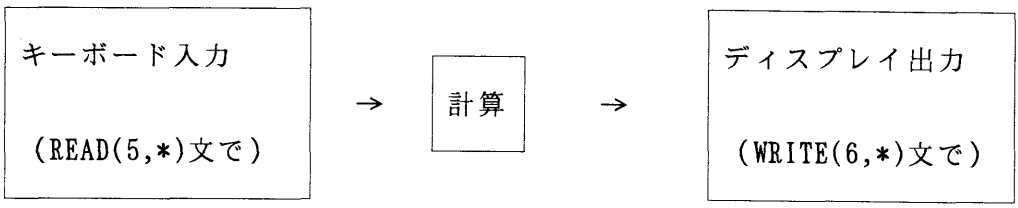
§５．ＦＯＲＴＲＡＮによるプログラミングの基本（その２：ファイル入出力）

これまで作成してきたプログラムでは、データの入出力は、

```
READ（５，＊）　A
WRITE（６，＊）　A
```

のように、キーボードから読み込み、ディスプレイに表示させるやり方で行なってきた。
つまり、



のような流れになっている。

例えば、次の例１のプログラムでは、５つの数値（実数）を読み込みそれぞれの値を３倍してディスプレイに表示させることになる。

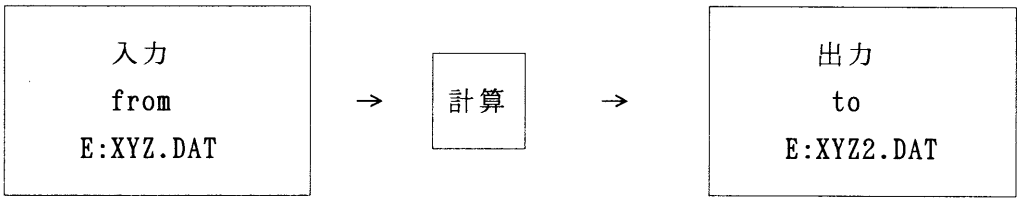
例１．PQR.FOR

```
      PROGRAM PQR
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION A(5),B(5)
C
      READ(5,*) (A(I),I=1,5)
C
      DO 100 I=1,5
      B(I)=A(I)*3.0
 100  CONTINUE
C
      WRITE(6,*) (B(I),I=1,5)
C
      DO 200 I=1,5
      WRITE(6,*) B(I)
 200  CONTINUE
C
      END
```

例１のプログラムで、キーボードから数値データを入力する代わりにＭＳ－ＤＯＳ上のファイルから数値を読み込ませたり、ディスプレイに結果を出力する代わりにファイルに結果を書き込んだりするにはどのようにすればよいであろうか。

この節では、ＭＳ－ＤＯＳ上のファイルからデータを読み込んだり、ファイルにデータを書き込んだりする方法について学ぶ。

例１で、XYZ.DAT（Ｅドライブ上）から変数Ａ（Ｉ）の値を読み込み、XYZ2.DATに変数Ｂ（Ｉ）の値を書き込むプログラムを考える。
流れとしては、



のようなものにしたい。次に示す例２のプログラムは、例１のプログラムをこの流れになるように書き換えたものである。

例２．XYZ.FOR

```
      PROGRAM XYZ
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION A(5),B(5)
C
      OPEN(1,FILE='E:XYZ.DAT')
      OPEN(2,FILE='E:XYZ2.DAT')
C
      READ(1,*) (A(I),I=1,5)
      DO 100 I=1,5
      B(I)=A(I)*3.0
 100  CONTINUE
C
      WRITE(2,*) (B(I),I=1,5)
      DO 200 I=1,5
      WRITE(2,*) B(I)
 200  CONTINUE
C
      CLOSE(1)
      CLOSE(2)
      END
```

例２の例１との違いは、

i)　ＯＰＥＮ文、ＣＬＯＳＥ文が入っている。

ii)　ＲＥＡＤ（１，＊），ＷＲＩＴＥ（２，＊）のように、ＲＥＡＤ文・ＷＲＩＴＥ文の中の番号が”５”、”６”ではない。

※　ファイル入出力を行うためには次の①～③の要領でプログラミングすればよい。

①　ＲＥＡＤ文・ＷＲＩＴＥ文が出て来るより前に、ＯＰＥＮ文を書く。プログラムの冒頭のＤＩＭＥＮＳＩＯＮ文やＤＡＴＡ文などの直後に書くとよい。

例２では、

ＯＰＥＮ（１，ＦＩＬＥ＝……）

ＯＰＥＮ（２，ＦＩＬＥ＝……）

といったＯＰＥＮ文が出てきている。

【書き方】　ＯＰＥＮ（番号，ＦＩＬＥ＝’ファイル名’）

・番号は１～４、７～９の間の整数を使うこと
・ファイル名には、クオーテーションマーク　’…’を忘れないこと
・ファイル名はドライブ名（例２ではＥ：）を必ず頭につけること（そのファイルがプログラムの存在するドライブにある場合でも必要）

