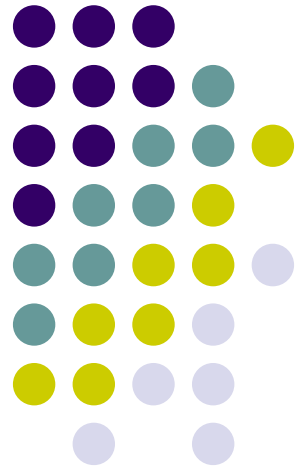
情報処理概論

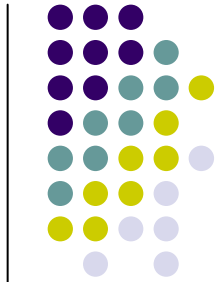
工学部 物質科学工学科
応用化学コース
機能物質化学クラス



第7回

2005年　6月　2日

今日の目標



- 5人分のテストの点数を入力して平均点を計算し、さらにそれぞれの点数と平均点の差を算出する。
  - 例えば5人の点数が
    50点、55点、60点、60点、75点
    の場合：
    平均：61点
    平均との差：　−11点、−6点、−1点、−1点、14点

- プログラムを最初から設計してみる

まず計算式を確認

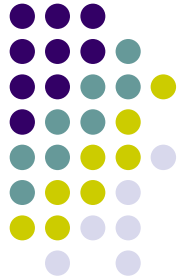
- 5人の点数がそれぞれ変数 a, b, c, d, e に格納されているとすると

平均点　＝ (a + b + c + d + e) / 5

平均点との差:

a ― 平均点
b ― 平均点
c ― 平均点
d ― 平均点
e ― 平均点

必要となりそうな変数に名前と型をつける

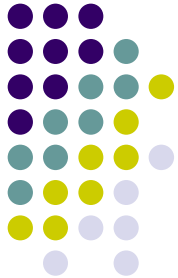


- 5人分の点数(整数):　a, b, c, d, e
  （実は、後述する”配列”を使ったほうが現実的)

- 平均点(実数): ave

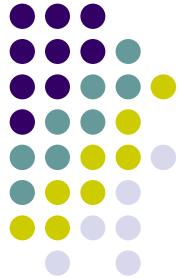
- 平均点との差は計算結果を直接 write文で表示すればよいので変数不要

実行イメージを考える



```
No. 1:
20
No. 2:
35
No. 3:
73
No. 4:
42
No. 5:
58
 Average =  45.6
   No. 1:  -25.6
   No. 2:  -10.6
   No. 3:  27.4
   No. 4:  -3.6
   No. 5:  12.4
```

大まかなプログラムの流れ



- 点数入力部分
- 平均点の計算部分
- 平均点の表示部分
- 平均点と各点数の差の表示部分

それぞれの部品の設計

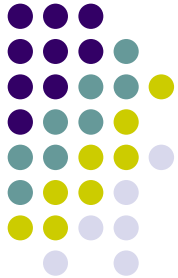
- 例1:　データ入力部分

```
write(*, *) 'No. 1:'
read(*, *) a
write(*, *) 'No. 2:'
read(*, *) b
...
```

- 例2:　平均点計算部分
  - ave は実数なので計算に注意

```
ave = (a + b + c + d + e) / 5.0D0
```

組み立て



- Fortran90 の文法にあわせて
  プログラム全体を仕上げる。
  - 変数、数学関数の宣言を忘れない。
    - 部品設計時に必要だと判明した変数があればそれも宣言する。

```
program average
   implicit none
   変数の宣言（必要に応じて）
   数学関数の宣言（必要に応じて）

   点数入力部分
   平均点の計算部分
   平均点の表示部分
   平均点との差の表示部分
   stop
end program
```

プログラム例

```
program average
  implicit none
  integer :: a, b, c, d, e
  real(8) :: ave

  write(*, *) 'No. 1:'
  read(*, *) a
  write(*, *) 'No. 2:'
  read(*, *) b
  write(*, *) 'No. 3:'
  read(*, *) c
  write(*, *) 'No. 4:'
  read(*, *) d
  write(*, *) 'No. 5:'
  read(*, *) e

  ave = (a + b + c + d + e) / 5.0d0

  write(*, *) 'Average = ', ave
  write(*, *) '  No. 1: ', dble(a) – ave
  write(*, *) '  No. 2: ', dble(b) – ave
  write(*, *) '  No. 3: ', dble(c) – ave
  write(*, *) '  No. 4: ', dble(d) – ave
  write(*, *) '  No. 5: ', dble(e) – ave
  stop
end program
```

