４　条件分岐

ここでは条件によって異なった処理をするプログラムを作成してみます。はじめに、二次方程式を実数の範囲で解くプログラムを作成します。判別式がゼロまたは正であって、実数解が存在する場合は解を求めて出力しますが、判別式が負の場合は何も出力しないようにします。FORTRAN の場合、条件分岐は IF 文を用いて行ないます。

プログラム：

FORTRAN

```
C  入力を求めるメッセージを標準出力に書き出す。
      WRITE(6,*) 'a x^2 + b x + c = 0:'
      WRITE(6,*) 'a, b, c?'

C  READ文で値を読みこむ。
      READ(5,*) A,B,C

C  判別式(根号の中身)を計算する。
      D = B**2 - 4. * A * C

C  ここからIF文が始まる。
C  (...)の中に書かれた条件を満たす場合だけ
C  ENDIFまでの範囲に書かれた命令を実行する。
C  ここでは、判別式Dの値が0以上の場合に限って、
C  二次方程式の解を計算する。
C  IF文の条件の指定の中で、
C  '.EQ.'は等しい、'.NE.'は等しくない、
C  '.GT.'は大きい、'.LT.'は小さい、
C  '.GE.'は以上、'.LE.'は以下という意味である。
      IF (D.GE.0.) THEN

C  二次方程式の2つの解を計算する。
C  関数SQRTで平方根を求めている。
        X1 = (- B - SQRT(D)) / (2. * A)
        X2 = (- B + SQRT(D)) / (2. * A)

C  結果を標準出力に書き出す。
C  このWRITE文もIF文の中にあるので、
C  判別式Dが負で方程式に実数解が存在しない場合には、
```

```
C  このプログラムでは何も出力されない。
        WRITE(6,*) 'x = ',X1,',',X2

C  ここでIF文が終了する。
      ENDIF

      STOP
      END
```

(参考) C

```
#include <stdio.h>
#include <math.h>

/*  関数 sqrt を参照するために、math.h を include しているので、
    コンパイル時には-lm オプションを指定する必要がある。  */

int main(void)
{

/*  変数を宣言する。  */
  float a, b, c, d, x1, x2;

/*  入力を求めるメッセージを標準出力に書き出す。  */
  printf( "%s¥n", "a x^2 + b x + c = 0:" );
  printf( "%s¥n", "a, b, c?" );

/*  関数 scanf で値を読みこむ。  */
  scanf( "%f,%f,%f", &a, &b, &c );

/*  判別式(根号の中身)を計算する。  */
  d = b * b - 4. * a * c;

/*  ここから if 文が始まる。
    (...)の中に書かれた条件を満たす場合だけ
    {...}の中に書かれた命令を実行する。
    ここでは、判別式 D の値が 0 以上の場合に限って、
    二次方程式の解を計算する。
    if 文の条件の指定の中で、"=="は等しい、"!="は等しくない、
    ">"は大きい、"<"は小さい、
    ">="は以上、"<="は以下という意味である。  */
```

```
  if( d >= 0. )
  {

/*  二次方程式の 2 つの解を計算する。
    関数 sqrt で平方根を求めている。  */
    x1 = ( - b - sqrt( d ) ) / ( 2. * a );
    x2 = ( - b + sqrt( d ) ) / ( 2. * a );

/*  結果を標準出力に書き出す。
    この関数 printf も if 文の中にあるので、
    判別式 D が負で方程式に実数解が存在しない場合には、
    このプログラムでは何も出力されない。  */
    printf( "%s%f%s%f¥n", "x =  ", x1, ", ", x2 );

/*  ここで if 文が終了する。  */
  }

  return 0;
}
```

実行例：

```
/home/snaoki> f77 prog04_1.f
/home/snaoki> ./a.out
 a x^2 + b x + c = 0:
 a, b, c?
1.,3.,2.
 x =  -2.,  -1.
/home/snaoki> ./a.out
 a x^2 + b x + c = 0:
 a, b, c?
1.,2.,5.
```

