# C Array

**Array** in C language is a *collection* or *group* of elements (data). All the elements of c array are *homogeneous* (similar). It has contiguous memory location.

C array is beneficial if you have to store similar elements. Suppose you have to store marks of 50 students, one way to do this is allotting 50 variables. So it will be typical and hard to manage. For example we can not access the value of these variables with only 1 or 2 lines of code.

Another way to do this is array. By using array, we can access the elements easily. Only few lines of code is required to access the elements of array.

#### Advantage of C Array

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

**2) Easy to traverse data**: By using the for loop, we can retrieve the elements of an array easily.

**3) Easy to sort data**: To sort the elements of 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 array, we can't exceed the limit. So, it doesn't grow the size dynamically like LinkedList which we will learn later.

## **Declaration of C Array**

We can declare an array in the c language in the following way.

1. data\_type array\_name[array\_size];

Now, let us see the example to declare array.

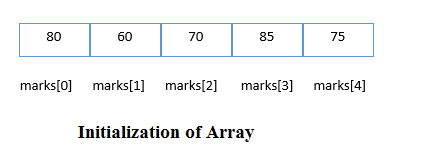
1. **int** marks[5];

Here, int is the *data\_type*, marks is the *array\_name* and 5 is the *array\_size*.

## **Initialization of C Array**

A simple way to initialize array is by index. Notice that **array index starts from 0** and ends with [SIZE - 1].

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



## **C array example**

1. #include <stdio.h>
2. #include <conio.h>
3. **void** main(){
4. **int** i=0;
5. **int** marks[5];//declaration of array
6. clrscr();
8. marks[0]=80;//initialization of array
9. marks[1]=60;
10. marks[2]=70;
11. marks[3]=85;
12. marks[4]=75;
14. //traversal of array
15. **for**(i=0;i<5;i++){
16. printf("%d \n",marks[i]);
17. }//end of for loop
19. getch();
20. }

#### Output

80

60

70

85

75

## **C Array: Declaration with Initialization**

We can initialize the c array at the time of declaration. Let's see the code.

1. **int** marks[5]={20,30,40,50,60};

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

1. **int** marks[]={20,30,40,50,60};

Let's see the full program to declare and initialize the array in C.

1. #include <stdio.h>
2. #include <conio.h>
3. **void** main(){
4. **int** i=0;
5. **int** marks[5]={20,30,40,50,60};//declaration and initialization of array
6. clrscr();
8. //traversal of array
9. **for**(i=0;i<5;i++){
10. printf("%d \n",marks[i]);
11. }
13. getch();
14. }

#### Output

20

30

40

50

60

# Two Dimensional Array in C

The two dimensional array in C language is represented in the form of rows and columns, also known as matrix. It is also known as *array of arrays*or *list of arrays*.

The two dimensional, three dimensional or other dimensional arrays are also known as *multidimensional* arrays.

## **Declaration of two dimensional Array in C**

We can declare an array in the c language in the following way.

1. data\_type array\_name[size1][size2];

A simple example to declare two dimensional array is given below.

1. **int** twodimen[4][3];

Here, 4 is the *row* number and 3 is the *column* number.

## **Initialization of 2D Array in C**

A way to initialize the two dimensional array at the time of declaration is given below.

1. **int** arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};

## **Two dimensional array example in C**

1. #include <stdio.h>
2. #include <conio.h>
3. **void** main(){
4. **int** i=0,j=0;
5. **int** arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
6. clrscr();
8. //traversing 2D array
9. **for**(i=0;i<4;i++){
10. **for**(j=0;j<3;j++){
11. printf("arr[%d] [%d] = %d \n",i,j,arr[i][j]);
12. }//end of j
13. }//end of i
15. getch();
16. }

#### 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

# Passing Array to Function in C

To reuse the array operation, we can create functions that receives array as argument. To pass array in function, we need to write the array name only in the function call.

1. functionname(arrayname);//passing array

There are 3 ways to declare function that receives array as argument.

**First way:**

1. return\_type function(type arrayname[])

Declaring blank subscript notation [] is the widely used technique.

**Second way:**

1. return\_type function(type arrayname[SIZE])

Optionally, we can define size in subscript notation [].

**Third way:**

1. return\_type function(type \*arrayname)

You can also use the concept of pointer. In pointer chapter, we will learn about it.

#### C language passing array to function example

1. #include <stdio.h>
2. #include <conio.h>
3. **int** minarray(**int** arr[],**int** size){
4. **int** min=arr[0];
5. **int** i=0;
6. **for**(i=1;i<size;i++){
7. **if**(min>arr[i]){
8. min=arr[i];
9. }
10. }//end of for
11. **return** min;
12. }//end of function
14. **void** main(){
15. **int** i=0,min=0;
16. **int** numbers[]={4,5,7,3,8,9};//declaration of array
17. clrscr();
19. min=minarray(numbers,6);//passing array with size
20. printf("minimum number is %d \n",min);
22. getch();
23. }

#### Output

minimum number is 3