***Array***

# **C 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.

## 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 the array.

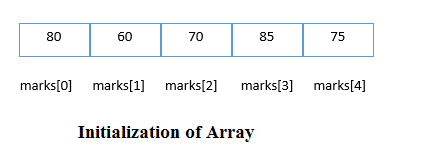
1. **int** marks[5];

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.

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. **int** main(){
3. **int** i=0;
4. **int** marks[5];//declaration of array
5. marks[0]=80;//initialization of array
6. marks[1]=60;
7. marks[2]=70;
8. marks[3]=85;
9. marks[4]=75;
10. //traversal of array
11. **for**(i=0;i<5;i++){
12. printf("%d \n",marks[i]);
13. }//end of for loop
14. **return** 0;
15. }

**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 the size**. So it may also be written as the following code.

1. **int** marks[]={20,30,40,50,60};

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

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

**Output**

20

30

40

50

60

## C Array Example: Sorting an array

In the following program, we are using bubble sort method to sort the array in ascending order.

1. #include<stdio.h>
2. **void** main ()
3. {
4. **int** i, j,temp;
5. **int** a[10] = { 10, 9, 7, 101, 23, 44, 12, 78, 34, 23};
6. **for**(i = 0; i<10; i++)
7. {
8. **for**(j = i+1; j<10; j++)
9. {
10. **if**(a[j] > a[i])
11. {
12. temp = a[i];
13. a[i] = a[j];
14. a[j] = temp;
15. }
16. }
17. }
18. printf("Printing Sorted Element List ...\n");
19. **for**(i = 0; i<10; i++)
20. {
21. printf("%d\n",a[i]);
22. }
23. }

## Program to print the largest and second largest element of the array.

1. #include<stdio.h>
2. **void** main ()
3. {
4. **int** arr[100],i,n,largest,sec\_largest;
5. printf("Enter the size of the array?");
6. scanf("%d",&n);
7. printf("Enter the elements of the array?");
8. **for**(i = 0; i<n; i++)
9. {
10. scanf("%d",&arr[i]);
11. }
12. largest = arr[0];
13. sec\_largest = arr[1];
14. **for**(i=0;i<n;i++)
15. {
16. **if**(arr[i]>largest)
17. {
18. sec\_largest = largest;
19. largest = arr[i];
20. }
21. **else** **if** (arr[i]>sec\_largest && arr[i]!=largest)
22. {
23. sec\_largest=arr[i];
24. }
25. }
26. printf("largest = %d, second largest = %d",largest,sec\_largest);
28. }

*Array* is a data structure that hold **finite sequential collection** of **homogeneous data**.

To make it simple let's break the words.

* **Array is a collection** - Array is a container that can hold a collection of data.
* **Array is finite** - The collection of data in array is always finite, which is determined prior to its use.
* **Array is sequential** - Array stores collection of data sequentially in memory.
* **Array contains homogeneous data** - The collection of data in array must share a same [data type](https://codeforwin.org/2017/08/data-types-in-c-programming.html).

We can divide arrays in two categories.

1. One-dimensional array (Or single-dimensional array)
2. [Multi-dimensional array](https://codeforwin.org/2017/10/multi-dimensional-array-c-declare-initialize-access.html)

**Why we need arrays?**

Let us understand the significance of arrays through an example.

Suppose, I asked you to write a program to input 1000 students marks from user. Finally [print average of their marks](https://codeforwin.org/2015/05/c-program-to-calculate-total-average-percentage-of-five-subjects.html).

To solve the problem you will declare 1000 integer variable to input marks. Call [I/O functions to input](https://codeforwin.org/2017/08/basic-input-output-c.html) marks in 1000 variables and finally find the average.

Think for a while how tedious will be to code if solved using above approach. Declare 1000 variables, take input in all variables, then find average and finally print its average. The above increases length of code, complexity and degrades performance. If you don't believe, try to code solution using above approach.

To solve above problem efficiently we use arrays. Arrays are good at handling collection of data (collection of 1000 student marks). Mind that in programming, we will always use some data structure (array in our case) to handle a collection of data efficiently.

**How to use arrays?**

Array and variable representation in memory(RAM).

Array in memory is stored as a continuous sequence of bytes. Like variables we give name to an array. However unlike variables, arrays are multi-valued they contain multiple values. Hence you cannot access specific array element directly.

For example, you can write sum = 432; to access sum. But, you cannot access specific array element directly by using array variable name. You cannot write marks to access 4th student marks.

In array, we use an integer value called **index** to refer at any element of array. Array index starts from 0 and goes till N - 1 (where N is size of the array). In above case array index ranges from 0 to 4.

