

情報処理概論

第4回 Fortran の基本 2

情報基盤研究開発センター 谷本 輝夫

今回の内容

- ▶ 前回の演習
- ▶ 今回の概要
- ▶ 変数
- ▶ 数学関数
- ▶ キーボードからの入力
- ▶ 表示方式（書式）の調整

前回の演習

- ▶ 整数、単精度実数、倍精度実数で同じ計算をして結果が変わる例を考え、実際にプログラムを作って確認する。

```
program error
  write(*, *) 1/3
  write(*, *) 1/3.0
  write(*, *) 1/3D0
  stop
end program error
```

```
$ gfortran -o enshu3-2 enshu3-2.f90
$ ./enshu3-2
      0
0.333333343
0.33333333333333331
```

今回の内容

- ▶ 前回の演習
- ▶ 今回の概要
- ▶ 変数
- ▶ 数学関数
- ▶ キーボードからの入力
- ▶ 表示方式（書式）の調整

今日の予習

- ▶ 以下のプログラムを入力し、コンパイルして、実行

```
program circle
  implicit none
  real(8) :: r, pi
  intrinsic atan
```

型宣言のない変数の利用を禁止する

r と pi という2つの変数(倍精度実数)を利用することを宣言

数学関数 atan を利用することを宣言

```
  pi = 4D0 * atan(1D0)
  write(*, *) 'Radius?'
  read(*, *) r
```

pi の値を計算して格納

メッセージの表示

キーボードから入力した値を r に格納

```
  write(*, '(A20, F10.4)') 'Area of Circle : ', pi*r*r
  write(*, '(A20, F10.4)') 'Volume of Sphere : ', &
    pi*r*r*r*4D0/3D0
```

pi と r に格納された値を参照して計算し、結果を表示
(表示結果の桁を揃えるために書式を指定)

```
stop
end program
```

実行例

- ▶ コンパイルして実行
 - ▶ ソースプログラムを `circ.f90` という名前のファイルで作成した場合

```
$ gfortran circ.f90 -o circ
$ ./circ
Radius?
5 ← キーボードから入力
   Area of Circle :    78.5398
   Volume of Sphere :  523.5988
```

今回の内容

- ▶ 前回の演習
- ▶ 今回の概要
- ▶ 変数
- ▶ 数学関数
- ▶ キーボードからの入力
- ▶ 表示方式（書式）の調整

変数の利用方法

- ▶ 宣言し、値を格納、格納した値を参照

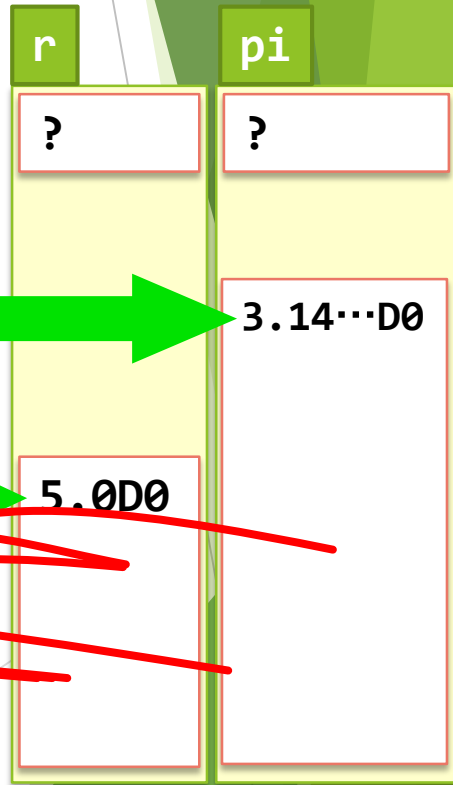
```
...  
real(8) :: r, pi  
...  
pi = 4D0 * atan(1D0)  
...  
read(*, *) r  
write(*, '(A20, F10.4)') 'Area of Circle : ', pi*r*r  
write(*, '(A20, F10.4)') 'Volume of Sphere : ', &  
pi*r*r*r*4D0/3D0
```

