CHE 208

ARRAYS: SUBSCRIPTED VARIABLES

Lecture-7

Lubna Ahmed

Problem

Suppose that a program is to be developed to

1. Read numbers obtained by all students in ChE208.
2. Find the average score obtained
3. Print a list of numbers greater than average score
4. Sort the list so that the numbers are is descending order

Solution to the Problem

The program is too long to be efficient

$\longrightarrow$

```
PROGRAM ChE208
REAL NUM1, NUM2…………….., NUM60, ANUM
READ*,NUM1,NUM2……………..,NUM60
ANUM=(NUM1+NUM2+………….+NUM60)/60
PRINT*,'AVERAGE NUMBER=',ANUM
IF (NUM1.GT.ANUM) THEN
        PRINT*,NUM1
ENDIF
IF (NUM2.GT.ANUM) THEN
        PRINT*,NUM2
ENDIF
…………………..
…………………..
IF(NUM60.GT.ANUM) THEN
        PRINT*,NUM60
ENDIF
and now you have to sort the numbers ?!!?
……………………….
END
```

Effective way to solve the problem

Effective way to solve the above problem

- To solve the problem efficiently, we need a data structure to store and organize the entire collection of numbers
- One such data structure is an array. (i.e. when we need to handle a large amount of data in the same way.) To avoid an enormously clumsy program, we can use subscripted variables, or arrays .

Effective way to solve the problem cont'd

- A single array can be used to process much more than one value. Also known as subscripted variable.
- Arrays can be extremely powerful tools. They permit us to apply the same algorithm repeatedly to many different data items with a simple do loop.
- Arrays are very powerful tool for manipulating data in Fortran. We can manipulate and perform calculations with individual elements of arrays one by one, with whole arrays at once, or with various subsets of arrays.
- We are going to discuss one-dimensional array in this lecture.

Introduction to Array

Array

•To store lists of data, all of the same type, use arrays.

- *e.g. lists of exam marks, or the elements of a position vector (x, y, z).*
- A matrix $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ is a two-dimensional array.

•An array is a group of variables, all of the same type, that is referred to by a single name.

•The values in the group occupies consecutive locations in the computer memory.

• An individual value within the array is called a array element . They are distinguished by a subscript or array index.

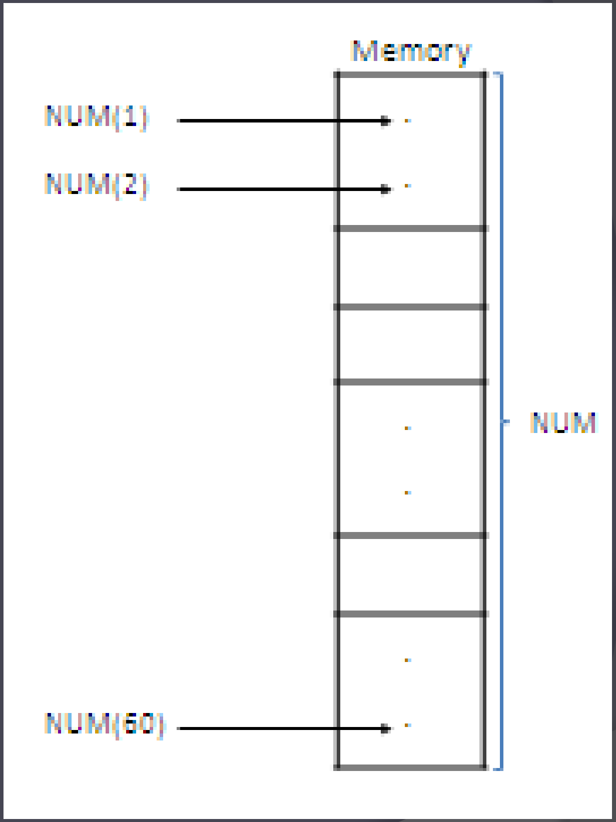
•Subscripts are consecutive integers.

Declaring Arrays

- An array may be of any type: real, integer, logical or character
- Arrays are named in the same way and have the same types as scalar variables.
- Arrays are declared, like variables, in type statements. These give their dimensions (sizes).

Example:

```
REAL, DIMENSION(60) :: NUM
```



Declaring Arrays cont’d

DIMENSION attribute:

- A Fortran 90 program uses the DIMENSION attribute to declare arrays.
- The DIMENSION attribute in the type declaration statement declares the size of the array being defined.

Examples:

1.
```
Real, Dimension(16) :: voltage
```

The elements in array voltage would be addressed as voltage(1), voltage(2), and so forth, up to voltage(16).

2.
```
Character(len=20), Dimension(50)::last_name
```

- Each of the elements in array last_name would be a 20 character –long variable , and the elements would be addressed as last_name(1), last_name(2), and so forth.

Declaring Arrays cont’d

DIMENSION attribute cont’d:

- The DIMENSION attribute requires three components in order to complete an array specification, *rank, shape, and extent.*

1.Rank:

- Arrays may be declared with more than one subscript , so they may be organized into two or more dimensions. These arrays are convenient for representing data that is normally organized into multiple dimensions , such as map information.
- The number of subscripts declared for a given array is called rank of array; i.e. the rank of an array is the number of “indices” or “subscripts.” The maximum rank is 7 (i.e., seven-dimensional).
- In previous examples, both array voltage and array last_name are rank-1 arrays, since they have only one subscript.

