# Ⅴ．　フォートラン（ＦＯＲＴＲＡＮ）によるプログラミング

## §１．プログラムのコンセプト

この章ではいよいよプログラムの実行のさせ方について実習する。実は前章でエディターを使って作成したプログラム「ＡＢＣ．ＦＯＲ」は、前にも断ったとおりＦＯＲＴＲＡＮ文法によって書かれた１つのプログラムなのである。

```
１行目　　PROGRAM ABC
２行目　　A=1.0
３行目　　WRITE(6,*) A
４行目　　END
```

プログラムは計算機内部での計算の命令手順を示したものである。プログラムの一般的なコンセプトを述べれば、

①　（特に断らない限り）上の行から順番に命令を実行する。
②　プログラムを書く人間が適当に変数名を定義する。
③　「何とか＝何とか」というような一見等式にみえるようなものは、実は方程式でも何でもなく、「右辺の値を左辺の変数に代入する」という操作を示したステートメントである。

ということにつきる。
例えば、上のプログラムを例に構成を考えてみよう。①に従って、１行目から実行される。まず一行目の"PROGRAM ABC"というのは、別に計算をするわけではなくて、このプログラム名が”ＡＢＣ”であることを宣言している（このように宣言をする文を宣言文という）。次は２行目である。②に従って、”Ａ”という名前の変数を勝手に定義し、③のコンセプトに従って変数Ａに１．０なる数値を代入している。次３行目の"WRITE(6,*) A"は変数Ａをパソコンのディスプレー上に表示させよという命令である。従って、この文が実行された時点で、パソコンの画面に

1.00000

という変数Ａの数値が表示されるはずである。最後の"END"はプログラムの末尾を表すステートメントで、ＦＯＲＴＲＡＮのプログラムはすべて"END"で終わる約束になっている。

## §２．プログラムの実行

さてこれで中身についての簡単な解説は終わって、このプログラムをいよいよ動かすことにしよう。プログラムを動かすためには、

（１）ＦＯＲＴＲＡＮ文法で書かれたプログラムをひとまずコンピューターの認識で

きる機械語に翻訳する。翻訳されたプログラムはもはやエディターなどで見ることはできない。この機械語に翻訳されたものを「オブジェクト・モジュール」という。（翻訳）

（２）その翻訳されたものとあらかじめ準備されているＦＯＲＴＲＡＮのライブラリーなどをくっつける。この結合によってできた機械語のファイルを「ロード・モジュール」という。（結合）

（３）（２）で作成されたロードモジュールは直ちにパソコン上で動く形になっているのでそれを実行させる。（実行）

以上の３段階の操作が必要なのである。何のことかわけがわからないと思うが、別にわからなくても実際上困ることはないので、無理にわからなくてもよい。言葉の意味もわからなくてもよい。むしろ、プログラムの実行の手順を鵜呑みにして、丸暗記する（身につける）ことがコツである。

## ＰＣ－ＦＯＲＴＲＡＮを使ったプログラムの実行（翻訳＋結合＋実行）

１．　A>FRTと入力する（ＰＣ－ＦＯＲＴＲＡＮを実行するための環境設定）。

２．　自動的にコマンドモードがＥドライブに移り、E>が表示される。

３．　E>DIR/Wと入力し（これは、ドライブ上に存在するファイルの名前のリストを表示させるコマンド）、（１）のＶＺエディターの操作で作成したプログラムが存在することを確認する（上の例では、ABC.FORというプログラムを作成したので、その名前が表示されるはずである）。

４．　E>FORTRAN ABC,,,　と入力する（ABCは（１）で作成したプログラム名ABC.FORのABCのこと。実行するプログラム名に応じてここを変えることになる。例えばCDE.FORを実行するのであれば、E>FORTRAN CDE,,,　と入力する）。

５．　４．の操作でプログラムの翻訳が終了する。"ABC　SUCCESSFULLY COMPILED"（翻訳成功の意味）と表示されるか確認せよ。
これがでないときは、プログラムのどこかに何らかのエラーが存在する。そのようなときは、（１）のエディターの操作を繰り返してチェックをせよ。

６．　E>LINK ABC,,,,　と入力する（コンマの数を間違えないように。今回はコンマは４つ）。
ここでいうABCとは、４．の操作で（自動的に作成されたオブジェクトモジュールABC.OBJのABCのことである。ABC.OBJのABCとはABC.FORの中身の１行目に存在する”PROGRAM ABC”のABCのことである。

７．　６．の操作で、プログラム（オブジェクトモジュール）の結合（ＬＩＮＫ）が

終了する。その結果、実行形式のロードモジュールABC.EXEがEドライブ上にできているはずである。

E>DIR/W と入力してABC.EXEがあることを確かめよ。

8． これで、いよいよプログラムを実行することができるようになった。E>ABC と入力せよ。ここでのABCとはABC.EXEのABCのことである。例題１のABC.FOR の実行（すなわちABC.EXE）であれば、”1.0000000”なる数値が実行結果として表示される。

9． RETURNキーを押すとコマンドモードE>に戻り、一連の作業が終了する。

以下に例を示す。

1.～2. 環境設定

```
A>FRT          ← 入力

A>ECHO OFF
A:¥FRT ¥FORTRN01.DAT
A:¥FRT ¥FORTRN02.DAT
A:¥FRT ¥FORTRN03.DAT
A:¥FRT ¥FORTRN04.DAT
A:¥FRT ¥FORTRN05.DAT
A:¥FRT ¥FORTRN06.DAT
        6 個のファイルをコピーしました.
A:¥FRT ¥FORTRN01.COD
A:¥FRT ¥FORTRN02.COD
A:¥FRT ¥FORTRN03.COD
A:¥FRT ¥FORTRN04.COD
A:¥FRT ¥FORTRN05.COD
A:¥FRT ¥FORTRN06.COD
        6 個のファイルをコピーしました.
ECHO ON

E>
```

← 無視する（気にしない）。

3. プログラムが存在することの確認

```
E>DIR/W        ← 入力
 ドライブ E: のディスクのボリュームラベルは MS-RAMDRIVE
 ディレクトリは E:¥

ABC      FOR    FORTRN01 DAT    FORTRN02 DAT    FORTRN03 DAT    FORTRN04 DAT
FORTRN05 DAT    FORTRN06 DAT    FORTRN01 COD    FORTRN02 COD    FORTRN03 COD
FORTRN04 COD    FORTRN05 COD    FORTRN06 COD
      13 個のファイルがあります.
  677888 バイトが使用可能です.

E>
```

