6 プログラミングの例

プログラミングは「習うより慣れろ」であるので、具体的例題を示す事にする。各人、実際にプログラミングを行い、「プログラムを作る」という一連の作業に早く習熟して欲しい。

6.1 例題

ファイルに次のようなデータが各行に一つずつ入っている時、これを読み込んでその平均値とそれぞれのデータの平均値からのズレ (偏差) を計算して出力するプログラムを作成せよ。

0.735	0.886	0.781	0.927	0.670	0.579	0.511	0.277	0.982	0.165
0.950	0.305	0.483	0.620	0.352	0.074	0.302	0.569	0.751	0.554
0.546	0.265	0.385	0.755	0.878	0.289	0.240	0.105	0.821	0.931
0.333	0.826	0.147	0.727	0.800	0.505	0.202	0.119	0.408	0.581
0.494	0.708	0.561	0.980	0.399	0.064	0.418	0.598	0.329	0.570

データは、

```
/home/teacher/z6wt01in/SAMPLE/weight.dat
```

に、サンプル・プログラムは同じディレクトリに

```
mean.f file_open.f read_data.f calc_mean.f data_output.f
```

として置いてある。

6.2 メインプログラム

細部の構造は後程決定することにして、まずメインプログラムを書き全体の流れを決める。メインプログラム calc_mean_value(ファイル名は mean.f) は以下のように、幾つかのサブルーチンを呼び出して作業を進める。

```
      program calc_mean_value
      implicit none
c constants:
      integer MAX_DATA                    ! 読み込むデータ数の最大値
      parameter(MAX_DATA=100)             ! 100 に定義
      integer ERR                         ! ファイル操作の状態を表す
      parameter(ERR=-1)                   ! 問題有り:-1(無し:0)
      integer LUNIN                       ! 入力ファイルの論理機番
      parameter(LUNIN=11)                 ! 11 を定義
c function:                               ! 組み込み関数以外は宣言する
      real calc_mean                      ! 平均値を計算するユーザー関数
c local variables:
      real    data_array(max_data)        ! データを格納する配列
      integer status                      ! ファイル操作の状態
```

```
      integer num_data/0/                    ! データの数
      real    mean/0.0/                      ! 平均値
c begin:
 1    continue
      call file_open('input',LUNIN,status)
      if (status.eq.ERR) goto 1
      call read_data(LUNIN,MAX_DATA,num_data,data_array)
      if (num_data.gt.0) then
         mean = calc_mean(num_data,data_array)
         call data_output(num_data,data_array,mean)
      else
         write(*,*) 'data empty'
      endif
      stop
      end
```

calc_mean_value はまず所定のファイルを開き (file_open)、データを読み込み (read_data)、平均値を計算 (calc_mean) した後、結果を出力する (data_output)。組み込み関数以外は型宣言が必要であるので、ユーザーが用意する関数 calc_mean(real 型) は宣言が必要である。

num_data, mean は変数宣言の際に初期値を定める。これは、一般的にプログラムを実行する際に、それぞれの変数が 0.0 に初期化されているとは限らないためである[1]。配列の初期化は

```
real array(10)/0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0/
```

のようにすれば良い。全ての要素を同じ値で初期化する場合は

```
real array(10)/10*0.0/
```

のようにすれば良い。

6.3 サブルーチン・関数

サブルーチン file_open

file_open は開くファイルの名前をユーザーに尋ね、所定の手続きのもとに開く。FORTRAN では、ファイルに対する入出力は対応する**論理機番**[2](LUN:Logical Unit Number) を指定して行われる。ファイルと論理機番を結びつける作業は open 文で行われる。

ここでは今後の利便性を考慮し、file_open は入出力共に扱える構造にする。mode という引数を用意し、mode の値により動作 (入力か出力か) を指定する。プログラムを読み易くするため、mode は文字変数とし、input または output という文字列を渡しそれに応じたファイルの開き方をする。以下に file_open の例を示す。内容を理解する必要はなく、機能を理解して欲しい。

