



Arrays in C

Rab Nawaz Jadoon

Assistant Professor

COMSATS IIT, Abbottabad

Pakistan

Department of Computer Science

DCS

COMSATS Institute of
Information Technology

Introduction to Computer Programming (ICP)

- C language provides a capability that enables the user to design a set of similar data types, called array.
 - For understanding the arrays properly, let us consider the following program:

```
main( )  
{  
    int x ;  
    x = 5 ;  
    x = 10 ;  
    printf ( "\nx = %d", x ) ;  
}
```

No doubt, this program will print the value of x as 10. Why so? Because when a value 10 is assigned to x, the earlier value of x, i.e. 5, is lost.

Thus, ordinary variables are capable of holding only one value at a time. However, there are situations in which we would want to store more than one value at a time in a single variable.

Example

- For example, suppose we wish to arrange the percentage marks obtained by 100 students in ascending order. In such a case we have two options to store these marks in memory:
 - Construct 100 variables to store percentage marks obtained by 100 different students, i.e. each variable containing one student's marks.
 - Construct one variable (called array or subscripted variable) capable of storing or holding all the hundred values.

Array Definition

- **array**
- **An array is a collective name given to a group of 'similar quantities'.**
- **OR**
 - Consecutive memory locations having the same name and same data type.

Note

These similar elements could be all ints, or all floats, or all chars, etc

Important points regarding array

- Usually, the array of characters is called a 'string',
 - whereas an array of ints or floats is called simply an array.
 - Remember that all elements of any given array must be of the same type. i.e. we cannot have an array of 10 numbers, of which 5 are ints and 5 are floats.

Array Declaration

- Like other variables an array needs to be declared so that the compiler will know what kind of an array and how large an array we want.
- For example `int marks[30];`
 - The bracket ([]) tells the compiler that we are dealing with an array.
 - Int specifies the type of the variable, just as it does with ordinary variables and the word marks specifies the name of the variable.
 - The [30] however is new. The number 30 tells how many elements of the type int will be in our array.

Accessing Elements of the array

- After array declaration, how individual elements in the array can be referred.
 - This is done with subscript, the number in the brackets following the array name.
 - This number specifies the element's position in the array.
 - All the array elements are numbered, starting with 0. Thus, marks[2] is not the second element of the array, but the third.

Entering data to an Array

- Here is the section of code that places data into an array:

```
for ( i = 0 ; i <= 29 ; i++ )  
{  
    printf ( "\nEnter marks " ) ;  
    scanf ( "%d", &marks[i] ) ;  
}
```


Reading Data from an Array

- We are getting the actual values of the arrays from its index, like,

```
for(int i=0;i<30;i++)  
{  
    printf("marks[%d] = %d", i, marks[i]);  
}
```

Simple program

- Let us try to write a program to find average marks obtained by a class of 30 students in a test:

```
//Simple array to calculate the AVG of Quiz of ten students
#include<stdio.h>
#include<conio.h>
main( )
{
    int avg, sum = 0 ;
    int i ;
    int marks[10] ; /*array declaration */
    for ( i = 0 ; i <10 ; i++ )
    {
        printf ( "\nEnter marks " ) ;
        scanf ( "%d", &marks[i] ) ; /* store data in array */
    }
    for ( i = 0 ; i <10 ; i++ )
        sum = sum + marks[i] ; /* read data from an array*/
    printf("\nThe sum is %d", sum);
    avg = sum / 10 ;
    printf ( "\nAverage marks = %d", avg ) ;
    getch();
}
```

Important points

An array is a collection of similar elements.

The first element in the array is numbered 0, so the last element is 1 less than the size of the array.

An array is also known as a subscripted variable.

Before using an array its type and dimension must be declared.

