INTRODUCTION OF C++ SECTION 4 PART(1)

Dr/Ghada Maher
Eng/ Hossam Medhat

C++ FUNCTIONS

- A function is a block of code which only runs when it is called.
- The function in C++ language is also known as procedure or subroutine in other programming languages.
- You can pass data, known as parameters, into a function.
- Functions are used to perform certain actions, and they are important for reusing code.
- There are two types of function:

1) Standard Library Functions: Predefined in C++ ,such as ceil(x), cos(x), exp(x), etc.
2) User-defined Function: Created by users.

ADVANTAGE OF FUNCTIONS IN C++

- There are many advantages of functions :

1) Code Reusability : By creating functions in C++, you can call it many times.

2) Code optimization : It makes the code optimized, we don't need to write much code.

3) reduces complexity of a big program , So we don't need to write the same code again and again.

C++ FUNCTIONS "CONT"

❖Create a Function :

➢**Syntax :**

```
void myFunction() {
  // code to be executed
}
```

➢**Example Explained :**

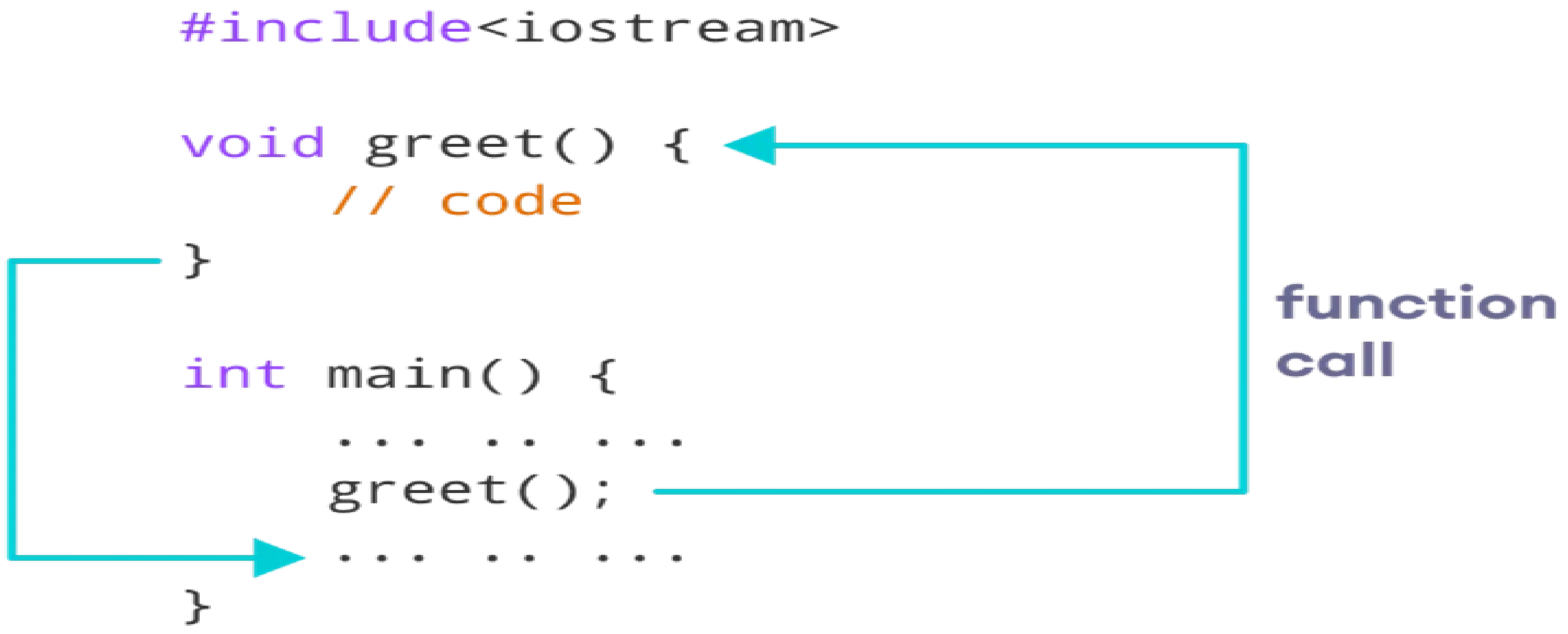
➢**myFunction()** is the name of the function

➢**void** means that the function does not have a return value.