②　ＲＥＡＤ文・ＷＲＩＴＥ文の中の番号（これを「装置指定子」という：従来の端末入出力の場合は、ＲＥＡＤ文は”５”、ＷＲＩＴＥ文は”６”を指定していた）を１～４、７～９の間の任意の番号に替える。”５”、”６”は、端末入出力用ということに決まっているので使わないこと。

例２では、

ＲＥＡＤ（１，＊）

ＷＲＩＴＥ（２，＊）

となっている。

①のＯＰＥＮ文の中の番号は、このＲＥＡＤ／ＷＲＩＴＥ文の中の番号（装置指定子）に合わせること。ＯＰＥＮ文の中のファイル名は、ＲＥＡＤ／ＷＲＩＴＥするファイル名を書く。

例２のＯＰＥＮ文は、

ＯＰＥＮ（１，ＦＩＬＥ＝’Ｅ：ＸＹＺ．ＤＡＴ’）

ＯＰＥＮ（２，ＦＩＬＥ＝’Ｅ：ＸＹＺ２．ＤＡＴ’）

となっている。ＲＥＡＤ（１，＊），ＷＲＩＴＥ（２，＊）となっているので、

ＲＥＡＤ（データの入力）は、Ｅドライブ上のＸＹＺ．ＤＡＴから、
ＷＲＩＴＥ（データの出力）は、Ｅドライブ上のＸＹＺ２．ＤＡＴにされる
わけである。
　要するに、１、２というのはファイル名をいちいち書く代わりに使われる指標
である。

③　ＲＥＡＤ／ＷＲＩＴＥ文の後にＣＬＯＳＥ文を書く。ＣＬＯＳＥ文は、ファイル入出力を終了する命令なので、ＣＬＯＳＥ文が出てきた後に、例えば
ＲＥＡＤ（２，…）のような文が出てきてはいけない。従って、ＣＬＯＳＥ文はプログラムの一番最後、ＥＮＤ文の直前に書くのがよい。

例２では、
　ＣＬＯＳＥ（１）
　ＣＬＯＳＥ（２）
となっている。

【書き方】　　ＣＬＯＳＥ（番号）

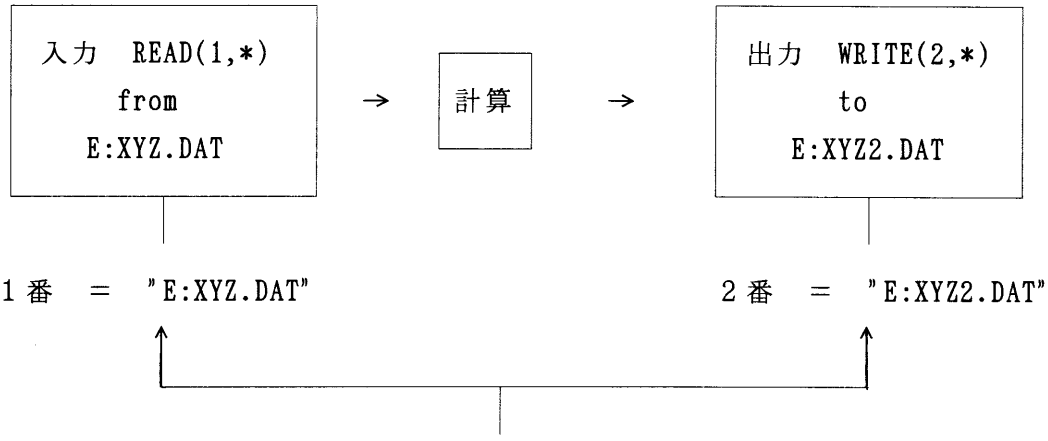
【注意】　当然の話であるが、ＲＥＡＤするファイル（例２ではＥ：ＸＹＺ．ＤＡＴ）は、プログラムを走らせる前に存在していなければならない。また、Ａ（Ｉ）に対応する数値データもその中になければならない。
　ＷＲＩＴＥするファイル（例２では、Ｅ：ＸＹＺ２．ＤＡＴ）は、予め存在していてもよいし、存在していなくてもよい。プログラムを走らせる段階で、Ｅ：ＸＹＺ２．ＤＡＴがなければ、自動的にこの名前のファイルが作られ、データが書き込まれる。もし、予めＸＹＺ２．ＤＡＴが存在すれば、プログラムを走らせた時点でデータの内容は書き換えられる。
【注意】　ＲＥＡＤするファイルとＷＲＩＴＥするファイルが同じであることは許されない。