← VZエディターで作成したABC.FORが存在している
残りのファイル（FORTRN....）は気にしない。

## 4.～5. プログラムの翻訳

```
E>FORTRAN ABC...                    ← 入力（コンマは3つ）
NEC FORTRAN77 Version  2.20
Copyright (C) 1985 by NEC Corporation


PROGRAM-NAME:FORTRAN2    REV: 2.20          NAME:ABC.FOR
DATE:93/04/20 TIME:14:38     PAGE:001

*** AREA SIZE ***

    CODE  AREA SIZE =    00086  (     134 )
    DATA  AREA SIZE =    00030  (      48 )
    CONST AREA SIZE =    00004  (       4 )
    TOTAL           =    000BA  (     186 )

    ABC        SUCCESSFULLY COMPILED  (8087 USE : NO)     ← このメッセージが出るか？

E>
```

- 出れば成功
- 出なければプログラムがどこか間違っている。

## 6. オブジェクトモジュール →（リンク（結合））ロードモジュール（実行形式）

```
E>LINK ABC....                      ← 入力（コンマは4つ）

Microsoft (R) 8086 Object Linker  Version 3.05
Copyright (C) Microsoft Corp 1983, 1984, 1985.  All rights reserved.
```

## 7. ロードモジュール（ABC.EXE）の存在の確認

```
E>DIR/W          ← 入力

 ドライブ E: のディスクのボリュームラベルは MS-RAMDRIVE
 ディレクトリは E:¥

ABC      FOR    FORTRN01 DAT    FORTRN02 DAT    FORTRN03 DAT    FORTRN04 DAT
FORTRN05 DAT    FORTRN06 DAT    FORTRN01 COD    FORTRN02 COD    FORTRN03 COD
FORTRN04 COD    FORTRN05 COD    FORTRN06 COD    ABC      OBJ    ABC      EXE
      15 個のファイルがあります.
  624640 バイトが使用可能です.
```

ABC.EXE が存在しているか？

## 8. 実行

```
E>ABC            ← 入力

      1.0000000  ←

STOP - PROGRAM TERMINATED.
PRESS <RETURN> TO COMMAND MODE.
```

プログラムの実行結果の表示。
（つまり、ABC.FOR の "WRITE(6,*) A" により A の値が表示されている（A=1.0000000））

## 9. コマンド モードへ戻る

```
E>
```

---

まとめると、プログラムの実行の手順は、

① E>FRT と入力。（初回のみ必要な操作。２回目以降のＰＣ－ＦＯＲＴＲＡＮの実行に際しては必要ない。）
② E>FORTRAN ABC,,, と入力。下線部は、プログラム名に応じて変えること（以下の下線部についても同様）。
③ ABC SUCCESSFULLY COMPILED が表示されることを確認する。表示されなければプログラムのエラーをＶＺエディターを使って修正する。
④ E>LINK ABC,,,, と入力。
⑤ E>ABC と入力。これで実行され、出力結果が画面に表示される。
⑥ RETURNキーをおすとコマンドモード E> に戻る。

◎プログラムを翻訳したときに出るエラーメッセージについて

作成したプログラムを翻訳したとき（つまり、FORTRAN ABC,,, などと入力したとき）、プログラムに文法上のエラーがなければ

「ＳＵＣＣＥＳＳＦＵＬＬＹ　ＣＯＭＰＩＬＥＤ」

というメッセージが表示されるが、そうでないときはエラーメッセージが出力される。パソコンでＦＯＲＴＲＡＮをやり始めのうちはエラーばかり出てくると思う。

【見方】

```
LINE L DIAGNO *** DIAGNOSTIC MESSAGE ***
   4 F  1101  SYMBOLIC NAME TOO LONG
   4 F  1002  UNRECOGNIZABLE STATEMENT
```

エラーメッセージ本文

行番号

エラーレベル（Ｗ：警告；　Ｆ：致命的；　Ｓ：重大）

エラーの出ている行番号に誤りがあるので、エディターを使ってプログラムを修正する。但し、表示されている行番号以外のステートメントに誤りの原因があることもある。

## §３．ファイルのフロッピーディスクへの保存

前にも述べたとおり、実習室のパソコンのＥドライブはＲＡＭディスクなので、電源ＯＦＦまたはＲＥＳＥＴでドライブ上のデータは全部消えてしまう。そこで、１日の作業を終わる前にＥドライブのデータをフロッピーにコピーした保存する必要がある。

その手順は非常に簡単であり、次の通り。

① フロッピーディスクをＣドライブ（ＰＣの２つのフロッピーディスク差込み口のうちの上の方）にさしこんで、固定する。

② A>COPY E:ABC.FOR C:　と入力する（これは、例えばABC.FORをセーブする場合である。場合に応じて下線部を変更すること）。
（これは、Ｅドライブ上のABC.FORなるファイルをＣドライブ、すなわち差し込だフロッピーディスクにコピーせよ、という意味）

※はじめに E> となっているときは、
E>COPY ABC.FOR C:
でよい（E:ABC.FOR の　E:　を省略できる）。
※Ｅドライブ上のすべてのプログラムを一度にセーブするときは
E>COPY *.FOR C:　　（あるいは　A>COPY E:*.FOR C:）
とすれば簡単である。

③ E>C:DIR/W （あるいは　A>C:DIR/W）を入力して、セーブしたファイルがＣドラブ上に存在すること（すなわちフロッピーディスクにコピーされたことを確認せよ）。

④ ＰＣからフロッピーを取り出す。

**練習問題**

例題２～４のプログラムをＶＺエディターを作成し、ＰＣ－ＦＯＲＴＲＡＮで実行させよ。さらにプログラムをフロッピーに保存せよ。

《例題２》　ファイル名はＡＢＣ２．ＦＯＲとせよ。

```
      PROGRAM ABC2
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      A=1.0
      WRITE(6,*) A
      READ(5,*) B
      WRITE(6,*) B
      END
```

《例題３》　ファイル名はＡＢＣ３．ＦＯＲとせよ。

```
      PROGRAM ABC3
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      A=0.0
      DO 333 I=1,100
      A=A+20.0
 333  CONTINUE
      WRITE(6,*) A
      END
```

