時にチエックされます。さて，ここで問題にしている副プログラム間の結合ですが，これらは次のリンクロード時に行われます。このリンクロードの意味は，副プログラムの結合処理を実行した後にプログラム全体を主記憶にロードする手続のことです。

副プログラム単位ごとの翻訳処理や，さらには副プログラムごとのアルゴリズムの正しさのテストを個々に完了しておくことはもちろん大切なことです。しかし，以上の手続きをもって正しいプログラムが出来上るわけではありません。言わゆる副プログラム間の結合処理

が正しくなければならないのです。もう少し具体的に言えば，副プログラムを引用する側と引用される側の相互の関係（例えば変数の受け渡しの方法）が矛盾なく行われていることが必要です。ここでは，JIS-FORTRAN（水準7000）[1]の文法をもとにNEAC FORTRAN-700[2),3)]におけるエラーメッセージ及びプログラムの具体的振舞いについて説明します。

2. 手続き副プログラムに関連する文法規則

副プログラム文，CALL文及びRETURN文は各々独立に使われることはなく，1つのプログラムの中でこれらは必ず対になって使われます。個々の文を誤りなく書く上では，単に標記法に気を付ければ充分です。しかし，これらの文が副プログラムの結合に重要な役割りを果すことからもわかるように，むしろそれら相互の関係に注意しなければなりません。まず，上記各文の標記法について述べます。

SOBROUTINE 文，FUNCTION 文，RETURN 文，CALL 文の形

SOBROOTINE 文は JIS FORTRAN（水準7000）において次の形とすることが定められています。

SUBROUTINE 文（SUBROUTINE statement）は，つぎのいずれかの形とする。

SUBROUTINE s (a_1, a_2 ……, a_n)

または

SUBROUTINE s

ここで，s は定義されるサブルーチンの英字名とする。a_1, a_2, ……, a_n, はいずれも仮引数とし，変数名か配列名か外部手続き名とする。

例：SUBROUTINE 文

```
SUBROUTINE MATMPY(L,A,M,B,N,C)
SUBROUTINE EXIT
```

この文は．サブルーチン副プログラムの先頭に置かなければなりません。ただし注釈行についてのみこの文の前に置いてもかまいません。

関数副プログラムは，JIS FORTRAN（水準7000）において次の形とすることが定められています。

関数副プログラムの定義　FUNCTION 文（FUNCTION statement）は，つぎの形とする。

t FUNCTION f (a_1, a_2, ……, a_n)

ここで，t は，INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL であるか空であり，それぞれ，定義される関数の型が整数型，実数型，倍精度実数型，複素数型，論理型であるか，または暗黙の型宣言に従うかを示す。

例：FUNCTION 文

```
FUNCTION BESSEL(N,X)
REAL FUNCTION J(N,X)
```

f は定義される関数の英字名とする。
a_1, a_2 ……, a_n はいずれも仮引数とし，これらは変数名，配列名または外部手続き名とする。

この文は，サブルーチン文と同じように関数副プログラムの先頭に置かなければならない。ただし，注釈行についてのみこの文の前に置いても良い。

RETURN文は副プログラムの最後の実行文です。RETURN文の実行により，実行の制御はこのRETURN文を含む副プログラムを引用した側のプログラムへ帰ります。RETURN文の形と意味は JIS FORTRAN（水準7000)において次のように定められています。

RETURN文 RETURN文(RETURN statement)は，つぎの形とする。

RETURN

RETURN文は，手続き副プログラムの論理的な終わりを示す。この文は，手続き副プログラムの中にだけしか現われないものとする。

例：RETURN 文

使 用 例：

```
REAL FUNCTION SIA(X,A)
      ⋮
SIA=Y+A
      ⋮
RETURN
END
```

サブルーチン副プログラムの中でこの文の実行は，そのサブルーチン副プログラムを引用したプログラム単位にもどることとする。

関数副プログラムの中でのこの文の実行は，その関数副プログラムを引用したプログラム単位にもどることであり，このときその関数の値が定まるものとする。

副プログラムにおいては少くとも1つのRETURN文がなくてはなりません。ただし，RETURN文は論理的に最後の部分ではあるが，必ずしも物理的に（つまり文の配置の上での順序として）最後でなくてもかまいません。

CALL文はSUBROUTINE副プログラムを引用するための文です。CALL文の形はJIS FORTRAN（水準7000）において次のように定められています。

CALL文 CALL文(CALL statement)は，つぎのいづれかの形とする。

CALLs (a_1, a_2, …, a_n)

または

CALLs

ここで，sはサブルーチンの名前とし，a_1, a_2, …, a_n は，いずれも実引数とする。

この文の実行によって，指定されたサブルーチンの引用が行なわれるものとする。指定されたサブルーチンの実行が終わったあとで，正常の実行順序にもどるものとする。

例：CALL文

```
CALL MATMPY( 5 ,X ,10, Y , 7 , Z )
CALL RANDOM
```

CALL文によりSUBROUTINE副プログラムを引用することによって，実引数又はCOMMON文で宣言された変数又は配列の値を再定義することができます。これについては，次の所で説明します。

