Arrays

Many applications require the processing of multiple data items that have common characteristics. In such a situation it is convenient to place such data items in an Array.

An array is a collection of similar data items that are sotred under a common name. A value in an array is identified by index or subscript enclosed in square brackets with array name.

The individual data items can be integers, floating point numbers, characters and so, on, but they must be the same type and same storage class.

Each array element is referred by specifying the array name with subscript, each subscript enclosed in square brackets and each subscript must be a non-negative integer.

Thus 'n' elements array 'Q' and the elements are

Q[0], Q[l], Q[2], Q[3], …, Q[n-1]

The arrays can be used to represent not only simple list of value but also task of data items in two and three or more dimensions.

Arrays can be classified into:

- One - dimensional arrays.
- Two - dimensional arrays.
- Multi - dimensional arrays.

One - dimensional arrays:

The collection of data can be stored under one variable I statement as well as loop statements name using only one subscript, such a variable is called the one-dimensional array.

Array declaration:

Arrays are declared in the same manner as ordinary variables except that each array name must have the size of the array.

Features of Arrays:

An array is a derived data type. It is used to represent a collection of elements of the same data type.

The elements can be accessed with base address (index) and the subscripts define the position of the element.

In array the elements are stored in continuous memory location. The starting memory location is represented by the array name and it is known as the base address of the array.

It is easier to refer the array elements by simply incrementing the value of the subscript

.

Array Initialization:

The values can be initialized to an array, when they are declared like ordinary variables, otherwise they hold garbage values.

The array can be initialized in the following two ways.

i) At compile time

ii) At run time

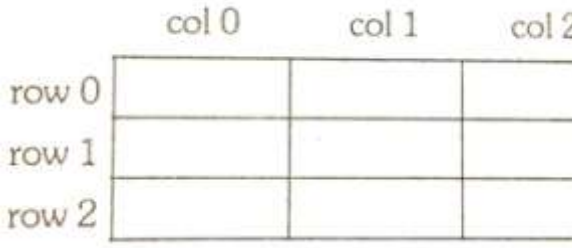
Entering data in to the array:

The data can be entered into an array using the input statement as well as loop statements.

Two dimensional array:

Two dimensional arrays are used in situation where a table of values need to be stored in an array.

These can be defined in the same fashion as in one dimensional arrays, except a separate pair of square brackets are required for each subscript.
Two pairs of square brackets are required for two dimensional array and three pairs required for three dimensional arrays and so on.
Two dimensional arrays are stored in a row-column matrix, where the left index indicates the row and the right indicates the column.
Ex: int a[3][3];
a is the array name, and it reserves 3 rows and 3 columns memory as shown below:



The individual elements are identified by index or subscript of an array, from the above example.
a[0][0] a[0][1] a[0][2]
a[1][0] a[1][1] a[1][2]
a[2][0] a[2][1] a[2][2]

Initialising a two dimensional array:
Like one dimensional array, the values can be initialised to the two dimensional arrays at the time of declaration.

Example:
int stud [4] [2] =
{
{ 9814,90 },
{ 9815, 95 },
{ 9816, 82 }
{ 9817, 88 }
}:
or
intstud [4] [2] = { 9814, 90, 9815, 95, 9816, 82, 9817, 88} ;
intstud [ ] [2] = { 9814, 90, 9815, 95, 9816, 82, 9817, 88 } ;
Remember that while initialising an array it is necessary to mention the second dimension, whereas the first dimension is optional.

Arrays to functions:
An entire array can be transferred to a function as a parameter. To transfer an array to a function, the array name is enough without subscripts as an actual parameters within the function call.
The corresponding formal parameters are written fashion, and must be declared as an array in parameters declaration.

Syntax:
void fun 1 (int , int [ ]) ;
main( )
{
int a[10], n;
……
……
fun 1 (n, a)
……
……
}
void fun 1 (x, ar [])
{
int x, ar [10] ;

Array of Characters (string):
A string is a collection of characters. A string constant is a one dimensional array of

characters terminated by a null (‘ \ o ’) character.

Example:

char name [ ] = {‘A’ , ‘N’ , T, ‘S’, ‘\ o’};

where ‘\ o’ is a null character and specifies end of the string.

Terminating the string with null ‘\ o’ is important, because it is the only way the functions that work with a string can know where the string ends.

A	N	I	S	\o

The last character is called ‘ \ o 1 null character, it looks like two characters but it is actually one character. Each character in the array occupies one byte memory and the last character is always ‘\ o’

C provides a shortcut method of initialise strings.

Example:

char name [ ] = “ANIS”1 ;

Note that in this declaration \ o is not necessary ‘C’ insert ‘ \ o ’ automatically at the end of the string.

Multidimensional Arrays:

Similarly, like one and two dimensional arrays, c language allows multidimensional arrays. The dimension with three or more called multi dimensional arrays.

Example:

int a[3] [3] [3];

float table b[4][4][4][4] ;

where,

‘a’ is the three dimensional array declared as integer type and can hold 27 elements.

‘b’ is the four dimensional array declared as float type and can hold 256 elements.

Arrays of more than three dimenions are not used often. Multidimensional arrays are slower than the single dimension in execution.