《例題４》　ファイル名はＡＢＣ４．ＦＯＲとせよ。

```
      PROGRAM ABC4
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION A(5)
      A(1)=23.0
      A(2)=3.0
      A(3)=4.0
      A(4)=99.99
      A(5)=13.2
      DO 333 I=1,5
      WRITE(6,*) A(I)
 333  CONTINUE
C
      DO 111 I=1,5
      A(I)=A(I)+0.01
 111  CONTINUE
      DO 222 I=1,5
      WRITE(6,*) A(I)
 222  CONTINUE
      END
```

## §４．ＦＯＲＴＲＡＮによるプログラミングの基本（その１）

§３までの実習でプログラムの見本さえ手元にあれば、一応、ファイルとして作ることとと走らせることはできるようになった（はずである）。これ以降では、
ＦＯＲＴＲＡＮ言語を用いた科学技術計算のための基本的な事柄について解説と演習を行

ない、プログラミングの基礎を身につけることを目指したい。

## §４－１　プログラムの構成

プログラムのコンセプトについてもう一度、

① プログラムは上の行から順番に命令を実行する。
② 変数名はプログラムを書く人間が適当に定義すればよい。
③ 「何とか＝何とか」というような一見等式にみえるようなものは、「右辺の値を左辺の変数に代入する」という操作を示している。

ということをもう一度銘記してほしい。

ＦＯＲＴＲＡＮの文法規則について一から説明していては５日間の実習ではとうてい時間がないし、またそれは必要ないことでもある。重要なことは、要点を要領よく知ることである。教官サイドとしては、実際の物質科学の研究のさまざまな局面で経験的に知ったコツのようなものを伝授したいと思う。それらは、ＦＯＲＴＲＡＮの文法の使い方の一部に過ぎなかったり、書き方の１つの流儀に過ぎなかったりするかもしれないが、ゴールへの最短距離であると思う。ＦＯＲＴＲＡＮプログラミングの基本は以下の通り。

① プログラムは（コメント文と文番号以外は）どの行も７カラム目から書き始めること！
　文番号は１～５カラム目の間に書く。

② プログラムの最後はＥＮＤ文で終わること。

③ プログラムの第１行目は、プログラム文を書くこと（プログラムファイル名として、例えば「XXX.FOR」に対しては、１行目で”ＰＲＯＧＲＡＭ　ＸＸＸ”と指定すること（例題１～４参照））。

　もしこれを指定しないと（つまり、プログラム文を入れ忘れると）、オブジェクトモジュールとして、FTMAIN.OBJ ができるので、リンクの際には、ロードモジュールとしてXXX.EXE を作るのであれば、 LINK FTMAIN,XXX,,, としなければならない。

④ ほとんどすべての科学技術計算は、倍精度（有効数字１６桁）で行うのが普通である。このためプログラムでは、プログラム文の次（すなわち２行目）に、

ＩＭＰＬＩＣＩＴ　ＲＥＡＬ＊８（Ａ－Ｈ，Ｏ－Ｚ），ＩＮＴＥＧＥＲ＊４（Ｉ－Ｎ）

なるステートメントを入れること。
　これは、頭文字がＡからＨ，ＯからＺまでの変数は、すべて倍精度の実数、それ以

外の（すなわち、頭文字がＩ，Ｊ，Ｋ，Ｌ，Ｍ，Ｎで始まる）変数は整数であること
を指定ものである。従って、プログラム中の変数のタイプはこの指定に合わせること。
　このような、倍精度の指定をしないと、単精度（有効数字８桁）の計算になってし
まう（電卓より精度が低い！）

⑤　変数は、実数・整数の区別を厳密に行うこと。

・実数型の場合、「Ａ＝１．０」　または
　　「Ａ＝１．０Ｄ０」または「Ａ＝１．０Ｅ０」（１．０×１０$^{0}$の意）
・整数型の場合、Ｉ＝１　（決して　Ｉ＝１．０　とはしない）
のように書くこと。

・整数→実数の変換は、例えば、Ａ＝ＲＥＡＬ（Ｉ）またはＡ＝ＦＬＯＡＴ（Ｉ）
・実数→整数の変換は、　　Ｉ＝ＩＮＴ（Ａ）
のように、ＲＥＡＬまたはＦＬＯＡＴ（実数化）、ＩＮＴ（整数化：整数部分だけ
を取り出す、小数部は切捨て）を使って行う。
例）Ｉ＝３のとき、Ａ＝ＦＬＯＡＴ（Ｉ）とすれば、Ａ＝１．０のようになり、
　　Ａ＝２．３のとき、Ｊ＝ＩＮＴ（Ａ）とすれば、Ｊ＝２のように小数部が切り捨
てられ、整数部のみが表示される。

⑥　１カラム目に「Ｃ」を入れれば、その行はコメント行（メモ用の行。その行は実行
しない）となる。

①～⑥に従って、今後、プログラムは以下の形式となる。　この3行は定番です！

```
      PROGRAM XXX
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      ↑ 6カラム空ける！
                    :
                    :
                    :
                    :
C     KOMENTOGYOU NANI WO KAITEMOYOI      ← １カラム目「Ｃ」はコメント行
                    :
_123__CONTINUE
    ↑ 文番号は１～５カラム目の間に書く！
                    :
                    :
      END ← プログラムの最後は「ＥＮＤ」文で終わる！
```

## §４－２　入出力文（ｃｆ．例題１、２）

**ＲＥＡＤ（５，＊）**　　パソコンキーボードより入力
**ＷＲＩＴＥ（６，＊）**　パソコンディスプレー（画面）に出力
（＊は、書式の指定をしないことを意味する）

書式とは、例えば「１．０」という数値を表示させる際に、
「１．００」　　と表示させるか
「１．００００」　　と表示させるか
「１．０Ｅ０」　　と表示させるか、などの違いをいう。
書式を指定するには、ＦＯＲＭＡＴ文を使う（後述）。

例題２をもう一度見てみよう。

```
PROGRAM ABC2
IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
A=1.0
WRITE(6,*) A
READ(5,*) B
WRITE(6,*) B
END
```

３行目で変数Ａが定義され、１．０なる値がＡに代入されている。４行目でＡの値がパソコンの画面に出力される。５行目では変数Ｂの値をキーボードから入力する。プログラムを走らせると５行目まで実行された時点でパソコン側はキーボードからの入力待ちになっているので、適当な実数の値をキーボードから入れてやる。例えば、「987.654321」と入れたとしよう。すると６行目の「WRITE(6,*) B」によってこの値がディスプレーに表示される。７行目でプログラム実行終了となる。