このプログラムの問題点

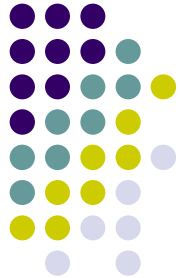
- 人数が変わると書き換えが大変
  - 例えば　100人になったら？
    → 作業量が非現実的

  - 以下のように添え字を使って計算できると便利

$$ave = \frac{\sum_{i=1}^{100} a(i)}{100}$$

”配列”を使えばできる

Fortran の配列変数

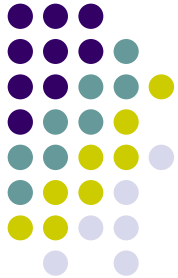


- 数学のベクトルや行列に似た概念
  - 名前と添え字（位置を示す番号）で値を格納、参照
  - 利用法は通常の変数とほぼ同様

```
a(1) = 1.0D0
b(4) = b(4) + 1
```

  - 宣言は少し違う

```
real(8), dimension(10) :: a, b, c
```



配列変数の代入，参照

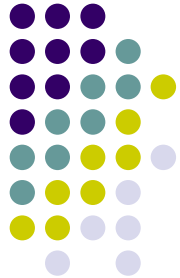
- 位置は整数で指定：
  - 整数変数や整数式も利用可能
    例）

```
a(i) = b(i * 2 + j)
```

  - 実数の計算でも位置は整数
    間違いの例）

```
a(1.0D0) = 2.0D0
```

配列変数と繰り返し

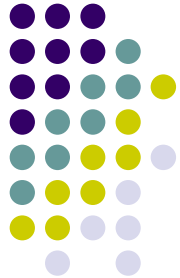


- 配列変数を繰り返し（do文）の中で用いることによって任意の数のデータを簡単に処理可能

  例) データ入力部分

```
do i = 1, 5
  write(*, *) 'No. ', i, ':'
  read(*, *) a(i)
end do
```

配列変数の宣言



- 型の直後に dimension で指定

```
型, dimension(範囲) :: 配列変数名
```

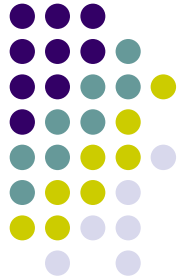
  - 型： Fortranで利用できる全ての型を指定可能
  - 範囲： 開始番号と終了番号，もしくは終了番号のみを指定

例1)
```
integer, dimension(0:10) :: score
```
score(0) ~ score(10) が利用可能

例2）
```
integer, dimension(10) :: score
```
dimension(1:10) と同じ

  - 配列変数名： 通常の変数と同様

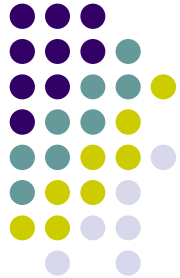
配列変数を使った平均値計算



- 一旦合計値を計算して、最後に人数で割る
  - 合計値用の変数(整数)：total
    - 最初 0 にしておいて，点数を加算していく

```
total = 0
do i = 1, 5
  total = total + a(i)
end do
ave = total / 5.0D0
```

配列を使ったプログラム例



```
program average
  implicit none
  integer, dimension(5) :: a
  integer :: total, i
  real(8) :: ave

  do i = 1, 5
    write(*, *) 'No. ', i, ':'
    read(*, *) a(i)
  end do

  total = 0
  do i = 1, 5
    total = total + a(i)
  end do
  ave = total / 5.0d0

  write(*, *) 'Average = ', ave
  do i = 1, 5
    write(*, *) '   No. ', a(i) - ave
  end do
stop
end program
```

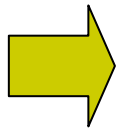
プログラム中の定数

- 何度も出てくる同じ数値に名前を付ける
  - 例） 前のページのプログラム中の 5 (及び 5.0D0)
    - あらかじめ以下の宣言で number という名前を付けると

```
integer, parameter :: number = 5
```

    - プログラム中で参照できる

```
integer, dimension(5) :: a
do i = 1, 5
do i = 1, 5
ave = total / 5.0d0
do i = 1, 5
```

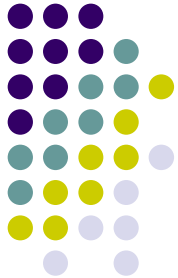


```
integer, dimension(number) :: a
do i = 1, number
do i = 1, number
ave = total / dble(number)
do i = 1, number
```

