
Lecture Note for Basic Data Analysis using Fortran90/95

Hisashi SATO (sato.hisashi@nagoya-u.jp)

Scope of this training class (このコースの目的)

Acquire minimum literacy of Fortran90/95 for basic data analysis. Students who want to study further, following books and websites would be useful.

- (1) 数値計算のための Fortran90/95 プログラミング入門, 牛島省(著), 森北出版株式会社
- (2) Fortran90 入門—基礎から再帰手続きまで, 新井親夫(著), 森北出版株式会社
- (2) <http://ace.phys.h.kyoto-u.ac.jp/~tomita/education/fortran90/sec0.html>
- (3) <http://www.cs.mtu.edu/~shene/COURSES/cs201/NOTES/fortran.html>

About Fortran (Fortran の簡単な紹介)

Fortran is the one of the oldest computer language; It was developed by John Bachus and his team in 1953. It passed through a sequence of evolution.

Fortran 66	The original standard
Fortran 77	Adds number of significant features
Fortran 90	Modernized with major revision
Fortran 95	Very minor revision
Fortran 2003	Adds object-oriented support and interoperability with C

Efforts are still underway for revisions. However, revisions after Fortran 90 have no points for most users.

Even after more than half century from the invention, Fortran is the primary language for some of the most intensive supercomputing tasks, such as climate modeling, computational fluid dynamics, and integrated earth-system modeling.

Preparation for training environment (実行環境の構築)

In this training course, we employ the G95, a free Fortran compiler. To get its install package, follow the following link on the website <http://www.g95.org/>: Top page→Downloads→Binaries→Information for Windows users→<http://ftp.g95.org/g95-MinGW.exe>. This install package will ask you to locate the install folder. Answer as you like (ex: C: ¥g95). For other questions, just reply “yes” or “ok”, basically.

この実習ではフリーの Fortran コンパイラである G95 を使用する。サイト <http://www.g95.org/>の トップページ→Downloads→Binaries→Information for Windows users→項目 "g95-MinGW.exe"内の <http://ftp.g95.org/g95-MinGW.exe> をクリック、インストーラーをダウンロードし(事前配布したライブラリ中にも入っています)、実行。インストール先のフォルダの問いについては、適当に邪魔にならないところを指定 ("C: ¥g95"とか)。その他の問いには、基本的に全て Yes と Ok でインストールを進める。

On somewhere you like, create a new folder whose name is “fortran”. Within this folder, create a shortcut of command prompt. Right click on the shortcut→Select “property” →”Shortcut” tab→”Working folder”. Delete all letters within this entry. By this modification, folder “fortran” becomes the working folder for command prompt that was launched by this short cut.

続いて適当な場所に、fortran という名前のフォルダを作成。このフォルダ内に、コマンドプロンプトのショートカットを作成。このショートカットを右クリックし、プロパティを選択、「ショートカット」タブ内の「作業フォルダー」を空欄にする。これで、このショートカットから起動されるコマンドプロンプトの作業フォルダはフォルダ fortran になる。

You can find more detailed information in the following web site (written in Japanese) for installing G95. <http://d.hatena.ne.jp/arakik10/20090213/p1>

A simple program (簡単なサンプルコード)

Create a new text file in the working folder. Rename the file to "hello.f90". Using a text editor, input following code in the file, and close it.

```
Program sample1
  write(*,*) "Hello"
End Program
```

Fortran compilers regard letters within double quotation as characters, not a variable

To compile this source code into an execution file, type "g95 hello.f90" in the command prompt. After a short time, you can find a new file whose name is "a.exe" in the working folder. Input "a.exe" in the command prompt, then it will write "Hello" on the screen.

Note1

"a.exe" is the default name for an executable file of G95. The default name depends on compiler. You can specify the name of execution file by adding a compile option as follows.

```
g95 hello.f90 -o hello.exe
```

In this example, file name of execution file will be "hello.exe".