宣言

値(3.14159...)の格納

値(例: 5)の格納

値の参照



変数の宣言

- ▶ プログラム中で使用する変数の名前とデータ型をプログラムの先頭で宣言

- ▶ 例)

```
real(8) :: r, pi
```

- ▶ 1つの変数について宣言は1回だけ
- ▶ 同じデータ型の変数は , で区切って列挙して良い
 - ▶ 別々に宣言しても良い

```
real(8) :: r  
real(8) :: pi
```

変数のデータ型

- ▶ その変数に格納できるデータ型
- ▶ この講義で利用する主なデータ型 :

データ型	宣言方法	1個のサイズ
整数	<code>integer</code>	4バイト
単精度実数	<code>real</code>	4バイト
倍精度実数	<code>real(8)</code>	8バイト
文字	<code>character</code>	1バイト
単精度複素数	<code>complex</code>	8バイト
倍精度複素数	<code>complex(8)</code>	16バイト

変数の宣言は必要か？

- ▶ 実は、宣言せずに変数を利用可能（**暗黙の型宣言機能**）
 - ▶ 何も宣言しない場合、変数の名前に応じて以下の通り型を設定
 - ▶ 変数名の最初の文字が $i \sim n$ であれば整数
 - ▶ それ以外は単精度実数
- ▶ でも、プログラムの入力ミスに気付きにくくなる

```
program triangle
  real(8) :: r, rad
  intrinsic cos, sin
  r = 4.5D0
  rad = 30.0D0 * 3.14159265 / 180.D0
  write(*,*) r*cos(rad)*(r + r*sin(rad))
  ...
```

入力ミスを探そう

暗黙の型宣言を無効にする implicit none

- ▶ 全ての変数について、宣言が必要
- ▶ 宣言されていない変数を使っているプログラムをコンパイルしようとすると、エラーが発生
 - ▶ 例) さっきのプログラムに implicit none を追加してコンパイル

```
program triangle
  implicit none
  real(8) :: r, rad, pi
  ...
```

```
$ gfortran triangle.f90 -o triangle
triangle.f90:7.23:
  write(*, *) r*cos(red)*(r + r*sin(rad))
                        1
Error: Symbol 'red' at (1) has no IMPLICIT
type
```

変数への値の格納（代入）

- ▶ = の左辺に指定した変数に、右辺の値（もしくは計算式の計算結果）を格納する。

- ▶ 例) `r = 4.5D0`

```
a = r * r * pi
```

- ▶ 左辺と右辺で型が違う場合：
左辺の変数のデータ型に合わせて自動変換される

- ▶ 実数を整数変数に代入する場合は切り捨て

- ▶ 例)

<code>real(8) :: r</code> <code>r = 5</code>	<code>integer :: r</code> <code>r = 5.5D0</code>
---	---

rはどんな値になる？

変数に格納された値の参照

- ▶ write文で表示

```
write (*, *) 'Radian: ', rad
```

- ▶ 計算式や関数で利用

```
menseki = 2D0 * r * cos(rad) * (r + sin(rad))/2D0
```

変数を使う上での注意 (1)

変数の名前

- ▶ 変数の名前に利用できるのは半角英数字と `_` だけ
 - ▶ 先頭は必ずアルファベット
 - ▶ 31文字以内

正しい変数名の例

```
real(8) :: r
```

```
real(8) :: information_a
```

```
real(8) :: abc123def456ghi789
```

間違った変数名の例

```
real(8) :: 345abc
```

```
real(8) :: _a
```

変数を使う上での注意 (2)

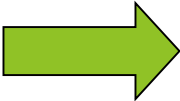
数学の数式とは違う

- ▶ Fortranの = は ← に近いイメージ：“等しい”ではなく“格納”


