



Pass by Value in Functions

Learn a way to pass the value of actual parameters to the function.

We'll cover the following



- Introduction
 - Basic syntax
 - Example program
 - Explanation
 - passValue function
 - main function

Introduction#

Suppose you have sent an email with an attached file to your friend. Your friend has downloaded the file and then made some changes to it. The original document can not be changed by any of the changes made by your friend because your friend has a copy of the original file.

Pass by value is just like sending a copy of a file to another person.

*In **pass by value**, when we call a function, we pass the copy of the actual parameters to the formal parameters in the function.*



In pass by value, the actual and formal parameters are stored in different memory locations. Any changes made in the formal parameters inside the function will not affect the values of actual parameters in the main function. In C++, by default, actual parameters are passed by value to the function.

Basic syntax

The general syntax for passing the value of a variable by value is given below:

```
#include <stdio.h>

return_type function_name ( number ) ;

.....

int main ( )
{
    int num = 10
    function_name ( num ) ;
    return 0;
}
```

Diagram illustrating the syntax for passing a value by value:

- The **formal parameter / parameter** is `number` in the function signature `function_name (number) ;`. It is represented by a box containing the value `10`.
- The **actual parameter / argument** is `num` in the function call `function_name (num) ;`. It is represented by a box containing the value `10`.

Example program

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

```
1  #include <iostream>
2
3  using namespace std;
```



```

3  using namespace std;
4  // function definition
5  void passValue(int number) {
6      // Multiply the number by 10
7      number = number * 10;
8      cout << "Value of number inside the function = " << number << endl;
9  }
10
11 int main() {
12     // Initialize variable
13     int number = 10;
14     cout << "Value of number before function call = " << number << endl;
15     // Call function
16     passValue(number);
17     cout << "Value of number after function call = " << number << endl;
18
19     return 0;
20 }

```



Output

1.01s

```

Value of number before function call = 10
Value of number inside the function = 100
Value of number after function call = 10

```

Explanation#

In the code above, we have two functions:

- passValue function
- main function

passValue function



Line No. 5: The `passValue` function takes a number of type `int` . It will perform its task and then returns nothing in output.

Line No. 7: Multiplies the `number` by `10` and stores the result in the `number` .

Line No. 8: Prints the updated value of the `number` .

main function

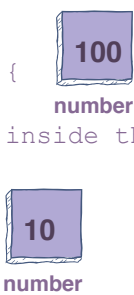
Line No. 13: Initializes a variable `number` .

Line No. 14: Prints the value of the `number` before the function call.

Line No. 16: Calls a function `passValue` . The program execution control is transferred to **Line No. 5**.

Line No. 17: Prints the value of the `number` after the function call.

```
void passValue (int number) {  
    number = number *10;  
    cout << "Value of number inside the function = " << number << endl;  
}  
  
int main ( ) {  
    int number =10;  
  
    cout << "Value of number before function call = " << number << endl;  
    passValue(number);  
  
    cout << "Value of number after function call = " << number << endl;  
  
    return 0;  
} Exit the main function
```



Output:

```
Value of number before function call = 10  
Value of number inside the function = 100  
Value of number after function call = 10
```



11 of 11



Quiz



Q What is the output of the following code?

```
void cube(int number) {  
    number = number * number * number;  
    cout << "number = " << number << endl;  
}  
  
int main() {  
    int number = 5;  
    cube(number);  
    cout << "number = " << number << endl;  
  
    return 0;  
}
```



(/learn)

☒ A) number = 5

number = 5



B) number = 125

number = 125

Your Answer



C) number = 125

number = 5

Explanation



In pass by value, the copy of the actual parameter is passed to the formal parameter. Therefore, any changes made in the value of formal parameter won't affect the actual parameter.



D) number = 5

number = 125

Submit Answer

Reset Quiz ↻

This sums up our discussion of passing values to functions. In the upcoming lesson, let's explore how we can pass a reference of value to a function.

Stay tuned!

← Back

Next →

C++ Function Parameters

Pass by Reference in Functions



Mark as Completed



Report an Issue