

C++ FUNCTION CALL BY POINTER

https://www.tutorialspoint.com/cplusplus/cpp_function_call_by_pointer.htm

Copyright © tutorialspoint.com

Advertisements

The **call by pointer** method of passing arguments to a function copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the passed argument.

To pass the value by pointer, argument pointers are passed to the functions just like any other value. So accordingly you need to declare the function parameters as pointer types as in the following function **swap**, which exchanges the values of the two integer variables pointed to by its arguments.

```
// function definition to swap the values.
void swap(int *x, int *y) {
    int temp;
    temp = *x; /* save the value at address x */
    *x = *y; /* put y into x */
    *y = temp; /* put x into y */

    return;
}
```

To check the more detail about C++ pointers, kindly check [C++ Pointers](#) chapter.

For now, let us call the function **swap** by passing values by pointer as in the following example –

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

// function declaration
void swap(int *x, int *y);

int main () {
    // local variable declaration:
    int a = 100;
    int b = 200;

    cout << "Before swap, value of a : " << a << endl;
    cout << "Before swap, value of b : " << b << endl;

    /* calling a function to swap the values.
     * &a indicates pointer to a ie. address of variable a and
     * &b indicates pointer to b ie. address of variable b.
     */
    swap(&a, &b);

    cout << "After swap, value of a : " << a << endl;
    cout << "After swap, value of b : " << b << endl;

    return 0;
}
```

When the above code is put together in a file, compiled and executed, it produces the following result –

```
Before swap, value of a :100
Before swap, value of b :200
```

```
After swap, value of a :200  
After swap, value of b :100
```