- ▶ 以下の式も OK (t の値を一つ増やす)

$$t = t + 1$$

- ▶ 変数 a と b に 0 を代入したい場合：

~~$a = b = 0$~~  $a = 0$
 $b = 0$

- ▶ a + b の結果を変数 c に代入したい場合：

~~$a + b = c$~~  $c = a + b$

今回の内容

- ▶ 前回の演習
- ▶ 今回の概要
- ▶ 変数
- ▶ 数学関数
- ▶ キーボードからの入力
- ▶ 表示方式（書式）の調整

Fortranの数学関数

- ▶ 主な数学関数は Fortran で利用可能。
 - ▶ 例)
 - ▶ 平方根 `sqrt`
 - ▶ 三角関数 `sin, cos, tan, asin, acos, atan`
 - ▶ 対数 `log, exp`
 - ▶ 数学関数以外にもさまざまな関数を用意：組み込み関数
 - ▶ 利用可能な他の関数についてはテキストを参照

数学関数の利用方法

- ▶ プログラム中で使用する関数の名前をプログラムの先頭であらかじめ宣言しておく
 - ▶ intrinsic 文
- ▶ 宣言した関数は任意の計算式の中で利用可能

```
program sample2
  intrinsic sqrt, exp
  write(*, *) sqrt(exp(2D0) * 12D0 - 0.5D0)
stop
end program
```

使用する関数の宣言

関数の利用

三角関数の利用

- ▶ 三角関数の角度はラジアンで指定する
 - ▶ 1.0 ラジアン = $180.0 / \pi$ 度
 - ▶ 1.0 度 = $\pi / 180.0$ ラジアン
- ▶ 例) 60度の cos
 - ▶ $\cos(60D0 * 3.14159265D0 / 180D0)$

π の算出方法

- ▶ 直接以下のように設定してもよいが

```
pi = 3.14159265358979264D0
```

- ▶ 以下のように計算で求めることもできる。

```
intrinsic atan
```

```
pi = 4D0 * atan(1D0)
```

今回の内容

- ▶ 前回の演習
- ▶ 今回の概要
- ▶ 変数
- ▶ 数学関数
- ▶ キーボードからの入力
- ▶ 表示方式（書式）の調整

キーボードからのデータ入力 read 文

- ▶ 利用方法 : read(*,*) 変数 1 , 変数2, ...
 - ▶ 例) 半径を入力し、円の面積を表示するプログラム

```
program circle
  implicit none
  real(8) :: r, pi
  intrinsic atan

  pi = 4D0 * atan(1D0)
  write(*, *) 'Hankei?'
  read(*, *) r
  write(*, *) 'Menseki : ', r * r * pi

  stop
end program
```

実行例

▶ 翻訳して実行

```
$ gfortran circ.f90 -o circ
$ ./circ
Hankei?
5
  Menseki : 78.5398163397448
$ ./circ
Hankei?
12.25
  Menseki : 471.435247579318
```

キーボードから入力

キーボードから入力

今回の内容

- ▶ 前回の演習
- ▶ 今回の概要
- ▶ 変数
- ▶ 数学関数
- ▶ キーボードからの入力
- ▶ 表示方式（書式）の調整

表示の桁揃え

- ▶ 例) 表示内容を列挙するだけだと桁はバラバラ

```
program sample3
  write(*, *) 'John', (51.0 + 63.0 + 72.0) / 3.0
  write(*, *) 'Paul', (6.0 + 15.0 + 8.0) / 3.0
  write(*, *) 'George', (83.0 + 76.0 + 91.0) / 3.0
  write(*, *) 'Richard', (67.0 + 82.0 + 86.0) / 3.0
stop
end program
```




```
John 62.0000000
Paul 9.66666698
George 83.3333359
Richard 78.3333359
```

書式の指定

- ▶ 書式を指定すると桁を揃えることができる

```
program sample4
  write(*, '(A10,1X,F5.2)') 'John', (51.0 + 63.0 + 72.0) / 3.0
  write(*, '(A10,1X,F5.2)') 'Paul', (6.0 + 15.0 + 8.0) / 3.0
  write(*, '(A10,1X,F5.2)') 'George', (83.0 + 76.0 + 91.0) / 3.0
  write(*, '(A10,1X,F5.2)') 'Richard' &
    , (67.0 + 82.0 + 86.0) / 3.0
stop
end program
```



```
John 62.00
Paul  9.67
George 83.33
Richard 78.33
```

実数を5桁の幅に表示
(ただし小数点以下は2桁まで表示)