➢**inside the function (the body),** add code that defines what the function should do

CALL A FUNCTION

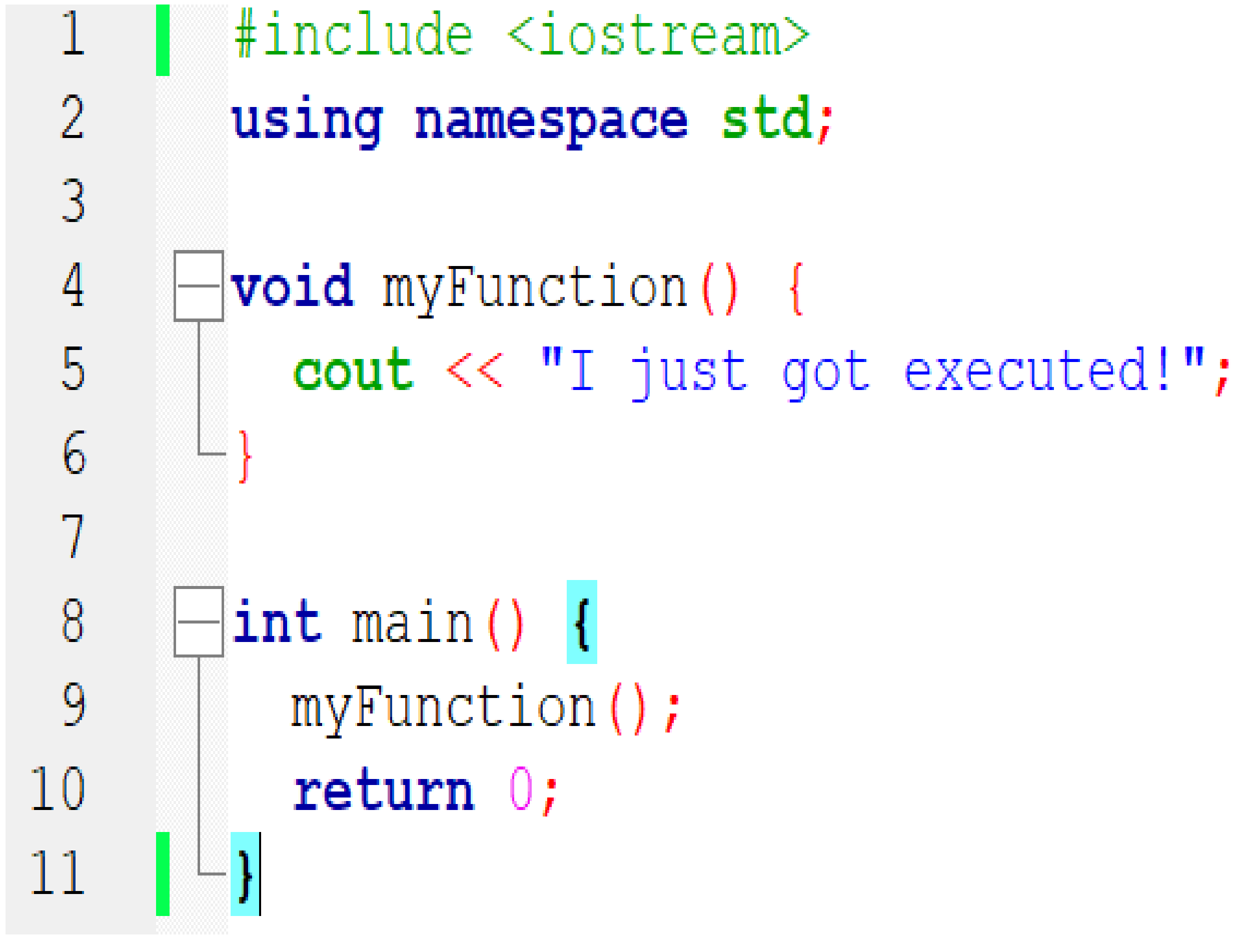
❖To call a function, write the function's name followed by two parentheses () and a semicolon ;



