Arrays and Functions

Let's get into the details of passing an array to a function.

We'll cover the following

Passing an array to a function#

To pass an array to a function, we just have to specify the array type, followed by an array name and square brackets in the function parameters.

```
ReturnType  FunctionName (  int ArrayName [ ] ) {

// Function Body

}

int  main (   ) {

// Function Body

FunctionName ( ArrayName )

}
```

Example program#

Let's write a program that takes an array in its parameters.

In the program, we will traverse the array elements. If the value of an array element is less than **50**, we will update the value at that index to **-1**.

Press the **RUN** button and see the output!

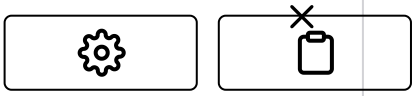
```
#include <iostream>

using namespace std;

// print_array function will print the values of an array
void print_array(int number[], int size) {
  for (int i = 0; i < size; i++) {
    cout << number[i] << " ";
  }
  cout << endl;
}

// modify_array function
void modify_array(int number[], int size) {
  // Traverse array
  for (int i = 0; i < size; i++) {
    // If value less tha 50 set it to -1
    if (number[i] < 50)
      number[i] = -1;
  }
  cout << "Values of array inside the function:" << endl;
  // Call print_array function
  print_array(number,size);
}

// main function
int main() {
  // Initialize size of an array
```



```
Output                                                    0.91s

 Values of array before function call:
 67 89 56 43 29 15 90 67
 Values of array inside the function:
 67 89 56 -1 -1 -1 90 67
 Values of array after function call:
 67 89 56 -1 -1 -1 90 67
```

In the code above:

modify_array function

`modify_array` function takes an array of type `int` and `int` value in its input parameters.

Line No. 16: Uses `for` loop to traverse array from **i** = **0** to **i** = **size-1**.

Line No. 18: Checks if the value of the `number[i]` is less than **50**. If true, then it executes **Line No. 19**.

Line No. 19: Sets `number[i]` to **-1**.

Line No. 23: Calls the `print_array` function to print the values of an array inside the function.

main function

Line No. 29: Initializes the `size` of an array.

Line No. 31: Initializes the values of the array `number` .

Line No. 35: Calls `print_array` function to print the values of an array.

Line No. 37: Calls the `modify_array` function.

Line No. 40: Calls `print_array` function to print the values of an array after calling the `modify_array` function.

Arrays are passed by reference#

In the code above, did you notice that any change made in the elements of an array inside the `modify_array` function is reflected in the `main` function?

This was not the case with variables because, by default, variables are passed by value.

When we pass an array to the function, we don't need to specify the size of an array in square brackets. This is because we need the size of an array when we are creating a new array. However, when we pass an array in the function, we are just passing an original array to the function. This means if we made any changes inside the function, we would see those changes outside the function. That is why we can say that by default, arrays are passed by reference.

Quiz

Q The general syntax for passing an array to the function is:

A) `function_name (array_name)`

B) `function_name (&array_name)`

Your Answer

C) `function_name (array_name);`

D) `function_name (&array_name);`

Submit Answer

Reset Quiz

That's all about one-dimensional arrays. Let's learn about the implementation of two-dimensional arrays in C++.

Back

Accessing an Array

Next

Creating a Two-Dimensional Array

Mark as Completed

Report an Issue