副プログラムの定義

ここでは，文法の解説を目的とするわけではないので副プログラムの定義について JIS FORTRAN (水準7000)の規定を抜粋するにとどめることにします。

関数副プログラムの定義

関数プログラム(function subprogram) は，つぎの制限のもとで構成する。

(1) 関数の英字名は，FUNCTION 文の中のこの関数の英字名を除いて，プログラム単位中の非実行文の中に現われてはならない

(2) 仮引数の英字名は,関数副プログラム中のEQUIVALENCE文，COMMON 文またはDATA 文の中に現われてはならない。

(3) 関数副プログラムは，結果を持ち帰らせるために，関数の値のほかに，その引数の値のうちのいくつかを定義したり，再定義したりしてもよい。

(4) 関数副プログラムは，BLOCK DATA文，SUBROUTINE 文，他のFUNCTION 文，または定義している関数を直接にあるいは間接に引用する文を含んではならない。

(5) 関数副プログラムは，少なくとも一つのRETURN文を含まなければならない。

(6) 関数の英字名は，副プログラムを定義する際に，変数名として現われなければならない。副プログラムの実行ごとに，この変数は定義されなければならない。また，1度定義したあとで引用したり，再定義したりしてもよい。この副プログラムの中にあるRETURN文のいずれかが実行されるときこの変数の値が，この関数の値として引用されるものとする。

例：関数副プログラム

使用例1：

```
FUNCTION TAN(X)
TAN=SIN(X)/COS(X)
RETURN
END
```

使用例2：

```
FUNTION COSH(X)
COSH=(EXP(X)+EXP(-X))/2.0
RETURN
END
```

サブルーチン副プログラムの定義

サブルーチン副プログラム(subroutione subprogram)は，つぎの制限のもとで構成する。

(1) サブルーチンの英字名は，SUBROUTINE 文中でサブルーチンの英字名として使われている場合を除き，そのサブルーチンを定義するサブルーチン副プログラム内の文の中に現われてはならない。

(2) 仮引数の英字名は，サブルーチン副プログラム内のEQUIVALENCE 文，COMMON 文，またはDATA 文の中に現われてはならない。

(3) サブルーチン副プログラムは結果を持ち帰らせるために，その引数のうちいくつかを定義したり再定

例：サブルーチン副プログラム

```
      SUBROUTINE QUAD(A,B,C,DISC,X1,X2)
C SOLUTION OF QUADRATIC EQUATION
C A*X**2+B*X+C=0
      S=-B/(2.0*A)
      DISC=S**2-C/A
      IF(DISC.LT.0.0) GO TO 1
      T=SQRT(DISC)
      X1=S+T
      X2=S-T
      RETURN
1     X1=S
      X2=SQRT(-DISC)
      RETURN
      END
```

義したりしてもよい。

(4) サブルーチン副プログラムは，BLOCKDATA文，FUNCTION文，他のSUBROUTION文および定義しようとするサブルーチンを直接にまたは間接に引用する文を含んではならない。

(5) サブルーチン副プログラムは，少なくとも一つのRETURN文を含まなければならない。

以上は，副プログラム，RETURN 文及びCALL 文に関する文法の概略です。さらに詳細な点については，後でバックについて解説しながら触れることにします。

フォートラン文法と言えども，規則の細部においては計算機の機種ごとに幾分違いがあります。FORTRAN-700 （NEAC 2200シリーズ モデル700用）においては，文法規則上次の事項が補足されています。[2)]

文法上の補足事項について

(1) CALL文の実引数は，文の番号を除き最大63個まで許される。この制限を超えるとコンパイル時に致命的エラーとなる。実引数として使われる文の番号の個数は63以下でなければならない。

(2) RETURNi 文のiの値が許される範囲を超えていると実行時にエラーメッセージが出されてUEPになる。許される範囲はつぎの通りである。

実引数となっている文の番号の個数 ： ℓ

仮引数となっている星印(*)の個数 ： m

$1 \sim \min(\ell, m)$ が許される範囲である。$\ell \neq m$であっても，エラー メッセージは出力されない。

(3) 手続きの仮引数の個数は最大63個まで許される。

(4) 引用側の実引数の個数と仮引数の個数は一致しなければならない。外部手続きの仮引数の個数と対応する実引数の個数が一致しなければリンクロード時に致命的エラーとなる。

(5) 外部手続きの型と引用側の型とは一致しなければならない。型が一致しなければリンクロード時に致命的エラーとなる。

(6) 実引数の型と対応する仮引数の型との一致は検査されない。実行時には，実引数の型の内部表現が仮引数の型の内部表現とみなされて実行される。

(7) 基本外部サブルーチンの引用における実引数の型が，決められている型と一致しなければコンパイル時に致命的エラーとなる。

これらの補足事項は，必ずしもNEAC 2200 Model 700以外にも通用するものではありません。しかし，少くともModel 700においては厳守されなければなりません。

3. 翻訳処理時のエラー・メッセージ

翻訳処理時にはコンパイラーにより行われるチエックは，主として文単位に見た表現上の形式か，せいぜい1つのプログラム単位内でのプログラム構造上の矛盾に関するものです。外部手続き副プログラム及びCALL文や外部関数を引用した算術代入文がいくつかの副プログラムの結合を問題にするという点から見れば，翻訳処理時のエラー・メッセージは今回の解説における中心的課題からは少しずれているかも知れません。しかし翻訳処理時の誤りに対して，これをデバッグする際に気をつけなければならないことは，文の修正において常に引用する側と引用される側のいずれも同時に調べながら変更しなければならないことです。

例えば，

```
CALL  8YSUB
```

で，翻訳処理時に誤りとしてエラー・メッセージが出力されたとしましょう。ただし，この時安易に8YSUB→MYSUBと変更すべきではなく，慎重を期して" SUBROUTINE MYSUBを引用するCALL文であり，しかもたしかにそのサブルーチンが有る"ことを確認することが必要です。このように例え結合処理時以前の段階であるとはいえ，やはり副プログラム間の結合に注意を払いながらデバッグをすすめなければなりません。