[1] コンパイラのオプションで、実行時にそれぞれの変数を 0.0 に初期化するように指定することが出来る場合もある。

[2] 標準入力であるキーボードには 5, 標準出力の画面には 6 が割り当てられているが、これまで見て来たように*で代用できる。ファイルには 5/6 の論理機番を指定しない方がよい。

```
      subroutine file_open(mode,lun,status)
      implicit none
c constants:
      integer ERR,NORMAL                  ! ファイル操作の状態を表す
      parameter(ERR=-1,NORMAL=0)          ! 問題無し:0、有り:-1
c inputs:
      character*(*) mode                  ! input または output
      integer lun                         ! 入出力ファイルの論理機番
c output:
      integer status                      ! ファイル操作の状態
c local variable:
      character*132 filename              ! ファイルの名前
c begin:
      status = NORMAL
      write(*,'(a,$)') 'file name : '
      read(*,'(a)') filename
      if ((mode.eq.'input').or.(mode.eq.'INPUT')) then
         open(unit=lun,file=filename,form='formatted',
     &        status='old',err=998)       ! 既存のファイルを開く
      else if ((mode.eq.'output').or.(mode.eq.'OUTPUT')) then
         open(unit=lun,file=filename,form='formatted',
     &        status='new',err=998)       ! 新規にファイルを作成
      else
         write(*,*) 'Illegal mode option detected ; ',mode
         status = ERR
      endif
      return
c error handling
 998  write(*,*) 'Can not open file : ',filename,' as ',mode
      status=ERR
      return
      end
```

今後、file_open をいろいろなプログラムから呼ぶときのために、file_open.f として保存しておけば良い。プログラムは

/home/teacher/z6wt01in/SAMPLE/file_open.f

に置いてある。

サブルーチン read_data

データを読み込むためのプログラムは以下のように書ける。

```
      subroutine read_data(lunin,size,num_data,array)
```

```
      implicit none
c constants:
      logical FOREVER                        ! 無限ループ生成用
      parameter(FOREVER=.true.)              ! 真を定義
c inputs:
      integer lunin                          ! 入力ファイルの論理機番
      integer size                           ! 最大データ数
c outputs:
      real    array(size)                    ! データを格納する配列
      integer num_data                       ! データの数
c local variable:
      real    value                          ! データを一時保存
c begin:
      num_data = 0                           ! データ数を初期化
      do while (FOREVER)
         read(lunin,*,end=999) value         ! ファイルの終り→ 999 へ
         num_data = num_data + 1
         array(num_data) = value
         if (num_data.gt.size) then
            write(*,*) 'Number of data exceeds limit: ',size
            goto 999
         end if
      end do
 999  continue
      return
      end
```

配列の受け方として**配列整合**のやり方を採用している。配列の大きさが動的に変化する (実行時に初めて定められる) ことを明示する事ができ、プログラムが分かり易くなる。サブルーチンの中では、配列の添字の値 (num_data) が上位で宣言された配列の大きさ (size) を越えてしまわないようにしている。このような心掛けは、安全なプログラムを作る上で有効である。

プログラムの中では、無限回まわるループが設定されており、データの読み込みの途中でファイルの終り (EOF:End Of File) を見付けたところでループから抜ける。そのために、read 文では end=という構文を用いている。この構文により、EOF に出会うと指定された文 (ここでは 999、continue は自分自身では何もしない) へ飛ぶ。

goto 文は乱用すると大変読み難いものになるので注意が必要である。ここでの例の様に例外処理の場合に限るように心がけること[a]。

[a] C 言語での break 文は一つ外側に抜けるだけなので、エラー処理が繁雑な場合などは非常に使い難い。

サブルーチン calc_mean

平均値を計算する部分を FORTRAN の function 文を使って書くと、例えば以下のように書ける。

```
      real function calc_mean(num_data,array)
      implicit none
c inputs:
      integer num_data                    ! データの数
      real    array(num_data)             ! データが入った配列
c local variable:
      integer i                           ! ループ変数
      real    sum                         ! データの総和
c begin:
      sum = 0.0
      do i=1,num_data
         sum = sum + array(i)             ! データの総和を計算
      end do
      calc_mean = sum / float(num_data)   ! 総和をデータ数で除算
      return
      end
```

サブルーチン data_output

計算結果を出力するプログラムは、例えば以下のように書ける。

```
      subroutine data_output(num_data,array,mean)
      implicit none
c inputs:
      integer num_data                    ! データの数
      real    array(num_data)             ! データが入った配列
      real    mean                        ! 平均値
c local variable:
      integer i                           ! ループ変数
c begin:
      do i=1,num_data
         write(*,'(I3,A1,2F9.4)')
     &        i,':',array(i),array(i)-mean
      end do
      return
      end
```

6.4 実行例

例の場合、プログラムのコンパイルは

```
frt -o mean mean.f file_open.f read_data.f calc_mean.f data_output.f
```

とすることにより実行ファイル mean が出来、実行すると、

```
% mean ↵
file name :  weight.dat ↵
1:  0.7350 0.2059
2:  0.8860 0.3569
3:  0.7810 0.2519
....
```

となる。

練習

本章で解説したプログラムを改良し、平均値と標準偏差を出力するプログラムを作成せよ。また、それぞれのデータの偏差値 (平均が 50、標準偏差が 10 になるように変換した値) を出力するプログラムを作成せよ。