【参考】　例２でできたＯＰＥＮ／ＣＬＯＳＥ文は、指定子をできるだけ少なくした最も簡単な形（指定子は、装置指定子とファイル名の２つだけしか与えていない）に書かれている。もっと厳密に書けば、

```
OPEN(1,FILE='E:XYZ.DAT',STATUS='OLD',ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(2,FILE='E:XYZ2.DAT',STATUS='NEW',ACCESS='SEQUENTIAL',FORM='FORMATTED')
```

となる。各指定子の意味は参考書を参照のこと。意味がわかれば、今日の授業でとりあげた以外のいろいろなファイル操作ができるようになるはずである。

まとめると例２の流れは、



当然２つ以上のファイルから読み込んだり、２つ以上のファイルに出力させたりすることもできる。

実行例：

予め、ＶＺエディターで、Ｅドライブ上に、ＸＹＺ．ＤＡＴを作成しておく。ファイルの中身は、

2.0　4.0　5.0　7.0　10.0

のように５つの値が入っている。

例２のプログラム（ＸＹＺ．ＦＯＲ）を作成する。プログラムを走らせる前には、ＸＹＺ２．ＤＡＴはまだ存在していない。

ディスプレイ表示

```
E>DIR/W

 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:¥

XYZ      FOR    XYZ      DAT    FORTRN01 DAT    FORTRN02 DAT    FORTRN03 DAT
FORTRN04 DAT    FORTRN05 DAT    FORTRN06 DAT    FORTRN01 COD    FORTRN02 COD
FORTRN03 COD    FORTRN04 COD    FORTRN05 COD    FORTRN06 COD
       14 個のファイルがあります.
```

プログラムの翻訳・結合（LINK）・実行を行うと、

```
E>XYZ

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

のように画面に表示され、ＸＹＺ２．ＤＡＴが自動的に作成される。
ＸＹＺ２．ＤＡＴの中身をＶＺエディターで見てみると、プログラムの
ＷＲＩＴＥ（２，＊）文で出力された数値データが書かれていることがわかる。

※問題１０
上の例を自分で実行させて、ファイル入出力ができることを確認せよ（ＸＹＺ．ＤＡＴは各自ＶＺエディターでプログラム実行前に作成しておくこと）。
（プログラム名　XYZ.FOR）

※問題１１
問題１０で作ったプログラムを、フロッピーディスク上のファイルとの間で入出力ができるように変更せよ。但し、計算結果は２つのファイル（XYZ2.DATおよび XYZ3.DAT）に出力させよ。作成したプログラムを実行させよ。（周知のこととは思うが、フロッピーディスクのドライブはＣ：またはＤ：である。予め　E>COPY XYZ.DAT C: として、XYZ.DATをフロッピーにコピーしておくこと）。（プログラム名　MON11.FOR）

さて、例２のようなプログラムを再び使って、今度はＦＧＨ．ＤＡＴなるファイルからデータを読ませるにはどうすればよいであろうか。当然、ＯＰＥＮ文を
　ＯＰＥＮ（１，ＦＩＬＥ＝’Ｅ：ＦＧＨ．ＤＡＴ’）
のように書き換えればよい。いま、そのように書き換えたプログラムの名前を仮に
FGH.FOR　とする。
しかし、同じたぐいの数値処理を行うのにもかかわらず、一旦作ったプログラムを書き換えて翻訳・結合・実行の手順を一からやり直すのはいかにも面倒であるし、それでは、１つの入力ファイルにつき１つのプログラムが必要なことになる（つまり、入力ファイルがXYZ.DATのときはXYZ.FORを走らせ、入力ファイルがFGH.DATのときにはFGH.FORを走らせることになり、同じようなプログラムが２つできてしまう）。
そこで、下の例３のようにプログラムを書いておけば、実行する段階でその都度入力ファイル名を自由に指定できることになり、便利である。例２のようなオーソドックスなファイル入出力のやり方のパターンに慣れれば、例３のような方法を使うことを勧める。

例３．XYZ.FOR（アンダーラインが例２からの変更箇所）

```
      PROGRAM XYZ
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION A(5),B(5)
      CHARACTER*15 IFLI
C
      WRITE(6,*) 'INPUT FILE NAME FOR INITIAL DATA'
      READ(5,101) IFLI
 101  FORMAT(A15)
C
      OPEN(1,FILE=IFLI)
      OPEN(2,FILE='E:XYZ2.DAT')
C
      READ(1,*) (A(I),I=1,5)
C
      DO 100 I=1,5
      B(I)=A(I)*3.0
 100  CONTINUE
C
      WRITE(2,*) (B(I),I=1,5)
C
      DO 200 I=1,5
      WRITE(2,*) B(I)
 200  CONTINUE
C
      CLOSE(1)
      CLOSE(2)
      END
```

【参考】　このプログラムで、CHARACTER*15 IFLI　とは、変数ＩＦＬＩは、数値ではなく、文字列１５文字をデータに持っているという意味である。（普通、変数は数値をデータとして持っている。例えば、Ａ＝１．０というステートメントがあれば、変数Ａは数値１．０をデータとして持っているわけである。例３の場合はＩＦＬＩは数値でなく、例えば'E:FGH.DAT'といった文字列をデータに持つことができるわけである。

READ(5,101) IFLI　とは、変数（文字列）ＩＦＬＩをキーボード（５番）から、文番号１０１番のＦＯＲＭＡＴ文に指定する書式で読み込めという意味である。ＦＯＲＭＡＴ文FORMAT(A15)は、文字型データ１５文字という書式を指定する。

以上何のことか意味がわからないかもしれないが、今は鵜呑みでよい。後ほど解説する。

例３のプログラムを実行させた例は以下の通り。