以下にFORTRAN-700において結合処理時に出力されるメッセージについて個々に解説しますが，各メッセージごとにこれらの点に気をくばりながら読んで下さい。

3.1 WARNING

WARNINGは，誤りの可能性を警告するのみであり，プログラムを実行する上から見ただけでは，必ずしも障害にはなりません。ところがFORTRANが数値計算に適しているという側面を考える時，単に実行の流れが順序正しくすすめられただけでは，はなはだ心配になる点が多くさんあります。特に，副プログラム等に関連して変数値の受け渡しが重要である場合はなおさらWARNINGといえどもおろそかにできません。WARNINGといえどもデバッグ時には充分に注意を払って下さい。

以下に副プログラムに関連したWARNINGについて具体的にそのメッセージと意味を説明します。

メッセージ番号	説明
008	WARING: RETURN STATEMENT IN MAIN PROGRAM ○ 主プログラム中にRETURN文がある。RETURN文はSTOP文として処理される。

主プログラムの実行の最後の部分はSTOP文でなければなりません。RETURN 文は副プログラムのみで使われます。したがって，この WARNING は当然のこととして理解されるでしょう。

メッセージ番号	説明
011	WARNING: FUNCTION "name" SHOULD BE SUPPLIED BY THE USER AS AN EXTERNAL FUNCTION ○ 組込み関数又は基本外部関数と同じ名前で，型の異なる外部関数を引用している。

（例）

```
DOUBLE  SIN,Z
Z =SIN(X,Y)
```

この時，SIN は組込み関数の１つの正弦関数ではなく，このプログラムで与えられた副プログラムSINが取られる。組込み関数SINは単精度であるのに，この例では倍精度であり，しかも引数が２つである点に注意して下さい。

メッセージ番号	説明
026	WARNING: RETURN OR STOP STATEMENT IS MISSING ○ プログラム単位内にRETURNもSTOP文も文がない。そのままコンパイルされる。ただし診断メッセージ番号０５５に該当する場合はRETURN文がジェネレートされる。

１つのプログラム単位には，必ずSTOP文（主プログラムの場合）又はRETURN 文（副プログラムの場合）が少くとも１つなければなりません。

メッセージ番号	説明
027	WARNING: RETURN I STATEMENT IN FUNCTION SUBPROGRAM ○ 関数副プログラム内にRETURN I 文がある。変数名または整定数を伴わないRETURN文としてコンパイルされる。

(例)

```
FUNCTION FUNC(X,I)
      ⋮
RETURN I
END
```

RETURN Iの形式はサブルーチン副プログラムにのみ許されるものです。したがって，これは誤りの例です。

メッセージ番号	説明
036	WARNING: ARGUMENT TYPE MISMATCH FOR FUNCTION "name"-CONVERTED ○ 基本外部関数あるいは組込み関数の実引数の型が，決められている型と一致しない。ただし算術代入文の代入の規則では許されている。実引数の型が算術代入文の代入の規則によって変換される。

(例)

```
I = 10
  ⋮
X = SIN(I)
```

組込み関数SIN（正弦関数）における実引数は実数型です。この例のように整数型である場合，WARNING 036 が出力され，Iは算術代入文の代入規則に従って処理されます。

3.2 FATAL ERROR

FATAL ERROR は前節の文法規則に明らかに違反するものに対して出力されます。このエラー・メッセージは文単位で出力されますが，副プログラムは相互に引用する側とされる側の両方が有ってはじめて意味があるわけで，FATAL ERROR に対する修正においてもこれら両面から調べる必要があります。この点について充分注意を払いながら以下の具体的メッセージ及びそれについての例を読んで下さい。

メッセージ番号	説明
151	RECURSIVE REFERENCE OF SUBPROGRAM ○ 副プログラム内で自分自身を引用している。この引用している文は削除されてコンパイルされる。プログラム実行時にこの削除された文に実行の制御が渡るとUEPになる。

(例)

```
SUBROUTINE MYSUB(X,Y)
      ⋮
CALL MYSUB(P,Q)
```

この例のように，フォートランでは副プログラムの回帰的引用は許されません。

メッセージ番号	説明
152	NUMBER OF LABEL ARGUMENTS EXCEEDS 63 ○ サブルーチン副プログラム引用時に，実引数として63個を越える文の番号を使用している。この文の削除されてコンパイルされる。プログラムの実行時にこの削除された文に実行の制御が渡るとUEP になる。

これは，前節における文法上の補足事項(1)に抵触することです。63個という数字はFORTRAN-700 固有のものです。

メッセージ番号	説明
153	SYNTAX ERROR IN CALL STATEMENT ○ CALL 文中で文法違反がある。 この文は削除されてコンパイルされる。プログラムの実行時にこの削除された文に実行の制御が渡るとUEPになる。

(例)

```
CALL MYSUB 1 (A , B+BCD, )
CALL MYSUB 2 (A・B)
CALL MYSUB 3 (     )
```

これらの例は，いずれもCALL文の形に違反するものです。

メッセージ番号	説明
154	NON-SUBROUTINE "name" REFERENCED AS SUBROUTINE ○ サブルーチン名でないものがCALL文でサブルーチン名として使われている。このCALL文は削除されてコンパイルされる。プログラムの実行時にこの文に実行の制御が渡るとUEPになる。

（例）

```
DIMENSION MATRIX (10,10)
    :
CALL MATRIX(ARG1,ARG2)
```

MATRIXは配列として宣言されています。それにもかかわらず，CALL文においてMATRIXはあたかもサブルーチンの如くに引用されてます。CALL文で引用できるのはサブルーチンに限られます。したがって文法違反となります。

