情報処理概論

第７回　配列２

情報基盤研究開発センター　谷本 輝夫

今日の内容

- 多次元配列
- 割付配列
  - 実行中に配列の大きさを決める
- 配列の一括操作
  - 代入
  - 表示
  - 計算

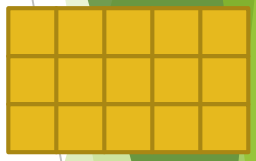
多次元配列の利用

- 1次元配列

宣言の例）
```
integer, dimension(5) :: a
```
使用例）
```
a(1) = 1
```

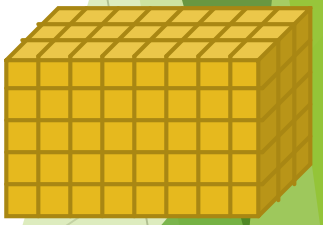
- 2次元配列

宣言の例）
```
real(8), dimension(3, 5) :: b
```
使用例）
```
b(3, 2) = 1.0D0d
```



- 3次元配列

宣言の例）
```
real(8), dimension(5, 8, 3) :: c
```
使用例）
```
c(4, 8, 2) = 2.0D0
```



- 4次元配列以上もOK

Fortran では，任意の次元の配列を利用可能

2次元配列の利用例

- 英語，数学，国語の３教科の点数を
  ５人分入力し，合計点の総平均を求める

```
$ ./score2
No. 1
Kamoku 1:
30
Kamoku 2:
60
Kamoku 3:
40
...
No. 5
...
 Average =
```

プログラム例（1/2）

```
program Score_management
  implicit none
  integer, parameter :: number = 5
  integer, parameter :: kamoku = 3
  integer :: i, j, total
  integer,dimension(number, kamoku) :: score
  real(8) :: ave
  intrinsic dble

  ! Input score data from keyboard
  do i = 1, number
    write(*, *) 'No.', i
    do j = 1, kamoku
      write(*, *) 'Kamoku ', j, ' : '
      read(*, *) score(i, j)
    end do
  end do
```

プログラム例（2/2）

```
  ! Calculate total and average
  total = 0
  do i = 1, number
    do j = 1, kamoku
      total = total + score(i, j)
    end do
  end do
  ave = dble(total) / dble(number*kamoku)

  write(*, *) 'Average = ', ave

stop
end program
```

今日の内容

- 多次元配列
- 割付配列
  - 実行中に配列の大きさを決める
- 配列の一括操作
  - 代入
  - 表示
  - 計算

実行時に配列の大きさを決めたい

- 例えば、前のプログラムで人数を変更したい
  - 方法１：毎回 emacs でプログラムを修正して，再度コンパイルして実行ファイルを作成

```
program score2
  implicit none
  integer, parameter :: number = 5
  integer, parameter :: kamoku = 3
  integer :: i, j, total
  integer,dimension(number, kamoku) :: score
```

人数が変わるたびに変更してコンパイル

  - 方法２：実行時に配列の大きさを決定

プログラム例（冒頭のみ）

```
program score3
  implicit none
  integer, parameter :: kamoku = 3
  integer :: i, j, total, number
  integer, dimension(:,:), allocatable :: score
  real(8) :: ave
  intrinsic dble

  ! Input number from keyboard
  write(*, *) 'Number: '
  read(*, *) number

  ! Set size of array A
  allocate(score(number, kamoku))
```

次元の数だけ指示

割り付け配列の宣言

大きさが決まった時点で割り付け

後は変更無し

実行例

- 人数を指定した後、英語，数学，国語の３教科の点数を人数分入力し，合計点の総平均を求める

```
$ ./score3
Number:
4
No. 1
Kamoku 1:
30
Kamoku 2:
60
Kamoku 3:
40
...
No. 4
...
 Average =
```

割り付け配列の使い方

- 宣言：大きさを指定せず，allocatable を追加

```
型, dimension(:), allocatable :: 配列変数名
```

- 大きさ（範囲）のかわりに ： を次元の数だけ記述
  - 例）３次元の割り付け配列の宣言

```
real(8), dimension(:,:,:), allocatable :: data
```

割り付け配列の使い方

- 大きさ（範囲）の指定

```
allocate(配列変数(大きさ))
```

  - 配列変数：allocatable付きで宣言した配列
  - 大きさ（範囲）の指定方法は配列を宣言するときと同様

- 例） 3次元配列の各次元の大きさ（範囲）を指定

```
allocate(data(100, 0:20, -50:50))
```

- 配列を利用する前に範囲を指定する

配列の大きさの指定方法：３通り

- 整数を直接指定

```
integer, dimension(5) :: a
```

- 定数を指定

```
integer, parameter :: n = 5
integer, dimension(n) :: a
```

- 実行時に指定

```
integer, dimension(:), allocatable :: a
...
  allocate(a(5))
```

今日の内容

- 多次元配列
- 割付配列
  - 実行中に配列の大きさを決める
- 配列の一括操作
  - 代入
  - 表示
  - 計算

配列操作の省略形

- 例）　配列 a の全要素に 0.0D0 を代入
  - do 文等で繰り返して参照,代入してもよいが…

do文で繰り返す方法

```
do i = 1, n
  do j = 1, n
    a(j, i) = 0.0D0
  end do
end do
```

