IISEE lecture for group training

Fortran programming for beginner seismologists Lesson 5

Lecturer

Tatsuhiko Hara

Reference
Introduction to FORTRAN90/95 by S. J. Chapman (New York: McGraw-Hill, 1998)

Array

- In numerical calculations, vectors and matrices are often used.
- In Fortran, we use “array” to store values of vectors, matrices, and multi dimensional quantities.

Rank-1 array (1)

- In declaration of a certain array, it is necessary to define the type, name, and size (number of the elements) of that array. The following is an example of rank-1 (or one dimensional) array:

```
real, dimension(3) :: x
```

  where the type is real, the size is 3, and its name is `x`. So `x` has 3 elements `x(1)`, `x(2)` and `x(3)`. For example, the array whose size is 3 can be used to store the values of 3 coordinates of the position, x, y, and z.

Rank-1 array (2)

- The array `x` declared by

```
real, dimension(0:10)::x
```

  has 11 elements, `x(0)`, `x(1)`, `x(2)`, ..., `x(10)`.

- The array `y` declared by

```
real, dimension(-10:10)::y
```

  has ___ elements, `y(-10)`, `y(-9)`, `y(-8)`, ..., `y(10)`.

Rank-1 array (3)

- The size can be specified by a named constant defined using the *PARAMETER* attribute of a type declaration statement:

```
integer, parameter:: nsize=10
real, dimension(nsize):: x
```

How can we put values to array? (1)

There are several ways to put values to an array:

a. Initialization in type declaration statements

```
real, dimension(3) :: x = (/1., 2., 3./)
```

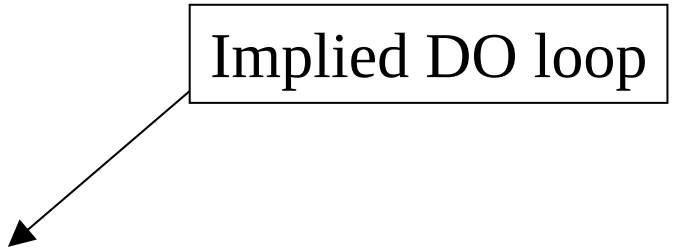
The number of elements in the constant must match the number of elements.

b. Direct assignment

```
x(1) = 1.0
x(2) = 2.0
x(3) = 3.0
```

c. Data statement

```
data (x(i)=1,3)/1., 2., 3.,/
```



If you initialize all of the elements of an array, you can write as:

```
data x/1., 2., 3.,/
```

In the above case, the size of `x` must be 3.

How can we put values to array? (2)

d. DO loop

```
do i=1, 3
    x(i) = i
end do
```

e. Read statement

```
do i=1, 3
    read(*,*) x(i)
end do
```

f. Implied Do loop

```
read(*,*) (x(i),i=1,3)
```

See pp. 213-216 of the reference for further details.

Calculation of summation using the iterative DO loop

We can efficiently perform summation calcululation such as $\sum_{n=1}^{4} n$

```
program cal_sum
implicit none
real :: sum
integer :: i
sum = 0.0
do i=1, 100
    sum = sum + i
end do
write(*,*) 'sum: ', sum
stop
end program cal_sum
```

How does *sum* change?

i	Sum (right hand)	Calculation (sum+i)	Sum (left hand)
1	0	0+1	1
2	1	1+2	3 (=1+2)
3	3	3+3	6 (=1+2+3)
4	6	6+4	10 (=1+2+3+4)

Initialization of arrays

EXERCISE 5-1

- Please compile the following program, and run it.

```
program array1
implicit none
integer, parameter:: nsize=100
real, dimension(nsize) :: x, y, z
integer :: i
do i=1, nsize
  write(*,*) i, x(i), y(i), z(i)
end do
stop
end program array1
```

- Please replace

```
real, dimension(nsize) :: x, y, z
```

by

```
real, dimension(nsize) :: x=0, y=0, z=0
```

and compile and run the modified program.

Whole array operations (1)

- It is possible to perform array operations in Fortran90.
- The operation is applied on an element-by-element basis.

Statements in codes	Operations for each element
`a+b`	a(i)+b(i)
`a-b`	a(i)-b(i)
`a*b`	a(i)*b(i)
`a/b`	a(i)/b(i)
`a**n`	a(i)**n

Both `a` and `b` are Rank 1 arrays. `n` is a variable.

