情報処理　（第11回）　(2017/6/26・茂木)

本日のテーマ

- 配列と data 文
- do 並びによる入出力

1　先週の演習の解答例

Q2・一様乱数のチェックその２

```
c23456
      program RandomCheck
      integer i, j, n
      integer ic(0:99) ! 0〜99 までのカウンター
      real x

      do j=0, 99        ! カウンターを 0 で初期化しておく
         ic(j)=0
      enddo

      n=100000
      do i=1, n
         x = rand()   ! この関数が呼ばれると 0〜1 の一様乱数を 1 つ発生する.
         j=100*x      ! 乱数の値から階級の番号を算定する.
         ic(j)=ic(j)+1   !例 0.00123 → 0.123 → 0 番目, 0.123 → 12.3 → 12 番目
      enddo

      do j=0, 99
         write(6,*)0.01*j,'〜',0.01*(j+1),' の確率は',1.0*ic(j)/n
      enddo
      end
```

Q3. ベクトルの内積

```
      program internalp
      real s, a(3), b(3) ! a,b は a(1)〜a(3), b(1)〜b(3) の 3 要素の一次元配列.
      a(1)=1.0
      a(2)=2.0
      a(3)=3.0
      b(1)=1.0
      b(2)=1.0
      b(3)=1.0

      s = a(1)*b(1)+a(2)*b(2)+a(3)*b(3)

      write(6,*)'a と b の内積は',s,' です. '
      end
```

2 data文

data 文で配列の初期化を行う例 (array1.f)

```
      program array1
      real s, a(3), b(3) ! a,b は a(1)～a(3), b(1)～b(3) の 3 要素の一次元配列.
      data a/1.0,2.0,3.0/, b/3*1.0/

      s = a(1)*b(1)+a(2)*b(2)+a(3)*b(3)

      write(6,*)'a と b の内積は',s,'です. '
      end
```

data 文はプログラムの開始時にあらかじめ変数に値を入れておく（初期化）ための宣言文で，たとえば，

```
real a(3), b(3)
data a/1.0,2.0,3.0/, b/3*1.0/
```

とすると，配列変数 a の値を，a(1)=3.0, a(2)=5.0, a(3)=1.0 に初期化します．配列変数 b の初期化は同じ値で初期化する場合の簡略記法で 1.0 が 3 つという意味です．b(1)=1.0, b(2)=1.0, b(3)=1.0 になります．

なお，data 文は配列だけでなく，通常の変数の初期化もできます．

```
integer i
real s
data s/0.0/, i/3/   ! 実数型変数 s を 0.0, 整数型変数 i を 3 で初期化
```

3 多次元配列

二次元配列（行列）の例 (array2.f)

```
c23456
      program array2
      implicit none
      integer i, j
      real D, a(3,3)   ! a は a(1,1), a(2,1), a(3,1), a(1,2),..., a(3,3) の
                       ! 9 つの要素の二次元配列.
      data a/1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0/      ! data 文で
                                                    ! 配列 a を初期化する


      s=a(1,1)+a(1,2)+a(1,3)+a(2,1)+a(2,2)+a(2,3)+a(3,1)+a(3,2)+a(3,3)

      write(6,*)'a(1,1) から a(3,3) までの和は',s,'です. '
      end
```

複数の添字を持つ多次元配列も可能です（7 次元まで）．array2.f では，3×3 の実数型二次元配列（数学では行列）を使用しています．二次元配列の宣言は次のようにします．

```
real a(3,3)
data a/1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0/
```

この宣言によって, a(1,1)=1.0, a(2,1)=2.0, a(3,1)=3.0, a(1,2)=4.0, a(2,2)=5.0, a(3,2)=6.0, a(1,3)=7.0, a(2,3)=8.0, a(3,3)=9.0 の 9 つの要素が使えるようになります.

この列挙した順番は記憶されるメモリ上の位置の順で, **左の添字が最も速く変化する順**です. 配列変数では, メモリ上にはこの順に連続して記憶され, 途中に別の変数が割り当てられる (allocate) ことはありません. この順番は例の二行目のように data 文で初期値を与えるときの順になるため重要ですので, 覚えておいてください.

