

Arrays & Pointers Arithmetic in C++ | C++ Tutorials for Beginners #13

In this tutorial, we will discuss arrays and pointer arithmetic in C++

What are Arrays in C++

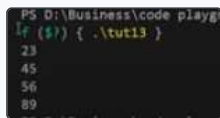
- An array is a collection of items which are of the similar type stored in contiguous memory locations.
- Sometimes, a simple variable is not enough to hold all the data.
- For example, let's say we want to store the marks of 2500 students; initializing 2500 different variable for this task is not feasible.
- To solve this problem, we can define an array with size 2500 that can hold the marks of all students.
- For example `int marks[2500];`

An example program for an array is shown in code snippet below.

```
int marks[] = {23, 45, 56, 89};
cout<<marks[0]<<endl;
cout<<marks[1]<<endl;
cout<<marks[2]<<endl;
cout<<marks[3]<<endl;
```

Code Snippet 1: Array Program 1

As shown in the code snippet, we initialized an array of size 4 in which we have stored marks of 4 students and then printed them one by one. The main point to note here is that array store data in continuous block form in the memory, and array indexes start from 0. Output for the following program is shown in figure 1.



```
PS D:\Business\code play> if ($?) { .\tut13 }
23
45
56
89
```

Figure 1: Array Program 1 Output

Another example program to declare an array is shown in code snippet 2.

```
int mathMarks[4];
mathMarks[0] = 2278;
mathMarks[1] = 738;
mathMarks[2] = 378;
mathMarks[3] = 578;

cout<<"These are math marks"<<endl;
cout<<mathMarks[0]<<endl;
cout<<mathMarks[1]<<endl;
cout<<mathMarks[2]<<endl;
cout<<mathMarks[3]<<endl;
```

Code Snippet 2: Array Program 2

As shown in code snippet 2, we have declared an array of size 4 and then assigned values one by one to each index of the array. Output for the following program is shown in figure 2.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
These are math marks
2278
738
378
578
```

Figure 2: Array Program 2 Output

To change the value at the specific index of an array, we can simply assign the value to that index. For example: `marks[2] = 333` can place the value `333` at the index `2` of the array. We can use loops to print the values of an array, instead of printing them one by one. An example program to print the value of the array with "for" loop is shown in code snippet 3.

```
for (int i = 0; i < 4; i++)
{
    cout<<"The value of marks "<<i<<" is "<<marks[i]<<endl;
}
```

Code Snippet 3: Array program with a loop

As shown in code snippet 3, we initialized an integer variable "i" with the value 0 and set the running condition of the loop to the length of an array. In the loop body, each index number and the value at each number is being printed. Output for the following program is shown in figure 3.

```
The value of marks 0 is 23
The value of marks 1 is 45
The value of marks 2 is 455
The value of marks 3 is 89
```

Figure 3: Array program with loop output

Pointers and Arrays

Storing the address of an array into pointer is different than storing the address of a variable into the pointer because the name of the array is an address of the first index of an array. So to use ampersand "&" with the array name for assigning the address to a pointer is wrong.

- &Marks --> Wrong
- Marks --> address of the first block

An example program for storing the starting address of an array in the pointer is shown in code snippet 4.

```
int* p = marks;
cout<<"The value of marks[0] is "<<*p<<endl;
```

Code Snippet 4: Pointer and Array Program

As shown in code snippet 7, we have assigned the address of array "marks" to the pointer variable "*p" and then printed the pointer "*p". The main thing to note here is that the value at the pointer "*p" is the starting address of the array "marks". The output for the following program is shown in figure 4.

```
The value of marks[0] is 23
```

Figure 4: Pointer and Array Program Output

As shown in figure 4, we have printed the value at pointer "*p", and it has shown us the value of the first index of the array "marks" because the pointer was pointing at the first index of an array and the value at that index was "23". If we want to access the 2nd index of an array through the pointer, we can simply increment the pointer with 1. For example: "(p+1)" will give us the value of the 2nd index of an array. An example program to print the values of an array through the pointer is shown in code snippet 5.

```
int* p = marks;
cout<<"The value of *p is "<<*p<<endl;
cout<<"The value of *(p+1) is "<<*(p+1)<<endl;
cout<<"The value of *(p+2) is "<<*(p+2)<<endl;
cout<<"The value of *(p+3) is "<<*(p+3)<<endl;
```

Code Snippet 5: Pointer and Array Program 2

As shown in code snippet 5, 1st we have printed the value at pointer "*p"; 2nd we have printed the value at pointer "(p+1)"; 3rd we have printed the value at pointer "(p+2)"; 4th we have printed the value at pointer "(p+3)". This program will output the values at "0, 1, 2, 3" indices of an array "marks". The output of the following program is shown in figure 5.

```
The value of marks 0 is 23
The value of marks 1 is 45
The value of marks 2 is 455
The value of marks 3 is 89
```

Figure 5: Pointer and Array Program 2 Output

Code as described/written in the video

```
#include<iostream>
using namespace std;

int main(){
    // Array Example
    int marks[] = {23, 45, 56, 89};

    int mathMarks[4];
    mathMarks[0] = 2278;
    mathMarks[1] = 738;
    mathMarks[2] = 378;
    mathMarks[3] = 578;

    cout<<"These are math marks"<<endl;
    cout<<mathMarks[0]<<endl;
    cout<<mathMarks[1]<<endl;
    cout<<mathMarks[2]<<endl;
    cout<<mathMarks[3]<<endl;

    // You can change the value of an array
    marks[2] = 455;
    cout<<"These are marks"<<endl;
    // cout<<marks[0]<<endl;
    // cout<<marks[1]<<endl;
    // cout<<marks[2]<<endl;
```

[← Previous](#)[Next →](#)