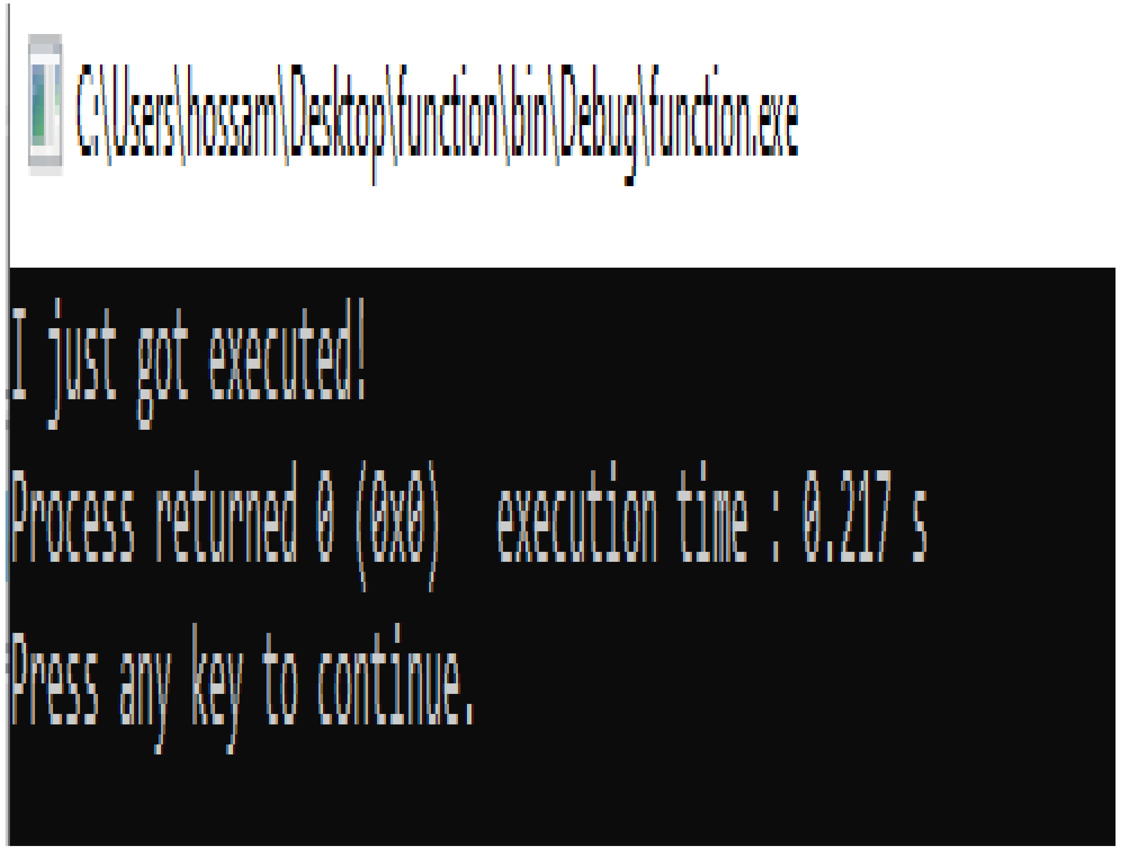
```
#include<iostream>

void greet() {
    // code
}

int main() {
    ... .. ...
    greet();
    ... .. ...
}
```

CALL A FUNCTION "CONT"

➢ **Example : myFunction() is used to print a text (the action), when it is called:**



