プログラム・ノート

(第2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)

――DO文――

研究開発部 磯 本 征 雄

1. はじめに

前回はフォートランにおける文相互の関連及びバッグについての一般的説明であった。そこでは同時にバッグのもついくつかの特徴及びデバッグのための一般的手法についてもふれておいた。ところがプログラム・デバッギングの際には，プログラムの中でのバッグをもつ文の位置とデバッグされるべき事項を具体的に知る必要がある。これらは，具体的現象の中から推察し，探索・決定するしかない。前回にも示したようにバッグの悪影響とその伝播の様子は具体的状況に強く依存しながらもなおかつ“規則的側面”をもっている。しかし個々のプログラムのもつ目的とプログラム中の文で表現しようとする事項の意味は密接に結びついているために，やはり“具体的プログラムの意味・内容”を離れて抽象的議論をする事は妥当ではない。このような理由から，“フォートランの文法”に即して“これに違反するプログラムの示す具体的症状”をもとにしてバッグ・デバッグの説明をおこなう。

ここで解説する事項または現象は，すべてが常に誤りとして取扱われるべきものではないかもしれない。プログラムの意味の上から，それらが誤りであるか否かは一概に速断できず，むしろプログラム作成者の判断に待つしかないものが多い。したがって以下にのべるすべての事項を承知の上で，敢て奇妙なプログラムをつくる場合についてはこの解説の言及の限りではない。

今回はDO文について解説する。その主な理由は次の事による。第1に使用頻度が高い。第2に主としてプログラムの重要な部分に位置する事が多い。第3にDO文に関係するバッグは複雑なために推察困難なものが多い。故にDO文から解説をすすめる事は，望ましいと思う。なおここで示す具体的事項は，すべてNEAC 2200シリーズ，MODEL 700システムにおけるFORTRAN 700でテストした結果である。したがって他のシステムについてはどの程度あてはまるかは明らかでない。

2. DO文に関する文法

フォートランの文法は，JIS-FORTRANとして規格化されている。

JIS-FORTRANには数段階の水準がある。ここでは最も新しく定められた水準7000を中心に話をすすめる。なお(NEAC) FORTRAN-700はこの水準7000に相当する文法である。DO

文の性質もFORTRAN水準7000に定められている。以下JIS FORTRAN（水準7000）におけるDO文についての説明文を紹介する。

2.1 JIS-FORTRAN（水準7000）におけるDO文

DO文は一群の実行文をくりかえし実行する事を定める文である。以下に文献1）からDO文に関する事項を抜粋し説明する。

DO文の表現形式とDO文に用いられるパラメーター，さらにはDO文の端末文に関しては次のように定められている。

DO文(DO statement)は，つぎのいずれかの形とする。

DO n i＝m_1, m_2, m_3

または

DO n i＝m_1, m_2

ここで，nは実行文の番号とする。番号nを持った文は，このDO文に対応する端末文(DO terminal statement)といい，DO文と同じプログラム単位内にあり，DO文よりも物理的にあとになければならない。端末文は，GO TO文，算術IF文，RETURN文，STOP文，PAUSE文またはDO文，もしくはこれらを含んだ論理IF文であってはならない。

iは整数型の変数名とし，制御変数という。

m_1を初期パラメタ(initial parameter)，m_2を終値パラメタ(terminal parameter)，m_3を増分パラメタ(incrementation parameter)といい，いずれも整定数か整数型のの変数の引用とする。m_3のない2番目の形のDO文を使った場合には，増分パラメタの値は1とみなす。DO文の実行の際には，m_1, m_2, m_3の値は，いずれもゼロより大きくなければならない。

例：DO文

文例：

```
DO 30 I=1, M, 2
DO 15 I=3, 10
```

使用例：

(1) つぎの例は $S=\sum_{i=1}^{100} R_i$ を求めるものである。

```
      ⋮
      SUM=0.0
      DO 20 I=1, 100
   20 SUM=SUM+A(I)
      ⋮
```

(2) つぎの例は$y_1, \cdots, y_n$からxに等しいものを探し，あればその添字の値を変数Mに与え，なければMを0とする。

```
      ⋮
      DO 10 I=1, N
      IF(X-Y(1))10, 20, 10
   10 CONTINUE
      M=0
      GO TO 30
   20 M=I
   30 …
```

DOの範囲：

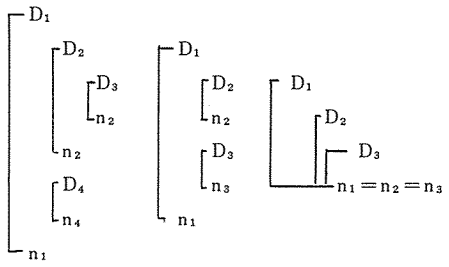
```
      ⋮
      DO 3 K=1, 10
    文 1 ┐
      ⋮  │ DOの範囲
  3 文 2 ┘
      ⋮
```

複数個のDO文においてくり返しの範囲が重複している場合には，DO文相互のくり返しの範囲に関して次に示されるいずれかの形式を守らなければならない。

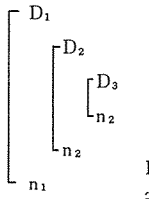
DO 文に対応する DO の範囲(range of DO)は，DO 文のあとに続く最初の文からその DO 文に対応する端末文までのすべての実行文の集まりとする。ある DO 文に対応する DO の範囲が別の DO 文を含む場合，二つの DO の範囲は入れ子(nest)をなしていなければならない。すなわち，内部の DO 文に対応する DO の範囲は，外部の DO の範囲に含まれなければならない。

完全入れ子(completely nested nest)は，いくつかの DO 文とそれらに対応する DO の範囲の集まりで，その中のどの二つの DO 文をとってもそれらの範囲は入れ子をなしていて，さらにこの集まりに属する DO の範囲はこの集まりに属さないいかなる DO の範囲とも入れ子をなさないものとする。

入れ子をなした範囲：



完全入れ子：



DO の範囲が一つの場合も完全入れ子となる。

DO 文が実行されるにあたって，制御変数の値は次の規則に従って変えられる。

DO 文は，繰返しを定義するのに使用する。DO 文の実行は，それに対応する DO の範囲の実行を引起こし，その際，つぎの五つの段階がとられるものとする。

第1段　制御変数に初期値パラメタの値が与えられる。この値は，終値パラメタの値より小さいか等しくなければならない。

第2段　DO の範囲が実行される。

第3段　第2段の結果，その DO 文に対応している端末文が実行され，その DO の範囲に含まれて端末文を共有する他の DO が，もしあるときには，それらがすべて満足されている（第4段参照）ならば，その DO の制御変数に増分パラメタの値が加えられる。

第4段　制御変数の値が終値パラメタの値より小さいか等しいならば第2段にもどる。制御変数の値が終値パラメタの値より大きいならば，この DO は満足されたといい，制御変数は不定となる。

第5段　DO が満足されたとき，この DO と端末文を共有する DO がすべて満足されているならば，端末文に続く最初の実行文の実行に移る。そうでないならば外側の DO の第3段に移る。

プログラム実行の流れが（DO が満足される前に）DO 文の範囲から出る時，DO の制御変数の値は次のようにとられる。

以下の部分では，GO TO 文または算術 IF 文を含んだ論理 IF 文は，それぞれ GO

TO 文または算術 IF 文とみなす。

GO TO 文または算術 IF 文の実行により DŌ の範囲から出るとき，つまり DŌ が満足されずに出るときには，DŌ の制御変数は，その直前に与えられた値を持っているものとする。

さらに DŌ 文には，DŌ 文の範囲から1度外に飛び出し，再び DŌ 文の範囲に返ってくる“拡張範囲”というものがある。“拡張範囲”は次のように定められている。

完全入れ子に関してつぎの条件が両方とも満たされる場合には，その完全入れ子に含まれるに含まれる DŌ は，拡張範囲 (extendeddrange of DŌ) を持つという。

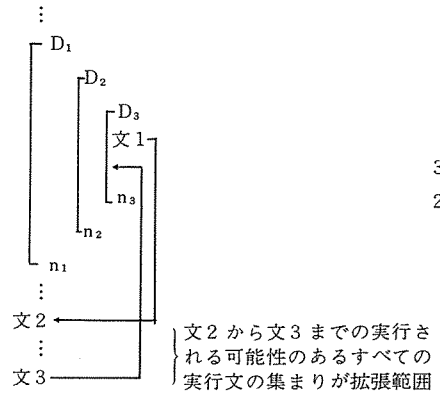
条件1　完全入れ子の一番内側の DŌ の範囲から完全入れ子の外側へ出る GO TO 文または算術 IF 文がある。

条件2　完全入れ子の外側から完全入れ子の一番内側の DŌ の範囲にはいる GO TO 文または算術 IF 文で，条件1に述べた文の実行のあとで実行される可能性があるものが，その完全入れ子と同一のプログラム単位内にある。

上の2条件が満たされる場合には，拡張範囲は，条件1の制御文と条件2の制御文の間で実行される可能性のあるすべての実行文の集まりとする。ここで，条件1の制御文は拡張範囲に含まれないが，条件2の制御文は含まれるものとする。

DŌ の拡張範囲内にその DŌ 文を含むプログラム単位内の他の DŌ 文や DŌ 形並びを含む入出力文があってはならない。拡張範囲について条件2で述べた制御文を除いて，GO TO 文あるいは算術 IF 文で DO の範囲外から範囲内にはいってはならない。