以上のプログラムでは、実数解が存在する場合のみ計算を行ないました。次に、複素数解になる場合にも計算を行ない結果を出力するプログラムを作成してみます。上のプログラムでは、条件が真である場合だけ処理を実行しましたが、条件が真の場合にはある処理を行ない、偽の場合には別の処理を行なうこともできます。FORTRAN の場合、「IF (～) THEN ～ ELSE ～ ENDIF」という形で書きます。

プログラム：

FORTRAN

```
C  入力を求めるメッセージを標準出力に書き出す。
```

```
      WRITE(6,*) 'a x^2 + b x + c = 0:'
      WRITE(6,*) 'a, b, c?'

C  READ文で値を読みこむ。
      READ(5,*) A,B,C

C  判別式(根号の中身)を計算する。
      D = B**2 - 4. * A * C

C  ここからIF文が始まる。
C  ここでは、'IF (...) THEN ... ELSE ... ENDIF'という構造になっている。
C  (...)の中に書かれた条件を満たす場合には
C  THENとELSEの間に書かれた命令を実行し、
C  そうでない場合にはELSEとENDIFの間に書かれた命令を実行する。
C  ここでは、判別式Dの値が0以上の場合には実数解を、
C  負の場合には複素数解を求める。
      IF (D.GE.0.) THEN

C  判別式が0以上の場合には、
C  二次方程式の2つの実数解を計算する。
        X1 = (- B - SQRT(D)) / (2. * A)
        X2 = (- B + SQRT(D)) / (2. * A)

C  結果を標準出力に書き出す。
C  このWRITE文も、
C  判別式Dが0以上で実数解が存在する場合だけ実行される。
        WRITE(6,*) 'x = ',X1,',',X2

C  IF文で指定した条件が真である場合には、
C  ELSEより前に書かれた命令が実行される。
C  偽である場合には、
C  ELSEより後ろに書かれた命令が実行される。
      ELSE

C  判別式が負の場合には、
C  二次方程式の2つの複素数解を計算する。
C  XRが実部、XIが虚部である。
        XR = - B / (2. * A)
        XI = SQRT(-D) / (2. * A)
```

```
C  結果を標準出力に書き出す。
C  このWRITE文も、
C  判別式Dが負で複素数解になる場合だけ実行される。
        WRITE(6,*) 'x = ',XR,' +-',XI,' i'

C  ここでIF文が終了する。
      ENDIF

      STOP
      END
```

（参考）C

```
#include <stdio.h>
#include <math.h>

/*  関数 sqrt を参照するために、math.h を include しているので、
    コンパイル時には-lm オプションを指定する必要がある。  */

int main(void)
{

/*  変数を宣言する。  */
  float a, b, c, d, x1, x2, xr, xi;

/*  入力を求めるメッセージを標準出力に書き出す。  */
  printf( "%s¥n", "a x^2 + b x + c = 0:" );
  printf( "%s¥n", "a, b, c?" );

/*  関数 scanf で値を読みこむ。  */
  scanf( "%f,%f,%f", &a, &b, &c );

/*  判別式(根号の中身)を計算する。  */
  d = b * b - 4. * a * c;

/*  ここから if 文が始まる。
    ここでは、"if (...) {...} else {...}"という構造になっている。
    (...)の中に書かれた条件を満たす場合には
    1 番目の{...}の中に書かれた命令を実行し、
    そうでない場合には 2 番目の{...}の中に書かれた命令を実行する。
    ここでは、判別式 D の値が 0 以上の場合には実数解を、
```

```
    負の場合には複素数解を求める。  */
  if( d >= 0. )
  {

/*  判別式が 0 以上の場合には、
    二次方程式の 2 つの実数解を計算する。  */
    x1 = ( - b - sqrt( d ) ) / ( 2. * a );
    x2 = ( - b + sqrt( d ) ) / ( 2. * a );

/*  結果を標準出力に書き出す。
    この関数 printf も、
    判別式 D が 0 以上で実数解が存在する場合だけ実行される。  */
    printf( "%s%f%s%f¥n", "x =  ", x1, ", ", x2 );

/*  if 文で指定した条件が真である場合には、
    1 番目の{...}の中に書かれた命令が実行される。
    偽である場合には、
    2 番目の{...}の中に書かれた命令が実行される。  */
  }else{

/*  判別式が負の場合には、
    二次方程式の 2 つの複素数解を計算する。
    xr が実部、xi が虚部である。  */
    xr = - b / ( 2. * a );
    xi = sqrt( - d ) / ( 2. * a );

/*  結果を標準出力に書き出す。
    この関数 printf も、
    判別式 D が負で複素数解になる場合だけ実行される。  */
    printf( "%s%f%s%f%s¥n", "x =  ", xr, " +- ", xi , " i" );

/*  ここで if 文が終了する。  */
  }

  return 0;
}
```