```
E>XYZ

INPUT FILE NAME FOR INITIAL DATA
E:FGH.DAT                                    ← 入力ファイル名をキーボード
                                               から入れる
STOP - PROGRAM TERMINATED.
PRESS <RETURN> TO COMMAND MODE.

E>
```

このように、INPUT FILE NAME FOR INITIAL DATA
と画面に表示された後、入力待ちの状態になっているので、ここで、入力ファイル名を入れてやれば、そのファイルからデータを読み取って実行される。

※問題１２

例３のプログラムをさらに、WRITE(2,*)文の出力ファイル名も実行時にキーボード入力で指定できるように変更せよ。

答)

```
      PROGRAM XYZ
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(5),B(5)
      CHARACTER*15 IFLI,OFLI
C
      WRITE(6,*) 'INPUT FILE NAME FOR INITIAL DATA'
      READ(5,101) IFLI
      WRITE(6,*) 'OUTPUT FILE NAME FOR FINAL DATA'
      READ(5,101) OFLI
 101  FORMAT(A15)
C
      OPEN(1,FILE=IFLI)
      OPEN(2,FILE=OFLI)
C
```

＜以下同じ＞

※問題１３

関数　ｙ＝ｓｉｎ　ｘ　において、２０個のｘの値　$x_n = n\pi / 10$　（ｎ＝１，２，…，２０）に対するｙの値，y_n，をそれぞれ求め、（x_n，y_n）の数値の組（２０点）を、フロッピーディスクに出力させるプログラムを作成し、実行させよ。

（プログラム名　MON13.FOR）

（ヒント：　ｓｉｎ　ｘ　はＦＯＲＴＲＡＮでは　ＳＩＮ（Ｘ）と書く。
ＤＯ文を使うこと。できるだけ配列変数を使うこと。）

※問題１４

問題１３でフロッピーディスクに作成したデータを読み込み、y_n（ｎ＝１，２，…，２０）をそれぞれ　$\pi / 10$　倍した値の総和を求め、結果をディスプレーに表示させるプログラムを作成せよ。プログラムを実行させよ（結果はいくらか？）。

（プログラム名　MON14.FOR）

（要するに、$S = \sum_{n=1}^{20} y_n \cdot \pi / 10$　を求める）

以上ＦＯＲＴＲＡＮ文法に関する練習問題（問題１５～問題２７）

※問題１５

級数　$\sum_{k=0}^{n} (2k+1)^2$

を求めるプログラムを作成せよ。（プログラム名MON15.FOR）

※問題１６

乗積　$\prod_{k=2}^{n} (k-1)^2$

を求めるプログラムを作成せよ。（プログラム名MON16.FOR）

※問題１７

４次式

$y = 1.3x^4 - 0.9x^3 + 6.3x^2 + 0.4x + 2.2$

の値を、０．１から０．２刻みで３．４までのｘに対して計算し、ｘとｙを出力するプログラムを作成せよ。（プログラム名MON17.FOR）

※問題１８

任意の自然数ｎに対し、（２ｎ－１）！！＝（２ｎ－１）（２ｎ－３）・…・３・１を計算するプログラムを作成せよ。（プログラム名MON18.FOR）

※問題１９

指数関数のＭａｃｌａｕｒｉｎ展開は、

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

である。上の展開式の右辺の無限級数を第ｍ項まで求めることによって、$e^{0.1}$ を近似的に求めるプログラムを作成せよ。ＦＯＲＴＲＡＮの組み込み関数を使えば e^x はＥＸＰ（Ｘ）と書けば求められる。級数の和として得られた値を、組み込み関数を使って得られた値と比較せよ。（プログラム名MON19.FOR）

※問題２０

次のような関係が知られている。