拡散範囲：



誤例：

```
    DO 30 I= 1 , 10
    DO 20 J= 1 , 20, 2
30  A= N(K, J)
20  A= M(I, J)
```

ここで文1と文3はGO TO 文か算術 IF 文である。

その他，DŌ 文に関して次の事項が定められている。

DŌ の制御変数は，その DŌ の範囲の実行中あるいはその拡張範囲の実行中に再定義されたり不定となったりしてはならない。また，その DŌ の三つのパラメータも一定でなければならない。

二つ以上のDO文が共通の端末文を持つ場合には，その端末文の番号は，それに対応する一番内側のDOの範囲またはその拡張範囲に含まれないGO TO文や算術IF文で使用してはならない。

参考1　DOが実行されているとき，DOの制御変数やそのパラメタを再定義することを制限したのは，処理系を効率よく働かせることを目的としたからである。

参考2　国際規格では“DOの拡張範囲に，そのDO文を含むプログラム単位内の，拡張範囲を持つ他のDO文があってはならない”となっている。

DO文の端末文として用いられるCONTINUE文に関しては次のように定められている。

(6) CONTINUE文

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

```
CONTINUE
```

この文の実行によって正常の実行順序が継続されるものとする。

例：CONTINUE文

使用例：

```
DO 10 I= 1 , 10
IF(K(J)－I)10, 20, 20
10 CONTINUE
    ⋮
```

文の効果：

```
    ⋮
   文 1
 CONTINUE
   文 2
    ⋮
```

この文は，つぎのものと同じ効果を持つ。

```
    ⋮
   文 1
   文 2
    ⋮
```

以上示された規格はプログラム作成における文法一般に関する事である。したがって具体的な計算機システムにおいてどのように処理されるかという点に関しては触れられていない。実際には次の点で計算機システムに依存する。

① コンパイル時におけるエラーに対する取扱い方法とエラー・メッセージの出力の仕方。

② DO文実行中に発生したエラーの症状。

③ DO文に対する具体的な処理方法。

主記憶装置上での位置や，制御変数・初期値パラメタ，終値パラメタ，増分パラメタ等の取扱いもこれに含まれる。

とくに項目③に関してはFORTRAN-700[2)]においては次のような補足事項がある。

FORTRAN-700 における文法上の補足事項

(1) DO文は最大63個の入れ子をなしてもよい。この制限を越えるとコンパイル時に致命

的エラーとなる。（7.1.3(2)(2)）を参照されたい。

(2) DO の拡張範囲に DO 文があったり，DO の範囲外から DO の範囲内に実行の制御が渡ってもエラーは検出されない。実行時に不確定な動作を起す。……（文法7.1.2(8)）

(3) 入出力文における DO 形並びは最大63重まで許される。DO 文の範囲内にある入出力文における場合は，DO の入れ子の回数と合計したものが63重を越えてはならない。
……（文法7.1.3(2)(2)）

(4) DATA 文の DO 形並びは，最大63重の入れ子をなしていてもよい。……（文法7.4）

(5) 2つの DOCHK 文により指定された範囲が重複してはならない。……（文法11(9)）

(6) 1つのプログラム内で配列要素の参照の回数と DO 文の算術演算子を含む初期値パラメタ，終値パラメタおよび増分パラメタの個数の合計が4095を越えると，コンパイルは中断される。 ……（その他）

（注）各項目の後にある数字は文献３）における参照すべき章，節の番号である。

これらは文法規則の上にさらに加えられた制限事項なので，プログラミングの際には注意しなければならない。

2.2 DO 文のバッグとその特徴

DO 文の性質は文法により規定されている。ところが文法で規定されていない手法又は文法に違反する手法を用いた場合には計算処理に異常を生じ概ねバッグとなる。このような DO 文に関するバッグを概観すれば次のようになる。

Ⅰ）DO 文は，DO 文の次にある実行文から DO 文で引用された文番号をもつ文までの間にある実行文のくりかえし実行を定義する。したがって DO 文の誤りは，これらのくりかえして定義された実行文全体に影響をもつため，その影響の範囲が他の文にくらべて広い（大きい）。

Ⅱ）DO 文には初期値パラメタ，終値パラメタと増分パラメタが必要である。これらのパラメタは定数，DO 文以前に定義された変数又は DO 文中での算術式のいずれかである。パラメタが変数又は算術式で与えられている場合には，これらの値が正整数であるか否かの判定が困難であり実行されるまでわからない。この結果，実行時になってはじめてエラーとなり，しかも計算機システムでは何らこの事実を検出しないままで計算を続行する事がある。

Ⅲ）DO 文の実行において，制御変数は，初期値（正整数でなければならない事になっている）からはじまって増分パラメタの大きさだけ加算がくりかえされ終値パラメタの値を超える時点までつづけられる。ところが実行中に副プログラムの引用で制御変数の値が変ったり，初期値・終値・増分パラメタが負値になった場合には文法違反である。しかし違反にもかかわらず異常処理を続行する事がある。

Ⅳ）DO 文が複数個重複して用いられる場合の実行の流れは複雑である。さらに拡張範囲が含まれている場合は，さらに複雑になる。このような場合には DO 文を単にくりかえしの定義文と考えず，むしろ広い意味で流れを制御する文であると考えたほうが，デバッグの

際には都合の良い場合も多い。

以上の事は，特にフォートラン初心者にとっては，プログラム作成時の混乱のもとになるようである。したがってこのような事実をつねに意識しながらプログラミングとデバッグの手続きをすすめる事が望まれる。これらの具体的説明は第3節，第4節でおこなうのでそこと合せて見ると良い。

3．コンパイル時のエラー・メッセージ

コンパイル時には各プログラム単位ごとに文法違反のチェックがなされる。この節では(NEAC) FORTRAN-700 における DO 文に関係したエラーメッセージについて説明する。個々のメッセージを文献 2）より抜粋すると共に，各々に例題と補足説明をつけておく。

3.1 WARNING

文法違反ではないが，時には間違いとなる可能性のある文については“WARNING:” が 出力される。WARNING の出力された文は計算処理という点のみからみれば問題にならない事も多いが，場合によっては論理的な誤りになっているので注意する必要がある。

以下に FORTRAN-700 におけるウオーニング・メッセージ及びそれについての説明を示す。

メッセージ番号	説明
*022	WARNING: UNREFERENCED STATEMENT LABEL “label” ○ 実行文あるいは FORMAT 文に付けられた文の番号が，いずれの文によっても参照されていない。そのままコンパイルされる。 （プログラム コード — 01_8）

（例）

```
内部番号                フォートラン文
0002                      DO  I=1, 10
0003                10    CONTINUE
エラー・メッセージ
0002          131  SYNTAX ERROR IN DO STATEMENT
              022  WARNING:  UNREFERENCED
                   STATEMENT LABEL  “10”
```

この例は上記 WARNING が出力される典型的なものである。内部番号0002の文において DO 10 I=1,10 を誤った。このため0002は FATAL ERROR となり，この結果0003を引用する文がなくなる。但し修正されるべきは DO 文である。この WARNING は以下の場合に生じる事が多い。

① プログラム作成過程で引用する側の文のみが除去され，引用される文の文番号がそのままに放置された。

② 引用する文がFATAL ERRORとなって無視されたために引用される事がなくなってしまった（上の例はこの場合に相当する。）

③ 引用する側又は引用される側のいずれかの文において番号をまちがえた。

第③番目の場合には後に重要なまちがいとなる。

*025	WARNING: UNDEFINED VARIABLE "name" ○ 未定義の変数名が引用されている。即ち代入文の左辺，READ文あるいはDECODE文の入力並び，EQUIVALENCE文，NAMELIST並びの要素，COMMONの要素，DŌの制御変数，実引数あるいは仮引数のいずれとしても現われないか，あるいはDATA文や型宣言文によって初期値を設定されていない変数名がWRITE文の出力並び，ENCODE文の出力並び，代入文の右辺のいずれかに現われた。そのままコンパイルされる。 （プログラム　コード　-01_8）

（例）

```
内部番号          フォートラン文
0004              GO 15 K=1, 10
0005              F (K) =FLOAT (K)
0006        15    CONTINUE
エラー・メッセージ
0004      286  ILLEGAL EXPRESSION SYNTAX
            022  WARNING:  UNREFERENCED STATEMENT
                           LABEL "15"
            025  WARNING:  UNDEFINED VARIABLE "K"
```

DŌ文のつづりを間違えてGŌとしたためFATAL ERRORとなり，内部番号0004のDŌ文が無視された。このために制御変数が未定義となった。

なお，メッセージ番号022のWARNINGについては前の所で説明した例と全く同じ事情による。

3.2 FATAL ERROR

明らかに文法違反であるためにフォートラン文としては許されない文については"FATAL ERROR"としてエラー・メッセージが出力される。この時FATAL ERRŌRとなった文は無視され（すなわち無かったものとして）コンパイルはつづけられる。ただしFATAL ERRŌRのあったプログラムをリンクし実行するか否かはシステムに対するOPTION指定で可能なも

のもある。この点については今問題にしている事項ではないので省略する。

メッセージ番号	説明
*131	SYNTAX ERROR IN DO STATEMENT ○ DO 文に文法違反がある。この DO 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。（プログラム コード — 04_8）

（例） (1) DO J=1, 10

(2) DO 25 K=L, M,

例1 は DO-LOOP の範囲を示すための端末文番号の指定がない。例2 は終値パラメターの後のコンマが余分についているかあるいは増分パラメターが欠けている。

132	BACKWARD STATEMENT REFERENCE IN DO STATEMENT ○ DO 文の端末文が，その DO 文よりも前にある。DO 文に実行の制御が渡ると UEP になる。（プログラム コード — 04_8）

（例）

```
内部番号          フォートラン文
0007                GO TO 20
0008           20   K=I
0009                DO 20 I=1, 10
0010           20   CONTI NUE
エラー・メッセージ
0009      132 BACKWARD STATEMENT REFERENCE IN DO
              STATEMENT
0010      407 DUPLICATE STATEMENT LABEL "20"
```

