**Course Name:** O Level (2nd Sem B2 and B3 Batch) **Subject:** C Language
**Topic:** Array **Date:** 23-April-2020

## What is an Array?

Array is a collection or group of similar data type elements stored in contiguous memory location. The individual data items can be characters, integers, floating points numbers and so on. Here contiguous memory allocation means array occupies contiguous bytes as needed in the memory.

**Or**

An array is a collection of similar elements. These similar elements could be all **int**s, or all **float**s, or all **char**s, etc. Usually, the array of characters is called a 'string', whereas an array of **int**s or **float**s 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 **int**s and 5 are **float**s.

## Types of an Array:

- Single / One Dimensional Array
- Multidimensional Array

**Single / One Dimensional Array :**

The array which is used to represent and store data in a linear form is called as 'single or one dimensional array'. Has a unique identifier for each element, called as a subscript or an index.

**Syntax:** <data-type> <array_name> [size];
Example: int a[5];

| A[index no.] | A[0] | A[1] | A[2] | A[n] |
|---|---|---|---|---|
| element | 5 | 7 | 2 | 12 |
| Memory address | 1000 | 1002 | 1004 | n |

**Memory allocation for Single / One dimensional array**

The array size could also be specified using a symbolic constant:

```
#define ARRAY_SIZE 25
int a[ARRAY_SIZE];
```

We can initialize array elements in declaration statements; e.g.,

```
int x[5] = {1, 2, 3, 4, 5};
int x[ ] = {2, 4, 6, 8};          /*This array has 4 elements*/
```

We cannot specify more initial values than there are array elements, but we can specify fewer initial values, and the remaining elements will be initialized to 0. e.g.,

```
int y[10] = { 2 };
double x[250 ] = { 0 };
```

Given that the array x is declared as:

```
int x[250];
```

To initialize a large array to a nonzero value - say 5, we can use a loop. e.g.,

```
for (k=0; k<=249; k++)
        x[k] = 5;
```

$\uparrow$

**k is a subscript**

Note: when referencing a particular element of an array use square brackets, not parenthesis or curly braces.

A subscript value for an array element can be specified as any ***integer*** expression.

**A Simple Program Using Array**

Write a program to find average marks obtained by a class of 30 students in a test.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int marks[30], i, avg, sum= 0 ;
clrscr() ;
for ( i = 0 ; i <= 29 ; i++ )
{
printf ( "\nEnter marks " ) ;
scanf ( "%d", &marks[i] ) ;                /* store data in array */
}
for ( i = 0 ; i <= 29 ; i++ )
sum = sum + marks[i] ;                     /* read data from an array*/
avg = sum / 30 ;
printf ( "\nAverage marks = %d", avg ) ;
getch ();
}
```

In the above example **marks** specifies the name of the variable. The number 30 tells how many elements of the type **int** will be in our array. This number is often called the 'dimension' of the array. The bracket ( [ ] ) tells the compiler that we are dealing with an array.

The **for** loop causes the process of asking for and receiving a student's marks from the user to be repeated 30 times. The first time through the loop, **i** has a value 0, so the **scanf( )** function will cause the value typed to be stored in the array element **marks[0],** the first element of the array. This process will be repeated until **i** becomes 29. This is last time through the loop, which is a good thing, because there is no array element like **marks[30]**.

In **scanf( )** function, we have used the "address of" operator (&) on the element **marks[i]** of the array, just as we have used it earlier on other variables **(&rate**, for example). In so doing, we are passing the address of this particular array element to the **scanf( )** function, rather than its value; which is what **scanf( )** requires.

**Things to remember about an arrays:**

- 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.

**Operation performed on Array**

There are number of operations that can be performed on arrays. These operations include:

- Transversal
- Insertion
- Search
- Deletion
- Merging
- Sorting

- Selection Sort
- Bubble Sort
- Insertion Sort

**Example of traversal an array.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
int  x[10], i;
for(i=0;i<10;i++)
{
printf("Enter the elements in array: \t");
scanf("%d", &x[i]);
}
printf("The Traverse of array is:\n");
for(i=0;i<10;i++)
printf("%d\n", x[i]);
getch();
}
```

**<u>Try Yourself:</u>**

1. What is an array?
2. Write all the types of initialization of array.
3. Write some advantages of array.