FORTRAN（と C）によるプログラミング

３　配列と反復

ここでは配列について反復処理をするプログラムを作成してみます。はじめに、3 個の要素を持つ配列を宣言して値を代入し、その後で、3 個の値を合計するためのプログラムを作ります。配列はプログラムの最初で宣言します。FORTRAN の場合、整数なら INTEGER、実数なら REAL で宣言します。合計を計算するときには、別に用意した変数にあらかじめゼロを代入しておき、配列の各要素の値を加算していきます。この過程で反復処理を行ないますが、FORTRAN の場合、DO 文を使います。

プログラム：

FORTRAN

```
C  配列の宣言は必ずプログラム単位の最初で行う。
C  ここでは実数の配列Aを宣言する。
C  INTEGERが整数、REALが実数である。
C  要素の個数は3個である。
      REAL A(3)

C  配列Aの各要素の値を代入する。
      A(1) = 1.
      A(2) = 3.
      A(3) = 5.

C  和の計算に変数Sを用いる。
C  初めに、Sの値にゼロを代入する。
      S = 0.

C  ここからDOループが始まる。
C  DO文から、DO文の番号と同じ行番号のついている行までの範囲を反復する。
C  ここでは、I=1、2、3について、3回反復する。
      DO 11 I=1,3

C  変数SにA(I)の値を加える。
        S = S + A(I)

C  ここでDOループが終了する。
C  CONTINUEと書かれている行では何も実行しない。
 11   CONTINUE

C  結果を標準出力に書き出す。
```

```
      WRITE(6,*) 'Sum =', S

      STOP
      END
```

（参考）C

```
#include <stdio.h>

int main(void)
{

/*  変数と配列の宣言は必ず関数の最初で行う。
    int が整数、float が浮動小数点である。  */
  int i;
  float a[3], s;

/*  配列 A の各要素の値を代入する。  */
  a[0] = 1.;
  a[1] = 3.;
  a[2] = 5.;

/*  和の計算に変数 S を用いる。
    初めに、S の値にゼロを代入する。  */
  s = 0.;

/*  ここから for ループが始まる。
    {}で囲まれている範囲を反復する。  */
  for (i=0; i<=2; i++)
  {

/*  変数 s に a[i]の値を加える。  */
    s = s + a[i];

/*  ここで for ループが終了する。  */
  }

/*  結果を標準出力に書き出す。  */
  printf( "Sum = %f¥n", s );

  return 0;
```

```
}
```

実行例：

```
/home/snaoki> f77 prog03_1.f
/home/snaoki> ./a.out
 Sum = 9.
```

配列の大きさを定数としてあらかじめ宣言しておくこともできます。FORTRAN の場合、PARAMETER 文で宣言しますが、この場合、定数の値は、そのプログラムの中では途中で変更することはできません。PARAMETER 文は配列の大きさ以外の定数を宣言するために使うこともできます。また、整数ではなく実数の定数も宣言できます。PARAMETER 文は、処理の開始よりも前に書く必要があります。

プログラム：

FORTRAN

```
C  PARAMETER文で定数IMAXを宣言する。
C  定数は変数とは異なり、値は固定である。
      PARAMETER (IMAX=3)

C  配列Aを宣言する。
      REAL A(IMAX)

C  配列Aの各要素の値を代入する。
      A(1) = 1.
      A(2) = 3.
      A(3) = 5.

C  変数Sにゼロを代入する。
      S = 0.

C  ここからDOループが始まる。
      DO 11 I=1,IMAX

C  変数SにA(I)の値を加える。
        S = S + A(I)

C  ここでDOループが終了する。
 11   CONTINUE

C  結果を標準出力に書き出す。
```

```
      WRITE(6,*) 'Sum =', S

      STOP
      END
```

（参考）C

```
#include <stdio.h>

int main(void)
{

/*  関数の最初で整数型変数 imax を宣言し初期値を与える。  */
  int imax=3, i;
  float a[imax], s;

/*  配列 A の各要素の値を代入する。  */
  a[0] = 1.;
  a[1] = 3.;
  a[2] = 5.;

/*  変数 s にゼロを代入する。  */
  s = 0.;

/*  ここから for ループが始まる。  */
  for (i=0; i<=imax-1; i++)
  {

/*  変数 s に a[i]の値を加える。  */
    s = s + a[i];

/*  ここで for ループが終了する。  */
  }

/*  結果を標準出力に書き出す。  */
  printf( "Sum = %f¥n", s );

  return 0;

}
```

実行例：

```
/home/snaoki> f77 prog03_2.f
/home/snaoki> ./a.out
 Sum = 9.
```

初期の段階で配列の中にあらかじめ値を入れておくこともできます。FORTRAN の場合、DATA 文を利用します。DATA 文によって初期の段階で変数や配列に代入されている値が決まりますが、この場合は、PARAMETER 文で宣言した定数とは異なり、あとで別の値を代入し変更することができます。DATA 文は PARAMETER 文と同様、処理の開始よりも前に書く必要があります。

プログラム：

FORTRAN

```
C  PARAMETER文で定数IMAXを宣言する。
      PARAMETER (IMAX=3)

C  配列Aを宣言する。
      REAL A(IMAX)

C  DATA文で配列Aの値を初期化する。
C  DATA文で指定した初期値はプログラム中で変更することができる。
      DATA A /1.,3.,5./

C  変数Sにゼロを代入する。
      S = 0.

C  ここからDOループが始まる。
      DO 11 I=1,IMAX

C  変数SにA(I)の値を加える。
        S = S + A(I)

C  ここでDOループが終了する。
 11   CONTINUE

C  結果を標準出力に書き出す。
      WRITE(6,*) 'Sum =', S

      STOP
      END
```