## §４－３　四則演算、組み込み関数

四則演算は「＋」「－」「＊」「／」の記号を用いて表現する。
ＡのＢ乗は「Ａ＊＊Ｂ」と表す。

「Ａ＊Ｂ／Ｃ＊Ｄ」　は、　（（Ａ＊Ｂ）／Ｃ）＊Ｄ　のことである（つまり数式の演算順序と同じ）。あやふやなときはためらわずに括弧を使うこと（間違ったプログラムを書くよりはるかにまし。括弧を使ってもエラーはでない）。

Ａの平方根は「ＳＱＲＴ（Ａ）」と表現する。
Ａの正弦は「ＳＩＮ（Ａ）」と表現する。このように「関数名（変数名）」のように表

せるものを組み込み関数という。前に出てきた、ＲＥＡＬ（…）、ＩＮＴ（…）も組込み関数の１つである。ＦＯＲＴＲＡＮで使用できる組み込み関数の例を次に示す。

## 組み込み関数一覧表

### ABS
機能：絶対値を与える。
書式：①IABS(<整数型数式>)
②ABS(<実数型数式>)
③DABS(<倍精度実数型数式>)
④CABS(<複素数型数式>)
関数の型：①INTEGER＊4
②REAL
③DOUBLE PRECISION
④COMPLEX
文例：B＝ABS(－4)

### ACOS
機能：逆余弦(アークコサイン)を与える。
書式：①ACOS(<実数型数式>)
②DACOS(<倍精度実数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
文例：A＝ACOS(1)

### AIMAG
機能：虚部を与える。
書式：AIMAG(<複素数型数式>)
関数の型：REAL
文例：C＝AIMAG(A)

### AINT
機能：小数点以下を切り捨てた整数値を与える。
書式：①AINT (<実数型数式>)
②DINT(<倍精度実数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
文例：A＝AINT(B)

### ANINT
機能：四捨五入した整数値を与える。
書式：①ANINT(<実数型数式>)
②DNINT(<倍精度実数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
文例：C＝ANINIT(A)

### ASIN
機能：逆正弦（アークサイン）を与える。
書式：①ASIN(<実数型数式>)
②DASIN(<倍精度実数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
文例：D＝ASIN(A)

### ATAN
機能：逆正接（アークタンジェント，$-\frac{\pi}{2}\leqq$結果$\leqq\frac{\pi}{2}$）を与える。
書式：①ATAN(<実数型数式>)
②DATAN(<倍精度実数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
文例：A＝ATAN(1)

### ATAN2
機能：逆正接（アークタンジェント，$-\pi\leqq$結果$\leqq\pi$）を与える。
書式：①ATAN2(<実数型数式1>,<実数型数式2>)
②DATAN2(<倍精度実数型数式1>,<倍精度実数型数式2>)
関数の型：①REAL
②DOUBLE PRECISION
cf. ●引数１＞０→結果＞０
●引数１＝０ AND
引数２＞０→結果＝０
●引数１＝０ AND
引数２＜０→結果＝$\pi$
●引数１＜０→ 結果＜０
●引数２＝０→ |結果|＝$\frac{\pi}{2}$
●引数１＝０ AND
引数１＝０ →エラー
文例：A＝ATAN(X, Y)

### CHAR
機能：数式を文字型へ変換する。
書式：CHAR(<整数型数式>)
関数の型：CHARACTER(1バイト)
文例：R＝CHAR (32)

### CMPLX
機能：数値を複素数型へ変換する。
書式：①CMPLX(<数式>)
数式→整数型，実数型，倍精度実数型，複素数型
②CMPLX(<数式1>, <数式2>)
数式１，数式２→整数型，実数型，倍精度実数型
cf.●数式１と数式２は同じ型
数式１→実部
数式２→虚部
関数の型：①，②COMPLEX
文例：C＝CMPLX(R)

### CONJG
機能：共役複素数を与える。
書式：CONJG(<複素数型数式>)
関数の型：COMPLEX
文例：A＝CONJG(C)

### COS
機能：余弦(コサイン)を与える。
書式：①COS(<実数型数式>)
②DCOS(<倍精度実数型数式>)
③CCOS(<複素数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
③COMPLEX
文例：C＝COS(A)

### COSH
機能：双曲線正弦(ハイパコサイン)を与える。
書式：①COSH(<実数型数式>)
②DCOSH(<倍精度実数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
文例：C＝COSH(A)

### DBLE
機能：数式を倍精度実数型に変換する。
書式：DBLE(<数式>)
数式→整数型，実数型，倍精度実数型，複素数型
関数の型：DOUBLE PRECISION
文例：R＝DBLE(I)

### DIM
機能：２数式の超過分を与える。
書式：①IDIM(<整数型数式1>,<整数型数式2>)
②DIM(<実数型数式1>,<実数型数式2>)
③DDIM(<倍精度実数型数式1>,<倍精度実数型数式2>)
関数の型：①INTEGER＊4
③DOUBLE PRECISION
cf. 数式１＞数式２→結果＝数式１－数式２
数式１≦数式２→結果＝０
文例：C＝IDIM(3, 1)

### DPROD
機能：倍精度化演算の積を与える。
書式：DPROD(<実数型数式1>,<実数型数式2>)
関数の型：DOUBLE PRECISION
文例：X＝DPROD(A, B)

### EXP
機能：ｅに対する指数関数の値を与える。
書式：①EXP(<実数型数式>)
②DEXP(<倍精度実数型数式>)

③CEXP(<複素数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
③COMPLEX
文例：A=EXP(C)

## ICHAR

機能：1バイト文字型を整数型に変換する。
書式：ICHAR(<1バイト文字型>)
関数の型：INTEGER＊4
文例：A=ICHAR(C)

## INT

機能：数式を整数型に変換する。
書式：①INT(<数式>)
数式→整数型，実数型，倍精度実数型，複素数型
②IFIX(<実数型>)
③IDINT(<倍精度実数型>)
関数の型：①INTEGER＊4
②INTEGER＊4
③INTEGER＊4
文例：I=INT(R)

## LOG

機能：自然対数を与える。
書式：①ALOG(<実数型数式>)
②DLOG(<倍精度実数型数式>)
③CLOG(<複素数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
③COMPLEX
文例：R=ALOG(A)