To access individual array element we use array variable name with index enclosed within square brackets [ and ]. To access first element of marks array, we use marks[0]. Similarly to access third element we use marks[2].

**How to declare an array?**

**Syntax to declare an array.**

data\_type array\_name[SIZE];

* data\_type is a [valid C data type](https://codeforwin.org/2017/08/list-data-types-c-programming.html) that must be common to all array elements.
* array\_name is name given to array and must be a [valid C identifier](https://codeforwin.org/2017/08/keywords-identifiers-c.html).
* SIZE is a [constant value](https://codeforwin.org/2017/08/constants-c-programming.html) that defines array maximum capacity.

**Example to declare an array**

int marks[5];

**How to initialize an array?**

There are two ways to initialize an array.

1. [Static array initialization](https://codeforwin.org/2017/10/c-arrays-declare-initialize-access.html#static-initialization) - Initializes all elements of array during its declaration.
2. [Dynamic array initialization](https://codeforwin.org/2017/10/c-arrays-declare-initialize-access.html#dynamic-initialization) - The declared array is initialized some time later during execution of program.

**Static initialization of array**

We define value of all array elements within a pair of curly braces { and } during its declaration. Values are separated using comma , and must be of same type.

**Example of static array initialization**

int marks[5] = {90, 86, 89, 76, 91};

**Note:** Size of array is optional when declaring and initializing array at once. The C compiler automatically determines array size using number of array elements. Hence, you can write above array initialization as.

int marks[] = {90, 86, 89, 76, 91};

**Dynamic initialization of array**

You can assign values to an array element dynamically during execution of program. First declare array with a fixed size. Then use the following syntax to assign values to an element dynamically.

array\_name[index] = some\_value;

**Example to initialize an array dynamically**

marks[0] = 90; // Assigns 90 to first element of marks array

marks[1] = 86; // Assigns 86 to second element of marks array

...

...

...

marks[4] = 91; // Assigns 91 to fifth element of marks array

Instead of hard-coding marks values, you can ask user to input values to array using scanf() function.

scanf("%d", &marks[0]); // Input an integer from user and assigns to first element of marks

scanf("%d", &marks[4]); // Input an integer from user and assigns to fifth element of marks

The array index is an integer value, so instead of hard-coding you can wrap array input code inside a [loop](https://codeforwin.org/2017/08/for-loop-in-c-programming.html).

int index;

// Run a loop from 0 to 4

for(index = 0; index < 5; index++)

{

// Replace the hard-coded index with 'index' variable

scanf("%d", &marks[index]);

}

The above code will run 5 times from 0 to 4. In each iteration it ask user to input an integer and stores it in successive elements of marks array.

**Example program to implement one-dimensional array**

Let us write a C program to declare an array capable of storing 10 student marks. Input marks of all 10 students and find their average.

/\*\*

\* C program to find average of marks using array

\*/

#include <stdio.h>

#define SIZE 10 // Size of the array

int main()

{

int marks[SIZE]; // Declare an array of size 10

int index, sum;

float avg;

printf("Enter marks of %d students: ", SIZE);

/\* Input marks of all students in marks array \*/

for(index=0; index<SIZE; index++)

{

scanf("%d", &marks[index]);

}

/\* Find sum of all marks \*/

sum = 0;

for(index=0; index<SIZE; index++)

{

sum = sum + marks[index];

}

/\* Calculate average of marks\*/

avg = (float) sum / SIZE;

/\* Print the average marks \*/

printf("Average marks = %f", avg);

return 0;

}

**Output -**

Enter marks of 10 students: 90 86 89 76 91 95 80 77 82 93

Average marks = 85.900002

**Array best practices**

* Arrays are fixed size, hence always be cautious while accessing arrays. Accessing an element that does not exists is undefined. You may get a valid value or the program may crash.For example, consider the below program.
* int array[5]; // Declare an integer array of size 5
* /\* Accessing sixth element of array which is undefined. \*/
* array[5] = 50;
* /\* Accessing -1th element of array is undefined \*/
* array[-1] = 10;
* Always take care of you loops when wiring it up with arrays. Be sure that the loop does not cross array index bounds.For example, consider the below loop that seems good but exceeds array index bounds.
* #include <stdio.h>
* #define SIZE 5
* int main()
* {
* int array[SIZE]; // Declare an array of size 5
* int index;
* /\*
* \* Valid array index start from 0 and goes till 4.
* \* However, the loop runs from 0 to 5 which exceeds
* \* array index bounds and the result is undefined.
* \*/
* for(index=0; index<=SIZE; index++)
* {
* scanf("%d", &array[index]);
* }
* return 0;

}

To overcome above array index bound, either use index<SIZE or use index<=SIZE-1.

* Always keep in mind that all array element store a value of similar type.