Whole array operations (2)

EXERCISE 5-2

- Please compile the following program, and run it.

```
program array2
implicit none
integer, parameter:: nsize=3
real, dimension(nsize) :: x=(/1.,2.,3./), y, z
integer i
write(*,*) 'array x:', x
write(*,*) 'array y?'
read(*,*) y
z = x + y
write(*,*) 'array z (=x + y):', z
do i=1, nsize
  z(i) = x(i) + y(i)
end do
write(*,*) 'array z (=x + y):', (z(i),i=1,nsize)
end program array2
```

Whole array operations (3)

EXERCISE 5-3

- Please compile the following program, and run it.

```
program array3
implicit none
integer, parameter:: nsize=10
integer i
real, dimension(nsize) :: x=(/(i,i=1,10)/),
   y=(/(2*i,i=2,20,2)/), z
write(*,*) 'x(i)'
write(*,*) x
write(*,*) 'y(i)'
write(*,*) y
write(*,*) '-x(i)'
write(*,*) -x
write(*,*) 'sqrt(y(i))'
z = sqrt(y)
write(*,*) z
stop
end program array3
```

Whole array operations (4)

EXERCISE 5-4

- Please compile the following program, and run it.

```
program array4
implicit none
integer, parameter:: nsize=3
real, dimension(nsize) :: x
real :: a
write(*,*) 'Coefficient a?'
read(*,*) a
write(*,*) 'array x?'
read(*,*) x
x = x + a
write(*,*) 'array x (=x + a):', x
x = x * a
write(*,*) 'array x (=x * a):', x
stop
end program array4
```

Whole array operations (5)

EXERCISE 5-5

- It is possible to calculate an inner product ($=\Sigma a_i * b_i$) using the function DOT_PRODUCT.

```
program array5
implicit none
integer, parameter:: nsize=3
real, dimension(nsize) :: x=(/1.,2.,3./), y=(/4.,5.,6./)
real :: s
integer :: i
s = dot_product(x,y)
write(*,*) s
s = 0.
do i=1, nsize
  s = s + x(i)*y(i)
end do
write(*,*) s
end program array5
```

- “DOT_PRODUCT” is one of the transformational intrinsic functions.

Subsets of arrays (1)

- It is possible to use a subset of an array (array section) in calculations.
- An array subset is specified with a subscript triplet (vector subscript):

```
subscript_1 : subscript_2 : stride
```

where

`subscript_1` is the first subscript to be included in the subset

`subscript_2` is the last subscript to be included in the subset

`stride` is the increment for the subscript

Subsets of arrays (2)

- EXERCISE 5-6
  Please compile the following program, and run it.

```
program subset_array
implicit none
integer, parameter:: nsize=10
integer :: i
integer, dimension(nsize) :: x=(/(i, i=1, 10)/)
write(*,*) x(:)
write(*,*) x(4:6)
write(*,*) x(3:)
write(*,*) x(:5)
write(*,*) x(2:8:2)
write(*,*) x(6:3:-1)
write(*,*) x(9:3:-2)
write(*,*) x(9:2:-2)
write(*,*) x(::3)
write(*,*) x(10:10)
stop
end program subset_array
```

Rank-2 array

- The following are examples of rank-2 (or two dimensional) arrays:

```
real, dimension(10,10):: a
integer, dimension(0:10, 0:10) :: b
```

- The followings are examples of rank-3 arrays:

```
real, dimension(5,10,3) :: c
integer, dimension(-10:10,-10:10,-5:5):: d
```

Read and *write* elements of rank-2 array

```
program array7
implicit none
integer, parameter:: msize=3, nsize=2
integer :: i, j
real, dimension(msize, nsize) :: x
do i=1, msize
  write(*,*) 'Input ', nsize, 'real numbers for ', 'row', i
  read(*,*) (x(i,j), j=1, nsize)
end do
do i=1, msize
  do j=1, nsize
    write(*,*) 'row', i, 'column', j, x(i,j)
  end do
end do
write(*,'(a20, 2i10)') 'column ', (j, j=1, nsize)
do i=1, msize
   write(*,'(a10, i10, 2f10.2)') 'row ', i, (x(i,j), j=1, nsize)
end do
write(*,*) 'x: ', x
stop
end program array7
```