メッセージ番号	説明
155	NUMBER OF ARGUMENTS EXCEPT LABELS EXCEEDS 63 ◦ サブルーチン副プログラムの実引数が文の番号を除いて63を越えているか、または関数副プログラムの実引数が63を越えている。この文は削除されてコンパイルされる。プログラムの実行時にこの削除された文に実行の制御が渡るとUEPになる。

これは，前節における文法上の補足事項(1)に抵触することです。63個という数字はFORTRAN −700固有のものです。

メッセージ番号	説明
233	ILLEGAL OR EXTRANEOUS INFORMATION FOLLOWING RETURN STATEMENT ◦ RETURN文に整定数または整数型の変数名以上のものがあるか，あるいは余分な情報がある。この文は削除されてコンパイルされる。プログラムの実行の制御が渡るとUEPになる。

（例）

```
RETURN  10.0
RETURN  I , 3
```

RETURN Iの形式はFORTRAN-700において許されるものですが，この時のIは整定数又は整数型変数名のみです。しかもただ1つしか許されません。

メッセージ番号	説明
571	ILLEGAL SUBPROGRAM NAME ◦ SUBROUTINE文，FUNCTION文あるいはENTRY 文に名前がないか，あるいは名前を示す英字名に誤まりがある。この文はないものとしてコンパイルされるが，この副プログラムはGO ファイル上に出力されない。

（例）

```
SUBROUTINE 8MYSUB (ARG1,ARG2)
FUNCTION(ARG1,ARG2)
```

最初の例では，サブルーチン名が数字ではじまっているために誤りです。これは英字からはじまるものにしなければなりません。第２例では関数名がありません。

メッセージ番号	説明
572	ILLEGAL SYNTAX IN SUBPROGRAM ARGUMENT LIST ◦ SUBROUTINE文，FUNCTION文あるいはENTRY文の仮引数として許されないものがあるか区切り記号に誤りがある。誤りのある仮引数はないものとみなされ，区切り記号の誤りは適当な仮定のもとにコンパイルされるが，この副プログラムはGO ファイル上に出力されない。

（例）

```
SUBROUTINE MYSUB (2.5, A *B)
ENTRY MYETRY (A. B+C)
```

これら２例共に，仮引数の形式が文法に定められたものになっていません。例えば，算術式を仮引数にすることはできません。

メッセージ番号	説明
573	"name" IS AN ILLEGAL DUMMY ARGUMENT ◦ 不正な仮引数がある。たとえばコモン要素名，NAMELIST名等である。不正な仮引数名はないものとしてコンパイルされるが，このプログラム単位はGOファイル上に出力されない。

（例）

(1)
```
SUBROUTINE MYSUB(NAM)
NAMELIST／NAM／A,B,C
```

(2) FUNCTION MYFUNC (＊, A)

第1の例ではNAMELIST名のNAMが仮引数として使われているが，これは許されません。

第2の例では，関数の仮引数として"＊"が用いられていますが，これも許されません。

メッセージ番号	説明
574	FUNCTION WITH NO ARGUMENTS ○ FUNCTION 文または，関数副プログラム内のENTRY文に仮引数がない。そのままコンパイルされるが，この関数副プログラムはGOファイル上に出力されない。

(例)

FUNCTION FUNC

FUNCTION 文においては必ず仮引数が付かなければなりません。この例では仮引数がありません。

メッセージ番号	説明
575	DUPLICATE DUMMY ARGUMENT ○ SUBROUTINE文，FUNCTION 文あるいはENTRY文において，同一の仮引数が2回以上使用されている。この副プログラウはそのままコンパイルされるが，GOファイル上に出力されない。

(例)

FUNCTION MYFUNC(ARG1,ARG2,ARG2)

仮引数ARG2が2回使用されているために誤りになります。

メッセージ番号	説明
577	DUPILICATE ENTRY NAME "name" ○ ENTRY文でENTRY名として使用されている英字名が，他のENTRY文，SUBROUTINE文，FUNCTION文のいずれかでENTRY名あるいは副プログラム名として使用されている。2回目以降に同一名を使用したENTRY文はないものとしてコンパイルされるが，この副プログラムはGO ファイル上に出力されない。

(例)

```
SUBROUTINE MYSUB(ARG1,ARG2)
      :
ENTRY MYSUB(ARG3,ARG4)
```

SUBROUTINE 名とENTRY名は相互に異る名前にしなければなりません。この例では同一の副プログラム名となっているために誤りです。

メッセージ番号	説明
578	NO VALUE RETURNED BY THIS FUNCTION 。関数副プログラムの中で関数値が定義されていない。そのままコンパイルされるが，この関数副プログラムはGO ファイル上に出力されない。

(例)

```
FUNCTION MYFUNC (X)
F=X**2
RETURN
END
```

関数副プログラムは関数名MYFUNC を定義するためのものです。この例のように，関数値の定義されない副プログラムは当然誤りです。

4. 結合処理時のエラー・メッセージ

CALL文とSUBROUTINE 文及び算術代入文とFUNCTION 文は副プログラムの結合の際のデータ転送の出入口に相当します。したがって，翻訳処理の完了した個々の副プログラムに対して，結合処理時にはCALL文とSUBROUTINE 文の結合の状況が問題になります。当然のことながら論理的かつ文法的矛盾さえなければ，誤りとはなりません。ところがうかつに誤りを犯すことはプログラミングにおいてはそれほど驚くにあたりません。