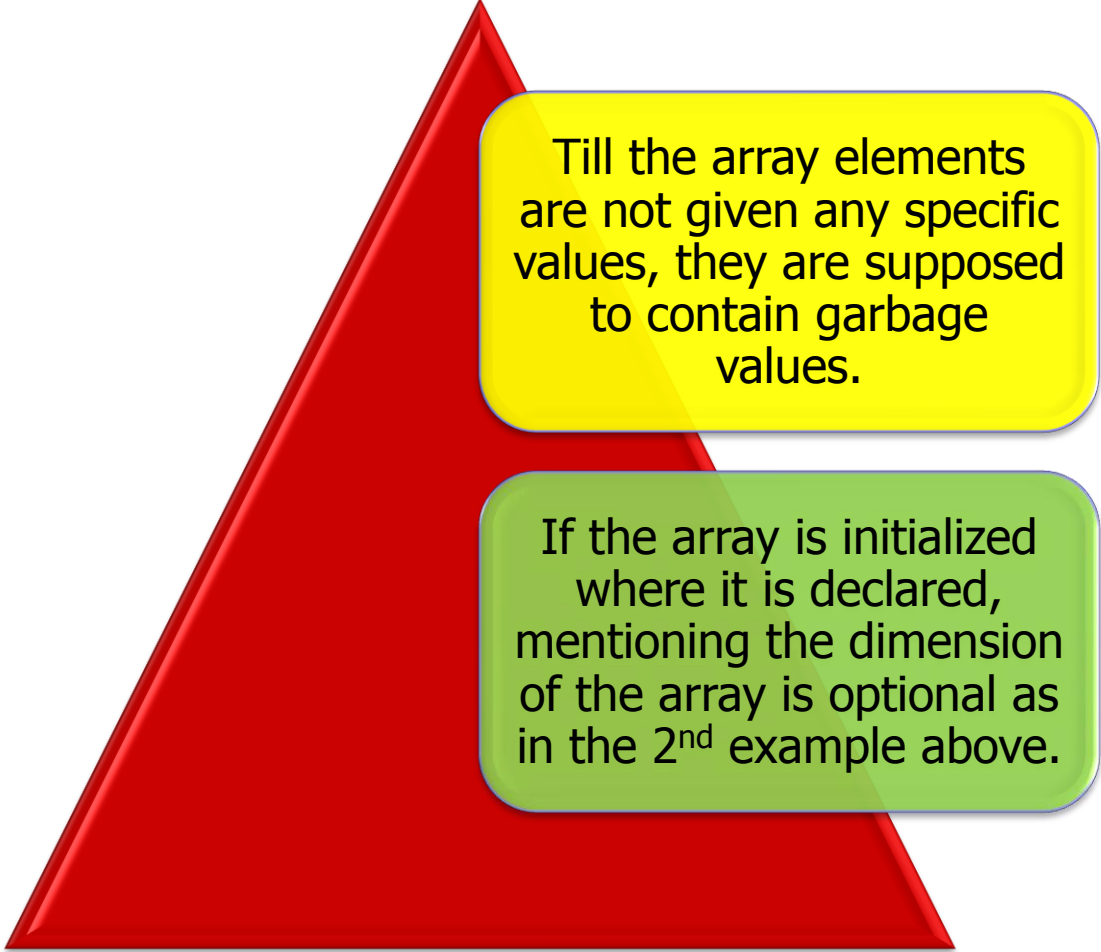
However big an array its elements are always stored in contiguous memory locations.

Array Initialization

- So far we have used arrays that did not have any values in them to begin with. We managed to store values at run time,
 - Let us now see how to initialize an array while declaring it.
 - Following are a few examples that demonstrate this.

```
int num[6] = { 2, 4, 12, 5, 45, 5 } ;  
int n[ ] = { 2, 4, 12, 5, 45, 5 } ;  
float press[ ] = { 12.3, 34.2 -23.4, -11.3 } ;
```

Important Points



Till the array elements are not given any specific values, they are supposed to contain garbage values.

If the array is initialized where it is declared, mentioning the dimension of the array is optional as in the 2nd example above.

Arrays in Memory

- Consider the following array declaration:
`int arr[8];`
 - 16 bytes get immediately reserved in memory, 2 bytes each for the 8 integers.
 - Since the array is not being initialized, all eight values present in it would be garbage values.
 - all the array elements would always be present in contiguous memory locations.

12	34	66	-45	23	346	77	90
65508	65510	65512	65514	65516	65518	65520	65522