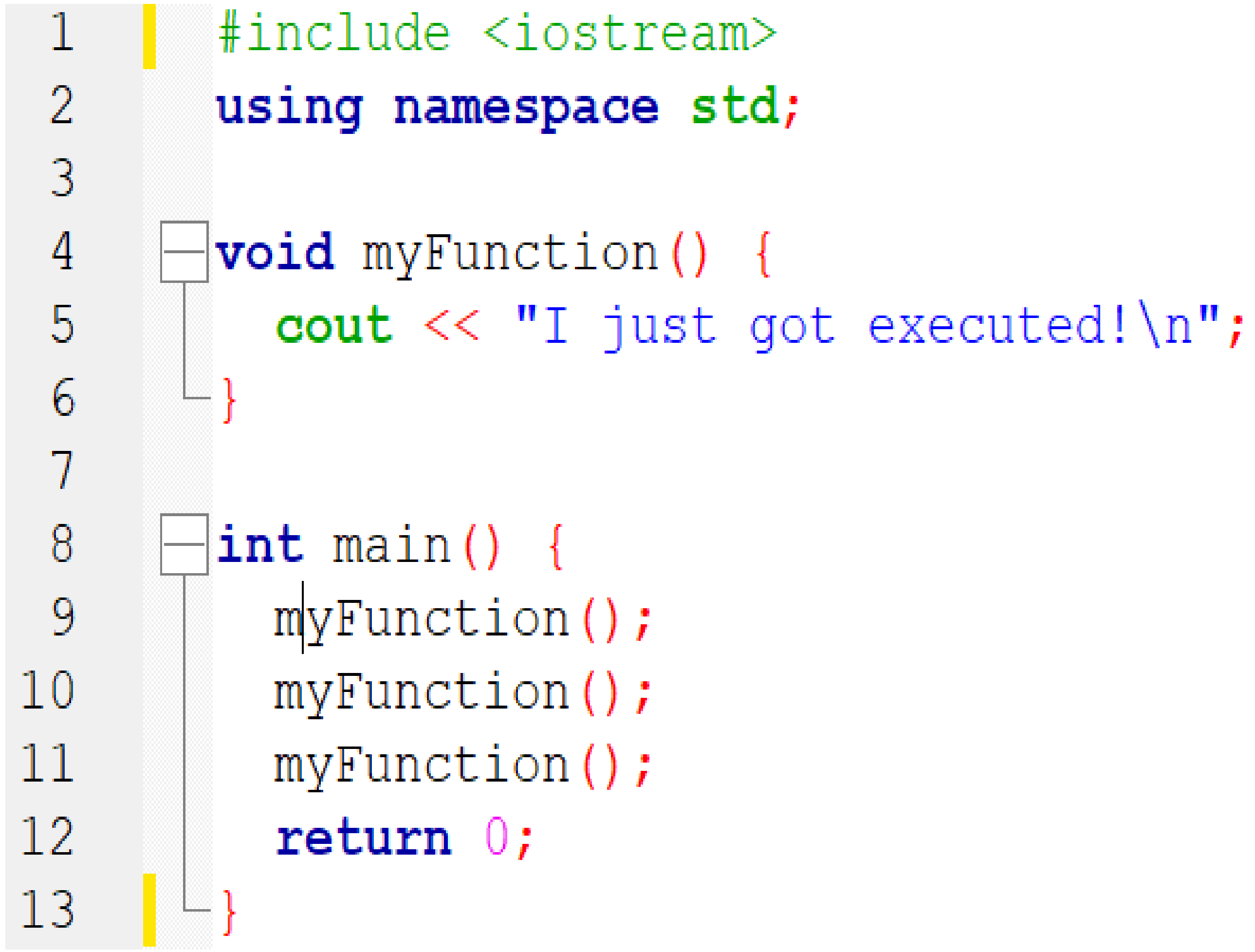
```cpp
#include <iostream>
using namespace std;

void myFunction() {
  cout << "I just got executed!";
}

int main() {
  myFunction();
  return 0;
}
```



