# 1. Fortran を使ってみる

## 1.1. Fortran とは

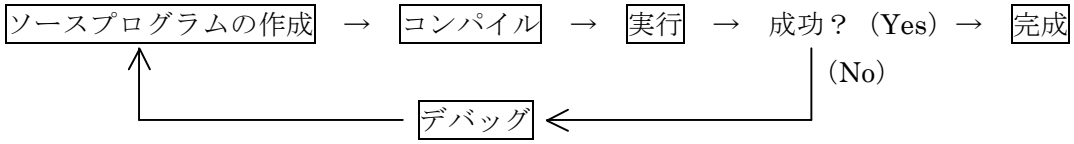
Fortran は科学技術計算向けに開発された言語です. スーパーコンピュータ用のプログラムはほとんど Fortran で記述されます. 最新の規格は Fortran2008 です. 実習で使用するのは gfortran です. Fortran に限らず, C や C++などの言語は, Cygwin をインストールする事で利用できます. Cygwin は Windows 上で Unix 環境を構築するフリーウェアです.

http://www.cygwin.com/

から入手できます. プログラミングに興味がある人は入れてみて下さい.

## 1.2. プログラムの作成手順

一般にプログラムを作成する手順は以下のようになります.




- **ソースプログラムの作成**
  まず, **コンピュータにやって欲しいことを人間にわかりやすい言語で書きます**. その命令の指示書がソースプログラムです. またソースプログラムを記述するための言語がプログラミング言語です.
- **コンパイル**
  次に, ソースプログラムを**コンピュータが理解できる言語に翻訳**します. これがコンパイルという作業です.
- **実行**
  最後にプログラムが正しくできたかどうか, 確かめるために実行して見ます.
- **デバッグ**
  期待通りに動かない場合は, プログラムを修正します. この作業をデバッグと呼びます. 「バグ」は「虫」という意味で, プログラムが思ったように動かないのは「虫」がいるせいで, その「虫」をとる(デバッグ)という意味です.

## 1.3. ソースプログラムの作成


それでは, 実際に簡単なプログラムを作ってみましょう. まずソースプログラムを作成します. プログラムは通常**テキスト・エディタ**と呼ばれるもので作ります. ここでは秀丸を使うことにします. 右図のアイコンをクリックすると秀丸の画面が開きます.

**秀丸を用いて**, 以下のプログラムを入力してください. **「！」以外で始まる各行は先頭に 6 文字程度スペース**を入れてください.

```
      program main
      Implicit none
!     This is a sample program
      write(6,*) 'Hello World!'
!
      stop
      end program main
```

プログラムを入力し終えたら，適当な名前(ここでは「hello.f90」とします)をつけて保存します．保存はメニューから「ファイル」→「名前をつけて保存 ...」とすると，ファイルを保存するダイアログが開くので，**「保存する場所」を必ず自分のホームディレクトリである「マイ ドキュメント」もしくは「'nafs¥u¥stu¥10'の S******(Y:)」としてください．**

### 1.4. ソースプログラムのコンパイル

次に，上で作成したプログラムをコンパイルします．この作業は，「Cygwin」のターミナルで行います．[スタート]→[全てのプログラム]→[Cygwin]→[Cygwin Bash Shell]とクリックしてください．そうするとターミナルが開きます．