$$\prod_{r=1}^{n-1} \sin\frac{r\pi}{n} \left(= \sin\frac{\pi}{n} \sin\frac{2\pi}{n} \cdots\cdots \sin\frac{(n-1)\pi}{n} \right) = \frac{n}{2^{n-1}}$$

左辺の乗積を求めるプログラムを作り、右辺の値と比較せよ。（プログラム名MON20.FOR）

※問題２１

３次元空間の２つの点の座標（x_1，y_1，z_1），（x_2，y_2，z_2）を読み込み、２点間の距離を求めるプログラムを作れ。（プログラム名MON21.FOR）

※問題２２

３角形の３辺の長さａ，ｂ，ｃを読み込み、ヘロンの公式によって面積を求めるプログラムを作成せよ。

（ヘロンの公式：

$S = \{w(w-a)(w-b)(w-c)\}^{1/2}$，$w = (a+b+c)/2$）

（プログラム名MON22.FOR）

※問題２３

フィボナッチ数列の最初の２０項を求めて出力するプログラムを作成せよ。

（フィボナッチ数列とは、最初の２項が$f_1=0$，$f_2=1$で始まり、以降は順次、反復公式

$f_n=f_{n-2}+f_{n-1}$

を用いて定められる数列のことである。）

（プログラム名MON23.FOR）

※問題２４

ｎ個の数値（実数）を読み込んで、その算術平均と幾何平均を求めるプログラムを作成せよ。（プログラム名MON24.FOR）

※問題２５

Ｎ個の整数を読み込ませて、その中の最大値と最小値をみつけるプログラムを、ＩＦ文を使って書け。

（プログラム名MON25.FOR）

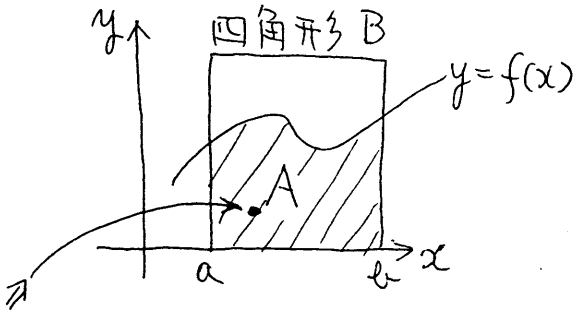
※問題２６

Ｐａ（パスカル）単位で表された圧力を読み込んで、ＧＰａ，ＭＰａ，ｋＰａ，Ｐａで表現して出力するプログラムを作成せよ。（Ｇ（ギガ）：10^9；　Ｍ（メガ）：10^6）

（プログラム名MON26.FOR）

※問題２７【上級者用応用問題：モンテカルロ法による数値積分】

積分 $\int_a^b f(x)\,dx$ を考える（ｆ（ｘ）≧０とする）。この図形上の意味は、区間ａ≦ｘ≦ｂでｘ軸と関数ｙ＝ｆ（ｘ）で囲まれた部分の面積である。その部分Ａを４角形Ｂで囲み、その中にランダムに点を打っていく。



打った点が、もし部分Ａに含まれているならカウントし、Ａに含まれないならカウントしないことにして、たくさんの点をランダムに打ち続ける。十分点を打った後、打った点

の合計N_Bと、カウントされた点の合計N_Aを調べる。図からわかるように、次の関係が成立する。

$$\frac{N_A}{N_B} \fallingdotseq \frac{\text{部分Aの面積}}{\text{部分Bの面積}}$$

長方形の部分Bの面積は簡単に求められ、左辺もすぐ計算できるので、部分Aの面積が（近似的に）求められることになる。このような方法をモンテカルロ法と呼ぶ。

モンテカルロ法を使って、原点を中心にして半径1の円を書き、その第1象限にある部分の面積を求めることによって、πの近似値を求めよ。このとき、区間［0，1］上の一様乱数を2つ作り、x，yとすると、$x^2+y^2 \leqq 1$なら、点（x，y）は円の中にある、ということを参考にしてプログラムを作れ。

0以上1未満の一様乱数は、組み込み関数　ＲＡＮＤＯＭ　を使って発生させよ。

（使い方：　　P=RANDOM(0)

　　Q=RANDOM(0)

と書くと、別の乱数Ｐ、Ｑが出て来る。）

（プログラム名MON27.FOR）

これまで作成してきたプログラムでは、データの入出力は、

```
ＲＥＡＤ（５，＊）　Ａ
ＷＲＩＴＥ（６，＊）　Ａ
```

のように書かれていた。

実は、ＲＥＡＤ／ＷＲＩＴＥ文の括弧の中の”＊”は、書式の設定をしないという意味だったのだ。書式とは、例えば、

Ａ＝１２３４５．６７８

と定義された変数Ａの値を（ディスプレイやファイルに）出力させるとき、

１．２３Ｅ４	と表示させるか
１２３４５．６	まで表示させるか
１２３４５．６７	まで表示させるか、

といった表示のさせ方のことである。これは単に表示のさせ方だけの問題である（Ａの数値自体は不変である）！

書式を指定するためには、ＦＯＲＭＡＴ文を使う。ＦＯＲＭＡＴ文は必ずＷＲＩＴＥ文やＲＥＡＤ文とセットにして使う。書き方は、例えば、