この例は同じ文番号20を2度使った。

DO 文から見た場合，DO 文より前に端末文の番号が現われている。当然この2つの事項は，共にエラーである。メッセージ番号132 のエラーは，このようにして現れる場合が多い。

*）UEP とは，Unusual End of Programの略である。プログラムが異常な状態になった場合には account list にこのメッセージが出力され，実行は中断される。

134	CONTROL VARIABLE "name" REDEFINED IN DO RANGE ○ DO 文の制御変数が，拡張範囲を除くその DO の範囲の中で再定義されている。即ちその制御変数が代入文の左辺，READ 文の入力並び，DECODE 文の入力並びのいずれかに現われているか，内側の DO 文の制御変数あるいは入出力文の DO 形並びの制御変数として使用されている。再定義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。　　（プログラムムコード　— 04_8）

（例）

```
内部番号          フォートラン文
0011              DO 25 I=1, 10
0012              I=1+5
0013           25 CONTINUE
エラー・メッセージ
0012      134 CONTROL VARIABLE "I" REDEFINED IN
              DO RANGE
```

DO の範囲内で制御変数を再定義してはならない。例では DO の制御変数 I を I=I+5 と再定義している。

135	LIMIT OF NESTED DO STATEMENTS EXCEEDED ○ DO 文の重ねが63重を越している。64番目以上の DO 文はすべて削除されてコンパイルされる。プログラムの実行時に64重目以上の DO 文に実行の制御が渡ると UEP になる。入出力文における DO 形並びも含む。　　（プログラム　コード　— 04_8）

この ERROR は前節の最後に述べた文法上の補足事項に対する文法違反である。

136	NON-INTEGER USED AS DO CONTROL VARIABLE ○ DO 文の制御変数が整数型の変数名でない。この DO 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 DO 文に実行の制御が渡ると UEP になる。 （プログラム　コード　— 04_8）

(例)　　内部文番号　　フォートラン文

```
0014            DO 30 F=1, 10
0015       30 CONTINUE
```

エラー・メッセージ

```
0014      136 NON- INTEGER USED AS DO CONTROL
                VARIABLE
```

制御変数は整数型でなければならないが，この例では実数型となっているため誤りである。

137	ILLEGAL NESTING OF DO LOOPS ○ DO の入れ子にくい違いがある。プログラムの実行時にくい違いのある DO 文の最初の端末文に実行の制御が渡ると UEP になる。 （プログラム　コード　— 04_8）

(例)

```
      DO 115 I=1, 10
       ⋮
      DO 116 J=1, 20
       ⋮
115 CONTINUE
       ⋮
116 CONTINUE
```

DO 文の範囲が115 に対するものと116 に対するものでくい違っている点で文法違反である。
(p. 3 の DO の範囲が重複する場合についての文法規則を参照せよ)

138	ILLEGAL STATEMENT TERMINATING A DO LOOP ○ DO の端末文として GO TO 文，算術 IF 文，RETURN 文，STOP 文あるいは DO 文かこれらのうちの1つを含んだ論理 IF 文が使用されている。この端末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。　（プログラム　コード　— 04_8）

(例)　　内部番号　　フォートラン文

```
0016            DO 26 I=1, 10
0017       26 GO TO 27
0018       27 K=I
```

エラー・メッセージ

0017　　138　ILLEGAL STATEMENT TERMINATING A DO LOOP

端末文がGO TO文になっている。もしGO TO文をDOの範囲内で使う必要があるならば，次のようにすればよい。

```
      DO 26 I=1, 10
         ⋮
      GO TO 27
   26 CONTINUE
         ⋮
   27 K=I
```

221	ILLEGAL STATEMENT TYPE FOLLOWS LOGICAL IF ○ 論理IF文に許されない文が含まれている。この論理IF文に含まれている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論理IF文の論理式の値が真のときUEPになる。 （プログラム　コード　－04_8）

（例）　　IF(.NOT. L(I)) DO 120 I=1, 15

論理IF文にはDO文を含んではならない。論理IF文に含まれてもよい文は，DO文と論理IF文を除く実行文である。

251	UNDEFINED STATEMENT LABEL REFERENCE TO "label" ○ 実行文かデバッグ文で参照されている文の番号が実行文で定義されていない。実行文の場合はその文が削除されてコンパイルされる。プログラムの実行時にその文に実行の制御が渡るとUEPになる。デバッグ文の場合はそのデバッグ文を含むプログラム単位はGOファイル上に出力されない。 （プログラム　コード　－04_8（実行文）） （　〃　－10_8（デバッグ文））

(例)　内部文番号　フォートラン文

```
0019          DO 35 I=1, 10
0020       35 COTNUE
```

エラー・メッセージ

```
0019      251 UNDEFINED STATEMENT LABEL REFERENCE
              TO "35"
0020      401 UNRECOGNIZABLE STATEMENT
```

内部文番号0020はCONTINUE文であるがつづりを間違えたためFATAL ERRORとなって無視された。このためDO文における文番号35は無定義となる。

252	ILLEGAL SYNTAX IN STATEMENT LABAL REFERENCE "label" ○ 参照された文の番号がゼロばかりより成るか6個以上の数字より成る。または文の番号として許されない文字が使用されている。実行文の場合はその文が削除されてコンパイルされる。プログラムの実行時にその文に実行の制御が渡るとUEPになる。デバッグ文の場合にはそのデバッグ文を含むプログラム単位はGOファイル上に出力されない。 （プログラム　コード　— 04_8（実行文）） （　〃　— 10_8（デバッグ文））

(例)　内部文番号　フォートラン文

```
0021          DO 0000 I=1, 10
0022       37 CONTINUE
```

エラー・メッセージ

```
0021      252 ILLEGAL SYNTAX IN STATEMENT LABEL
              REFERENCE
          022 WARNING: UNREFERENCED STATEMENT
              LABEL "37"
```

DO文における引用すべき端末文番号がすべてゼロより成る事はゆるされない。WARNINGはDO文の誤りが原因で生じたものである。

278	SYNTAX ERROR IN SUBSCRIPT EXPRESSION FOR { "DO-PARAM" / "name" } ○ 配列要素の添字式かまたはDOのパラメタの形式に誤りがある。DATA文における場合は，この配列要素を含む並びとそれに対応する定数の並びはないものとしてコンパイルされ，この文を含むプログラム単位はGOファイル上に出力されない。実行文中における場合とDO文のパラメータの場合は，この文は削除されてコンパイルされ，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。デバッグ文における場合は，デバッグ文は削除されてコンパイルされ，プログラムの実行時にそのデバッグ文により指定された範囲内の実行文に実行の制御が渡るとUEPになる。 備考：DATA文中における添字式の許される形式は，実行文とデバッグ文におけるそれより範囲がせまい。 （プログラム　コード　— 04_8（実行文 / デバッグ文）） （　〃　— 10_8（DATA文）） 注：メッセージ中の"DO-PARAM"はDOのパラメータの形式の誤りの場合である。

（例１）　内部文番号　　フォートラン文

```
0001          DIMENSION IMAX (10)
                    ⋮
0023          IMAX (1) =10
0024          DO 40 I=1, IMAX (1)
0025       40 CONTINUE
```

エラーメッセージ

```
0024      278 SYNTAX ERROR IN SUBSCRIPT EXPRESSION
              FOR "DO-PARAM"
0024      131 SYNTAX ERROR IN DO STATEMENT
```

DO文のパラメタに配列要素を用いる事はFORTRAN-700では禁止されている。

（例２）　内部文番号　　フォートラン文

```
0026          DO 42 I=1, 20/3
0027       42 CONTINUE
```

エラー・メッセージ

```
0026      278 SYNTAX ERROR IN SUBSCRIPT EXPRESSION
              FOR "DO-PARAM"
```

DO 文における初期値パラメタ，終値パラメタ及び増分パラメタは算術式で表わせる。但し割算は許されない。

279	NON-INTEGER SUBSCRIPT EXPRESSION FOR { "DO-PARAM" / "name" } ○ 配列要素の使用で，添字式の中に整数型でない定数あるいは変数名が引用されているか DO 文のパラメタ中で整数型でない変数が引用されている。DATA 文における場合はその配列要素を含む並びがないものとしてコンパイルされ，EQUIVALENCE 文における場合は，この配列要素を含む並びがないもとしてコンパイルされ，いずれの場合もこの文を含むプログラム単位は GO ファイル上に出力されない。実行文における場合は，その実行文が削除されてコンパイルされ，プログラムの実行時にその文に実行の制御が渡ると UEP になる。デバッグ文における場合は，そのデバッグ文は削除されてコンパイルされ，プログラムの実行時にそのデバッグ文により指定された範囲内の実行文に実行の制御が渡ると UEP になる。 （プログラム　コード　— 04_8（実行文／デバッグ文）） （　〃　— 10_8（非実行文）） 注：メッセージ中の "DO-PARAM" は DO のパラメタに関する誤りの場合である。

（例１）

内部文番号　フォートラン文

```
0028          DO 44 I=1, 10. 5
0029       44 CONTINUE
```

エラー・メッセージ

```
0028    279 NON- INTEGER SUBSCRIPT EXPRESSION
            FOR "DO- PARAM"
```

（例２）

内部文番号　フォートラン文

```
0030          F=10. 6
0031          DO 45 I=1, F
0032       45 CONTI NUE
0033          STOP
0034          END
```

