C++ Function Parameters

Get acquainted with actual parameters, formal parameters, and the default values of the parameters.

We'll cover the following

Function parameters

We can declare the variables inside the function definition as parameters. We specify the list of parameters separated by a comma inside the round brackets. In C++, we have:

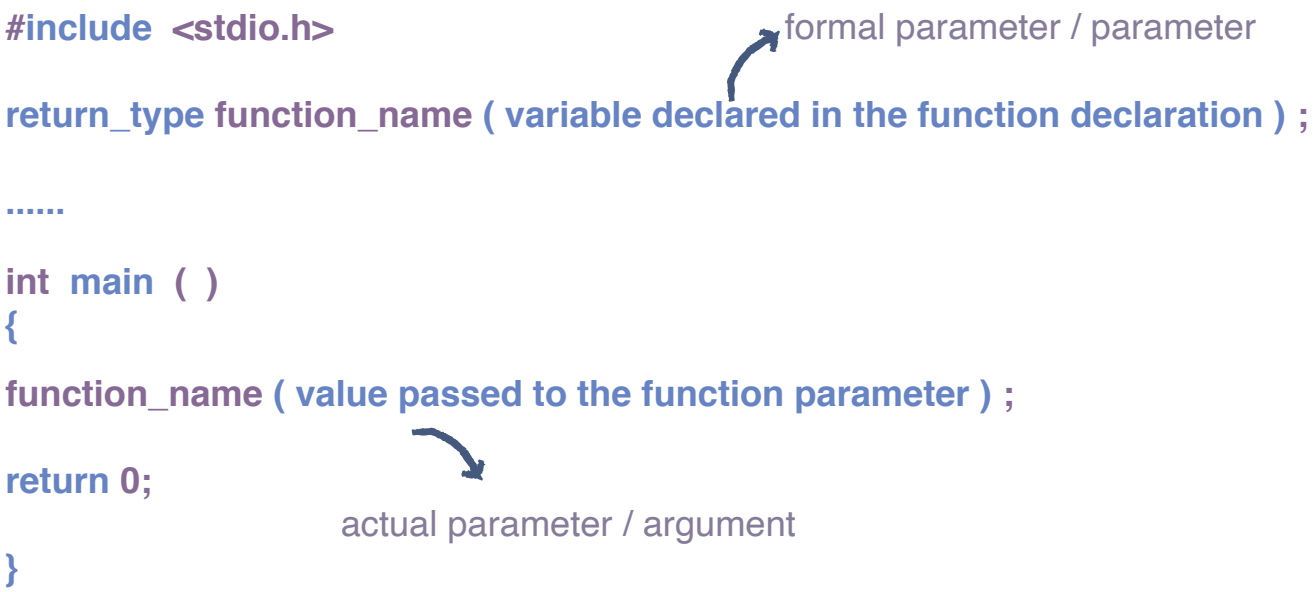
- Formal parameters
- Actual parameters

Formal parameters

Formal parameters are the variables defined in the function definition. These variables receive values from the calling function. Formal parameters are commonly known as **parameters**.

Actual parameters

Actual parameters are the variables or values passed to the function when it is called. These variables supply value to the called function. Actual parameters are commonly known as **arguments**.



Example program

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

```
1  #include <iostream>
2  using namespace std;
3
4  // Function definition
5  int make_juice ( int water , int fruit){
```

```
// Define new  variable juice of int type
  int juice ;
// Adds water in apple and saves the output in juice
  juice = water + fruit;
// Prints text on the screen
  cout << "Your juice is ready" << endl ;
 // Returns juice value in output
  return juice;

}


int main() {
  // Declares a variable juice_glass
  int juice_glass;
  // Calls function make_juice and save its output in juice_glass
  juice_glass = make_juice ( 2 , 5);
  // Prints value of juice_glass
  cout << "Number of juice glass = " << juice_glass << endl;
  return 0;
}
```

```
Output                                                    1.27s

 Your juice is ready
 Number of juice glass = 7
```

In the above program:

Line No. 5: We defined the function `make_juice` . In the `make_juice` definition, we declare the variables `water` and `fruit` that take integer values. These are the **formal parameters**.

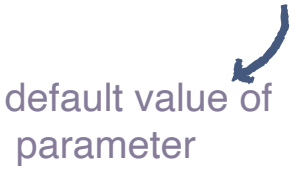
Line No. 22: In the `main` function, we call the function `make_juice` . `make_juice` takes **2** and **5** inside the round brackets. Here, **2** and **5** are the

actual parameters.

Default parameter values#

If we provide fewer or no arguments to the calling function, the default values of the parameters are used. We specify the default values in the function declaration using an equal sign `=`.

```
#include  <stdio.h>

return_type function_name ( formal parameter / parameter = value ) ;

.....

int  main  (  )
{

function_name ( actual parameter / argument ) ;

return 0;

}
```



Example program

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

```
 8  // Adds water in apple and saves the output in juice
 9    juice = water + fruit;
10  // Prints text on the screen
11    cout << "Your juice is ready" << endl ;
12   // Returns juice value in output
13    return juice;
14
15  }
16
17
```

```
int main() {
  // Declares a variable juice_glass
  int juice_glass;

  // Calls function make_juice without any actual paramters
  juice_glass = make_juice ( );
  cout << "Number of juice glass = " << juice_glass << endl;
  // Calls function make_juice with only one actual paramters
  juice_glass = make_juice (5);
  cout << "Number of juice glass = " << juice_glass << endl;
    // Calls function make_juice and save its output in juice_glass
  juice_glass = make_juice ( 2 , 5 );
  cout << "Number of juice glass = " << juice_glass << endl;

  return 0;
}


```

Output 1.11s

```
Your juice is ready
Number of juice glass = 4
Your juice is ready
Number of juice glass = 8
Your juice is ready
Number of juice glass = 7
```

Explanation#

In the code above:

Line No. 23: If we call the function without specifying the actual values of the `water` and `fruit`, the compiler uses the default values of the parameters.

Line No. 26: If we call the function with one actual parameter, the compiler

Line No. 26: If we call the function with one actual parameter, the compiler uses the actual value for `water` and the default value for `fruits`.

Line No. 29: If we specify the actual values for both `water` and `fruit`, the compiler uses their actual values.

> 📝 If we specify the default value of the parameters, the parameters following it must have a default value. Otherwise, you get an error. However, it is not necessary to assign the default values to the parameters preceding it.

Passing actual parameters to the function#

We can pass the actual parameters to the function in the following two ways:

- Pass by value
- Pass by reference

Quiz

Q What is the output of the following code?

```
int number_sum (int num1 = 30 , int num2 ){
return num1 + num2;
}

int main() {
  int sum = number_sum (20) ;
  cout << sum ;
  return 0;
```

}

A) 60

B) 50

C) Null

Your Answer

D) Generates an error

Explanation

If you specify the default value of the parameters, then the parameters following it must have a default value. Otherwise, you will get an error.

Submit Answer

Reset Quiz

Let’s dig deeper into passing parameters to functions in the upcoming lesson.

See you there!