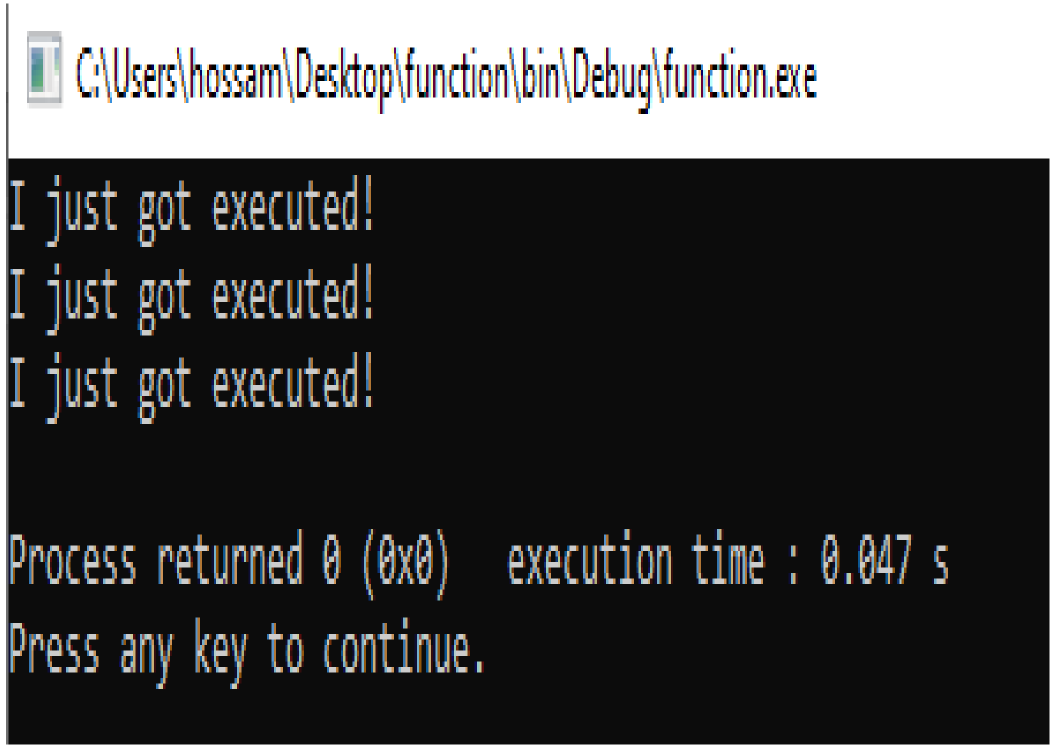
```
C:\Users\hossam\Desktop\function\bin\Debug\function.exe
I just got executed!
Process returned 0 (0x0)   execution time : 0.217 s
Press any key to continue.
```

CALL A FUNCTION "CONT"