例えば図4.1を見て下さい。図4.1のプログラムを実行すれば，結合処理時に誤りとなります。このプログラムの問題点は，

(1) 副プログラム側又は引用側のいずれかにおける型宣言に誤りがある

ことなのです。すなわち，主プログラムで引用する際の関数F(X)は倍精度宣言されております。ところが，引用された側の外部関数F(X)は単精度になっています。つまり

(2) 関数副プログラムにおいて外部手続き側の型と引用側の型が一致していない

図4.1　結合処理時に誤りとなるプログラムの例

```
CCC  MAIN PROGRAM
     DOUBLE F,Y
C         :
     X = 1.0
     Y = F(X)
C         :
     STOP
     END
C    SUBPROGRAM
     FUNCTION   F(X)
C         :
     F = X**2
C         :
     ○
     END
```

結合処理時のエラー・メッセージ

LLK66I ***** TYPE / ADDCON UNMATCH F

のです。このことはプログラムを主記憶上にロードする時の状況を考えてみれば，どのような誤りであるかがわかるでしょう。

さて，この結果

(3)　結合処理時にエラー・メッセージ

LLK66I ***** TYPE／ADDCON UNMATCH F

が出力される

このように型の差異は，単精度が1語長であるのに対して倍精度が2語長であるための記憶領域の大きさに関係しています。つまり，このままでは語長に関する矛盾から主記憶上へのロードがうまくゆかないのです。

引数の語数に直接関係することに，もう一つ

(4)　外部手続き側の仮引数と引用側の実引数の個数が違っている

場合があります。この結果は，(3)のエラー・メッセージを出力させることになります。この(4)の引数の個数の違いは，引数となっている変数における(1)の型宣言の誤りや引数として使われた配列において

(5)　配列宣言に誤りがある

場合にも，(4)における見掛け上の語数の違いと同様の結表を生むことになります。

図4.2　引数の個数の違う場合の例

```
CCC  MAIN PROGRAN
     DATA  X,Y,Z/1.0,2.0,3.0/
C          :
     CALL SUB (X,Y,Z,T)
C          :
     CALL SUB (X,T)
C          :
     STOP
     END
C    SUBPROGRAM
     SUBROUTINE SUB (X,Y,T)
C          :
     T=X*Y
C          :
     RETURN
     END
```

結合処理時のエラー・メッセージ

```
LLK66I ***** TYPE /ADDCON UNMATCH SUB
LLK66I ***** TYPE /ADDCON UNMATCH SUB
```

これらに対する具体例を図4.2に示しました。この例では，サブルーチンの仮引数は3個あります。ところが，主プログラムでの引用において，最初のCALL文では実引数が4個あり，第2番目のCALL文では実引数が2個あります。これらはいずれも誤り(4)に相当し，(3)に示したエラー・メッセージが2回出力される結果になったのです。