エラー・メッセージ

```
0031    279 NON- INTEGER SUBSCRIPI EXPRESSION
            FOR "DO- PARAM"
```

DO 文のパラメタは整数型でなければならない。例1は実数型定数の例である。例2は実数型変数の例である。

以上は特に DO 文にのみ関係した ERROR であった。最後に DO 文に限らず FORTRAN 文全般に可能性のあるエラー（又はエラー・メッセージ）について示す。

255	FORTRAN NAME LONGER THAN 6 CHARACTERS ○ 英字名が6文字を越えている。 実行文の場合にはその文が削除されてコンパイルされる。プログラムの実行時にその文に実行の制御が渡ると UEP になる。非実実行文の場合左端より6文字とられてコンパイルされるが，その文を含むプログラム単位は GO ファイル上に出力されない。デバッグ文の場合にはそのデバッグ文は削除されてコンパイルされ，プログラムの実行時にそのデバッグ文により指定された範囲内の文に実行の制御が渡ると UEP になる。 （プログラム　コード　— 04_8（実行文／デバッグ文）） （　　〃　　— 10_8（非実行文））

256	VARIABLE NAME “name” FOLLOWED BY LEFT PARENTHESES ○ 変数名の後に左かっこがある。実行文の場合にはその文が削除されてコンパイルされる。プログラムの実行時にその文に実行の制御が渡ると UEP になる。非実行文の場合にはその文が削除されてコンパイルされるが，その文を含むプログラム単位は GO ファイル上に出力されない。デバッグ文の場合にはそのデバッグ文は削除されてコンパイルされ，プログラムの実行時にそのデバッグ文により指定された範囲内の文に実行の制御が渡ると UEP になる。 例： DO 10 I=1, 10 A=I (J) ⋮ 10 CONTINUE （プログラム　コード　— 04_8（実行文／デバッグ文）） （　　〃　　— 10_8（非実行文））

400	ILLEGAL CHARACTER IN STATEMENT ○ 文の中で FORTRAN 用でない文字が使用されている。ただし文字定数または文字欄記述子として使われた場合は除く。この文字は削除されてコンパイルされるが，この文を含むプログラム単位は GO ファイル上に出力されない。 （プログラム　コード　— 10_8）

401	UNRECOGNIZABLE STATEMENT ○ FORTRANのいかなる文にも該当しない文である。この文は削除されてコンパイルされるが，この文を含むプログラム単位はGOファイル上に出力されない。 （プログラム　コード　— 10_8）

701	NUMBER OF ARRAY ELEMENT REFERENCE AND DO PARAMETERS EXCEEDS THE LIMIT ○ 配列要素の参照の回数とDO文の算術演算子を含む初期値パラメタ，終値パラメタおよび増分パラメタの個数が合計4095を越えた。コンパイルは中断される。 （プログラム　コード　— 20_8）

4．実行時におけるDO文のふるまい

DO文のプログラム中での表現形式は

```
      DO  10  I=IMIN, IMAX, ISTEP
                    ⋮
   10  CONTINUE
                    ⋮
```

である。但し　　I；制御変数

IMIN；初期値パラメタ，

IMAX；終値パラメタ，

ISTEP；増分パラメタ。

実行時におけるDO文のふるまいについては2節の文法説明においてすでに基本的なものは示された。

実行時におけるDO文の不測の事態をつぎの4つの事項について詳しく検討してみたい。

(1) 初期値パラメタ，終値パラメタ，及び増分パラメタは，いずれも定数の場合には，バッグの発見はコンパイル時になされる。しかし，これらパラメタが変数又は算術式で表わされている場合には実行時に到るまでバッグになり得るか否かの判断がつかない。

(2) 次に制御変数に関しては“再定義は禁止”されているが，しかしこのチェックはプログラム単位内についてのみなされる。このため副プログラム文がDOの範囲で引用されている場合には，制御変数の再定義は副プログラムを通してなされる事がある。しかも計算機によるチェックはされないのでプログラム作成者自身でしらべるしかない。

(3) さらにDO文はしばしば配列の値の定義の際に利用される。このためDO文の異常が配

列要素の値の異常として表面にあらわれる事がある。

(4) 最後に問題になるのは単純GO TO文，算術IF文，論理IF文でのGO TO文，割当型GO TO文及び計算型GO TO文のいずれかにより DOの範囲外から DOの範囲内へ（拡張範囲としてではなく）飛び込みをおこなった場合の DO文のふるまいである。

以上，4つの点が DO文における（または DO文にかかわる）実行時のバッグとして経験され，問題となりやすいものである。

本節においては DO文において発生するプログラミング上の上記4つの問題点を，具体例をあげながら，明らかにする。

4.1 初期値パラメタ，終値パラメタと増分パラメタについて

フォートラン文法においては DO文のパラメタが変数又は算術式の場合には，その値の正負については実行されるまで確認できない。ここでは，このパラメタの不確定性がプログラムのバッグとしてどのように現れてくるかを調べる。

図4.1.1 初期値パラメタの値に対する DO文のふるまいを調べるための例題。

```
      ⋮
      DO  10  I=IMIN, 10
   10 WRITE (6, 300)  I
  300 FORMAT (1H , 10X, ' I=', 15)
      STOP
      END
```

最初に初期値パラメタの様々の値に対する DO 文のふるまいについて調べる。図4.1.1には，このテストのために用いたプログラムを示した。制御変数Iは WRITE文により出力される。この制御変数の変化により DOの範囲の実行されるようすを見ることができる。図4.1.2には，初期値パラメタ IMINの値の3例について図4.1.1のプログラムで出力された結果を示した。

図4.1.2 例題図4.1.1における初期値パラメタ IMINの種々の値に対するWRITE文による出力結果

IMIN=－4の場合	IMIN=4の場合	IMIN=14の場合
I=－4	I=4	I=14
I=－3	I=5	
I=－2	I=6	
I=－1	I=7	
I=0	I=8	
I=1	I=9	
I=2	I=10	
I=3		
I=4		
I=5		
I=6		
I=7		
I=8		
I=9		
I=10		

初期値パラメタが負の場合（IMIN＝－4）

最初に制御変数（I）は初期値パラメタの値（＝－4）に取られDOの範囲を実行する。端末文（文番号10）に達すると制御変数に増分パラメタ（＝1）が加えられ，さらに終値パラメタ（＝10）と比較されて制御変数が終値パラメタより小さいか等しければDOの範囲のはじめにもどる。

以下この操作がくりかえされる。制御変数（I）が終値パラメタ（＝10）を超えた時点でDOの範囲の実行は終る。FORTRAN-700では初期値パラメタが負になった事はシステムではチェックされていない。（但し4.1.3節においてこの事実と必ずしも合致しない現象が示されるのでそこも参照する事）

初期値パラメタ（IMIN＝4）が終値パラメタ（＝10）を超えない正整数の場合

最初に制御変数は初期値パラメタに等しく取られDOの範囲を実行する。端末文に達すると制御変数（I）に増分パラメタの値（＝1）が加えられ，さらに終値パラメタと制御変数を比較して制御変数が終値パラメタより小さいか等しければDOの範囲の最初に帰る。制御変数が大きければDOの範囲の実行をやめる。この結果 I＝4，5，…，9，10とDOの範囲が実行された。

初期値パラメタ（IMIN＝14）が終値パラメタ（＝10）より大きい場合

最初に制御変数（I）は初期値パラメタ（IMIN＝14）に等しく取られ，DOの範囲が実行される。端末文（文番号10）に達すると制御変数に増分パラメタの値（＝1）が加えられ，さらに制御変数（I＝15）は終値パラメタと比較され，終値パラメタより大きい事が確認されてDOの範囲の実行は終る。とくにFORTRAN-700では，初期値パラメタが終値パラメタを超える場合には，制御変数が初期値パラメタの値に等しい場合についてのみ1回実行される。

終値パラメタ，増分パラメタについても上記説明と同様の議論になるので省略する。

このようにコンパイル時にエラー・メッセージが出力されないにもかかわらず文法違反を犯している事がある。あるいは文法違反ではないが，パラメタの誤りのためDO文の制御変数の値が予期しているものとは違ったふるまいをする事がある。このように“予期に反する”場合が“バッグ”であると言えよう。したがって図4.1.2のような状況に対してバッグであるか否かは，プログラムの背景にある意味に依存するので，プログラム作成者の判断によるほかない。要約すれば，バッグについて次のようにまとめる事ができる。