実行例：

```
/home/snaoki> f77 prog04_2.f
/home/snaoki> ./a.out
```

```
 a x^2 + b x + c = 0:
 a, b, c?
1.,3.,2.
 x =  -2.,  -1.
/home/snaoki> ./a.out
 a x^2 + b x + c = 0:
 a, b, c?
1.,2.,5.
 x =  -1. +-  2. i
```

ある条件を満たすまで処理を反復させることもできます。ここでは級数の計算を例にして説明します。

たとえば、指数関数を級数で表してみます。指数関数 exp(x)は、マクローリン展開（ｘ＝０のまわりでのテイラー展開）によって、

$$\exp x = 1 + x + \frac{1}{2}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \cdots$$

と書けます。これを指数関数の定義とみなすこともできます。以下では、このように級数によって定義された指数関数の値を、各項の和を計算することによって求めてみます。

無限個の要素の和が指数関数の真の値に収束するので、一定の精度で関数の値を求めるために、各要素の絶対値が一定値より小さくなるまで和を計算していきます。FORTRAN の場合、CONTINUE 文と GO TO 文を利用します。ループを終了するための条件を満たすまで、CONTINUE 文と GO TO 文で挟まれたループの中の処理を反復します。条件を満たしたときだけ、GO TO 文でループの外に出て、反復処理を終了します。このような処理を行なうときは無限ループを作らないように注意する必要があります。

プログラム：

FORTRAN

```
C  入力を求めるメッセージを標準出力に書き出す。
      WRITE(6,*) 'y = exp(x):'
      WRITE(6,*) 'x?'

C  READ文で入力値を読みこむ。
      READ(5,*) X

C  入力値Xの絶対値が10より大きい場合は、
C  正しく計算できない可能性があるので何もしないで終了する。
C  絶対値は関数ABSによって求められる。
      IF (ABS(X).GT.10.) STOP

C  指数関数の値を、テイラー級数として計算する。
```

```
C  求める級数をYとする。
C  はじめにYにゼロを代入する。
      Y = 0.

C  指数をNとする。
C  指数がゼロの項から計算するので、Nにゼロを代入する。
      N = 0

C  ここからループが始まる。
C  ループの中では各項の値を計算し、変数Yに加えていく。
 11   CONTINUE

C  Nの階乗の値を求める。
C  オーバーフローを防ぐため、
C  整数ではなく実数で計算する。

C  はじめに変数Cを1.0にする。
      C = 1.

C  変数Cに1,2,...,Nを順にかけていく。
      DO 21 M=1,N
        C = C * REAL(M)
 21   CONTINUE

C  各項の値を求める。
C  XのN乗をNの階乗で割った値を求め、変数Y1に代入する。
      Y1 = X**N / C

C  変数Y1の値の絶対値が10^-6以下になったら、
C  行番号19に飛んで計算を終了する。
      IF (ABS(Y1).LE.1.0E-6) GO TO 19

C  変数Yに変数Y1の値を加える。
      Y = Y + Y1

C  指数Nの値をひとつ進める。
      N = N + 1

C  行番号11に戻って処理を反復する。
```

```
C  行番号11とこの行の間の処理を、
C  終了する条件を満たすまで反復することになる。
      GO TO 11

C  終了する条件を満たすと、この行に飛んでくる。
 19   CONTINUE

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

      STOP
      END
```