上記の例はいずれも引数に関する誤りですが，別の状況として，

(6) 外部手続きにおける関数名又はサブルーチン名が誤りである。

場合や，逆に

(7) 引用側の関数名又はサブルーチン名が誤っている

場合があります。これらはいずれも，単純なミス・パンチやコーディング・シートの読み誤りによって起こることが多いでしょう。

これらはいずれもリンケージ・ローダーにより

(8) 引用側に対応する外部手続きが無い

ものとして処理されます。そして，

(9) 結合処理時にエラー・メッセージ

LLK17I ***** UNRESOLVED ENTRIES

LLK38I ***** UNRESOLVED SUBPROGRAM

が出力されます。これらのエラーは，単にプログラマーの作成したプログラムに限らず，基本外部関数等の計算機システムに予め組込まれた外部手続きについても全く同様です。

なお，これに類似した誤りに

(10) 同じ名前の副プログラム名が２つ以上ある

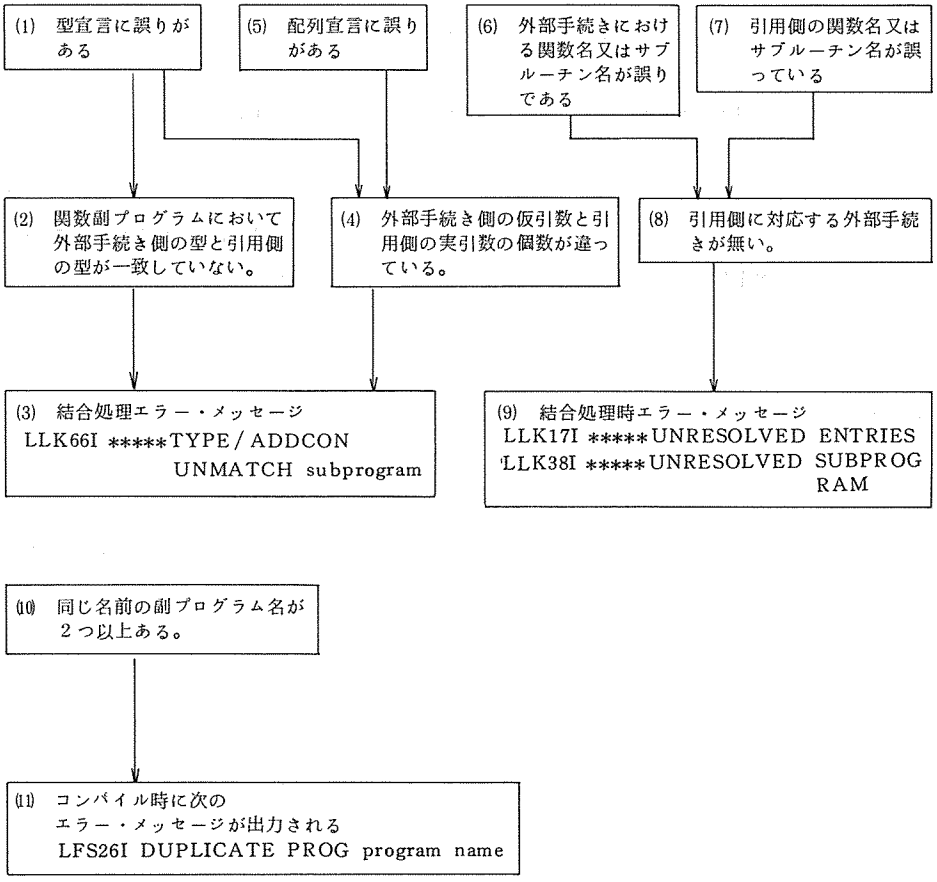
場合があります。この場合は

(11) 翻訳処理時に次のエラー・メッセージが出力されます

LFS26I DUPLICATE PROG Program name

これらを原因→結果の順に整理すれば図4.3のようにまとめられます。

図4.3 結合処理時の誤り



5. 実行時のCALL 文・副プログラムに関する誤動作

まずRETURN文について調べてみます。一つの副プログラムの中でRETURN 文の置かれるべき場所は，実行文の後ならばどこであってもかまいません。この結果，第7カラムより英字R・E・T・U・R・Nがこの順序に並べられている限り文法的な誤りとはなりません。しかも，一つの副プログラム中にあるRETURN 文の数は制限されていません。さらに，実行がRETURN 文に達するとその副プログラムは終了し，実行の制御はその副プログラムを引用した元のプログラムへ帰ります。これを別の面から見ると，

(12) RETURN 文の置かれている位置に誤りがある

場合であっても文法的には正しいプログラムであり，そのために

(13) 外部手続き引用において，正しい値が得られない

ことにより，プログラムの誤りに気付くことになります。このような誤りは，不注意な行の入れかわり等によりおこるものなのでデバッキングの際に時々見逃してしまう恐れがあります。このように副プログラムの出口に相当するRETURN文も決しておろそかにできないものです。

次に副プログラムの入口に相当するSUBROUTINE 文やFUNCTION 文及び引用に関連するCALL文について調べてみましょう。これらは，一口で言えば実引数と仮引数間の一対一の対応関係が重要です。図5.1にはサブルーチンSUB を使ったプログラム例を示しました。

このプログラムが正しいか否かは，総てその意図にかかっています。これらの引数におけるCALL文とSUBROUTINE 文の対応関係は次のようになっています。

図5.1 引数の順序に関する例題プログラム

```
C     MAIN PROGRAM
C       :
      CALL SUB ( X , Y , Z )
        :
      STOP
      END
C     SUBPROGRAM
      SUBROUTINE SUB  ( Y , Z , X )
C       :
      RETURN
      END
```

引用側		外部手続き側
X	⟷	Y
Y	⟷	Z
Z	⟷	X

つまり，Ｘ，Ｙ及びＺの英字自体には意味はなく，それらの順序が意味をもっているのです。実引数及び仮引数の最左側から順次相互に対応するのです。このために，

⑭　実引数の順序に関する単純な誤りがある

場合において，あるいは

⑮　仮引数の順序に関する単純な誤りがある

場合においても，これらはいずれも

⑯　実引数と仮引数の間で個々の配列名及び変数名の対応が一致していない

ことになります。この結果として，⑬外部手続き引用において正しい値が得られないことになります。

図5.2　実引数と仮引数の型の違う場合の例

```
C         MAIN PROGRAM
          DATA   X,Y,Z/1.0,2.0,3.0/
          DATA   I,J,K/1,2,3/
C               :
          CALL SUB (X,Y,Z,T)
          WRITE (6,100) X,Y,Z,T
  100     FORMAT (1H3,'TEST 1'/4X,'X,Y,Z=',4E19.8)
C               :
          CALL SUB (I,J,K,T)
          WRITE (6,200) I,J,K,T
  200     FORMAT (1H3,'TEST 2'/4X,'I,J,K=',3(I15,4X),E15.8)
C               :
          STOP
          END
C         SUBPROGRAM
          SUBROUTINE SUB(X,Y,Z,T)
          T=X+Y+Z
          X=X+1.0
          Y=Y+4.0
          Z=Z+10.0
          RETURN
          END
```

上記プログラムによる出力結果

```
TEST 1
 X,Y,Z,T= .20000000E+01  .60000000E+01  .13000000E+02  .60000000E+01
TEST 2
 I,J,K,T= 7036874417665  70268744177667  87960930222084  .0
```

外部手続き引用の際に，特にFORTRAN-700においては，実引数の型と対応する仮引数の型との一致は検査されていません。このために，実行時には実引数の型の内部表現が仮引数の型の内部表現とみなされて実行されます。例えば，図5.2を見て下さい。第1番目の引用においては引用側の実引数と外部手続き側の仮引数の型と順序は正しく一致しています。この結果はTEST 1として正しく出力されています。ところが第2番目の引用例においては，実引数の左側3つは整数型ですが仮引数の左3つは実数型になり，一致していません。今の場合，仮引数の型の差異を無視すれば，第1の例と第2の例で引数の数値は同じです。ところが第2の例に対する引用の結果は，TEST 2の所に書き出されているように全く違った数値になっています。このように