(1) DOの制御変数のふるまいの異常は，初期値パラメタ，終値パラメタ又は増分パラメタのすべて又はいずれかの誤りに原因をもつ。

(2) 初期値パラメタ，終値パラメタ及び増分パラメタにおける誤りは，

(2) DO文自体の誤り。

(3) DO 文のパラメタに関係のある変数の代入文の誤り。

(4) DO 文のパラメタに関係のある変数の入力の誤り。

(5) 主記憶に記憶されたプログラム部分の破壊。

のいずれかに原因を持つ。

さらに DO の制御変数のふるまいの異常は，次のような状況又は症状を引き起こす。

(6) DO のくりかえしが異常に多い。

このために

(7) CPU-TIME が異常に長くなる。

あるいは逆に

(8) DO のくりかえしが異常に少い。

この結果

(9) CPU-TIME が異常に短かくなる。

しかも(6)及び(8)に共通しておこす事は

(10) 変数値，配列値の異常

である。

上記説明文中～～部分はバッグ及びその症状でありそこにつけられた番号はバッグ整理のための番号である。これらを図式化すれば図４.1.3になる。

図４.1.3　DO 文のパラメターにかかわるバッグ

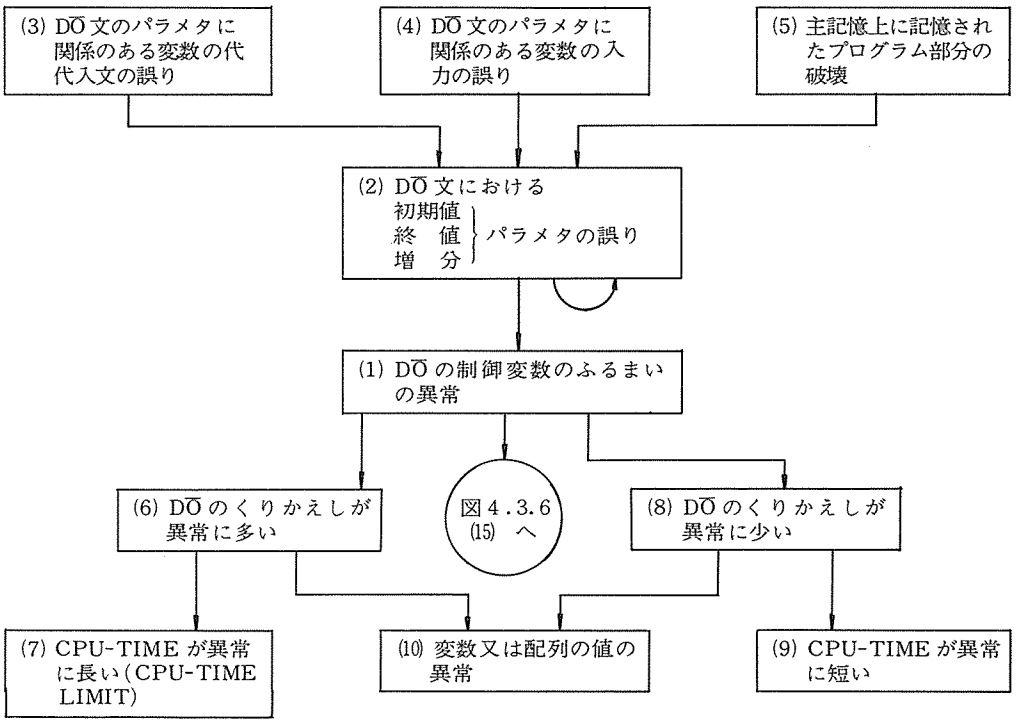


図4.1.1及び図4.1.2の側から図4.1.3の結論を引きだすための論理には飛躍があるが単に説明を煩雑にするだけなのでここでは説明を省く。

4.2 DO文における制御変数の再定義

DOの範囲内でDO文の制御変数を再定義してはならない。すなわち制御変数Iを用いたDOの範囲内で

I＝e　（ただしeは整数型算術式）

なる代入文があらわれてはいけない。（制御変数が左辺にあらわれてはいけない。）この誤りはコンパイル時にプログラム単位についてのみ調べられエラー・メッセージ番号134の出力により知る事ができる。この事はDOの範囲内で実行の流れが副プログラムに移る場合については適用されない。ここでは，副プログラムの引用の際に副プログラム文中の引数として用いられた制御変数，初期値パラメタ及び終値パラメタが副プログラム中で再定義される場合についてそのふるまいを示す。

図4.2.1　DOの範囲内で制御変数，初期値パラメタ及び終値パラメタの値が再定義される場合の例題

主プログラム

```
      DATA  IMIN, IMAX/ 1, 10/
      DO  10  I=IMIN, IMAX
      A=F (I, IMIN, IMAX)
      WRITE (6, 100)  A, I, IMIN, IMAX
   10 CONTINUE
  100 FORMAT (1H , '  A=', F10. 5, 5X, 'I=', I5, 5X, 'IMIN=', I5, 5X,
     'IMAX=', I5)
      STOP
      END
```

副プログラム

```
      FUNCTION  F (I, IMIN, IMAX)
                :
      RETURN
      END
```

図4.2.1は主プログラム及び関数副プログラムFより成るプログラムの例である。関数副プログラムの引数としてはDOの制御変数I，初期値パラメターIMIN及び終値パラメターIMAXが取られている。ただしIMIN及びIMAXはDATA文により

```
DATA  IMIN, IMAX/1, 10/
```

で与えられている。制御変数及びDO文のパラメタを仮引数として用いる事は誤りではない。しかし副プログラム中で制御変数Ｉを再定義した場合には，

「制御変数は初期値からはじまって順次増分パラメタの値づつ増加して終値パラメタ・の値づつ増加して終値パラメタに達するようには変化しない」

結果になる。この事を具体的に示したのが図４.２.２と図４.２.３である。図４.２.２及び図４.２.３における出力結果は，いずれも図４.２.１の主プログラム中にあるWRITE文によるものである。

図４.２.２　関数Ｆの中で制御変数Ｉが
Ｉ＝Ｉ＋１
により再定義される例。上に示した副プログラム及び出力結果は，図４.２.１のプログラムによる。

副プログラム

```
FUNCTION  F (I, IMIN, IMAX)
F=FLOAT (I)
I=I+1
IMIN=IMIN
IMAX=IMAX
RETURN
END
```

主プログラム（図４.２.１）による出力結果

```
A=    1. 00000      I=     2      IMIN=    1      IMAX=    10
A=    3. 00000      I=     4      IMIN=    1      IMAX=    10
A=    5. 00000      I=     6      IMIN=    1      IMAX=    10
A=    7. 00000      I=     8      IMIN=    1      IMAX=    10
A=    9. 00000      I=    10      IMIN=    1      IMAX=    10
```

図４.２.３　関数Ｆの中で制御変数Ｉが

Ｉ＝Ｉ－１

により再定義される例。副プログラムＦは図４.２.１により引用される。
下段の出力結果は図４.２.１の主プログラム中のWRITE文による。

```
  副プログラム
FUNCTION  F (I,  IMIX,  IMAX)
F=FLOAT (I)
I=I−1
IMIN=IMIN
IMAX=IMAX
RETURN
END
```

主プログラム（図４.２.１）による出力結果

```
A=    1. 00000      I=    0      IMIN=    1      IMAX=    10
A=    1. 00000      I=    0      IMIN=    1      IMAX=    10
                      ⋮
              （無限にくりかえされる）
                      ⋮
                      ⋮
                      ⋮
                      ⋮
```

図４.２.２における例は，関数Ｆの中で制御変数Ｉが

Ｉ＝Ｉ＋１

によって再定義されたものである。このプログラムの振舞いは次のようになる。まず主プログラムにおいてDO̅の制御変数Ｉは初期値パラメタ　（＝１）に等しくとられる。次に関数Ｆの引用によりプログラムの実行は関数Ｆに移る。関数Ｆの中で　Ｉ＝Ｉ＋１　により制御変数は２に変って再び実行は主プログラムに帰る。この時に実行されたWRITE文の出力が出力結果の第１行目である。変数Ａは，プログラム実行の流れが正しく副プログラムに移った事を確認するために出力した。実行が端末文10 CONTINUEに達すると制御変数（Ｉ＝２）は増分パラメタ　（＝１）だけ増加して３となりDO̅の範囲の最初の実行文になる。以下この操作がＩＩ＞10までつづけられる。この結果が図４.２.２の出力結果である。制御変数Ｉが１つ飛びになっているのは，説明したように，制御変数が関数中で１加えられ，さらに端末文で１加えられた結果である。

図４.２.３には，関数Ｆの中で制御変数Ｉが

$$I = I - 1$$

によって再定義された場合の例である。プログラム実行の流れは図４.２.３の場合と同じなので省略する。図４.２.２と図４.２.３との違いは，前者の制御変数がDOの範囲の実行を１回終るごとに２づつ増加するのに対して後者は全く増加しない点である。図４.２.３における出力結果に示したようにWRITE文の所での制御変数Ｉの値は常にゼロである。この結果，制御変数は永久に終値パラメタに到達せずDOの範囲の実行が無限にくりかえされる。

最後に，初期値パラメタと終値パラメタが，DOの範囲内で引用された副プログラム中で再定義される例を図４.２.４に示す。パラメタの再定義にもかかわらず，DO文の実行はDO文実行直前に定められたパラメタの値に従ってなされている。FORTRAN-700に関する限り，パラメタの再定義は深刻なエラーになる心配はないであろう。

以上の説明をまとめれば次のようになる。

(11) 副プログラムの引用の際に，コモン領域*)又は仮引数**)にDOの制御変数を含んでいる。かも制御変数は副プログラム中で再定義されている。

＊）コモン領域とは，主プログラム及び副プログラムの集まりに対して共通に用いられる変数や配列の値を記憶するために，プログラム間に共通にとられた領域である。

＊＊）仮引数とは，引用するプログラムから引用されるプログラム値を転送するためにとられた変数又は配列である。通常，副プログラム名の右にカッコでくくって示される。

図４.２.４ 4初期値パラメタ(IMIN) 及び終値パラメタ(IMAX) が副プログラムFの中で再定義される例。

副プログラム

```
FUNCTION F (I, IMIN, IMAX)
F=FLOAT (I)
I=I
IMIN=IMIN+2
IMAX=IMAX−2
RETURN
END
```

主プログラム(図４.２.１)による出力結果

A=	1. 00000	I=	1	IMIN=	2	IMAX=	9
A=	2. 00000	I=	2	IMIN=	4	IMAX=	7
A=	3. 00000	I=	3	IMIN=	6	IMAX=	5
A=	4. 00000	I=	4	IMIN=	8	IMAX=	3
A=	5. 00000	I=	5	IMIN=	10	IMAX=	1
A=	6. 00000	I=	6	IMIN=	12	IMAX=	−1
A=	7. 00000	I=	7	IMIN=	14	IMAX=	−3
A=	8. 00000	I=	8	IMIN=	16	IMAX=	−5
A=	9. 00000	I=	9	IMIN=	18	IMAX=	−7
A=	10. 00000	I=	10	IMIN=	20	IMAX=	−9