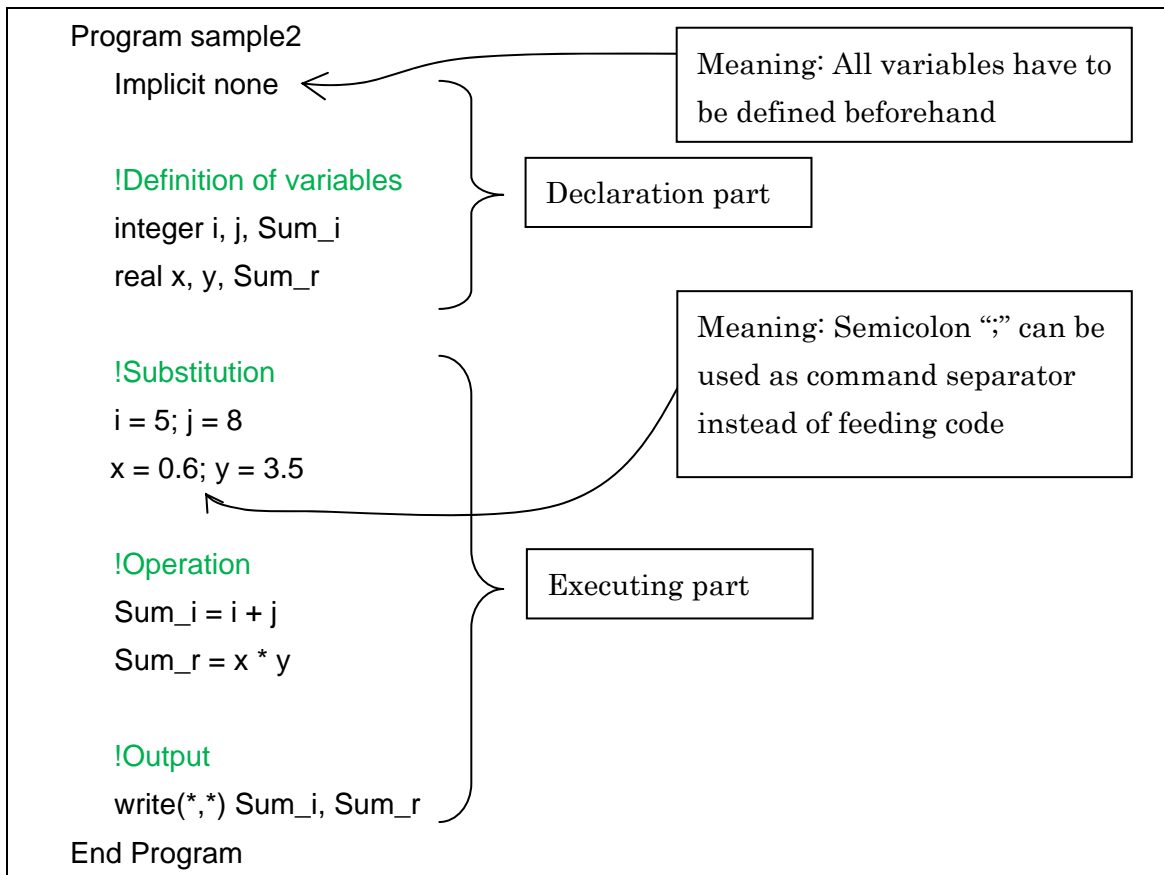
Note2

Here's are tips on the Windows command prompt and the Unix command line.

Up/down arrow keys	: Recall commands beginning with the most recent one
Tab key	: Suggests names for files and folders
exit	: Terminate command prompt

Definition of variables, four arithmetic operations (変数定義、四則演算)

In the following sample code, we will learn (1) how to define variables, and (2) how to substitute numbers to variables. This program code is composed of declaration part and executing part. Declaration part has to be preceded to execution part. Lines those start with "!" are comments. As Fortran compilers ignore all characters between "!" and line feed code, you can omit to input these lines.