4 DO 並びによる入出力

行列とベクトルの積（array3.f）

```
c23456
      program array3
      implicit none
      integer i, j
      real A(3,3), x(3), y(3)

      write(6,*)'3x3 行列 A とベクトル x を入力して下さい'
      read(5,*) ((A(i,j),j=1,3), i=1,3)   ! DO 並びで行列とベクトルを読込
      read(5,*) (x(i),i=1,3)              !  i,j の順番にも注意

      y(1)=A(1,1)*x(1) + A(1,2)*x(2) + A(1,3)*x(3)
      y(2)=A(2,1)*x(1) + A(2,2)*x(2) + A(2,3)*x(3)
      y(3)=A(3,1)*x(1) + A(3,2)*x(2) + A(3,3)*x(3)
                                                  ! DO 並びで結果を出力
      write(6,*)' 答は', (y(i), i=1, 3), ' です. '
      end
```

DO 並びは, READ 文, WRITE 文, DATA 文などで配列変数を用いるときの省略記法で, DO 文と同じように, DO 変数とその初期値, 終了値, 増分を丸括弧内に与えます. 増分を省略すると 1 と見なされます. 用いる DO 変数は整数型変数でなければなりません. また, 殆どの場合 1 になるため, 増分を付けない形式が普通です.

例 1. 一次元配列 x(1)〜x(3) に数値を入力する例

```
integer j
     ...略...
read(5,*)(x(j),j=1, 3)      ← read(5,*)x(1),x(2),x(3) と書いたのと
                              同じことになる
```

例 2. 二次元配列 C(1,1)〜C(3,3) の値を出力する例

```
do i=1, 3    ! 二次元配列 C の出力
   write(6,*)(C(i,j),j=1,3)  ← write(6,*)C(i,1),C(i,2),C(i,3) と
                               書いたのと同じことになる
enddo
```

出力は以下の順になります.

```
C(1,1) の値　C(1,2) の値　C(1,3) の値　改行
C(2,1) の値　C(2,2) の値　C(2,3) の値　改行
C(3,1) の値　C(3,2) の値　C(3,3) の値　改行
```

例 3. 二次元配列 C(1,1)〜C(3,3) の値を出力する例・ネスト（入れ子）

例.2 の 3 行は，DO 並びをネストすると下の例のように 1 行で書くことができます．なお，この例では書式を付けないと読みにくい出力になるので，FORMAT 文で書式を指定しています（教科書 10.6〜10.9 節参照・詳しい説明は次週）.

```
      write(6,1000)((C(i,j),j=1,3),i=1,3)
 1000 format(3f10.4)    ! 1 行に小数点以下 4 桁，計 10 桁の実数値（f）を 3 つ書く
                           という意味
```

例 4. DO 並びの前後で別の変数の入出力も行う例

例えば下のような read 文に対して, 1.0，2.0，3.0，4.0，5.0 と入力すると a=1.0, x(1)=2.0, ...，x(3)=4.0，s=5.0 となります.

```
      read(5,*) a, (x(j),j=1, 3), s
```

5　データの入力

サンプルプログラム array3.f を実行すると，2 つの read 文に対してそれぞれデータを入力する必要があります．一つ一つ手で入力しても構いませんが，データが大きくなると入力も面倒で，かつ，入力ミスも生じます.

そこで，入力するデータをあらかじめ別のファイルに入力しておいて，プログラムの実行時にそのファイルからデータを読み込むようにすると便利です．これは，コマンドプロンプトの「リダイレクト」という機能を使うと簡単で便利です（より高機能のファイル入出力は後日説明します）.

下のように，**入力データファイルをメモ帳で打ち込みます**．このファイルは Fortran のプログラムではないので，左に空白を入れる必要もありませんし，ファイル名（この例では in）も自由です.

入力データファイル in

```
1.0, 2.0, 3.0          ←行列 A
4.0, 5.0, 6.0
7.0, 8.0, 9.0
1.0, 2.0, 3.0          ←ベクトル x
```

array22.f の実行－リダイレクトの使用

```
C:¥work¥T66160¥20170626>f77 array3.f -o array3          ←コンパイル（普段通り）

C:¥work¥T66160¥20170626>array3 < in     ←実行・データはファイル in から読み込む.
 3x3 行列 A とベクトル x を入力して下さい     ← write(6,*) による出力は普段通り画面へ
 答は   14.   32.   50. です.
```

練習問題

(1) プログラム例 `array1.f` の内積の計算の部分を `DO` ループを用いたプログラムに書き換えよ.

(2) プログラム例 `array2.f` の要素の和を求める部分を `DO` ループを用いたプログラムに書き換えよ.

(3) 次の二つの3×3行列 $\boldsymbol{A}$ と $\boldsymbol{B}$ の和（$\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B}$）を求め，$\boldsymbol{C}$ を出力するプログラムを作成せよ.

$$\boldsymbol{A} = \begin{bmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 5.0 & 6.0 \\ 7.0 & 8.0 & 9.0 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 11.0 & 12.0 & 13.0 \\ 14.0 & 15.0 & 16.0 \\ 17.0 & 18.0 & 19.0 \end{bmatrix}$$

(4) プログラム例 `array3.f` の行列とベクトルの積の部分を `DO` ループを用いたプログラムに書き換えよ.