## LOG10

機能：常用対数を与える。
書式：①ALOG10(<実数型数式>)
②DLOG10(<倍精度実数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
文例：C=ALOG10(A)

## MAX

機能：2つ以上の数式の中で最大値を与える。
書式：①MAX 0(<整数型数式>,<整数型数式>…)
②AMAX 1 (<実数型数式>,<実数型数式>…)
③DMAX 1 (<倍精度実数型数式>, <倍精度実数型数式>…)
④AMAX 0(<整数型数式>, <整数型数式>…)
⑤MAX 1 (<実数型数式>,<実数型数式>…)
関数の型：①INTEGER＊4
②REAL
③DOUBLE PRECISION
④REAL
⑤INTEGER＊4
文例：A＝MAX 0 (X, Y, Z)

## MIX

機能：2つ以上の数式の中で最小値を与える。
書式：①MIN 0(<整数型数式>,<整数型数式>…)
②AMIN 1 (<実数型数式>,<実数型数式>…)
③DMIN 1(<倍精度実数型数式>,<倍精度実数型数式>…)
④AMIN 0 (<整数型数式>,<整数型数式>…)
⑤MIN 1 (<実数型数式>, <実数型数式>…)
関数の型：①INTEGER＊4
②REAL
③DOUBLE PRECISION
④REAL
⑤INTEGER＊4
文例：F＝MIN 0 (A, B, C)

## MOD

機能：剰余を与える。
書式：①MOD(<整数型数式>,<整数型数式>)
②AMOD(<実数型数式>,<実数型数式>)
③DMOD(<倍精度実数型数式>,<倍精度実数型数式>)
関数の型：①INTEGER＊4
②REAL
③DOUBLE PRECISION
文例：C=MOD(A, B)

## NINT

機能：四捨五入の結果を整数で与える。
書式：①NINT(<実数型数式>)
②IDNINT(<倍精度実数型数式>)
関数の型：①INTEGER＊4
②INTEGER＊4
文例：I=NINT(R)

## RANDOM

機能：0以上1未満の一様乱数を与える。
書式：RANDOM(<整数型数式>)
関数の型：REAL
文例：R=RANDOM(0)

## REAL

機能：数式を実数型に変換する。
書式：①REAL(<実数>)
数式→整数型，実数型，倍精度実数型，複素数型
②FLOAT(<整数型数式>)
③SNGL(<倍精度実数型数式>)
関数の形：①REAL
②REAL
③REAL
文例：D=REAL(A)

## SIGN

機能：符号の付けかえを行なう。
書式：①ISIGN(<整数型数式1>,<整数型数式2>)
②SIGN(<実数型数式1>,<実数型数式2>)
③DSIGN (<倍精度実数型数式1>,<倍精度実数型数式2>)
関数の型：①INTEGER＊4
②REAL
③DOUBLE PRECISION
cf.数式2≧0→|数式|
数式2＜0→－ 数式|
文例：J=ISIGN(I, 0)

## SIN

機能：正弦(サイン)を与える。
書式：①SIN(<実数型数式>)
②DSIN(<倍精度実数型数式>)
③CSIN(<複素数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
③COMPLEX
文例：C=SIN(X)

## SINH

機能：双曲線正弦(ハイパサイン)を与える。
書式：①SINH(<実数型数式>)
②DSINH(<倍精度実数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
文例：X=SINH(A)

## SQRT

機能：平方根(スクエアルート)を与える。
書式：①SQRT(<実数型数式>)
②DSQRT(<倍精度実数型数式>)
③CSQRT(<複素数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
③COMPLEX
文例：X=SQRT(A)

## TAN

機能：正接(タンジェント)を与える。
書式：①TAN(<実数型数式>)
②DTAN(<倍精度実数型数式>)
関数の型：①REAL
②DOUBLE PRECISION
文例：X=TAN(A)

## TANH

機能：双曲線正接(ハイパタンジェント)を
書式：①TANH(<実数型数式>)
②DTANH(<倍精度実数型数式>)
関数の型：1 REAL
2 DOUBLE PRECISION
文例：A=TANH(R)

例）

```
      PROGRAM XYZ
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      A=1.0E0
      B=4.0E0
      C=A+B+3.0E1
      D=SQRT(C)
      E=D**2-(A+B)
      F=SIN(E)-2.0**(1.0/3.0)
      F=F-4.0
      WRITE(6,*) F
      END
```

これは結局、

のような演算を行っているわけである。
「Ｆ＝Ｆ－４．０」というのは奇妙に思うかもしれないが、右辺の値、つまり、
（その前の行までのＦの値（SIN(E)-2.0**(1.0/3.0)））－４．０
を左辺の変数Ｆの値としてに新たに定義するというコンセプトを思い起こせば理解できよう。もちろんプログラムの最後の３行を、

```
      G=F-4.0
      WRITE(6,*) G
      END
```

のように書いても同じ結果が得られるのはいうまでもない。

※問題１　　整数$x_1$，$x_2$，$x_3$を読み込んで、
$y = 13.5x_1 + 4.8x_2 + 5.2 / x_3$
を求め、その整数部を出力させよ。プログラム名はMON1.FORとせよ。
（プログラムの中では、整数型変数は頭文字が、ＩからＮで始まるアルファベットでないといけないことに注意せよ。変数名を「Ｘ１」などとすればそれは実数型となる。問題の式と同じ変数名をプログラム中に定義しなければならない理由はない。）

※問題２　　２次方程式
$$ax^2 + bx + c = 0$$
の解を求めるプログラムを書け。プログラム名はMON2.FORとせよ。
（方程式の各次の係数ａ，ｂ，ｃをキーボードから入力すれば、解の公式に従って、２つの解がディスプレーに出力されるようなプログラムを作ればよい。）