(17) 実引数と仮引数の相互の間の型が一致しない

場合においてもやはり，(13)外部手続き引用において正しい値が得られないのです。図5.2の例では，すべての引数が外部手続き側で再定義されていますが，このことは余り問題になりません。さらにほかの例として，外部手続き側でDO文による制御変数として，あるいはREAD文等により値を再定義する場合であっても，これら算術代入文による再定義とその効果は全く同じです。　ところが，

(18) 外部手続きにおいて入力用に使われた仮引数が副プログラムの中で再定義されている場合に，予想外の副作用を引き起こし思わぬバッグをつくり出すことがあります。図5.3に

図5.3　関数引用における望ましくない副作用の例

```
C       MAIN PROGRAM
        DATA X / 1.0 /
        Y = FUNC (X) * ( FUNC(X) + 1.0 )
        WRITE ( 6,100) X , Y
  100   FORMAT (1H3,'TEST OF FUNCTION'/4X,'X,Y=',2E15.8)
        STOP
        END
C       SUBPROGRAM
        FUNCTION FUNC (X)
        FUNC = X** 2
             :
        X = X + 1.0
             :
        RETURN
        END
```

上記プログラムによる出力結果

```
 TEST OF FUNCTION
 X , Y =  .30000000E+01    .50000000E+01
```

は，この副作用を示した具体的プログラムの例が示してあります。関数FUNCでは，引用される度に仮引数Xに1.0が加えられます。この例は非常に単純化されたプログラムであるためにあまり明確なことはいえませんが，演算途中の作業用変数に対して誤って仮引数と同じ変数名を使った等の結果として，このような引数の再定義を不本意にやってしまうこともしばしばあります。つまり，

(19) 外部手続き引用後にも値が保存されるべき実引数の値が変ってしまった

ような場合の誤りは決して少ない例ではないのです。したがって図5.3の関数副プログラムがもし非常に大きいものであるならば，この種の誤りの恐れを常にもつでしょう。この結果，プログラムの実行過程で

(20) 外部手続き引用時に，再定義をしていないはずの変数値が変ってしまった

とか，

(21) 連続して外部手続きを引用すると結果が正しくない

などの誤りを引きおこします。これについて，図5.3のプログラム例に則して説明しましょう。第3行目の算術代入文

Y＝FUNC (X)＊ (FUNC (X) + 1.0)

は，この例では実際に次のように処理されています。

Y＝1.0 ＊ (4.0＋1.0)

つまり，第1回目のFUNC引用においては，

FUNC (X)＝1.0　　但し　X＝1.0

になっています。ところがFUNC (X)＝1.0が帰ってくる時点でのXの値はさらに1.0が加えられて X＝2.0 となっています。したがって，2回目のFUNC 引用において，

FUNC (X)＝4.0　　但し　X＝2.0

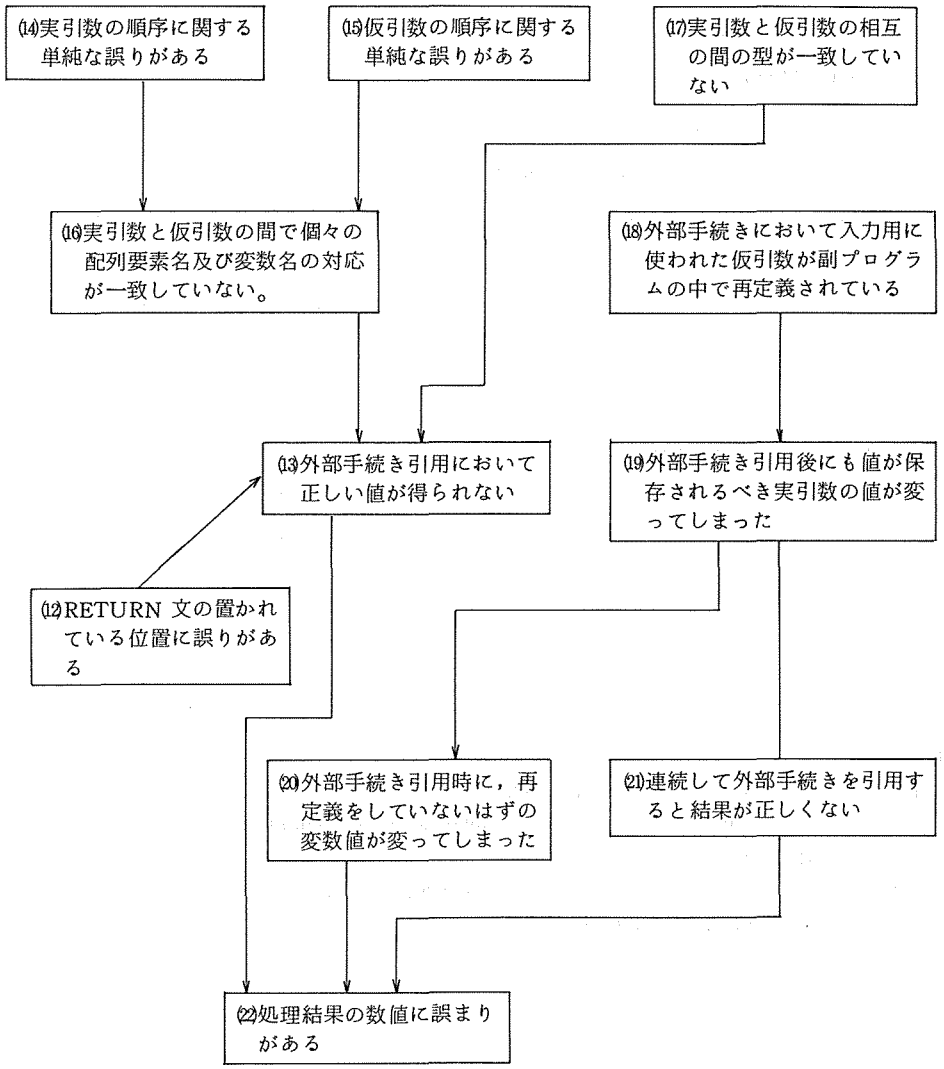
となっているのです。さらにこの時にXの値には1.0が加えられ X＝3.0 となっています。これらの結果，図5.3の下に見るような結果となるのです。このような，副プログラムの副作用の問題は，プログラム作成時におけるテスト段階で充分に確認しておくことが必要です。