省略形

```
a = 0.0D0
```

do 文を使わずに配列の全体や部分に対する操作を記述できる

配列全体への一括代入

- 全部に同じ値を代入

```
a = 0.0D0
```

- 各要素に任意の値を代入

```
a = (/1.0D0, 0.2D0,  -3.5D0, 0.0D0, 12.5D0/)
```

- 規則的な値を代入

```
a = (/ (2 * i, i = 1, 5) /)
```

以下の do文と同じ意味

```
do i = 1, 5
  a(i) = 2 * i
end do
```

部分配列への一括代入

- 部分配列の指定方法

```
配列変数名(開始番号:終了番号：間隔)
```

  - 例１） ５番目から１０番目の要素に 1.0D0 を代入

```
a(5:10) = 1.0D0
```

  - 例２）偶数の要素に, 1, 3, 5, 7, 9, … のように
    奇数を順に代入

```
a(2:n:2) = (/ (i, i = 1,n,2) /)
```

初期値の設定にも利用可能

- 宣言時に初期値を設定

```
integer, dimension(5) :: a=0D0
real(8), dimension(2, 2) :: b=(/ 1d0,0d0,0d0,1d0 /)
```

多次元配列での注意

- 格納される順番は左の次元が優先

```
a(1:2, 1:3) = (/ 1, 2, 3, 4, 5, 6/)
```

```
a(1, 1) = 1   a(1, 2) = 3   a(1, 3) = 5
a(2, 1) = 2   a(2, 2) = 4   a(2, 3) = 6
```

- 以下の形式は 1 次元配列以外では使えない

```
a(1:2, 1:3) = (/ (i, i=1,6) /)
```

配列の表示

- 全部の要素を列挙
  - 多次元配列の場合、表示も左側の次元が優先

```
  integer, dimension(2,3) :: a
...
  write(*, *) a
```

```
a(1, 1)  a(2, 1)  a(1, 2)  a(2, 2)  a(1, 3)  a(2, 3)
```

- 部分配列も表示可能

```
  write(*, *) a(2, : )
```

```
  write(*, *) score(2:3, 1:2)
```

配列の表示(書式付き)

- 1行に表示する項目数を書式で指定できる

```
  integer, dimension(2,3) :: a
...
  write(*, ’(3I5)’) a
```

5桁の整数を1行に3個ずつ

a(1, 1)	a(2, 1)	a(1, 2)
a(2, 2)	a(1, 3)	a(2, 3)

配列の計算

- 形状が同じ配列同士の計算を簡単に記述

- 配列 a に配列 b をコピー

```
a = b
```

- 配列 a に配列 b の各要素を2倍したものを格納

```
a = b * 2
```

- 配列 c に配列 a と配列 b の和を格納

```
c = a + b
```

部分配列の計算

- 部分配列でも同様の計算が可能

```
  integer, dimension(2, 3) :: a
  integer, dimension(2, 2) :: b
...
  a(1:2,1:2) = b
```

左右で形状が同じなのでOK

同じ配列同士の計算に注意

- 以下の二つは結果が違う

```
  integer, dimension(5) :: a=(/1,2,3,4,5/)
...
  a(2:5) = a(1:4)
```

４つの要素への代入が同時に行われる

```
  integer, dimension(5) :: a=(/1,2,3,4,5/)
...
  do i = 1, 4
    a(i + 1) = a(i)
  end do
```

４つの要素への代入が順番に行われる

配列計算の関数

- 配列全体や部分配列の計算を行う関数が多数用意されている
  - 例１）配列の全要素の総和

```
total = sum(a)
```

  - 例２）ベクトルの内積
    - この例では、部分配列

```
p = dot_product(a(1:3), b(4:6))
```

- 他にも、最大値、最小値、行列積等

今日の演習

- 演習（提出不要）
  - 以下の二つの操作が違う結果になることを確認する

```
  integer, dimension(5) :: a=(/1,2,3,4,5/)
...
  a(2:5) = a(1:4)
```

```
  integer, dimension(5) :: a=(/1,2,3,4,5/)
...
  do i = 1, 4
    a(i + 1) = a(i)
  end do
```

今回の演習

- 以下のプログラムを作成
  - 「ｎ人の3教科分の成績を入力すると、3教科の合計点が最も高かった人の番号を表示する」
- 人数 ｎ は実行時に入力する
- 時間に余裕がある人は、以下にも挑戦
  - 「最も合計点の低かった人の番号と点数を表示」
  - 「合計点の順に番号を表示」

実行例

```
$ ./test
Number:
2
No. 1
Kamoku 1:
30
Kamoku 2:
60
Kamoku 3:
40
No. 2
Kamoku 1:
50
Kamoku 2:
30
Kamoku 3:
100
No. 2 is the top!
```

ランタイムエラー

- 実行時にランタイムエラーが起こる場合、配列の添え字の指定が間違っていて、配列の範囲外を参照していることが多い
  - 例：a(5) の大きさの配列を確保していて、a(6) を読もうとしている場合
- なので、配列の宣言、参照範囲などを確認すること