1桁分の空白表示

文字列を10桁の幅に表示

書式の指定方法

- ▶ write文の括弧内で指定

```
write(*, '(書式1,書式2, ...)' 表示内容1, 表示内容2, ...)
```

- ▶ 書式

- ▶ 表示する内容それぞれについて、データ型と桁数を指定

- ▶ 書式の指定記号

表示内容	書式	指定内容
整数	I w	w 桁で整数を表示
実数	F $w.m$	全体 w 桁、小数点以下 m 桁で実数を表示
実数	E $w.m$	全体 w 桁、小数点以下 m 桁で実数を指数形式により表示（指数形式 = $0.???? \times 10^?$ の形式）
文字列	A w	w 桁で文字列を表示
空白	nX	n桁の空白を表示（表示内容とは無関係に指定可能）

書式の便利な使い方 1

- ▶ 同じ書式が続く場合、以下のようにまとめて指定できる

F10.5, F10.5, F10.5



3F10.5

- ▶ 同じ書式パターンが続く場合 () でまとめられる

A6, I4, A6, I4, A6, I4



3(A6, I4)

- ▶ 文字列データは書式中に記述しても良い

```
write(*, '(A6, F5.2)') 'Ave =', (50.0 + 60.0 + 70.0) / 3.0
```



```
write(*, '(" Ave = ", F5.2)') (50.0 + 60.0 + 70.0) / 3.0
```

書式指定の注意点

- ▶ 書式指定と表示内容を1対1で対応させる

```
write(*, '(I3, A10, 1X, 3I4, A6, F5.2)') &  
1, 'John', 50, 60, 70, 'Ave=', (50.0 + 60.0 + 70.0)/3.0
```

書式指定と表示内容の対応ができていない場合、
どうなる？

- 書式の数が足りない場合
 - 書式の数が多すぎる場合
 - 書式の型と表示内容の型が違う場合
- 等

書式の便利な使い方 2

- ▶ 同じ書式を何度も使う場合、1箇所で定義した書式を共有可

```
program sample5
  write(*, 100) &
    1, 'John', 'Math=', 50, 'Eng=', 60, 'Chem=', 70 &
    , (50.0 + 60.0 + 70.0)/3.0
  write(*, 100) &
    2, 'Paul', 'Math=', 6, 'Eng=', 15, 'Chem=', 8 &
    , (6.0 + 15.0 + 8.0)/3.0
  write(*, 100) &
    3, 'George', 'Math=', 83, 'Eng=', 76, 'Chem=', 91 &
    , (83.0 + 76.0 + 91.0)/3.0
  write(*, 100) &
    4, 'Richard', 'Math=', 67, 'Eng=', 82, 'Chem=', 86 &
    , (67.0 + 82.0 + 86.0)/3.0
  100 format(I3, A10, 1X, 3(A6, I4), ' Ave=', F5.2)
stop
end program
```

100番の書式
を参照

100番の書式

format文の利用法

- ▶ 先頭に行番号を指定

行番号 format(書式)

- ▶ write文等から参照
- ▶ 同じ行番号を複数の format文に使用しない

演習 4

- ▶ キーボードから数値を入力させ、それをもとに計算をして結果を出力するプログラムを作成する。
 - ▶ 例) 半径を入力し、その半径の円に内接する正三角形の面積を計算し、表示するプログラム

```
Radius?
```

```
100
```

```
Area of the inscribed triangle is      12990.38
```

- ▶ 冒頭のプログラムを修正しても良い
- ▶ 表示する桁数も適切に調整する

今までに多かったミス

- ▶ ファイル名の間違い。末尾が .f90 になっていない。

```
$ emacs test
```

```
$ emacs test .f90
```

- ▶ プログラムや変数名として使えない記号を使っていた

```
program test.f90
```

```
integer :: a:b
```

- ▶ 乗算記号 * を忘れていた

```
2.0 ( 3.0 + exp(4.0))
```

- ▶ , と . を打ち間違えた

```
30,0 * 3,14159265
```

- ▶ プログラム中に全角文字が入っていた
 - ▶ 半角/全角 キーの押し忘れ (特に空白)
 - ▶ 文字列には、全角文字が入っていてもよい