Function and Pointers

In this lesson, you will see how to pass pointers to the functions.

We'll cover the following

Passing pointers to the function

In a previous lesson, we discussed two ways of passing actual parameters to the formal parameters in the function.

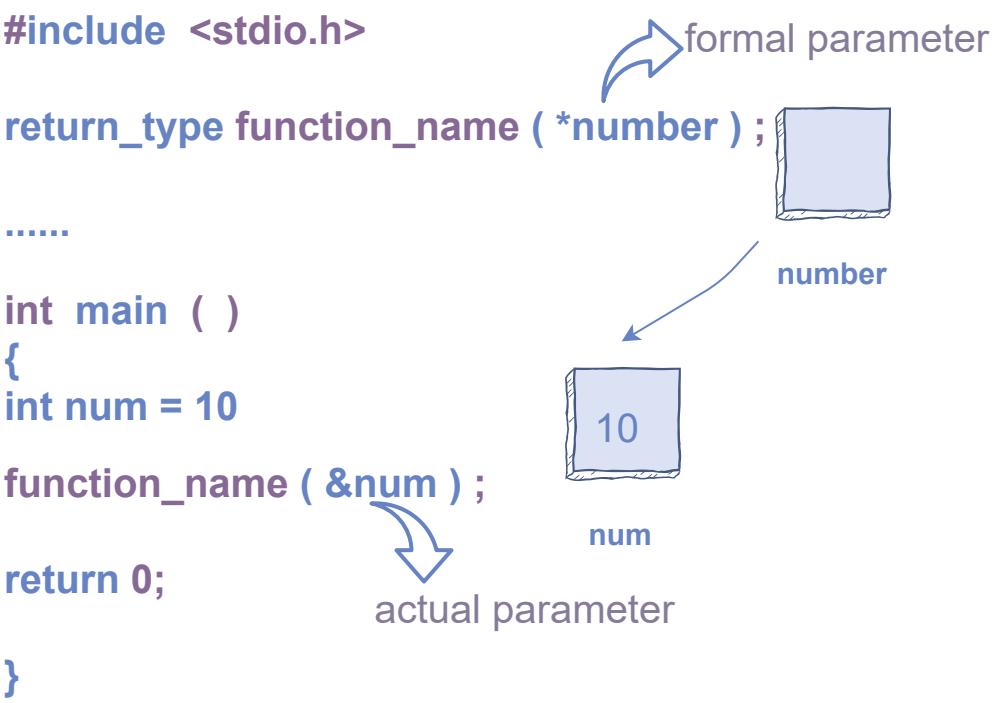
- Pass by value
- Pass by reference

However, there is another way to pass arguments to the function that is passed by reference with a pointer parameter.

The function pointer parameter receives the address of the parameter. Then, it uses the dereference operator to access the value of the variable.

Basic syntax

The general syntax for passing a pointer to the function parameter is given below:



When we want to pass the pointer, we will declare a function parameter as pointers. To declare a function parameter as a pointer, use an asterisk `*` before the function parameter. Then, we will pass the address of the variable in the actual parameter using the address-of `&` operator.

Example program

```
#include <iostream>

using namespace std;
// function definition
void passPointer(int *number) {
  // Multiply the number by 10
  *number = *number * 10;
  cout << "Value of number inside the function = " << *number << endl;
}

int main() {
  // Initialize variable
  int num = 10;
  cout << "Value of number before function call = " << num << endl;
  // Call function
  passPointer(&num);
  cout << "Value of number after function call = " << num << endl;

  return 0;
}
```

Explanation

In the code above, we have two functions:

- passPointer function
- main function

passPointer function

Line No. 5: The `passPointer` function receives an address of the `int` value and stores it in the `number`. Since the function is of type `void`, no value is returned.

Line No. 7: Multiplies the value that the pointer `number` is pointing to by `10` and stores the result in the location pointed by the `number`.

Line No. 8: Prints the value pointed by the `number`.

main function

Line No. 13: Initializes a variable `num`

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

Line No. 16: Calls a function `passPointer` and passes the address of the `num` to function. The execution control is transferred to **Line No. 5**.

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

Illustration

i By default, pointers are passed by value. When we call the function, the value of the address is copied to the pointer variable. So, if we change the value of the pointer inside the function, we cannot see that change outside the function body.

Quiz

Consider the function given below:

```
void passPointer(int *number) {
  int value = 13;
  number = &value;
  *number = *number + 14;
```

```
  *number = *number + 14;
}
```

Suppose `num = 10` and we called the function `passPointer(&num)`, what would be the value of `num` after the function call?

Retake Quiz

Let’s get our hands dirty with a few challenges related to pointers.