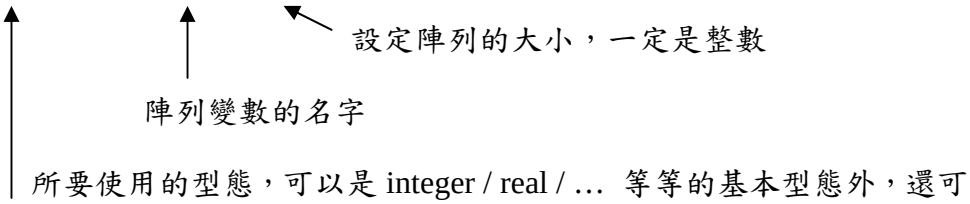
Fortran

Chapter 6　陣列

6-1　矩陣的宣告與使用

陣列的宣告法：

DataType :: name (Size)



or　DataType, dimension (Size) :: name

Example:

(1) integer :: A(10)　　宣告 10 個元素的整數陣列 A，其元素的取用範圍為 1~10，i.e. A(1),A(2),…,A(10)

(2) integer :: A(-1:8)　　宣告 10 個元素的整數陣列 A，其元素為 A(-1),A(0),A(1),A(2),…,A(10)

(3) integer :: A(0:10: 2)　　宣告 6 個整數元素陣列，其元素為 A(0),A(2),A(4),A(6),A(8),A(10)

(4) integer :: A(10:0:-2)　　宣告 6 個整數元素陣列，其元素為 A(10),A(8),A(6),A(4),A(2),A(0)

(5) integer :: A(3,3)　　宣告一個 3x3 的二維陣列

(6) integer :: A(3,3,3)　　宣告一個 3x3x3 的三維陣列

(7) integer :: A(0:3,0:3)　　宣告一個 4x4 的矩陣

6-2　設定陣列的初值

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

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

※括號和斜號之間不能有空白

(2) integer :: a(5) = (/ 3, i = 1, 5 /)

a(1) = a(2) = a(3) = a(4) = a(5) = 3

integer :: a(5) = (/ i, i = 1, 5 /)

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

(3) integer :: a(5) = (/ 0, (i, i = 2, 4), 5 /)

a(1) = 0, a(2) = 2, a(3) = 3, a(4) = 4, a(5) = 5

6-3　陣列在電腦記憶體中的儲存方法

(1) one-dimension

integer :: A(5)

元素在記憶體的連續區塊中的排列情況為

A(1)→A(2)→A(3)→A(4)→A(5)

integer :: A(0:10:2)

元素在記憶體的連續區塊中的排列情況為

A(0)→A(2)→A(4)→A(6)→A(8)→A(10)

(2) two- dimension

integer :: A(3,3)

A(1,1)	A(1,2)	A(1,3)
A(2,1)	A(2,2)	A(2,3)
A(3,1)	A(3,2)	A(3,3)

The first row

The first column

陣列 A 在記憶體的排列情況為

A(1,1) → A(1,2) → A(1,3)　← 先放 first column

→A(2,1) → A(2,2) → A(2,3)　← 再放 second column

→A(3,1) → A(3,2) → A(3,3)　← 其次 third column

~column major 排列法

Example: compute sum = $\sum_{i=1}^{3}\sum_{j=1}^{3} A(i, j)$

(i)
```
sum=0
do i = 1, 3
   do j = 1, 3
        sum = sum + A(i,j)
   end do
end do
```

not good,因為 cpu 要不斷的在記憶中跳躍式的來讀取資料，無法使用到快取記憶體（Cache）的便利（Cache 會自動抓取鄰近近的記憶體資料到 Cache 中）

(ii)
```
sum=0
do i = 1, 3
   do j = 1, 3
        sum = sum + A(j,i)
   end do
end do
```

better than (i)，充份利用 cache，程式執行的速度會較快

6-4 Fortran90 新增有關矩陣的功能

6-4-1 整個矩陣的運作

(1) a = b

```
do i = 1, k
   a(i) = b(i)
end do
```

(2) a = b + c

```
do i = 1, k
      a(i) = b(i) + c(i)
end do
```

(3) a = b - c

```
do i = 1, k
       a(i) = b(i) - c(i)
end do
```

(4) a(1:3) = b(4:6)