SUM function

This program calculates $S = \sum_i \sum_j x_{ij}$, $A_i = \sum_j x_{ij}$, $B_j = \sum_i x_{ij}$ for x_{ij} $(i = 1, m;\ j = 1, n)$

```
program array8
implicit none
integer, parameter:: msize=3, nsize=2
integer :: i, j
real, dimension(msize, nsize) :: x
real, dimension(msize) ::        x1
real, dimension(nsize) ::        x2
real :: s
do i=1, msize
  write(*,*) 'Input ', nsize, 'real numbers for ', 'row', i
  read(*,*) (x(i,j), j=1, nsize)
end do
write(*,'(a20, 2i10)') 'column ', (j, j=1, nsize)
do i=1, msize
   write(*,'(a10, i10, 2f10.2)') 'row ', i, (x(i,j), j=1, nsize)
end do
s = sum(x)
write(*,*) 'sum of x_ij', s
x1 = sum(x,dim=2)
x2 = sum(x,dim=1)
write(*,*) 'a_i', x1
write(*,*) 'b_j', x2
stop
end program array8
```

Matrix calculation using rank-2 arrays

Statements in codes	Operations for each element
`A+B`	a(i,j)+b(i,j)
`A-B`	a(i,j)-b(i,j)
`A*B`	a(i,j)*b(i,j)
`A/B`	a(i,j)/b(i,j)
`A**n`	a(i,j)**n
`alpha*A`	alpha*a(i,j)
`Matmul(A,B)`	$\sum_k a_{ik} b_{kj} \, (= c_{ij})$
`Transpose(A)`	a_{ji}

Both `A` and `B` are rank-2 arrays. `alpha` is a variable.

Multiplication of matrices

```
program array9
implicit none
integer, parameter:: msize=3
integer :: i, j
real, dimension(msize, msize) :: x, y, z
do i=1, msize
  read(*,*) (x(i,j), j=1, msize)
end do
do i=1, msize
  read(*,*) (y(i,j), j=1, msize)
end do
z = matmul(x,y)
do i=1, msize
  write(*,*) (z(i,j), j=1, msize)
end do
stop
end program array9
```

```
program array9a
implicit none
integer, parameter:: msize=3
integer :: i, j, k
real, dimension(msize, msize) :: x, y, z
do i=1, msize
  read(*,*) (x(i,j), j=1, msize)
end do
do i=1, msize
  read(*,*) (y(i,j), j=1, msize)
end do
z = 0.
do i=1, msize
  do j=1, msize
    do k=1, msize
      z(i,j) = z(i,j) + x(i,k)*y(k,j)
    end do
  end do
end do
do i=1, msize
  write(*,*) (z(i,j), j=1, msize)
end do
stop
end program array9a
```

Multiplication of matrix and vector

```
program array10
implicit none
integer, parameter:: msize=3
integer :: i, j
real, dimension(msize, msize) :: x
real, dimension(msize) :: y, z
do i=1, msize
  read(*,*) (x(i,j), j=1, msize)
end do
read(*,*) (y(j), j=1, msize)
z = matmul(x,y)
write(*,*) (z(j), j=1, msize)
stop
end program array10
```

```
program array10a
implicit none
integer, parameter:: msize=3
integer :: i, j
real, dimension(msize, msize) :: x
real, dimension(msize) :: y, z
do i=1, msize
  read(*,*) (x(i,j), j=1, msize)
end do
read(*,*) (y(j), j=1, msize)
z = 0
do i=1, msize
  do j=1, msize
      z(i) = z(i) + x(i,j)*y(j)
  end do
end do
write(*,*) (z(j), j=1, msize)
stop
end program array10a
```

Exercise 5-7

- Please compile the programs, `array 7`, `array 8`, `array 9`, and `array 10`, which are given in the previous slides, and run their executable files.

Exercise 5-8

Make a program to calculate the angle for two vectors $\mathbf{x} = (1,2,3)$ and $\mathbf{y} = (4,5,6)$, which is defined by

$$\cos\vartheta = \frac{\mathbf{x}\cdot\mathbf{y}}{|\mathbf{x}||\mathbf{y}|}.$$

Rotation matrix

- Rotation matrix on a plane given below is a typical example of two dimensional array:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Exercise 5-9

a. Make a program to rotate unit vectors (1,0) and (0,1) by an angle θ
b. Make a program to calculate **RR**. Then compare **R** computed for 2θ