## Array:

An array is defined as the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc. It also has the capability to store the collection of derived data types, such as pointers, structure, etc. The array is the simplest data structure where each data element can be randomly accessed by using its index number.

C array is beneficial if you have to store similar elements. For example, if we want to store the marks of a student in 6 subjects, then we don't need to define different variables for the marks in the different subject. Instead of that, we can define an array which can store the marks in each subject at the contiguous memory locations.

By using the array, we can access the elements easily. Only a few lines of code are required to access the elements of the array.

## Properties of Array:

The array contains the following properties.

- Each element of an array is of same data type and carries the same size, i.e., int = 4 bytes.
- Elements of the array are stored at contiguous memory locations where the first element is stored at the smallest memory location.
- Elements of the array can be randomly accessed since we can calculate the address of each element of the array with the given base address and the size of the data element.

## Advantage of C Array:

1) Code Optimization: Less code to the access the data.

2) Ease of traversing: By using the for loop, we can retrieve the elements of an array easily.

3) Ease of sorting: To sort the elements of the array, we need a few lines of code only.

4) Random Access: We can access any element randomly using the array.

**Disadvantage of C Array:**

1) Fixed Size: Whatever size, we define at the time of declaration of the array, we can't exceed the limit. So, it doesn't grow the size dynamically like LinkedList which we will learn later.

**One Dimensional array in C:**

**Declaration of C Array:**

data_type array_name[array_size];

example to declare the array.

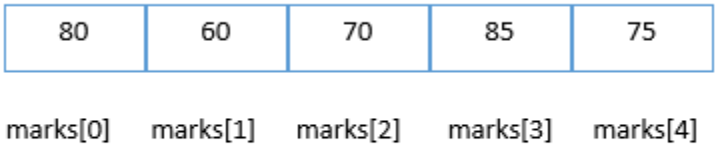
int marks[5];

Here, int is the data_type, marks are the array_name, and 5 is the array_size.

**Initialization of C Array:**

The simplest way to initialize an array is by using the index of each element. We can initialize each element of the array by using the index. Consider the following example.

```
marks[0]=80;          //initialization of array
marks[1]=60;
marks[2]=70;
marks[3]=85;
marks[4]=75;
```




**Initialization of Array**

**Array example:**

```
#include<stdio.h>
int main(){
int i=0;
int marks[5];//declaration of array
marks[0]=80;//initialization of array
marks[1]=60;
marks[2]=70;
marks[3]=85;
marks[4]=75;
//traversal of array
for(i=0;i<5;i++){
printf("%d \n",marks[i]);
}//end of for loop
return 0;
}
```

**Output:**

```
80
60
70
85
75
```

**Array Declaration with Initialization:**

We can initialize the c array at the time of declaration.

```
int marks[5]={20,30,40,50,60};
```

In such case, there is no requirement to define the size. So it may also be written as the following code.

```
int marks[]={20,30,40,50,60};
```

**Example:**

```
#include<stdio.h>
int main(){
int i=0;
int marks[5]={20,30,40,50,60};//declaration and initialization of array
for(i=0;i<5;i++){
printf("%d \n",marks[i]);   //traversal of array
}
return 0;
}
```

**Output**

```
20
30
40
50
60
```

**Two-Dimensional Array in C:**

The two-dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns. However, 2D arrays are created to implement a relational database lookalike data structure. It provides ease of holding the bulk of data at once which can be passed to any number of functions wherever required.

**Declaration of two-dimensional Array**:

The syntax to declare the 2D array:
data_type array_name[rows][columns];
int twodimen [4][3];

Here, 4 is the number of rows, and 3 is the number of columns.

**Initialization of 2D Array**:

In the 1D array, we don't need to specify the size of the array if the declaration and initialization are being done simultaneously. However, this will not work with 2D arrays. We will have to define at least the second dimension of the array. The two-dimensional array can be declared and defined in the following way.
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};

**Two-dimensional array example:**

```
#include<stdio.h>
int main(){
int i=0,j=0;
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
//traversing 2D array
for(i=0;i<4;i++){
for(j=0;j<3;j++){
printf("arr[%d] [%d] = %d \n",i,j,arr[i][j]);
 }//end of j
}//end of i
return 0;
}
```

**Output:**

```
arr[0][0] = 1
arr[0][1] = 2
arr[0][2] = 3
arr[1][0] = 2
arr[1][1] = 3
arr[1][2] = 4
arr[2][0] = 3
arr[2][1] = 4
arr[2][2] = 5
arr[3][0] = 4
arr[3][1] = 5
arr[3][2] = 6
```