## §４－４　整数と実数の区別について再度注意

繰り返して言うが、ＦＯＲＴＲＡＮでは変数の

整数型　（小数点がない）
実数型　（小数点がある）
文字型　（英語の文字）
など

変数のタイプの区別を厳密に行う。

プログラムの冒頭で

「ＩＭＰＬＩＣＩＴ　ＲＥＡＬ＊８（Ａ－Ｈ，Ｏ－Ｚ），ＩＮＴＥＧＥＲ＊４（Ｉ－Ｎ）」

と宣言した以上、

頭文字がＡ－Ｈ，Ｏ－Ｚで始まる名前を持つ変数は、倍精度（１６桁）実数型

頭文字がＩ－Ｎで始まる名前を持つ変数は、整数となる。

初心者は、１つのステートメント（文）に整数型と実数型の変数が混在するような書き方はしないようにせよ。

１つのステートメントに整数型と実数型が混在してよいのは、

整数型→実数型　の変換：　e.g.,　Ｂ＝ＦＬＯＡＴ（Ｉ）

実数型→整数型　の変換：　e.g.,　Ｉ＝ＩＮＴ（Ｂ）

のように、ＦＬＯＡＴやＩＮＴを含む文だけしかないと心得よ。

例えば、上のようなＩＭＰＬＩＣＩＴ宣言文をしたのなら、

```
M＝7
N＝2
A＝M／N
```

のようなプログラムは書かないこと。なぜなら３行目は左辺が実数型、右辺に整数型が入っているからでこのような混在はよくない。

この計算の結果、Ａ＝３．５という値が得られると思うのは間違いである。Ａ＝３．０になっているはずである。なぜなら、Ｍ／Ｎは７／２の整数部だけを取り出すことになるので（３．５→３、つまり小数部分切捨て）、３になるのである。Ａは、この３という整数を実数型として扱うことになるので３．０になるわけである。このようなことをいちいち考えるのは慣れるまではいささかややこしい。また、整数型と実数型を混在させたステートメントを書くと、今の例のように、初心者は実際の値（３．０）とは違う値（３．５）がでてくると期待しがちである。

従って、

＊　実数型だけの文を書く。
＊　整数型だけの文を書く。

というふうにプログラムを書いて欲しい。

上の例でいえば、

```
M＝7
N＝2
ＰM＝ＦＬＯＡＴ（M）
ＰN＝ＦＬＯＡＴ（N）
A＝ＰM／ＰN
```

のように、整数M，Ｎをあらかじめ”ＦＬＯＡＴ（　）”を使って実数化し、変数Ａ（実数型）が現れる文は、ＰＮ，ＰＭ（実数型）を使って書くようにする。

また、整数型の変数が現れるステートメントは整数型の変数だけを使って書くこと。例えば、

```
Ｊ＝M－（M／２）＊２
```

のようなステートメントがそうである。例えば、Ｍ＝３１のとき（Ｍ／２）は１５．５の整数部分１５である。従って、（Ｍ／２）＊２は３０となるので、右辺の値は１、すなわちＪ＝１となる（このステートメントは、問題７（４ページ）に使える）。

整数→実数　の変換は　ＦＬＯＡＴ

実数→整数　の変換は　ＩＮＴ

を使って行うこと。

同様に、数字についても、実数型・整数型の区別をして欲しい。

数字　２　　は　整数型

数字　２．０　や　２．０Ｅ０　は　実数型

である。Ａ＝２と書いたり、Ｊ＝３．０と書いたりしないように（Ａ＝２．０，Ｊ＝３と書くべし）。

従って、例えば問題２のプログラムの中で、２次方程式の根の公式を使う部分では、

```
Ｘ１＝（－Ｂ＋ＳＱＲＴ（Ｂ＊＊２－４＊Ａ＊Ｃ））／（２＊Ａ）
```

と書くのではなく、

```
Ｘ１＝（－Ｂ＋ＳＱＲＴ（Ｂ＊＊２－４．０＊Ａ＊Ｃ））／（２．０＊Ａ）
```

と書いて欲しい（２乗の２は整数でよい）。

同様に、上の例でも整数型の計算部分に、

```
Ｊ＝M－（M／２．０）＊２．０
```

というようなマネはしないこと。

組み込み関数ＦＬＯＡＴとＩＮＴを使えば、いま述べたような書き方は守れるはずである。

## §４－５　ＤＯ文……繰り返し計算の方法（ｃｆ．例題３、４）

「計算の繰り返し」はコンピューターの最も得意とする演算形態である。ある計算の繰り返しをさせるためには、その計算を記述したステートメントを「ＤＯ文、ＣＯＮＴＩＮＵＥ文」で挟んでやればよい。例を示す。

```
      ＤＯ　３３３　Ｉ＝１，１００
              :
              :
３３３　ＣＯＮＴＩＮＵＥ
```

この部分が ← 繰り返される

「ＤＯ　３３３　Ｉ＝１，１００」と「３３３　ＣＯＮＴＩＮＵＥ」で挟まれた部分の計算が、１００回繰り返される。そのとき、繰り返しの１回目は変数Ｉが１，２回目は２，１００回目は１００となり、Ｉが１から１００まで繰り返しの度に１つずつ増えてゆく。

文番号「３３３」は「ＤＯ文」と「ＣＯＮＴＩＮＵＥ文」のブロックを認識するための単なる示標であって、任意の整数を用いてよい。

例題３をもう一度見てみよう。

```
      PROGRAM ABC3
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      A=0.0
      DO 333 I=1,100
      A=A+20.0
  333 CONTINUE
      WRITE(6,*) A
      END
```

「Ａ＝Ａ＋２０．０」という、右辺の値を新たに左辺の値（すなわちＡ）とせよ、という意味の演算が、１００回繰り返されることになる。

```
      ＤＯ　３３３　Ｉ＝１，１００
      Ａ＝Ａ＋２０．０
３３３　ＣＯＮＴＩＮＵＥ
```

の部分を、ＤＯ文を使わずに書き直せば、

```
Ａ＝Ａ＋２０．０　　……このときＩ＝１
Ａ＝Ａ＋２０．０　　……このときＩ＝２
      :                    :
      :                    :
```

} １００回

```
      A＝A＋２０．０          ……このときＩ＝１００
```

のようになる。こう書いたことと同じことをやっているのである。
このＤＯ－ＣＯＮＴＩＮＵＥの計算の結果、Ａの値は１回繰り返す毎に
２０．０ずつ加算されて、最終的に２０００．０になる。
ちなみにＤＯ－ＣＯＮＴＩＮＵＥは、ＢＡＳＩＣ言語では、「ＦＯＲ－ＮＥＸＴ文」に対応する。このようにＤＯ－ＣＯＮＴＩＮＵＥ文は、コンピューターの最も得意とする、繰り返し演算を行うためにある。これを使えば、ΣやΠのような計算が簡単にできることが察せられるであろう。