**まず，自分のホームディレクトリに移動します．そのためにターミナルで，**

**$ cd /cygdrive/c**

**と打ってください($はプロンプトで実際に打つのは，ls だけです)．この作業は Cygwin Bash Shell を起動するたびに毎回必要です．**

次に，本当にソースプログラムができたかどうか確かめます．ターミナルで，

$ ls

と打ってみて下さい．ファイルのリストが表示されます．その中に「hello.f90」があれば，きちんと保存されたことになります．

**コンパイルするためのコマンドは，**

| gfortran -o 実行ファイル名　ソースプログラム名 |
|---|

です．「実行ファイル名」は後で実際に実行するファイルの名前です．上の例では，

$ gfortran -o hello hello.f90

となります．エラーが出ずに終了すれば「hello.exe」という実効ファイルが作成されます．

### 1.5. プログラムの実行

次に，プログラムを実行してみます．そのためにはターミナルで，

| ./実行ファイル名 |
|---|

とします．上の例では，

$ ./hello

です．これまでのすべての作業が成功していれば，
Hello World!
と出力されます．

## 2. Fortran の文法

### 2.1. Fortran の書式(自由フォーマット)

ここでは，Fortran90 の書式を説明します．Fortran77 とは多少異なります．
Fortran90 のプログラムの基本形は，

```
program プログラム名
implicit none
型宣言文
プログラム本体
stop
end program プログラム名
```

です．Hello.f90 の例では，「main」がプログラム名です．型宣言文はありません．プログラム本体は，「write(6,*) 'Hello World!'」だけです．
**「!」記号よりあとは，コメント文(非実行文)でコンピュータには無視されます**
(‘ ’記号の中は文書なので別)．

### 2.2. データの型と型宣言文

#### 2.2.1. データの型

データの型には以下のものがある．
　整数型，実数型，倍精度実数型，複素数型，倍精度複素数型，論理型，文字型
人は１つの「x」という変数を，臨機応変に「整数」だと思ったり，「複素数」だと思ったりできるがコンピュータにはそれができない．そのためあらかじめ「x」は「整数」なのか「実数」なのか「複素数」なのかなどを決めてやらなければいけないのである．

#### 2.2.2. 変数データの型の決定

変数データの型の宣言は以下の 3 通りがあり，この順に優先される
(1) 宣言文
　明示的に変数の型を決めてやる．下の例は倍精度実数の指定．
　(例) real(8) :: x, y, z
(2) Implicit 文
　特定のアルファベットから始まる変数は特定の型を持つという指定の仕方．下の例は，

a から h および o から z で始まる変数は倍精度実数とする指定．プログラムの先頭によく用いられる．

（例） implicit real*8 (a-h,o-z)

(3) 暗黙の約束(Default implied typing)

何も指定しない場合’i，j，k，l，m，n’で始まる変数は整数型でそれ以外は実数型と扱われる．この約束を取り消すには**「implicit none」**と指定する．

(2)，(3)は Fortran77 の名残りで現在でも通用するが，今後は最初に，**「implicit none」と指定して，使う変数は全て型宣言する**方がよい．

### 2.2.3. 型宣言文

それぞれの型は以下の宣言文により指定する

| | |
|---|---|
| 整数型 | **integer** :: ix, iy, iz |
| 実数型 | **real** :: x, y, z |
| 倍精度実数型 | **double precision** :: x,y,z |
| 複素数型 | **complex** :: x, y, z |
| 倍精度複素数型 | **complex(kind(0d0))** :: x, y, z |
| 論理型 | **logical** :: x, y, z |
| 文字型 | **character** :: word |

**「単精度」は有効数字がおよそ 7 桁，「倍精度」はおよそ 15 桁である．**

### 2.2.4. 定数データの型

定数の型は以下のように指定する．

| | |
|---|---|
| 整数 | 4 |
| 単精度実数 | 2. |
| | 2.0 |
| | 2.0e0 |
| 倍精度実数 | 2.0d0 |
| 単精度複素数 | cmplx(1., 1.) |
| | cmplx(1.0, 1.0) |
| | cmplx(1.0e0, 1.0e0) |
| 倍精度複素数 | cmplx(1.0d0, 1.0d0) |

「2.0e0」と「2.0d0」はともに，「$2.0\times10^{0}$」という意味である．しかし，**「2.0e0」は，有効数字 8 桁までしか保証されない．つまり，2.0000000123 でも 2.0e0 である．「2.0d0」とかくと，有効数字はおよそ 15 桁である**．このことは，特に演算を行うときに重要である