```
        ＷＲＩＴＥ（６，１２３）　Ａ
１２３　ＦＯＲＭＡＴ（……）
```

のように、ＷＲＩＴＥ文の括弧の中でＦＯＲＭＡＴ文の文番号を指定し、ＦＯＲＭＡＴ文は括弧の中で書式の指定をする。すると（……）の中の指定に応じて、Ａの値が、１．２３Ｅ４と出力されたり、１２３４５．６７と出力されたりすることになる。

ＦＯＲＭＡＴ文はそのプログラムの中だとどこに書いてもよい。つまりＷＲＩＴＥ文がでてくるずっと前にも書いてもよいし、ずっと後でもよい（もちろんプログラムの先頭部の宣言文の前や、ＥＮＤ文のあとに書いてはダメ）。

ＦＯＲＭＡＴ文の指定では、出力する変数が、整数型か実数型か、あるいは文字型かによって指定の書き方（記号）が違っている。とりあえず以下の例を見てほしい。

例１．整数型変数の出力

```
PROGRAM SYO1
IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
J=123456789
K=987654321
```

```
      WRITE(6,*)   J,K
      WRITE(6,100) J
      WRITE(6,110) J
      WRITE(6,120) J
      WRITE(6,130) J
      WRITE(6,140) J,K
      WRITE(6,150) J,K
      WRITE(6,160) J,K
      WRITE(6,170) J,K
  100 FORMAT(3X,I4)
  110 FORMAT(3X,I9)
  120 FORMAT(6X,I9)
  130 FORMAT(3X,I12)
  140 FORMAT(3X,2I9)
  150 FORMAT(3X,I9,I9)
  160 FORMAT(3X,I9,3X,I9)
  170 FORMAT(3X,2(I9,3X))
      END
```

例１の例えば１１０番のＦＯＲＭＡＴ文に出てきた”３Ｘ”というのは、空白を３カラム文あけよという意味で、Ｉ９は整数を９桁で表示させよという意味である。この結果、

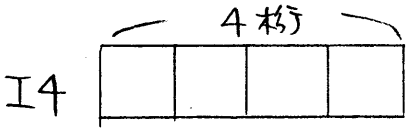
ＷＲＩＴＥ（６，１１０）　Ｊ

では、空白が３カラム出力された後に（同じ行に）、Ｊの値が９桁で表示されることになる。

整数型変数を出力させるには、このような「Ｉ型編集記述子」を使って指定する。

例えば、

Ｉ４と指定すると、次のようになる。



１７０番の　２（Ｉ９，３Ｘ）　は、　Ｉ９，３Ｘ，Ｉ９，３Ｘ　と同じ意味である。つまり、「はじめの数値（Ｊ）が９桁で表示された後、空白３カラム、次の数値（Ｋ）が９桁で表示、空白３カラム」という書式で出力される。

つまり、

ｎＩｗ

ｎは使用するＩ型編集子の個数、ｗは桁数。

例２．実数型変数の出力