Declaring Arrays cont'd

DIMENSION attribute cont'd:

2.Shape: The shape of an array indicates the number of elements in each "dimension." It is the combination of its rank and the extent of the array in each dimension.

3. Extent:

- The number of elements in a given dimension of an array is called the extent of the array in that dimension.
- The rank and shape of an array is represented as $(s_1, s_2, \ldots, s_n)$, where n is the rank of the array and s_i $(1 \leq i \leq n)$ is the number of elements in the i-th dimension.
- Thus two arrays have the same shape if they have the same rank and the extent of the array in each dimension.

Declaring Arrays cont'd

DIMENSION attribute cont'd:

- The size of an array is the total number of elements declared in that array. Therefore, the size of array voltage is 20 and that of array last_name is 50.

Array constants:

- An array constant is an array consisting entirely of constants.
- It is defined by placing the constant values between special delimiters called array constructors.
- For example, the following expression defines an array constant containing five integer elements:

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

Input/output of Arrays

Three ways:

- Use a DO loop containing an input/output statement
- Use only the array name in an input/output statement
- Use an implied DO loop in an input/ output statement

1. USING DO LOOP

```
DO i=1,60
READ*, NUM(i)
END DO
```

Input

```
DO i=1,60
PRINT*, NUM(i)
END DO
```

Output

Input/output of Arrays cont'd

2. USING NAME OF THE ARRAY

```
READ*, NUM
```

```
PRINT*, NUM
```

3. USING IMPLIED DO LOOP

(i/o-list, control variable = initial-value, limit)
or
(i/o-list, control variable = initial-value, limit, step size)

```
READ*, NUM(i), i=1,50
```

```
PRINT*, NUM(i), i=1,50
```

Initializing Arrays with Data

- One must initialize the values in an array before using them, just as with ordinary variables. If an array is not initialized , the contents of the array elements are undefined.
- All or part of an array may be initialized in a DATA statement. There are a number of possibilities, e.g.

```
REAL, DIMENSION(10) :: A, B
DATA A / 5*0, 5*1 /                     ! first 5 zero, last 5 one
DATA B(1:5) / 4, 0, 5, 2, -1 /          ! section 1:5 only
```

Initializing Arrays with Assignment statement

- Initial values may be assigned to the array by using assignment statements, either element by element in a Do Loop or altogether with an array constructor
- Example: The following Do loop will initialize all of the elements of array array1 to 0.0 one element at a time

```
Real, Dimension (10):: array1
Do i=1,10
    array1(i)= 0.0
End Do
```

The following assignment statement accomplishes the same function all at once using an array constructor:

```
Real, Dimension (10):: arrray1
Array1= (/0.,0.,0.,0.,0.,0.,0.,0.,0.,0./)
```

Initializing Arrays in type declaration statement

- To initialize an array in a type declaration statement, we use an array constructor to declare its initial values in that statement.
- Example: The following statement declares a five element integer array array2, and initializes the elements of array2 to 1,2,3,4,and 5.

```
Integer, Dimension (5) :: array2=(/1,2,3,4,5/)
```

Array Expression

Array Expressions

The following are examples of array expressions:

```
X + Y               ! result has elements X(I) + Y(I)
X * Y               ! result has elements X(I) * Y(I)
X * 3               ! result has elements X(I) * 3
X * SQRT( Y )       ! result has elements X(I) * SQRT( Y(I) )
X == Y              ! result has elements .TRUE. if X(I) == Y(I),
                              .FALSE. otherwise
```

Allocate statement

- A variable is specified as dynamic with the ALLOCATABLE attribute. In particular, a one dimensional array may be specified as:

  REAL, DIMENSION(:), ALLOCATABLE :: X

- Although its rank is specified, it has no size until it appears in an ALLOCATE statement, such as

  ALLOCATE(X(N))

- When it is no longer needed, it may be deallocated:

  DEALLOCATE(X)

Solution of discussed problem

```
PROGRAM NUMBER
IMPLICIT NONE
INTEGER I, NOS
REAL:: SUM=0, ANUM=0
REAL, DIMENSION(100) :: NUM
PRINT*,'HOW MANY STUDENTS‘
READ*,NOS
PRINT*,'ENTER NUMBERS OBTAINED'
READ*,(NUM(I),I=1,NOS)
DO I = 1,NOS
SUM=SUM+NUM(I)
END DO
ANUM=SUM/NOS
PRINT*, ‘AVERAGE NUMBER OBTAINED’, ’=’, ANUM
DO I = 1,NOS
IF (NUM(I).GT. ANUM) THEN
PRINT*, NUM(I)
END IF
END DO
END PROGRAM NUMBER
```

Summary

- Arrays are useful for representing and processing large amounts of data.
- An array is a collection of subscripted variables with the same name.
- Members of an array are called elements.
- The number of elements in an array is its size. The size may be zero.
- Upper and lower bounds of array subscripts are specified with the DIMENSION attribute in the array type specification statement.
- An array subscript may be any valid numeric expression (rounded if necessary).
- The number of dimensions of an array is its rank. An array may have up to seven dimensions.
- A dynamic variable (which may be an array) is specified with the ALLOCATABLE attribute and may be allocated memory while a program is running. The memory may be deallocated later.

End of Lecture…

Class Assignment

1. Write a program to calculate the squares of integers from 1 to 10, using assignment statements/ type specification statements to initialize the values in array number.

Class Assignment

2. Write a Program that reads a set of measurements and calculates the mean and the standard deviation of the input data set, the data set can be positive, negative, or zero.