【注意】　一般に、標準のＦＯＲＴＲＡＮでは、変数の初期値は、必ずしもゼロではない。従って、ΣやΠをＤＯ－ＣＯＮＴＩＮＵＥ文を使って求めるときは、ＤＯ文より前に、変数の値を初期化しておくこと（例題３の３行目のことである。これを入らなければ変数Ａは繰り返しの前にいったいいくらなのか設定されていない）。
くどいようだがもう少し例をあげよう。
例１．

```
      DO 10 I=1,3
      READ(5,*) Q
      WRITE(6,*) Q
10    CONTINUE
```

上のＤＯループは、

```
      READ(5,*) Q
      WRITE(6,*) Q          } ← Ｉ＝１　（繰り返し１回目）
      READ(5,*) Q
      WRITE(6,*) Q          } ← Ｉ＝２　（繰り返し２回目）
      READ(5,*) Q
      WRITE(6,*) Q          } ← Ｉ＝３　（繰り返し３回目）
```

と書きかえても同じである。要するに、ＤＯ文とＣＯＮＴＩＮＵＥ文ではさまれている部分が３回（Ｉ＝１，３）繰り返されているわけである。この例では数値Ｑをキーボードから読ませてディスプレーに出力する操作が３回繰り返されることになる。

例２．

```
      M=0
      DO 300 K=1,5
      M=M+K
```

```
200  CONTINUE
```

これは、書き換えると、

```
      M=0
      M=M+1                ←  K＝1      左辺のMは  M＝0＋1
      M=M+2                ←  K＝2      左辺のMは  M＝1＋2
      M=M+3                ←  K＝3      左辺のMは  M＝3＋3
      M=M+4                ←  K＝4      左辺のMは  M＝6＋4
      M=M+5                ←  K＝5      左辺のMは  M＝10＋5
```

と同じである。この結果、１＋２＋３＋４＋５（$=\sum_{k=1}^{5} k$）という級数が求められたことがわかる。

例３．

```
      L=1
      DO 200 J=1,4
      L=L*J
200   CONTINUE
```

上のようなループは、

```
      L=1
      L=L*1                ←  J＝1      左辺のLは  L＝1＊1
      L=L*2                ←  J＝2      左辺のLは  L＝1＊2
      L=L*3                ←  J＝3      左辺のLは  L＝2＊3
      L=L*4                ←  J＝4      左辺のLは  L＝6＊4
```

と同じである。この結果、１＊２＊３＊４（$=\prod_{J=1}^{4} j$）という乗積が求められることがわかる。

このようにＤＯループを使えば、同じ作業の繰り返しをしたり、いろいろな級数（Σ）や乗積（Π）を求めることができる。

※問題３　　ａ＝１，２，…，２０に対して、その平方根を求め出力せよ。
（プログラム名MON3.FOR）

※問題４　　１からｎまでの自然数の和$\sum_{i=1}^{n} i$，２乗和$\sum_{i=1}^{n} i^2$　をもとめるプログラムを作れ。
（プログラム名MON4.FOR）

※問題５　　１０！を求めるプログラムを作れ。（プログラム名MON5.FOR）

※問題６　　ある実数ｘを読み込み、$x^{(1/2n)}$をｎ＝１，２，…，１０に対して順に出力するプログラムを作れ。
（プログラム名　MON6.FOR）

## §４－６　配列変数（ｃｆ．例題４）

普通、変数Ａ，Ｂといえば、それぞれＡ＝２．３，Ｂ＝４．５のような１つの値を持つわけであるが、配列変数とは、はじめに例えば、”ＤＩＭＥＮＳＩＯＮ　Ａ（５）”と宣言することによって、Ａに５つの添え字をつけることができる。
意味がよくわからないかもしれないが、例題４を見てみよう。

```
      PROGRAM ABC4
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION A(5)
      A(1)=23.0
      A(2)=3.0
      A(3)=4.0
      A(4)=99.99
      A(5)=13.2
      DO 333 I=1,5
      WRITE(6,*) A(I)
 333  CONTINUE
C
      DO 111 I=1,5
      A(I)=A(I)+0.01
 111  CONTINUE
      DO 222 I=1,5
      WRITE(6,*) A(I)
 222  CONTINUE
      END
```

ここで、Ａ（１），Ａ（２），…，Ａ（５）はそれぞれ全く別の変数と考えてよい。つまり、

$a_1 = 23.0$
$a_2 = 3.0$
:
:

$a_5$＝１３．２

と書いても同じことである（単に添え字を５つつけて５つの変数を定義しただけ！）。

配列変数は、例えば、多量のデータを処理する場合（これもコンピューターのオハコ）などに使われる。観測量ｙのデータが例えば１０００点あるとき、１０００個もの変数名をそれぞれ与えることは実際上できない。このようなとき、

ＤＩＭＥＮＳＩＯＮ　Ｙ（１０００）

とすれば１０００個のデータに対して、（添え字付き）変数を割り振ることができ、便利である。

配列変数を使うためには、

① 配列変数は、ＤＩＭＥＮＳＩＯＮ文で宣言しなければならない。
② ＤＩＭＥＮＳＩＯＮ文はプログラムの冒頭（ＩＭＰＬＩＣＩＴ文の次）に書くこと。

という約束を守る。

くどいようだがもう一度説明する。例えば、プログラムの冒頭、ＩＭＰＬＩＣＩＴ文の次に

「ＤＩＭＥＮＳＩＯＮ　Ａ（５），Ｂ（５）」

と宣言することによって、変数Ａ，Ｂはそれぞれ５つの成分をもつ配列変数となる。配列変数とは成分を持つ変数のことである。いまの宣言をすることによって、

Ａ（１），Ａ（２），…，Ａ（５），
Ｂ（１），Ｂ（２），…，Ｂ（５），

のように１０個の変数を使うことになったと考える。つまり、Ａ，Ｂには実は添え字が付いていて、$A_1$，$A_2$，…，$A_5$，$B_1$，$B_2$，…，$B_5$ のようになっていると考える。だから当然、例えば$A_1$と$A_2$は違う値を持っている。下のプログラムを参照されたい。

```
PROGRAM EXAM
IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
DIMENSION A(5)
A(1)=2.3
A(2)=3.4
A(3)=4.3
A(4)=7.7
A(5)=9.2
WRITE(6,*) A(1)
WRITE(6,*) A(2)
WRITE(6,*) A(3)
WRITE(6,*) A(4)
WRITE(6,*) A(5)
END
```