この結果として

(12) DO の範囲において制御変数の増分及び制御変数の変化域が初期値パラメタ，終値パラメタ及び増分パラメタで指定したものと一致しない。

これら不一致の様子は，次のように再定義の違いとして区別できる。

(13) DO 文の制御変数が I=I−N (N７０) の形式で再定義されている。

この場合には制御変数の値が前進しないために

(6) DO のくりかえしが異常に多い。

結果になる事が多い。一方，次の状況もある

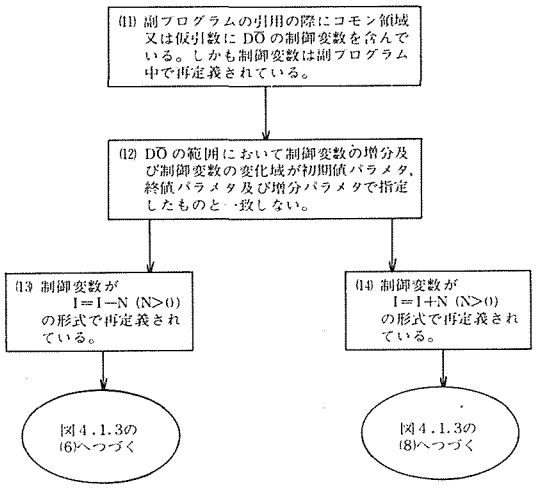
(14) DO 文の制御変数が I=I+N(N＞０) の形式で再定義されている。

この場合には制御変数の値が増分パラメタの値以上に前進するために，

(8) DO のくりかえしが異常に少い。

結果になる。ただし(6)と(8)は図４.１.３ですでにあらわれたものである。これらを図式化したものが図４.２.５である。

図4.2.5　制御変数の再定義にかかわるバッグ



4.3　DOの範囲中で配列要素の値を定義する事について

配列は一連の変数の系列をまとめたものである。このように変数を配列としてまとめた場合，各配列要素は数学的に同じ性質（又は同じ関数）で表わせられることが多い。したがってこのような配列の特徴及びDO文のもつ機能から，DO文を用いて配列要素の値を次々と定義する事はしばしばおこなわれる。一方配列の大きさは宣言文で定められ，実行にあたっては引用又は代入文のいずれの場合においても定義域をはみだしてはならない。ここでは，これらの規則がまもられなかった場合の現象について解説する。

図4.3.1　DO文の実行によって配列要素の値を定義する。

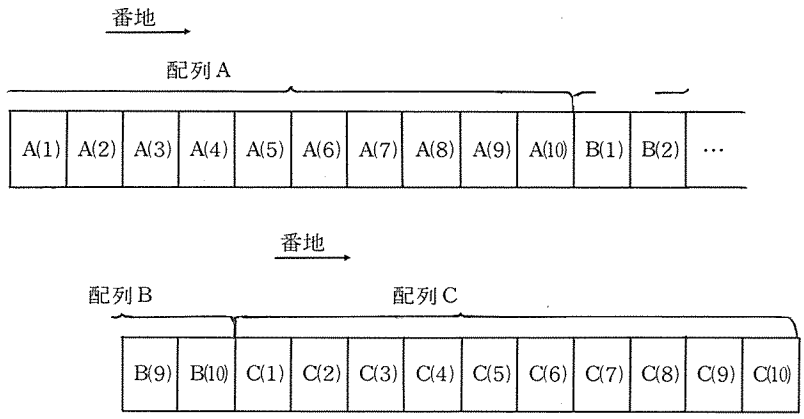
```
      INTEGER     A (10), B (10), C (10)     INTEGER
      DATA   (A (I), I=1, 10), (C (J), J=1, 10) / ▼A1▼, ▼A2▼, ▼A3▼,
             ▼A4▼, ▼A5▼, ▼A6▼, ▼A7▼, ▼A8▼, ▼A9▼, ▼A10▼, ▼C1▼, ▼C2▼,
             ▼C3▼, ▼C4▼, ▼C5▼, ▼C6▼, ▼C7▼, ▼C8▼, ▼C9▼, ▼C10▼ /
      DATA  NMIN, NMAX/1, 10/
      DO  10  N=NMIN, NMAX
   10 B (N) =N
      WRITE (6, 100)    (A (I), I=1, 10), (B (J), J=1, 10), (C (K), K=1, 10)
  100 FORMAT (1H, 5X, ▼A (I) =▼, 10 (2X, A3, ▼,▼ ) /
     1                5X, ▼B (I) =▼, 10 (2X, I3, ▼,▼ )/
     2                5X, ▼C (I)=▼, 10 (2X, A3, ▼ ,▼ ) )
```

上図プログラムによる出力結果

```
A (I) =  A1 ,   A2 ,   A3 ,   A4 ,   A5 ,   A6 ,   A7 ,   A8 ,   A9 ,   A10
B (I) =    1,     2,     3,     4,     5,     6,     7,     8,     9,    10
C (I) =  C1 ,   C2 ,   C3 ,   C4 ,   C5 ,   C6 ,   C7 ,   C8 ,   C9 ,   C10
```

例題として配列A(10), B(10)及びC(10)に各々値を定義するプログラムを考える。配列A(10)とC(10)はDATA文*)により定義する。配列B(10)の値をDO文により与える。これをプログラム化したのが図4.3.1である。DO文の初期値パラメタ及び終値パラメタは各々DATA文により与えられている。配列A, BとCの値はプログラム実行の後にWRITE文により出力される。図4.3.1の後半にはこのプログラムによって出力された結果が示されてある。なおプログラム中で配列A, B, Cの相対的位置は図4.3.2に示したとうりである。配列要素はA(1)からはじまりA(2), A(3)…A(9), A(10)とすすむ。A(10)のすぐ隣りからB(1), B(2), …

図4.3.2　プログラム図4.3.1における配列A(10), B(10), C(10)の主記憶装置上での相対的位置



とはじまりB(9), B(10)と続く。B(10)の後にはC(1), C(2), …が配置され, …C(9), C(10)と並ぶ。

図4.3.1のプログラムにおいて問題になるのはDO文の制御変数Nが配列Bの定義域を越えてN≦0またはN≧11になる場合である。この場合配列Bに対しては定義されていない領域へ強制的に値を代入しようとするため異常状態になる。この状態は次の3つの場合に分けて考える事ができる。

CASE1; 初期値パラメタが負になる。

*) DATA文は引数又はコモン文中にない変数又は配列の初期値を定義するのに用いられる。通常, 代入文の代用にされる。したがって, 整数型・実数型・文字型等いずれの型に対しても用いる事ができる。

例　　DATA NMIN, NMAX/−5, 10/

CASE2; 終値パラメタの値が配列Bの大きさを少し越える。

例　　DATA NMIN, NMAX/1, 12/

CASE3; 終値パラメタの値が配列Bの大きさを充分に越える。

例　　DATA NMIN, NMAX/1, 40/

図4.3.1のプログラムにおけるDATA文を各々CASE1,2及び3で置き換えた場合について以下に調べる。

CASE1：初期値パラメタが負になる。

図4.3.3　プログラム図4.3.1において初期値パラメターを負にした場合の出力結果。

DATA NMIN, NMAX/−5, 10/

出力結果

A (I) =A1 , A2 , A3 , A4 , ÇÇÇ, A6 , A7 , A8 , A9 , A10
B (I) =　0,　0,　0,　0,　0,　0,　0,　0,　0　0
C (I) =C1 , C2 , C3 , C3 , C4 , C5 , C6 , C7 , C8 , C9 , C10

DŌ文のパラメタは　DATA NMIN, NMAX/−5, 10/　で与えられる。このデータ文により与えられたパラメタによる 図4.3.1 のプログラムの実行結果は図4.3.3に示した出力結果により知る事ができる。図4.3.1の出力結果との差異は，配列要素 A(5) の値が壊されている事と，配列Bの要素の値がすべてゼロである事である。配列要素A(5)は，図4.3.2 に示した主記憶上での相対位置から見るとB(−5) に相当する所にある（但しフォートランでは添字に対して負の整数を用いる事は禁止されている)。すなわち初期値 I=−5 の実行によってB(−5)の所，すなわち A(5) に値が代入された事になる。制御変数が負であるため以後のDŌの範囲の実行は中断されているためB(1), B(2), …, B(10) の値は与えられずゼロのままにのこされた。*

CASE2: 終値パラメタが配列Bの大きさを少し越える

DŌ文のパラメタは DATA NMIN, NMAX/1, 12/ で与える。 このデータ文を用いて

*備考）この事は4.1節の説明といく分矛盾する点である。しかし両者の間にはコンパイラーによる処理方法に差異があるためにこのような差異が生じたものと思われる。この点は，ここでの主題ではないので深く立入る事はしない。

図4.3.4　プログラム図4.3.1において終値パラメターを配列Bの大きさより少し大きくした場合の出力結果

```
                DATA NMIN, NMAX/1, 12/
                出力結果
A(I)=A1 , A2 , A3 , A4 , A5 , A6 , A7 , A8 , A9 , A10
B(I)=  1,   2,   3,   4,   5,   6,   7,   8,   9,  10
C(I)=000, 000, C3 , C4 , C5 , C6 , C7 , C8 , C9 , C10
```