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

(5) a(1:5:2) = 3

a(1) = 3, a(3) = 3, a(5) = 5

(6) a(1:10) = a(10:1:-1)

把 a(1~10)的內容給反轉

(7) a = b * c

```
do i = 1, k
    a(i) = b(i) * c(i)
end do
```

(8) a = b / c

```
do i = 1, k
    a(i) = b(i) / c(i)
end do
```

6-4-3 可變大小的陣列

Fortran 90 中的陣列可以等到程式執行中再來決定它所要使用的長度

Example:

```
Program ex0709
implicit none
integer :: students, i
integer :: allocatable :: grades( : )
write(*,*) 'How many strdents in this class?'
read(*,*) students
allocate(grades(students))
do i = 1, students
   write(*, '(1X, A25, I2, A2)') 'Input grades of number', I, ':'
   read(*,*) grades(i)
end do
do i = 1, students
   if (grades(i) < 60) then
      write(*, '(1X, A10, I2, A8)') number', I, 'fail!'
   end if
end do
stop
end program ex0709
```

表示矩陣"可變大小"

說明：

(a) integer, allocatable :: grades(:)

≡ integer, allocatable, dimension(:) :: grades

只有一個冒號表示矩陣只有一個維度，要宣告兩個維度的矩陣則為 grades(:,:)

(b) allocate(grades(students))

設定矩陣大小為 students

allocate(grades(m:n:inc))

自定矩陣的上下限範圍及增值量(inc)

宣告完成後，要先經由 allocate 這個敘述來設定出陣列的大小後才能使用陣列。

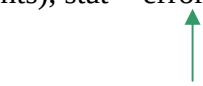
allocate：問電腦的記憶體要求空間來儲存資料

deallocate：將 allocate 所得到的陣列記憶體空間釋放

Example:

```
Program ex0710
implicit none
integer :: students, i
integer :: allocatable :: grades( : )
write(*,*) ‘How many strdents in this class A ?’
read(*,*) students
allocate(grades(students))
do i = 1, students
   write(*, ‘(1X, A25, I2, A2)’) ‘Input grades of number’, I, ‘:’
   read(*,*) grades(i)
end do
do i = 1, students
   if (grades(i) < 60) then
      write(*, ‘(1X, A10, I2, A8)’) number’, I, ‘fail!’
   end if
end do
deallocate(grades)
write(*,*) ‘How many strdents in this class B ?’
read(*,*) students
do i = 1, students
   write(*, ‘(1X, A25, I2, A2)’) ‘Input grades of number’, I, ‘:’
   read(*,*) grades(i)
end do
do i = 1, students
   if (grades(i) < 60) then
      write(*, ‘(1X, A10, I2, A8)’) number’, I, ‘fail!’
   end if
end do
stop
end program ex0710
```

allocate(grades(students), stat = error)



Error 是事先宣告好的整數變數，做 allocate 這個動作時會經由 stat 這個敘述傳給 error 一個數值。If error = 0，then allocate 陣列成功。Otherwise，allocate 失敗。電腦的記憶不足

deallocate(grades(students), stat = error)

6-4-4 陣列的庫存函數

1. dot_product(vector_a, vector_b)

   Calculates the dot product of two equal-sized vectors.

2. matmul(matrix_A, matrix_B)

   Performs matrix muttiplication on to conformable matrices.

3. maxloc(array, mask) ⇔ minloc(array, mask)

   找出陣列中最大值的所在位置，傳回一個整數。Mask 的運算方法和陣列在 where 中所使用的邏輯運算一樣。

   e.q. b = maxloc(a, a<50)

   找出陣列中小於 50 的最大數值的所在位置

4. maxval(array, mask) ⇔ minval(array, mask)

   returns the maximum value in array among those elements for which mask was true.

5. reshape(data_source, shape)

   把資料"整型"好後，再傳給一個陣列，常用在設定陣列初值時使用。

6. sum(array, mask)

   calculates the sum of the elements in array for which the mask is true. If mask is not present, it calculates the sum of all of the elements in the array.

   e.q. integer :: a(3,3) = reshape((/1,2,3,4,5,6,7,8,9/), (/3,3/))

$$a = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

   e.q. integer :: b(2,3) = reshape((/1,2,3,4,5,6/), (/2,3/))

$$b = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

7. transpose(matrix)

   returns the transpose of matri