```
      PROGRAM SY02
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      P=12345.67
      Q=67890.12
      WRITE(6,*)   P,Q
      WRITE(6,100) P
      WRITE(6,110) P
      WRITE(6,120) P
      WRITE(6,130) P
      WRITE(6,140) P
      WRITE(6,150) P
      WRITE(6,160) P,Q
      WRITE(6,170) P,Q
  100 FORMAT(3X,F4.0)
  110 FORMAT(3X,F6.0)
  120 FORMAT(6X,F8.2)
  130 FORMAT(3X,E4.0)
  140 FORMAT(3X,E6.0)
  150 FORMAT(3X,E15.7)
  160 FORMAT(3X,2(F21.10,2X))
  170 FORMAT(3X,2(E21.10,2X))
      END
```

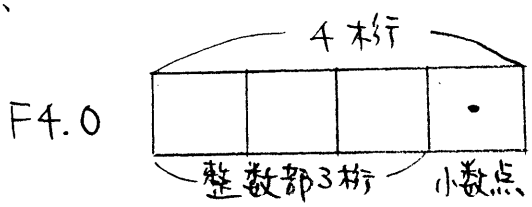
実数型変数を出力させるには、「Ｆ型編集記述子」や「Ｅ型編集記述子」を使って指定する。Ｅ型は、出力結果を指数表示で行うもので、Ｆ型は単なる小数として表示するものである。

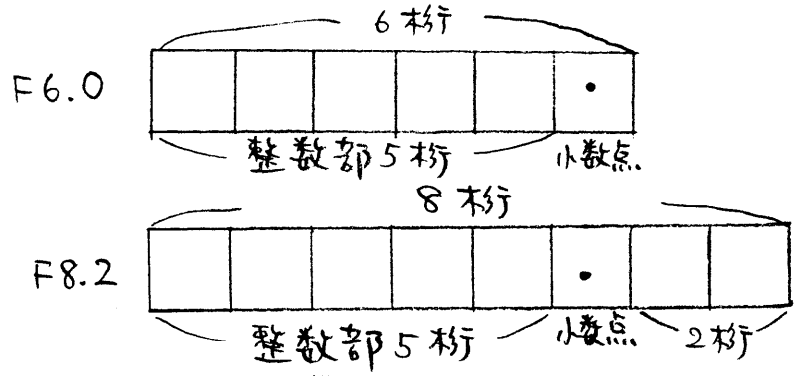
Ｆ型編集記述子は次のように編集を行う。

ｎＦｗ．ｄ

ｎは使用するＦ型編集子の個数、ｗは全桁数、ｄは小数部の桁数。小数点も桁数に含まれるので、整数部の桁数はｗ－１－ｄになる。

例えば、

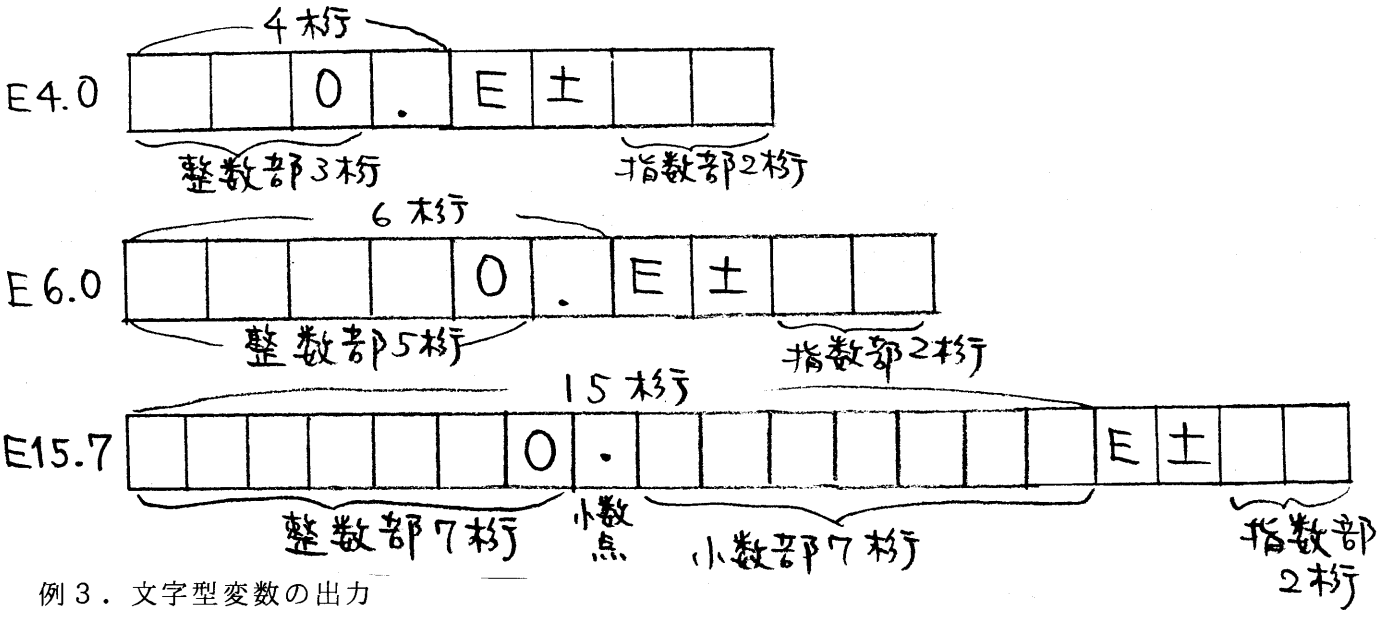




E型編集記述子は次のように編集を行う。

nEw．d

nはE型編集子の個数、wは全桁数、dは小数の桁数



例３．文字型変数の出力

```
      PROGRAM SY03
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      CHARACTER*15 IJK,PQR
      IJK='ABCDEFGHIJ'
      PQR='0123456789'
      WRITE(6,100) IJK
      WRITE(6,110) IJK
      WRITE(6,120) IJK,PQR
      WRITE(6,130) IJK,PQR
  100 FORMAT(3X,A6)
  110 FORMAT(3X,A10)
  120 FORMAT(3X,A10,2X,A10)
```