図4.3.1のプログラムを実行した場合の結果は，図4.3.4において出力結果として示されている。配列AとBの値は正しい。ところが配列要素C(1)とC(2)の値が壊されている。配列BとCとの間の相対的位置は図4.3.2に示したとうりである。一方DO文の実行においては配列Bに対して

B(1), B(2), …, B(9), B(10), B(11), B(12)

と代入されてゆく。B(11)とB(12)は宣言文で定義されてはいないが，しかし，相対的位置から見て，

B(11)≡C(1)

B(12)≡C(2)

である。このためC(1)とC(2)の値が変えられてしまった。

CASE3：終値パラメタの値が配列Bの大きさを充分に越える

DO文のパラメタは　DATA NMIN, NMAX/1, 40/で与える。このデータ文を用いて図4.3.1のプログラムを実行した場合の結果は，図4.3.5に示した。ERROR MESSAGEの意味は，“FORMAT文が誤りである”という事である。前例題で見たように，これは元々正しい文なので，一見奇異な事である。しかしDO文の実行が

B(1), B(2), …, B(10), B(11), …, B(40)

と進められてゆく事を考えれば理解できよう。すなわちB(11), B(12)…, B(20)と次々と配列Cの領域へ値が代入され，B(21), B(22), …, B(40)はさらに配列Cの領域を越えて主記憶のさらに先へと値が強制的に入れられる。そして，やがてプログラムが記憶されている部分へも代入されてゆくためにプログラムは壊される。この結果壊された部分へプログラムの実行が移

図4.3.5　プログラム図4.3.1において終値パラメタを配列Bの大きさよりも充分大きな値に取った場合の出力結果

```
                DATA NMIN, NMAX/1, 40/
            次のエラー・メッセージが出力され実行は中断された。
**132D  INVALID FORMAT CODE ON FILE 06
        ERROR DETECTED IN NFSB1SQWTF        00025154
              CALLED FROM MAIPG       0006  00004434
****  MRMOOI     UEP 0010222 DEPENDENT
```

るとエラーとなる。

このように配列の値の代入とDO文が関係し合った場合の誤りはプログラムの構成及び実行の状況により現れ方が違ってくる。以上を整理すれば，次のようになる。

ここで取上げた誤りとは

(15) DOの範囲内で，配列に対して宣言文で定義した領域以外の所へ代入しようとした

場合であった。この原因としては，次の2つの場合が考えられる。

(16) 配列宣言の誤りのため，配列の大きさを小さくとりすぎた。

(2) DO文における初期値パラメタ，終値パラメタ，増分パラメタの誤り。

エラー(15)において，その程度を次のように2つに分けて考えると便利である。

(17) 配列への代入が，宣言文で定義した領域よりわずかにはみ出した。

18 配列への代入が，宣言文で定義した領域をはるかに越えてしまった。

さて，エラー(17)の場合には，図4.3.3及び図4.3.4の例に相当するだろう。したがってこの場合には，

(19) 問題となっている配列の近傍にある配列又は変数の値が変られてしまった。

一方，エラー(18)に対しては図4.3.5がその例である。この場合には

(20) プログラム部分が壊れた。

ために，

(21) 実行が中断される。

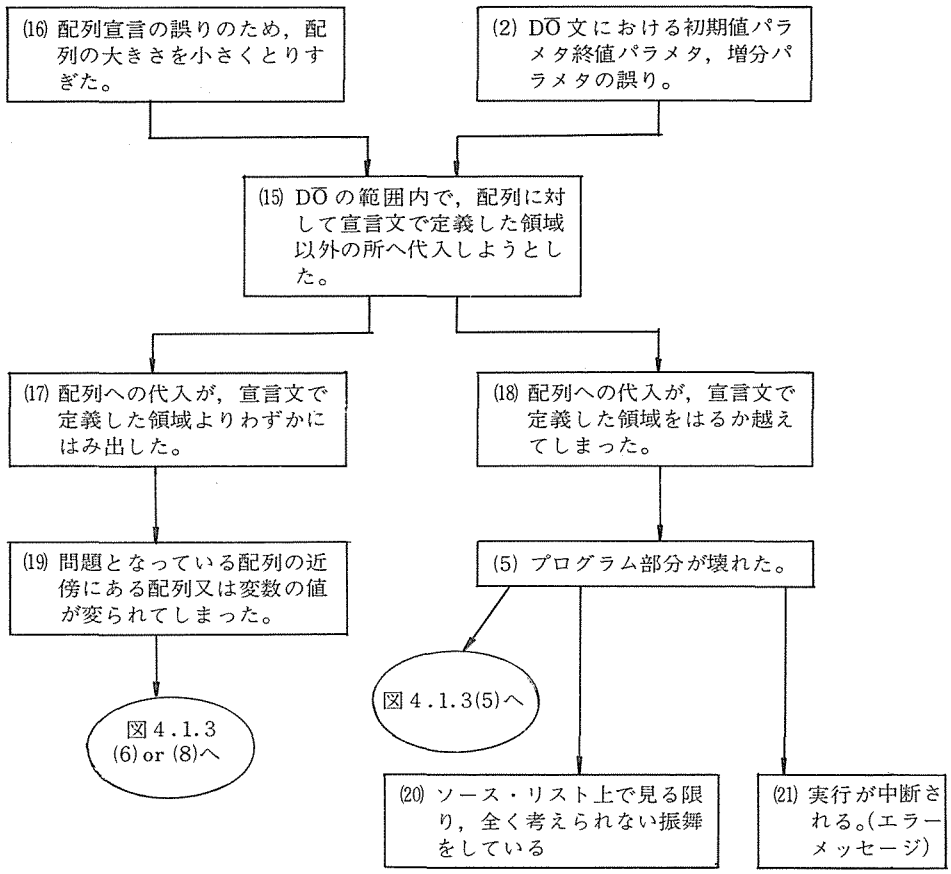
場合とか，

(22) ソース・リスト上で見る限り，全く考えられない振舞をしている。

等の状況になる。

これらを図式化してまとめたものが図4.3.6である。

図4.3.6 DO の範囲中で配列要素の値を定義する際に生ずるエラー



4.4 拡張範囲以外の DO の範囲外から DO の範囲へ飛び込む場合について

JIS FORTRAN（水準7000）では，DO の範囲外から範囲内へ DO 文を飛び越してはいってはならない。図4.4.1には，その流れを概略図として示した。拡張範囲と違って，DO の範囲からの飛び出しはなく，単に飛び込むだけである。図4.4.2には DO の範囲への飛び込みをもつフォートラン・プログラムの具体例を示した。

DO の範囲への飛び込みについては FORTRAN-700 では FATAL ERROR とはならない。このために実行時に思わぬバッグとして確認される可能性をもつ。このような理由により，DO の範囲外（拡張範囲は除外）から DO の範囲内へ飛び込む場合の状況について調べておくことは重要である。

図4.4.1 DO 文を飛び越して DO の範囲外から範囲内へはいる例。
文1は IF文又は GO TO 文。但し拡張範囲とは異なる。

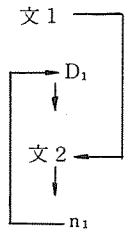


図4.4.2　拡張範囲以外の DO の範囲外から DO の範囲へ飛び込む場合のプログラム

```
      I=I1
      GO TO 80
      DO 90 I=1, 10
      IA=I
   80 IB=I*10
      WRITE (6, 200)  I, IA, IB
   90 CONTINUE
  200 FORMAT (1H ,  ▼I=▼, I2, 1X, ▼IA=▼, I2, 1X, ▼IB=▼,  I3)
```

DO の範囲外から範囲内への飛び込みにおいて最も重要な事は，DO の範囲内へ飛び込む直前における DO の制御変数に相当する変数の値である。図4.4.2に示したプログラムを用いて，最初に制御変数 I （または I 1）を次のように与えてその状況をしらべた。

(a)　制御変数 I を DO 文の終値パラメタより大きくした。（I 1＝14）

(b)　制御変数 I を DO 文の初期値パラメタと終値パラメタの間の値にした。（I1＝1）

(c)　制御変数 I を DO 文の初期値パラメタより小さく（又は負の値に）した。（I1＝－2）

これらに対するテスト結果は図4.4.3に示した。図中の出力結果は図4.4.2のプログラムにおける WRITE 文で出力されたものである。(a), (b) と (c) において共通な点は，次の通りである。

○DO の範囲の実行は，飛び込みを受けた文からはじまる。

○制御変数の値は初期値パラメタに関係なく，DO の範囲に飛び込む直前の値からはじまる。

○実行が DO 文の端末文に達すると制御変数に増分パラメターの値が加えられ，以後はあ

図4.4.3　プログラム図4.4.2における実行結果。変数 I 1 を入力とし，出力は DO の範囲にある WRITE 文によりなされたものである。FORTRAN 700 におけるコンパイラーは初期にすべての変数の値をゼロにセットする。このため IA の値は各々第1行目では実行されなかったためにゼロのままになっている。

(a)　I1＝14

```
I=14  IA=0  IB=140
```

(b)　I1＝1

```
I= 1  IA= 0  IB= 10
I= 2  IA= 2  IB= 20
I= 3  IA= 3  IB= 30
I= 4  IA= 4  IB= 40
I= 5  IA= 5  IB= 50
I= 6  IA= 6  IB= 60
I= 7  IA= 7  IB= 70
I= 8  IA= 8  IB= 80
I= 9  IA= 9  IB= 90
I=10  IA=10  IB=100
```