Here, variables i, j, and k were defined as integer numbers (整数), while variables x, y, and z were defined as real number of single precision (単精度の整数). If you substitute a real number into an integer variable and/or substitute a integer number into a real variable (like followings), G95 automatically converts type of number appropriately.

```

Sum_i = x * y
Sum_r = i + j

```

However, if you intend to do so, you should employ authorized expressions as follows.

```

Sum_i = int (x * y)
Sum_r = real(i + j)

```

In this case, x=0.6, y=3.5, and Sum_i =4

In this case, i=5, j=8, and Sum_r =13.0

Here, function “int” converts a real number into integer number, and function “real” converts an integer number into real number of single precision.

Note 1: Other operation characters

-, subtraction; /, division; **, multiplication

Note 2: Rules for variable name

Variable names have to start with an alphabetical character. Fortran does not distinguish capital letters and lower-case letters.

Note 3: Frequently used functions

ABS (x)	absolute value	絶対値
INT (x)	truncate	切捨て
MOD (x,y)	overmeasure	剰余
MAX (x,y,z,...)	maximum value	最大値
MIN (x,y,z,...)	minimum value	最小値
SQRT (x)	square root	平方根
EXP (x)	exponential	指数
LOG (x)	natural log	自然対数
LOG10 (x)	common log	常用対数
SIN (x)	sine	サイン
COS (x)	cosine	コサイン

Evaluation order of expressions, and how to change it with parenthesis (演算子の優先順位)

Operators within parenthesis have the highest priority for evaluation, while + and – have the lowest priority. For nested parenthesis, the deepest parenthesis has the highest priority. For operators at same priority, evaluation is conducted from the leftmost one to the right direction.

```
Program sample3
  Implicit none
  real    x, y, z
  x=0.2; y=1.5; z=2.0
  write(*,*) x + y / z    ! result will be 0.95
  write(*,*) (x + y) / z  ! result will be 0.85
End Program
```

Drill 1 (演習問題 1)

Make a program that displays perimeter (周辺長のこと) and area of a triangle whose sides are 2.9, 3.1, and 4.1cm. For calculating area of a triangle, Heron's formula would be useful.

$$S = \sqrt{s \times (s - a) \times (s - b) \times (s - c)}$$

where S is the area. a , b , and c are side lengths. s is the one-half of perimeter.

Conditional sentence (条件文)

Following example shows how to use a conditional sentence. Whether logical formula in parenthesis is true or false controls which section in the construction is evaluated.

Program sample4

Implicit none

real x

x = 1.41421356

if (x>1.0) then

write(*,*) "Variable x is larger than 1.0"

else

write(*,*) "Variable x is smaller than 1.0"

endif

if (x>2.0) then

write(*,*) "Variable x is larger than 2.0"

elseif (x>2.0) then

write(*,*) "Variable x is smaller than 2.0, but larger than 1.5"

else

write(*,*) "Variable x is smaller than 1.5"

endif

End Program

If logical formula
x>1.0 is true, this line
will be executed

In other cases, this
line will be executed

Following are relational operators (関係演算子) of Fortran90

<code>e1 < e2</code>	True when $e1 < e2$
<code>e1 > e2</code>	True when $e1 > e2$
<code>e1 == e2</code>	True when $e1$ is equal to $e2$
<code>e1 /= e2</code>	True when $e1$ is NOT equal to $e2$
<code>e1 <= e2</code>	True when $e1 \leq e2$
<code>e1 >= e2</code>	True when $e1 \geq e2$

Following are logic operators (論理演算子) of Fortran90

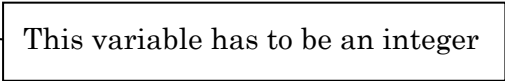
<code>f1 .and. f2</code>	True when both formulas $f1$ and $f2$ are true
<code>f1 .or. f2</code>	True when $f1$ or $f2$ is true

Iteration statement (繰り返し文)