（参考）C

```
#include<stdio.h>
#include<math.h>

int main(void)
{

/*  変数を宣言する。  */
  int n, m;
  float x, y, y1, c;

/*  入力を求めるメッセージを標準出力に書き出す。  */
  printf( "%s¥n", "y = exp(x):" );
  printf( "%s¥n", "x?" );

/*  scanf 文で入力値を読みこむ。  */
  scanf( "%f", &x );

/*  入力値 x の絶対値が 10 より大きい場合は、
    正しく計算できない可能性があるので何もしないで終了する。
    絶対値は関数 fabs によって求められる。  */
  if (fabs(x) > 10.){return 0;}

/*  指数関数の値を、テイラー級数として計算する。  */

/*  求める級数を y とする。  */
/*  はじめに y にゼロを代入する。  */
```

```
  y = 0.;

/*  指数を n とする。
    指数がゼロの項から計算するので、n にゼロを代入する。  */
  n = 0;

/*  ここからループが始まる。
    ループの中では各項の値を計算し、変数 y に加えていく。
    while 文の条件を(1)と書き、常に真になるようにして、ループを作る。  */
  while( 1 ){

/*  n の階乗の値を求める。
    オーバーフローを防ぐため、
    整数ではなく浮動小数点で計算する。  */

/*  はじめに変数 c を 1.0 にする。  */
    c = 1.;

/*  変数 c に 1,2,...,n を順にかけていく。  */
    for (m=1; m<=n; m++){
      c = c * m;
    }

/*  各項の値を求める。
    x の n 乗を n の階乗で割った値を求め、変数 y1 に代入する。  */
    y1 = pow( x, n ) / c;

/*  変数 y1 の値の絶対値が 10^-6 以下になったら、
    while 文から抜けて計算を終了する。  */
    if (fabs( y1 ) < 1.0e-6){break;}

/*  変数 y に変数 y1 の値を加える。  */
    y = y + y1;

/*  指数 N の値をひとつ進める。  */
    n++;

/*  while とこの行の間の処理を、
    終了する条件を満たすまで反復することになる。  */
  }
```

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

  return 0;
}
```

実行例：

```
/home/snaoki> f77 prog04_3.f
/home/snaoki> ./a.out
 y = exp(x):
 x?
5.
 y =    148.41318
```

IF 文では、AND や OR を用いて、２つ以上の条件を組み合わせて使うことができます。たとえば、N が 1 以上 12 以下のときにある処理を実行したい場合には、「IF ((N.GE.1).AND.(N.LE.12)) THEN ～ ENDIF」とします。また、N が 5 以下または 22 以上のときにある処理を実行したい場合には、「IF ((N.LE.5).OR.(N.GE.22)) THEN ～ ENDIF」とします。C では、「if (n >= 1 && n<= 12) { ～ }」、「if (n<=5 || n>=22) { ～ }」とします。

課題：ユークリッドの互除法を用いて、２つの自然数の最大公約数を計算するプログラムを作成せよ（report04.f[.c]）。FORTRAN では i÷j の余りは、「MOD(I,J)」で計算できる。C では「i % j」のようにする。

（参考）**ユークリッドの互除法**

ユークリッドの互除法は、２つの自然数の最大公約数を計算するためのアルゴリズムである。まず、２つの自然数のうち、大きいほうを小さいほうで割り、その余りを求める。余りが０であれば、割った数（小さいほうの自然数）が最大公約数である。余りが０でない場合は、ひとつ前の計算で割る数であった数を割られる数、余りであった数を割る数として、次の割り算を行う。割り算の結果、余りが０であれば、そのときの割る数が最大公約数である。余りが０になるまで同様の処理を反復すれば、任意の２つの自然数の最大公約数を求めることができる。