```
130   FORMAT(3X,A12,2X,A12)
      END
```

周知の通り、冒頭のＩＭＰＬＩＣＩＴ文で、頭文字がＡ～Ｈ，Ｏ～Ｚで始まる変数は倍精度実数型、頭文字がＩ～Ｎで始まる変数は４バイト整数型であることを宣言している。ある変数が、「文字型」であることを宣言するには、ＣＨＡＲＡＣＴＥＲ文を使う。例３に示すとおり、

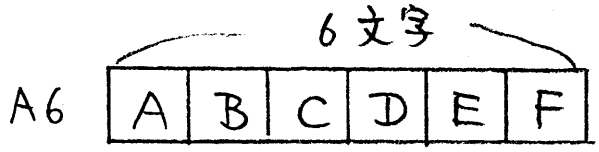
　ＣＨＡＲＡＣＴＥＲ＊ｎ　ＩＪＫ

は変数ＩＪＫにｎ個の文字数をもつ文字列が代入されることを宣言する。例３では実際には、ＩＪＫには１０個の文字数をもつ文字列’ＡＢＣＤＥＦＧＨＩＪ’が代入されている。このような文字型変数を実際にどのように使うかという具体例は、配布資料４の例３を参照。

文字型変数を出力させるには、「Ａ型編集記述子」を使って指定する。
Ａ型編集記述子は次のように編集を行う。

ｎＡｗ

ｎは使用するＡ型編集子の個数、ｗは出力する文字数。
例えば、



例４．任意の文字列の出力

```
      PROGRAM SY04
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      WRITE(6,*)  'ABCDEFGHIJ'
      WRITE(6,100)
  100 FORMAT(3X,'ABCDEFGHIJ')
      A=1.0
      WRITE(6,110) A
      WRITE(6,120) 'A=  ',A
  110 FORMAT(3X,'A=',2X,E12.5)
  120 FORMAT(3X,A4,E12.5)
      END
```

任意の文字列を出力させるには例４のような方法がある。

※問題２８

例１～４のプログラムを作成し、実行させ、どのように出力されるかを見よ。出力結果の画面コピーをとれ。

＊＊＊＊の表示は、まともに数値の表示ができなかったことを意味している（変数の桁数よりＦＯＲＭＡＴ文で指定した桁数が少なかった、など）。その原因を考えよ。

ＲＥＡＤ文に対しても、同様の書式指定ができる。但し、数値を入力する際には、指定した書式で入力させなければならない。
実際上、文字型変数の入力以外は、ＲＥＡＤ（５，＊）　のように書式指定なしで
ＲＥＡＤ文を書いておいた方が、どのような書式の数値データでも入力できて便利である。

フロッピーディスク上のファイルなどにＷＲＩＴＥ文で出力させた数値データを、別のプログラムでＲＥＡＤ文で読み込む場合、出力時のＷＲＩＴＥ文の書式と、入力時のＲＥＡＤ文の書式は同じにしておいた方が（初心者は）無難である。特に、入力データよりも少ない桁数の書式で読み込んだ場合、入力された数値の正しさは保証されないので注意が必要である。これを避けるため、例えば、

```
    DO 100 I=1,N
    WRITE(1,200) A(I),B(I)
100 CONTINUE
200 FORMAT(2X,E12.5,2X,E12.5)
```

で出力された数値Ａ（Ｉ），Ｂ（Ｉ）を読み込む場合は、

```
    DO 300 I=1,N
    READ(1,400) A(I),B(I)
300 CONTINUE
400 FORMAT(2X,E12.5,2X,E12.5)
```

のように同じ書式で読み込むのが無難。もっともこの場合は、
ＲＥＡＤ（１，＊）　Ａ（Ｉ），Ｂ（Ｉ）　で読み込んでも問題は生じない。

ここでとりあげたＦＯＲＭＡＴの指定は、数ある書式指定の中のごく一部である。詳しくは、ＦＯＲＴＲＡＮの文法書を参照されたい。

これまでに

① 整数と実数の区別
② ステートメントの意味（右辺の値を左辺に代入するということ、組込み関数
③ 繰り返し計算の方法（ＤＯ文、配列変数）
④ 条件判断・分岐の方法（ＩＦ／ＧＯ　ＴＯ文）
⑤ データの入出力の方法（ＲＥＡＤ／ＷＲＩＴＥ／ＯＰＥＮ／ＣＬＯＳＥ／ＦＯＲＭＡＴ文）

を習得した。実はこれで科学技術計算に必要最小限の文法はマスターしたのだ！

以上で、ＦＯＲＴＲＡＮの解説を終了する。これで、ＦＯＲＴＲＡＮを使って必要最小限のプログラミングはできるようになった（はずである）。科学技術計算をするための最短コースを短時間で走ってきたので、多くの詳細を省略したし、科学技術計算の経験上、最も有用と思われる１つのやり方を勧めてきたに過ぎない。例えば、まだ他に、サブルーチン（副プログラム）、文関数定義文など、全く取り上げていない事柄もあるし、宣言文、条件文やＤＯ文についてももっといろいろな使い方があるが実習では取り上げなかった。それらについては、ＦＯＲＴＲＡＮの参考書を見て各自で補ってもらいたい。

次章では、科学技術計算の初歩についてとりあげる（例えば、データ解析や微分や積分をコンピューターで行うにはどうすればよいか）。