さて，ここに示した実行時の誤りは，最悪の場合いずれも

(22) 処理結果の数値に誤りがある

ことにより確認されます。これら上記の事項を原因→結果の順序で流れ図にまとめれば，図5.4となります。

図5.4　実行処理時のプログラム誤動作



6. ま と め

今回は，副プログラムの結合の問題に焦点を合せて解説しました。副プログラムの結合において特に注目しなければならないのは，副プログラム間のデータの転送です。これには大きく分けて

① ファイルを中介とする方法

② COMMON 文による方法

③ 引数による方法

があります。 ①は次回（第８回目）解説予定であり，②はすでに第３回目（センター・ニュース №１５）で解説しました。③こそが今回の主題であったのです。これら①，②及び③の個々の特徴はさておき，いずれも変数の値を変えることなくプログラムの異る部分間でのデータ転送をおこなうものです。フオートランが四則演算を重視する言語であるが由に，このような単なるデータ転送に関する事項に余り注意を払わないプログラマーも決して少くありません。

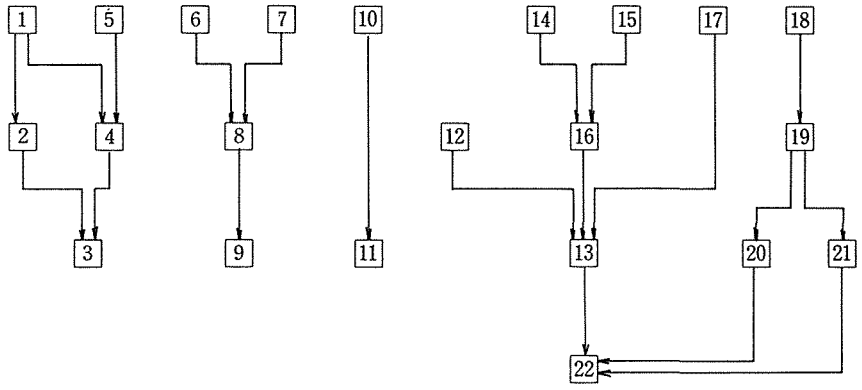
ここであげたプログラム例及びそれらに関する解説は，いずれも多くのプログラマーがしばしば犯している誤りを要約したものです。これら総てを誤りの例としてのみ理解されることは必ずしも筆者の思うところではありませんが，いずれもCALL文，外部手続き副プログラムの間のデータ転送の状況の誤解されやすい側面を抽出したものであることを了解して下さい。

引用文献

1. JIS ハンドブック，情報処理，日本規格協会（1973）
2. NEAC シリーズ2200オペレーティングシステムMOD Ⅳ EX／Ⅶ FORTRAN-700，文法説明書，NEC日本電気株式会社
3. NEAC シリーズ2200オペレーティングシステム MODⅣ EX／Ⅶ FORTRAN-700，プログラミング説明書，NEC 日本電気株式会社

付　　録

CALL文.
FUNCTION 文,
SUBROUTIN文
及びRETURN文の結合処理及び実行時の誤りの因果関係



結合処理時

(1) 型宣言に誤りがある。

(2) 関誠副プログラムにおいて，外部手続き側の型と引用側の型が一致していない。

(3) 結合処理時エラー・メッセージ

LLK66I ***** TYPE / ADDCON UNMATCH subprogram name

(4) 外部手続き側の仮引数と引用側の実引数の個数が違っている。

(5) 配列宣言に誤りがある。

(6) 外部手続きにおける関数名又はサブルーチン名が誤りである。

(7) 引用側の関数名又はサブルーチン名が誤っている。

(8) 引用側に対応する外部手続きが無い。

(9) 結合処理時エラー・メッセージ

LLK17I ***** UNRESOLVED ENTRIES subprogram name

LLK38I ***** UNRESOLVEO SUBPROGRAM

コンパイル時（補足）

(10) 同じ名前の副プログラム名が２つ以上ある。

(11) コンパイル時に次のエラー・メッセージが出力される。

LFS26I DUPLICATE PROG program name

実　行　時

(12) RETURN文の置かれている位置に誤りがある。

(13) 外部手続き引用において正しい値が得られない。

(14) 実引数の順序に関する単純な誤りがある。

(15) 仮引数の順序に関する単純な誤りがある。

(16) 実引数と仮引数の間で個々の配列名及び変数名の対応が一致していない。

(17) 実引数と仮引数の相互の間の型が一致していない。

(18) 外部手続きにおいて，入力用に使われた仮引数が副プログラムの中で再定義されている。

(19) 外部手続き引用後にも値が保存されるべき実引数の値が変ってしまった。

(20) 外部手続き引用時に，再定義をしていないはずの変数値が変ってしまった。

(21) 連続して外部手続きを引用すると結果が正しくない。

(22) 処理結果の数値に誤りがある。