

# Call by Value & Call by Reference in C++ | C++ Tutorials for Beginners #16

In this tutorial, we will discuss call by value and call by reference in C++

## Call by Value in C++

Call by value is a method in C++ to pass the values to the function arguments. In case of call by value the copies of actual parameters are sent to the formal parameter, which means that if we change the values inside the function that will not affect the actual values. An example program for the call by value is shown in Code Snippet 1.

```
void swap(int a, int b){ //temp a b
    int temp = a;        //4   4 5
    a = b;                //4   5 5
    b = temp;             //4   5 4
}
```

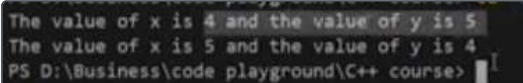
**Code Snippet 1: Call by Value Swap Function**

As shown in Code Snippet 1, we created a swap function which is taking two parameters "int a" and "int b". In function body values of the variable, "a" and "b" are swapped. An example program is shown in Code Snippet 2, which calls the swap function and passes values to it.

```
int main(){
    int x =4, y=5;
    cout<<"The value of x is "<<x<<" and the value of y is "<<y<<endl;
    swap(x, y);
    cout<<"The value of x is "<<x<<" and the value of y is "<<y<<endl;
    return 0;
}
```

**Code Snippet 2: Passing Values to Swap Function**

As shown in Code Snippet 2, we have initialized two integer variables "a" and "b" and printed their values. Then we called a "swap" function and passed values of variables "a" and "b" and again printed the values of variables "a" and "b". The output for the following program is shown in figure 1.



```
The value of x is 4 and the value of y is 5
The value of x is 5 and the value of y is 4
PS D:\Business\code playground\C++ course>
```

**Figure 1: Call by Value Swap Function Output**

As shown in figure 3, the values of "a" and "b" are the same for both times they are printed. So the main point here is that when the call by value method is used it doesn't change the actual values because copies of actual values are sent to the function.

## Call by Pointer in C++

A call by the pointer is a method in C++ to pass the values to the function arguments. In the case of call by pointer, the address of actual parameters is sent to the formal parameter, which means that if we change the values inside the function that will affect the actual values. An example program for the call by reference is shown in Code Snippet 3.

```
// Call by reference using pointers
void swapPointer(int* a, int* b){ //temp a b
    int temp = *a;              //4   4 5
    *a = *b;                    //4   5 5
    *b = temp;                  //4   5 4
}
```

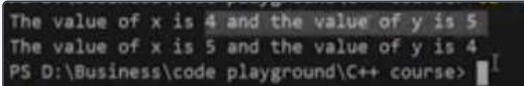
**Code Snippet 3: Call by Pointer Swap Function**

As shown in Code Snippet 3, we created a swap function which is taking two pointer parameters "int\* a" and "int\* b". In function body values of pointer variables, "a" and "b" are swapped. An example program is shown in Code Snippet 4, which calls the swap function and passes values to it.

```
int main(){
    int x =4, y=5;
    cout<<"The value of x is "<<x<<" and the value of y is "<<y<<endl;
    swapPointer(&x, &y); //This will swap a and b using pointer reference
    cout<<"The value of x is "<<x<<" and the value of y is "<<y<<endl;
    return 0;
}
```

#### Code Snippet 4: Passing Values to Call by Pointer Swap Function

As shown in Code Snippet 4, we have initialized two integer variables “a” and “b” and printed their values. Then we called a “swap” function and passed addresses of variables “a” and “b” and again printed the values of variables “a” and “b”. The output for the following program is shown in figure 2.



**Figure 2: Call by Pointer Swap Function Output**

As shown in figure 2, the values of “a” and “b” are swapped when the swap function is called. So the main point here is that when the call by pointer method is used it changes the actual values because addresses of actual values are sent to the function.

## Call by Reference in C++

Call by reference is a method in C++ to pass the values to the function arguments. In the case of call by reference, the reference of actual parameters is sent to the formal parameter, which means that if we change the values inside the function that will affect the actual values. An example program for a call by reference is shown in Code Snippet 5.

```
void swapReferenceVar(int &a, int &b){ //temp a b
    int temp = a; //4 4 5
    a = b; //4 5 5
    b = temp; //4 5 4
}
```

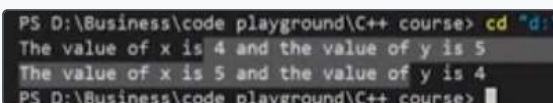
#### Code Snippet 5: Call by Reference Swap Function

As shown in Code Snippet 5, we created a swap function that is taking reference of “int &a” and “int &b” as parameters. In function body values of variables, “a” and “b” are swapped. An example program is shown in Code Snippet 6, which calls the swap function and passes values to it.

```
int main(){
    int x =4, y=5;
    cout<<"The value of x is "<<x<<" and the value of y is "<<y<<endl;
    swapReferenceVar(x, y); //This will swap a and b using reference variables
    cout<<"The value of x is "<<x<<" and the value of y is "<<y<<endl;
    return 0;
}
```

#### Code Snippet 6: Passing Values to Call by Reference Swap Function

As shown in Code Snippet 6, we have initialized two integer variables “a” and “b” and printed their values. Then we called a “swap” function and passed variables “a” and “b” and again printed the values of variables “a” and “b”. The output for the following program is shown in figure 3.



**Figure 3: Call by Reference Swap Function Output**

As shown in figure 3, the values of “a” and “b” are swapped when the swap function is called. So the main point here is that when the call by reference method is used it changes the actual values because references of actual values are sent to the function.

## Code as described/written in the video

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

int sum(int a, int b){
    int c = a + b;
    return c;
}

// This will not swap a and b
void swap(int a, int b){ //temp a b
    int temp = a;        //4   4  5
    a = b;                //4   5  5
    b = temp;             //4   5  4
}

// Call by reference using pointers
void swapPointer(int* a, int* b){ //temp a b
    int temp = *a;          //4   4  5
    *a = *b;                //4   5  5
    *b = temp;              //4   5  4
}

// Call by reference using C++ reference Variables
// int &
void swapReferenceVar(int &a, int &b){ //temp a b
```

[← Previous](#)[Next →](#)