In the following sample code, lines within Do~EndDo construction are repeated. The integer i is initialized to 1 at the beginning of in the first repeat, and automatically increases by 1 for at the beginning of each repeat. When the integer i reaches to 10, this loop ends.

```
Program sample5
  Implicit none
  integer i
  real    x

  x = 0.0
  Do i=1, 10
    x = x + 0.1
    write(*,*) i, x
  EndDo
  write(*,*) "Done!"
End Program
```



In the next example, lines within “Do while () ~ EndDo” construction are repeated while condition in the parenthesis is fulfilled.

```

Program sample6
  Implicit none
  integer i

  i = 1
  Do while (i<10)
    write(*,*) i
    i = i + 1
  EndDo
  write(*,*) "Done!"
End Program

```

Arrays (配列変数)

Array deals linear sequence of elements stored consecutively in memory. Each element is specified with integer number in the parenthesis. A character ":" in the parenthesis stands for all elements in the dimension.

```

Program sample7
  Implicit none
  integer i, j
  integer array1 (5) !One-dimensional array (corresponds to a vector)
  real array2 (3,3) !Two-dimensional array (corresponds to a matrix)

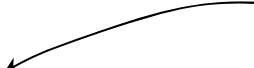
  Do i=1, 5
    array1(i) = i*10
  EndDo
  write(*,*) array1(:)

  Do i=1,3
    Do j=1,3
      array2(i,j) = real(i) / real(j)
    EndDo
  EndDo

  Do i=1,3

```

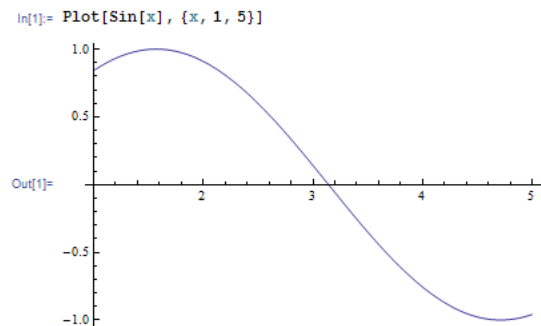
Here, colon ":" means all elements in that dimension




```
write(*,*) array2(i,:)
EndDo
End Program
```

Drill 2 (演習問題 2)

x is a real number between 1.0 and 5.0.
Find x that satisfies $\sin(x) = 0.0$ using the bisection method (二分法). Estimation error must be less than 0.0001. Right figure is the sine curve (Range of independent variable is 1.0 to 5.0).



Output to file (ファイルへのデータ書き出し)

This sample code creates a new file “matrix_out.txt” in the working folder, and writes elements of array *matrix* on the file and monitor. Here, number “10” is a device number. Device numbers have to be integer numbers those are larger than 0.

```
Program sample7
Implicit none
integer matrix(5,10)
integer i, j

Do i=1,5
  Do j=1,10
    matrix(i,j) = i + j
  EndDo
EndDo

Open(10,file='matrix_out.txt', status="NEW")
Do i=1,5
  write(10,*) matrix(i,:) !Write to device number 10
  write(*,*) matrix(i,:) !Write to standard device (monitor)
EndDo
Close(10)
```

End Program

Input from file (ファイルからのデータ読み込み)

This sample code reads elements of a array from the file "matrix_out.txt", which was created by the previous sample code.

```
Program sample8
  Implicit none
  integer matrix(5,10)
  integer i

  Open(10,file='matrix_out.txt', status="OLD")
  Do i=1,5
    read(10,*) matrix(i,:) !Input from device number 10
  EndDo
  Close(10)

  Do i=1,5
    write(*,*) matrix(i,:)
  EndDo
End Program
```

Drill 3 (演習問題 3)

A file "**dat_Population.txt**" contains time series of world population from 1750 to 2150 for each of 7 world areas; Each line contains populations for World, Africa, Asia, Europe, South Africa & Caribbean, North America, and Oceania. By adding row 2-7 of this data file, make a data file "**dat_PopulationTot.txt**" that contains time series of TOTAL world population.