人数の変更が一箇所でできる

数字の意味が分かりやすくなる

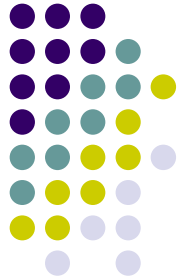
パラメータ



- プログラム中の定数を定義

- 配列の宣言に利用できるので便利
  - 変数は配列の宣言には利用できない

パラメータを利用したプログラム例



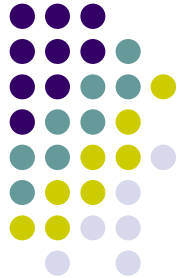
```
program average
  implicit none
  integer, parameter :: number = 5
  integer, dimension(number) :: a
  integer :: total, i
  real(8) :: ave
  intrinsic dble

  do i = 1, number
    write(*, *) 'No. ', i, ':'
    read(*, *) a(i)
  end do

  total = 0
  do i = 1, number
    total = total + a(i)
  end do
  ave = total / dble(number)

  write(*, *) 'Average = ', ave
  do i = 1, number
    write(*, *) '   No. ', a(i) - ave
  end do
stop
end program
```

パラメータの利用



- 効果:
  - 値を変更する際に一箇所の修正で済む
  - 値の意味を分かりやすくする

- 利用方法:
  - 宣言

```
型, parameter :: パラメータ名 = 値
```

    - 例） 5 という値に number というパラメータ名をつける

```
integer, parameter :: number = 5
```

  - 変数との違い
    - 値は変更できない.

```
write(*, *) 'Average = ', dble(total)/dble(number)
```

    - 配列の宣言にも利用可能

```
real(8), dimension(number) :: score
```

多次元配列

- 今回扱ったのは1次元配列だけだが
  Fortranでは2次元以上の配列も同じように扱える
  - 例) ５×１０の2次元配列を宣言

```
real(8), dimension(0:4, 10) :: a
```

a(0, 1) ~ a(4,10)を利用可能

  - 例) 3次元配列に値を格納

```
b(1, 5, 2) = 2.0D0
```

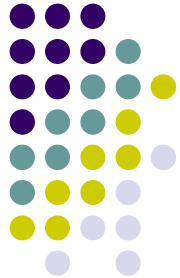
演習： 多項式の計算

- 実数 $a_0, a_1, a_2, a_3, a_4, a_5, x$ を入力し
  以下の多項式の値を計算するプログラムを作成

$$a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$$

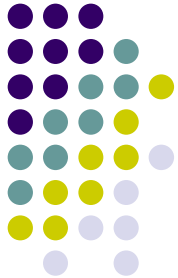
- $a_0, a_1, a_2, a_3, a_4, a_5$ は配列に格納すること
  - 添え字が0から始まる点に注意
- do 文を使って計算すること

必要となりそうな変数に名前と型をつける



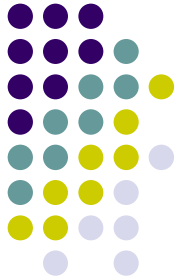
- 例えば...
  - 多項式の係数:　配列 a
  - 変数:　x
  - do文の制御変数: i
  - 計算結果: result

実行イメージを考える



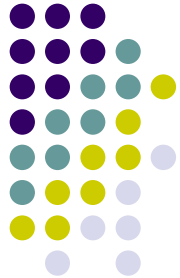
```
 a 0:
1.0
 a 1:
2.0
 a 2:
3.0
 a 3:
4.0
 a 4:
5.0
 a 5:
6.0
 x:
10.0
 Result =  654321.0000000000
```

大まかなプログラムの流れ



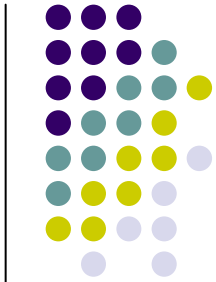
- 係数入力部分
- xの入力部分
- 多項式の計算部分
- 結果の表示部分

それぞれの部品の設計



- 過去に作ったプログラムを参考にする
  - 係数の入力，xの入力
    - レポートや平均点計算のプログラム等を参考
  - 多項式の計算
    - 平均点計算プログラムの合計値計算部分を参考

組み立て



- Fortran90 の文法にあわせて
  プログラム全体を仕上げる。
  - 変数、数学関数の宣言を忘れない。
    - 部品設計時に必要だと判明した変数があればそれも宣言する。

```
program average
   implicit none
   変数の宣言（必要に応じて）
   数学関数の宣言（必要に応じて）

   係数入力部分
   x の入力部分
   多項式の計算部分
   計算結果の表示部分
   stop
end program
```