この例では、A（１）～A（５）の値がはじめに与えられていて、それがWRITE文でディスプレーに出力されるわけである。この例でわかるように、A（１），A（２），…，A（５）は全く別の数値を持つ別の変数と考えられる。従って、上のプログラムを書く代わりに、配列変数を使わないで、

```
    PROGRAM EXAM2
    IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
    P=2.3
    Q=3.4
    R=4.3
    S=7.7
    T=9.2
    WRITE(6,*) P
    WRITE(6,*) Q
    WRITE(6,*) R
    WRITE(6,*) S
    WRITE(6,*) T
    END
```

と書いても同じである。では、なぜあえて配列変数を書くかというと、
一番上のプログラムEXAMは、実は

```
      PROGRAM EXAM
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION A(5)
      DATA A/2.3,3.4,4.3,7.7,9.2/
      DO 100 I=1,5
      WRITE(6,*) A(I)
100   CONTINUE
      END
```

ともっと短く書き換えることができるのである。こちらのプログラムの方がはるかに簡単だ！例えば、１００個もの多数のデータを読み込ませたり計算させたり出力させたりするとき、変数の名前を１００個もいちいち名づけるのは不便である。そこで、配列変数を使うわけである。

実は配列変数はＤＯ文と併用したときに最も威力を発揮する。例えば、

```
      DO 100 I=1,5
      WRITE(6,*) B(I)
100   CONTINUE
```

上のＤＯループは、

```
WRITE(6,*) B(1)          ←  Ｉ＝１   (繰り返し１回目)
WRITE(6,*) B(2)          ←  Ｉ＝２   (繰り返し２回目)
WRITE(6,*) B(3)          ←  Ｉ＝３   (繰り返し３回目)
WRITE(6,*) B(4)          ←  Ｉ＝４   (繰り返し４回目)
WRITE(6,*) B(5)          ←  Ｉ＝５   (繰り返し５回目)
```

と同じである。やはりＤＯ／ＣＯＮＴＩＮＵＥ文ではさまれている部分が５回繰り返されているが、Ｉの値が、１回目はＩ＝１、２回目はＩ＝２、というように、５まで１ずつ増えてゆくことがわかることと思う（それが、Ｉ＝１，５　の意味なのだ！）。

このような理解の上で、例題４をもう一度見てほしい。

※問題７　　１５個の実数を順にキーボードから読み込み、その総和を出力するプログラムを作れ。
(プログラム名MON7.FOR)

## §４－７　条件文（ＩＦ）

例１．

```
      PROGRAM REI1
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      READ(5,*) III
      J=0
      IF(III.GT.100) J=101
      IF(III.LE.50) J=1
      WRITE(6,*) J
      END
```

ＩＦ文は、条件を判断するための文である。上の例では、ＩＩＩ＞１００ならばＪ＝１０１となり、ＩＩＩ＝５０ならばＪ＝１となる。つまり、ＩＦ（……）の条件（…の部分のことを論理式という）が満たされる場合には、実行文（上の例では、J=101やJ=1のこと）が実行される。

①　ＩＦ文はこのように、論理式と一緒に使う。

| | |
|---|---|
| ＞ | ＧＴ |
| ≧ | ＧＥ |
| ＜ | ＬＴ |

≦　　ＬＥ
＝　　ＥＱ
≠　　ＮＥ

② ＩＦ（Ａ．ＮＥ．Ｂ）　のような形に書くこと（ピリオドで区切る）。

※問題８　整数を読み込み、それが偶数であるか奇数であるかを判断するプログラムを考えよ（偶数なら１００を、奇数なら９９９を画面に表示させよ）。
（プログラム名MON8.FOR）

例２．

```
PROGRAM REI2
IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
READ(5,*) III
J=0
I=0
IF(III.GT.100) THEN
J=101
I=201
END IF
WRITE(6,*) J,I
END
```

例２のような場合は、論理式が満足されるときに、ＩＦ（…）　ＴＨＥＮ　と
ＥＮＤ　ＩＦで挟まれた部分が実行されることになる。１つのＩＦに対して、実行文が２行以上存在するときは、このような形に書けばよい。これをブロックＩＦ文という。

③ ブロックＩＦ文の形式は、

```
ＩＦ（……）　ＴＨＥＮ
　　……
　　……
ＥＮＤ　ＩＦ
```

である。

## §４－８　ＤＡＴＡ文

つまらないことであるが、変数の初期値はＤＡＴＡ文を使っても書くことができる。例えば、前節の例２の場合、４行目、５行目の「J=0」「I=0」をステートメントとして書

くかわりに

```
PROGRAM REI2
IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
DATA J,I/0,0/
READ(5,*) III
IF(III.GT.100) THEN
J=101
I=201
END IF
WRITE(6,*) J,I
END
```

と書き換えができる。

① ＤＡＴＡ文は、

ＤＡＴＡ　ＡＢＣ／ｅ／

の形式である（これは変数ＡＢＣの初期値がｅの場合）。

配列変数に対しては、例えば１０００次元の変数　Ａ（１０００）に対しては、すべての要素を０．０に設定する場合は、

ＤＡＴＡ　Ａ／１０００＊０．０／

と書けばよい。

## §４－９　ＧＯ　ＴＯ文

ＧＯ　ＴＯ　文は飛び越しを命令する。

例３．

```
      PROGRAM REI3
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      READ(5,*) A
      IF(A.NE.0.0) GO TO 123
      III=9999
      WRITE(6,*) III
      STOP
123   CONTINUE
      B=1234.0/A
      WRITE(6,*) B
      END
```

例３では、０．０を読み込ませた場合、ゼロで割り算はできないので、９９９９を表示させて終了する仕組みになっている。
「ＧＯ　ＴＯ」の分岐の行き先は、「ＣＯＮＴＩＮＵＥ文」で受ける。
要点は

飛ばすとき：　　ＧＯ　ＴＯ　ｅ　　（ｅは文番号の数字）
受けるところ：　ｅ　ＣＯＮＴＩＮＵＥ

ということである。

※問題８ａ　問題４のプログラムを、ＤＯ文を使わずに、「ＧＯ　ＴＯ」と「ＣＯＮＴＩＮＵＥ」を使って書き換えよ。
（プログラム名MON8A.FOR）

※問題９　問題１のプログラムで、$x_3 = 0$を入力してもエラーがでないようにプログラムを修正せよ（$x_3 = 0$のときは割り算を実行しないようにせよ）。
（プログラム名MON9.FOR）