(c)　I1＝－2

```
I=-2  IA= 0  IB=-20
I=-1  IA=-1  IB=-10
I= 0  IA= 0  IB=  0
I= 1  IA= 1  IB= 10
I= 2  IA= 2  IB= 20
I= 3  IA= 3  IB= 30
I= 4  IA= 4  IB= 40
I= 5  IA= 5  IB= 50
I= 6  IA= 6  IB= 60
I= 7  IA= 7  IB= 70
I= 8  IA= 8  IB= 80
I= 9  IA= 9  IB= 90
I=10  IA=10  IB=100
```

たかもそれまでDŌの範囲が正しく実行されたが如くにDŌの範囲の実行が文法に従って進められる。

一方，上記の例を整理すれば次のようになる。

(22) DŌの範囲内へ，拡張範囲以外のDŌの範囲外から飛び込みがある。

事がバッグの原因又はバッグであった場合には，

(23) DŌの制御変数が初期値パラメタ以外の値からはじまつている。但し増分パラメタの加算は端末文で毎回正しくおこなわれている。

という症状をひきおこす。エラー(23)について制御変数の値の面からみるとさらに次のように細かく分けて考える事ができる。

(24) DŌ文の制御変数が初期値パラメタより小さい値からはじまっている。

この場合には，さらに{図4.3.6の(15) / 図4.1.3の(6)}へとつながってゆく。

(25) DŌ文の制御変数が初期値パラメタと終値パラメタの間の値からはじまっている。

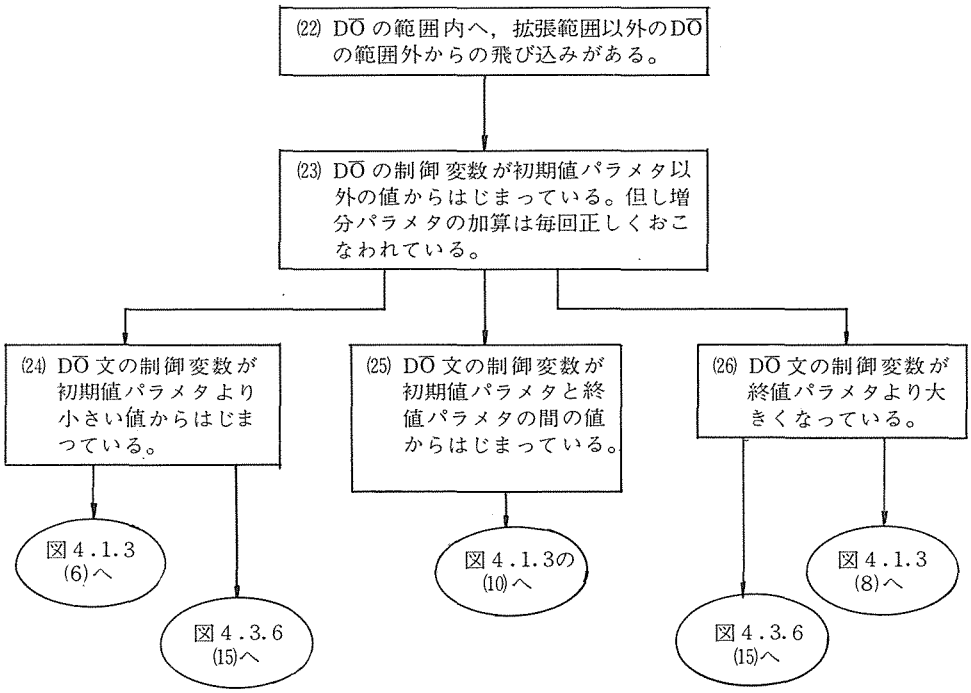
この場合には図4.1.3(10)へとつながる。

(26) DŌ文の制御変数が終値パラメタより大きくなっている。

この場合にも{図4.1.3の(8) / 図4.3.6の(15)}へとながる。

以上を図式化したのが図4.4.4である。

図4.4.4 DŌの範囲内へ拡張範囲以外の範囲外から飛び込した時のエラー



5. ま と め

DO 文に関係したバッグについて解説した。第2節の文法説明，第3節のコンパイル時のエラー・メッセージについては余り問題はないだろう。これらは，プログラム上においてもほとんど局所的であり，またエラーとしても発生段階で発見されるので修正は容易である。第4節の実行時のエラーの問題についてはプログラミングをおこなう者をしばしば悩ます事である。この解説も特に第4節を中心に考えた。

第4節の図4.1.3，図4.2.5，図4.3.6及び図4.4.4はバッグ発生の際の現象を原因→結果の順序で矢印をつけてまとめたものである。これらを実際のデバッグの場で応用する時には，結果→原因の順序に図中の矢印を逆向きに進んでゆくとよい。そしてデバッグされるべきプログラムの示す状態を選びながら進んでゆくとデバッグの具体的手法が明らかになる。ただしここで示した流れ図は，いかなる処理（処置）をすべきかではなく，いかなる状態がいかなる状態の結果として惹き起こされたかを示したにすぎない。したがって示された状況がプログラム作成者に都合の悪いものであるか否かの判断は，作成者自らが判断しなければならないと考えている。なお文中で“異常”とか“～すぎる”等は，すべて文法規則に従った時の状態を基準に考えた。なお4節の流れ図の全体を附録としてまとめておいた。

以後数回にわたって他のフォートラン文についても同様の解説を続けてゆく予定ですが，より多くの計算機利用者からの声を期待しております。

〔教育広報委員会からのお願い〕

この解説に関連して，読者からFORTRANについての自分の経験，疑問点などをご提示頂き，Forum欄を充実したいと思います。どうぞご協力下さい。

参考文献

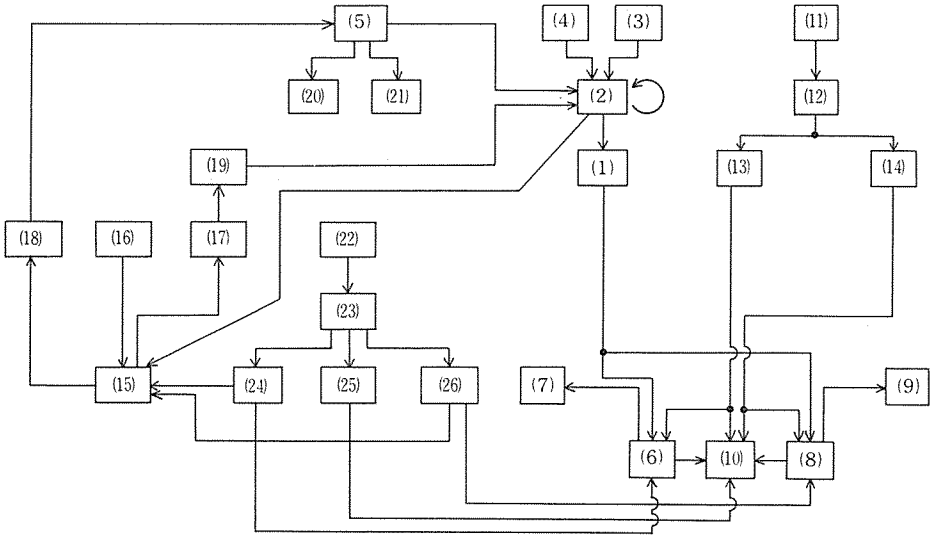
1）JISハンドブック，情報処理，1973，日本規格協会

2）NEAC-シリーズ2200 オペレーティングシステム MODIV EX/Ⅶ FORTRAN700 プログラミング説明書；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 MOD Ⅳ EX/Ⅶ FORTRAN 700 文法説明書；ＮＥＣ日本電気株式会社

附　録

図４.１.３，図４.２.５，図４.３.６及び図４.４.４のまとめ



エラーの番号及び内容

(1) DO の制御変数の振舞いの異常。
(2) DO 文における初期値，終値，増分パラメタの誤り。
(3) DO のパラメタに関係のある変数(正整数)の代入文の誤り。
(4) DO 文のパラメタに関係のある変数の入力の誤り。
(5) 主記憶上に記憶されたプログラム部分の破壊。
(6) DO のくりかえしが異常に多い。
(7) CPU-TIME が異常に長い。
(8) DO のくりかえしが異常に少い。
(9) CPU-TIME が異常に短かい。
(10) 変数又は配列の値の異常。
(11) 副プログラムの引用の際にコモン領域又は仮引数に DO の制御変数を含んでいる。しかも制御変数が副プログラム中で再定義されている。
(12) DO の範囲において制御変数の増分及び制御変数の変化域が初期値，終値及び増分パラメタで指定したものと一致しない。
(13) 制御変数が I=I−N（N＞０）の形式で再定義されている。
(14) 制御変数が I=I+N（N＞０）の形式で再定義されている。
(15) DO の範囲内で，配列に対して宣言文で定義した領域以外の所へ代入しようとした。
(16) 配列宣言の誤りのため配列の大きさを小さくとりすぎた。
(17) 配列への代入が，宣言文で定義した領域よりわずかにはみ出した。
(18) 配列への代入が，宣言文で定義した領域をはるかに越えた。
(19) 問題となっている配列の近傍（宣言文中で）にある配列又は変数の値が変られてしまった。
(20) ソース・リスト上で見る限り，全く考えられれない振舞をしている。
(21) 実行が中断される。
(22) DO の範囲内へ，拡張範囲以外の DO の範囲外からの飛び込みがある。
(23) DO の制御変数が初期値パラメタ以外の値からはじまっている。但し増分パラメタの加算は毎回正しくおこなわれている。
(24) DO 文の制御変数が初期値パラメタより小さい値からはじまっている。
(25) DO 文の制御変数が初期値パラメタと終値パラメタの間の値からはじまっている。
(26) DO 文の制御変数が終値パラメタより大きくなっている。