## 2.3. 数値演算

### 2.3.1. 代表的な数値演算

Fortran には，代表的な演算として以下のものがある．

| 関数 | 意味 | 関数 | 意味 |
|---|---|---|---|
| + | + | - | - |
| * | × | / | ÷ |
| x**n | $x^n$ | sin(x) | sin$x$ |
| cos(x) | cos$x$ | tan(x) | tan$x$ |
| atan(x) | Arctanx | exp(x) | $e^x$ |
| log(x) | log$x$ | sqrt(x) | $\sqrt{x}$ |
| abs(x) | $\|x\|$ | sign(x) | x/\|x\| |

### 2.3.2. データ型の重要性

データの型は演算を行うときに重要である．

[問題１] 以下の４つのプログラムを作成して，実行結果を考察しなさい．

プログラム１

```
program main
implicit none
integer :: ix,iy,iz
ix=1.0d0
iy=3.0d0
iz=ix/iy
write(6,*) 'iz=',iz
stop
end program main
```

プログラム２

```
program main
implicit none
real :: ix,iy,iz
ix=1.0d0
iy=3.0d0
iz=ix/iy
write(6,*) 'iz=',iz
stop
end program main
```

プログラム３

```
program main
implicit none
real :: x
x=atan(1.0d0)
write(6,*) 'x=',x
stop
end program main
```

プログラム４

```
program main
implicit none
double precision :: x
x=atan(1.0d0)
write(6,*) 'x=',x
stop
end program main
```

### 2.3.3. 代入文(プログラムでの＝の意味)

Fortran に限らず，**プログラム言語で「＝」は「右辺を左辺に代入する」という意味を持ち，数学における「右辺と左辺は等しい」という意味と異なる**．例えば，

```
1       program main
2       implicit none
3       real :: x
4       x=1.0d0
5       x=x+1.0d0
6       write(6,*) 'x=',x
7       stop
8       end program main
```

というプログラムを考えてみる．数学的には「x=x+1.0d0」から「0.0d0=1.0d0」となり変になるが，答えは x=2.0d0 である．4 行目で x に 1.0d0 が代入され，5 行目の右辺で 1.0d0 である x に 1.0d0 を足して，右辺は 2.0d0 となりその値を改めて左辺の x に代入するのである．

**型の異なる変数に代入をしたときは左辺の型が有効になる**．例えば単精度を倍精度に代入した場合，倍精度として扱われる．しかし有効数字が増えるわけではない．**バグのもとになるのでやら無いこと**．

### 2.4. 簡単な入出力

ファイルからデータを読み込むときは「read 文」，ファイルにデータを書き込むときは「write 文」を用いる．ファイルの指定には装置番号と呼ばれる番号を用いる．ここではファイルの１つである，キーボードからデータを読み込んで，コンソール(画面)に結果をかき出すもっとも簡単な方法を示す．より詳しい文法は後で習う．デフォルトでは，**キーボードは装置番号 5 番に，コンソールは 6 番に割り付けられている**．

(例) キーボードから変数 x に数値を入力する．

```
read(5,*) x
```

(例) 変数 x の値をコンソールにかき出す

```
write(6,*) x
```

(例) 変数 x の値をコンソールに「x=」の後にかき出す．

```
write(6,*) 'x=',x
```

今まで習ったことを使ったプログラム例を 2 つ示す.

(例 3-１) キーボードから'x', 'y'の値を読み込んで'x+y'の値を返す.

```
program main
implicit none
double precision :: x,y,z
read(5,*) x
read(5,*) y
z=x+y
write(6,*) 'z=',z
stop
end program main
```

(例 3-２) πの値を出力する

```
program main
implicit none
double precision :: pi
pi=4.0d0*atan(1.0d0)
write(6,*) 'pi=',pi
stop
end program main
```

(例 3-2)はプログラム中でπの値を作るときの常套手段である.

[問題２] 華氏温度(Fahrenheit temperature) $T_F$ とセ氏温度(Cerucius temperature) $T_C$ の間には, $T_C = 5(T_F-32)/9$ 関係がある. 上のプログラムを参考にして, $T_F$ をキーボードから読み込んで, コンソールに $T_C$ を返すプログラムを作れ.