FORTRAN 77
Chapter 4

Satish Chandra

www.satish0402.weebly.com

ARRAYS AND SUBSCRIPTED VARIABLES

- A variable can store a single value at a time. There may be cases where we have to deal with a set of values. Using so many variables for storing all these values is a difficult practice.
- In such case we use subscripted variable that refers to an array or a group of quantities by a single name.

SUBSCRIPTED VARIABLES

- A subscript can only be an integer constant, an integer variable or an integer expression. If the subscript is an expression it should be greater than one when evaluated.
- The general form of a subscripted variable is an integer or a real variable name followed by subscripts enclosed within parentheses.
- For example: v(i, j, k) is a subscripted variable, where v is a variable name which may be either integer or real and i , j, k are subscripts.

ARRAYS

- Many scientific computations use vectors and matrices. The data type Fortran uses for representing such objects is the array.
- A one-dimensional array corresponds to a vector, while a two-dimensional array corresponds to a matrix.

One Dimensional Arrays

- The simplest array is the one-dimensional array, which is just a linear sequence of elements stored consecutively in memory. For example, the type declaration

  real a(20)

  declares a as a real array of length 20. That is, a consists of 20 real numbers stored contiguously in memory.
- By convention, Fortran arrays are indexed from 1 and up. Thus the first number in the array is denoted by a(1) and the last by a(20).

One Dimensional Arrays

- However, you may define an arbitrary index range for your arrays using the following syntax:

  real b(0:19), c(-162:237)
- Here, b is exactly similar to a from the previous example, except the index runs from 0 through 19. But c is an array of length 237-(-162)+1 = 400.

One Dimensional Arrays

- The type of an array element can be any of the basic data types. Examples:

```
integer i(10)
logical aa(0:1)
double precision x(100)
```

- Each element of an array can be thought of as a separate variable. You reference the i^{th} element of array a by a(i).

One Dimensional Arrays

- A code segment that stores the 10 first square numbers in the array:

```
      integer i(10), sq(10)
      do 100 i = 1, 10
      sq(i) = i**2
100   continue
```


One Dimensional Arrays

- Reading an array, in the do loop

```
      integer  i(100), x(100)
      do 10 i = 1, 100
      read(*,*) x(i)
10    continue
```

- Alternatively, we use the single statement

```
read(*,*) (x(i), i=1,100)
```

- This is called an **implied do statement**.

Problems 1:

1. Arrange a set of numbers in ascending order.
2. Read a set of marks and write the number of marks above average.

Two-dimensional arrays

- Matrices are very important in linear algebra. Matrices are usually represented by two-dimensional arrays. For example, the declaration

```
real A(3,5)
```

defines a two-dimensional array of 3*5=15 real numbers. It is useful to think of the first index as the row index, and the second as the column index.

Two-dimensional arrays

- Hence we get the graphical picture:

```
(1,1) (1,2)  (1,3) (1,4) (1,5)
(2,1) (2,2) (2,3) (2,4) (2,5)
(3,1) (3,2) (3,3) (3,4) (3,5)
```

Two-dimensional arrays

- Two-dimensional arrays may also have indices in an arbitrary defined range. The general syntax for declarations is:

  name (low_index1 : hi_index1, low_index2 : hi_index2)
- The total size of the array is then

  size = (hi_index1 - low_index1 + 1) * (hi_index2 - low_index2 + 1)

Two-dimensional arrays

- Example 1: Nested do loops

```
      real a(3,3)
      integer i, j
      do 20 j = 1, 3
            do 10 i = 1, 3
            a(i,j) = real(i)/real(j)
10          continue
20    continue
      stop
      end
```


Two-dimensional arrays

- Example 2: Nested do loops

```
      real a(3,3)
      integer i, j
      do 20 j = 1, 3
            do 10 i = 1, 3
            read (*,*) a(i, j)
10          continue
20    continue
      stop
      end
```


Two-dimensional arrays

- Example 3: Implied do statement

```
      real a(3,3)
      integer i, j
      read (*,*) ((a(i, j), i = 1,3), j = 1,3)
      stop
      end
```


Multi-dimensional arrays

- Fortran 77 allows arrays of up to seven dimensions. The syntax and storage format are analogous to the two-dimensional case, so we will not spend time on this.
- The implied do may also be used for reading n-dimensional arrays, where n<=7 .
- The rules for writing read statement with implied do apply without any change to write statements also.

Multi-dimensional arrays

- Example 1: Nested do loops

```
      real a(3,4,5)
      integer i, j, k
      do 30 k = 1, 5
            do 20 j = 1, 4
                  do 10 i = 1, 3
                  read (*,*) a(i, j, k)
10                continue
20          continue
30    continue
      stop
      end
```

Multi-dimensional arrays

- Example 2: Implied do statement

```
real a(3,4,5)
integer i, j, k
read (*,*) (((a(i, j, k), i = 1,3), j = 1,4), k=1,5)
stop
end
```


The dimension statement

- There is an alternate way to declare arrays in Fortran 77. The statements

```
real a, x
dimension x(50)
dimension a(10,20)
```

  are equivalent to

```
real a(10,20), x(50)
```

- To make these variables integer type, we use the statement integer a(10,20), x(50).

The dimension statement

- With the declaration statement dimension x(100), computer reserves 100 memory cells with names x(1), x(2), x(3), …. , x(100).
- The statement dimension is a non-executable statement which provides information to the compiler that x is a subscripted variable with 100 components and each is of type integer.
- This dimension statement is considered old-fashioned style today.

CHAPTER 4

ENDS