（参考）C

```
#include <stdio.h>

int main(void)
{

/*  関数の最初で整数型変数 imax を宣言し初期値を与える。  */
  int imax=3, i;

/*  浮動小数点型配列 a を宣言し初期値を与える。  */
  float a[3] = {1., 3., 5.};
  float s;

/*  変数 s にゼロを代入する。  */
  s = 0.;

/*  ここから for ループが始まる。  */
  for (i=0; i<=imax-1; i++)
  {

/*  変数 s に a[i]の値を加える。  */
    s = s + a[i];

/*  ここで for ループが終了する。  */
  }

/*  結果を標準出力に書き出す。  */
  printf( "Sum = %f¥n", s );

  return 0;

}
```

実行例：

```
/home/snaoki> f77 prog03_3.f
/home/snaoki> ./a.out
 Sum = 9.
```

2 次元の配列を宣言することもできます。ここでは、行列とベクトルとの積を計算するプログラムを作成します。例として、

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$$

行列A　ベクトルb　ベクトルc

という計算を実行するプログラムを書いてみます。

プログラム：

FORTRAN

```
C  PARAMETER文で定数IMAX、JMAXを宣言する。
      PARAMETER (IMAX=3,JMAX=2)

C  配列A、B、Cを宣言する。
C  配列AはIMAX行JMAX列の行列に対応する。
C  また配列Bは要素数JMAXのベクトル、
C  配列Cは要素数IMAXのベクトルに対応する。
      REAL A(IMAX,JMAX),B(JMAX),C(IMAX)

C  DATA文で配列A、Bを初期化する。
C  2次元配列の場合、
C  A(1,1),A(2,1),...,A(IMAX,1),A(1,2),A(2,2),...
C  の順である点に注意する。
      DATA A / 0., 1., 0.,
     +         -1., 0., 0./
      DATA B / 2.,-1./

C  行列AとベクトルBの積を計算する。

C  ここからDOループ11が始まる。
      DO 11 I=1,IMAX

C  C(I)にゼロを代入する。
        C(I) = 0.

C  ここからDOループ12が始まる。
        DO 12 J=1,JMAX

C  行列Aの(I,J)成分とベクトルBの第J成分の積を、
C  C(I)に加える。
          C(I) = C(I) + A(I,J) * B(J)
```

```
C  ここでDOループ12が終了する。
 12     CONTINUE

C  ここでDOループ11が終了する。
 11   CONTINUE

C  結果を書き出す。

C  ここからDOループ21が始まる。
      DO 21 I=1,IMAX

C  C(I)の値を標準出力に書き出す。
        WRITE(6,*) 'c[',I,']=',C(I)

C  ここでDOループ21が終了する。
 21   CONTINUE

      STOP
      END
```

(参考) C

```
#include <stdio.h>

int main(void)
{

/*  変数 imax、jmax を宣言する。  */
  int imax=3, jmax=2, i, j;

/*  配列 a、b、c を宣言する。
    配列 a は imax 行 jmax 列の行列に対応する。
    また配列 b は要素数 jmax のベクトル、
    配列 c は要素数 imax のベクトルに対応する。  */
  float a[3][2] = { 0., -1.,
                    1.,  0.,
                    0.,  0.};
  float b[2] = { 2., -1.};
  float c[3];
```

```
/*  行列 a とベクトル b の積を計算する。  */

/*  ここから for ループが始まる。  */
  for (i=0; i<=imax-1; i++)
  {

/*  c[i]にゼロを代入する。  */
    c[i] = 0.;

/*  ここから for ループが始まる。  */
    for (j=0; j<=jmax-1; j++)
    {

/*  行列 a の(i,j)成分とベクトル b の第 j 成分の積を、
    c[i]に加える。  */
      c[i] = c[i] + a[i][j] * b[j];

/*  ここで for ループが終了する。  */
    }

/*  ここで for ループが終了する。  */
  }

/*  結果を書き出す。  */

/*  ここから for ループが始まる。  */
  for (i=0; i<=imax-1; i++)
  {

/*  c[i]の値を標準出力に書き出す。  */
    printf( "c[%d] = %f¥n", i+1, c[i] );

/*  ここで for ループが終了する。  */
  }

  return 0;

}
```

実行例：

```
/home/snaoki> f77 prog03_4.f
/home/snaoki> ./a.out
  c[ 1]= 1.
  c[ 2]= 2.
  c[ 3]= 0.
```

課題：ベクトル（x, y, z）と回転角 θ （° ）を入力すると、ベクトル（x, y, z）を z 軸の周りに角度 θ だけ回転したベクトル（x', y', z'）を出力するプログラムを作成せよ（report03.f[.c]）。ただし、z 軸の周りの回転を表す行列Ａは、

$$A = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

である。FORTRAN では、正弦関数は「SIN(THETA)」、余弦関数は「COS(THETA)」のように書く。引数 THETA はラジアンである点に注意すること。実数（浮動小数点）の有効数字は 10 進数で 7 桁程度なので、円周率のような定数は 8 桁程度で表現するとよい。この課題では、行列Ａの各要素の値は回転角を入力するという処理の後でなければ確定できないので、DATA 文を使用したり、配列の宣言と同時に初期値を代入したりすることはできない。なお、C の場合、正弦関数は「sin(theta)」、余弦関数は「cos(theta)」である。C においては、math.h の include することと、コンパイル時に-lm オプションを指定することを忘れないこと。