➢ **Example : A function can be called multiple times:**



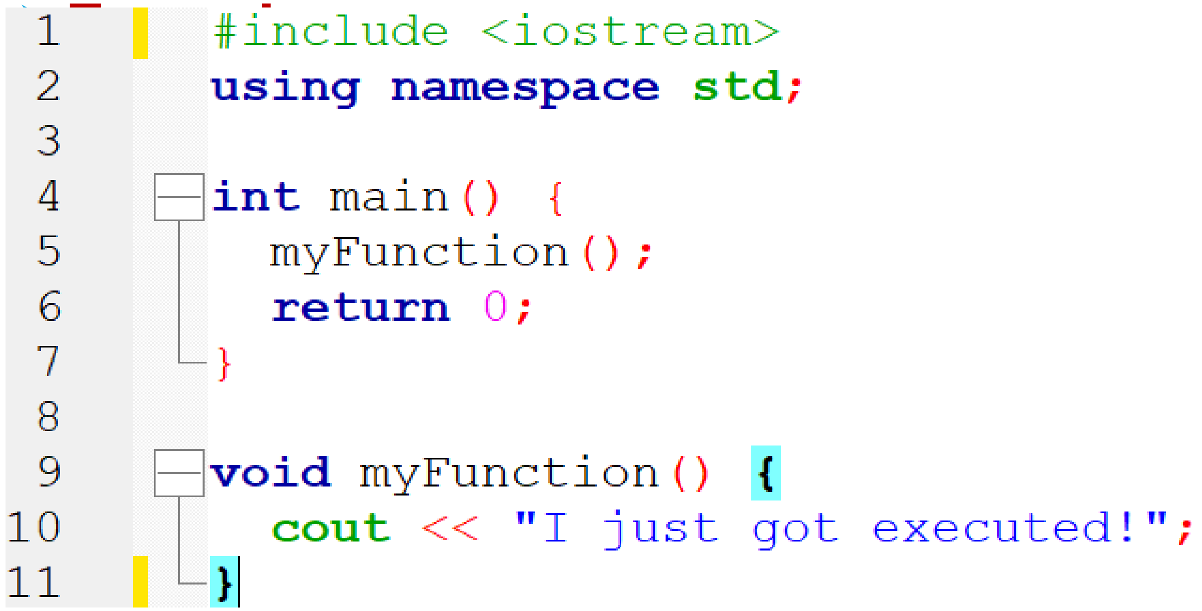
```cpp
#include <iostream>
using namespace std;

void myFunction() {
  cout << "I just got executed!\n";
}

int main() {
  myFunction();
  myFunction();
  myFunction();
  return 0;
}
```



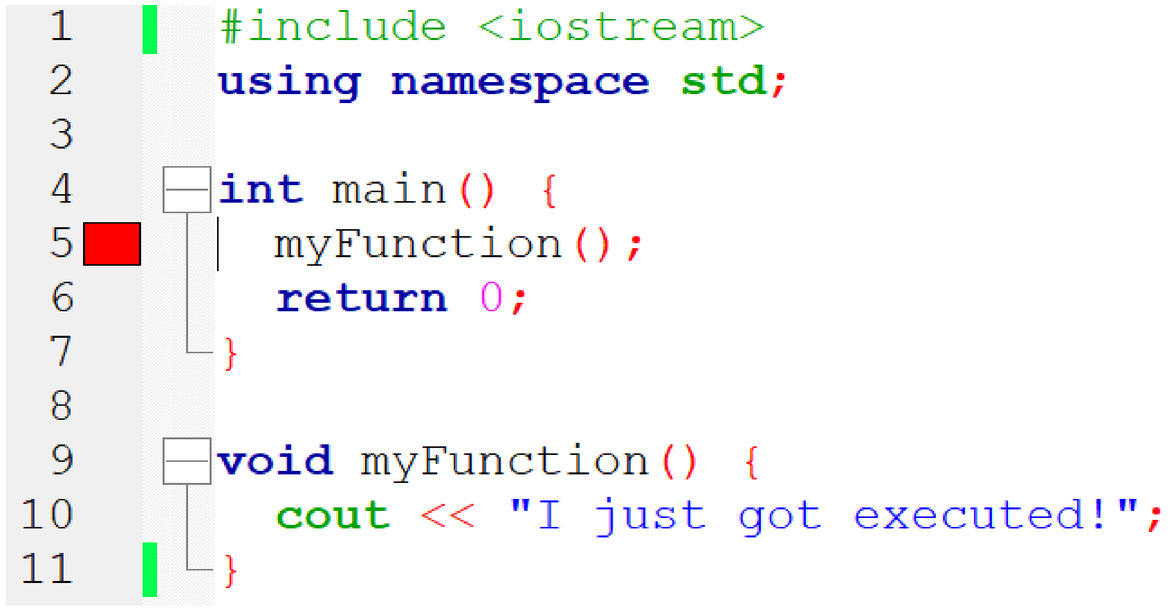
```
C:\Users\hossam\Desktop\function\bin\Debug\function.exe
I just got executed!
I just got executed!
I just got executed!

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

Call a Function "Cont"

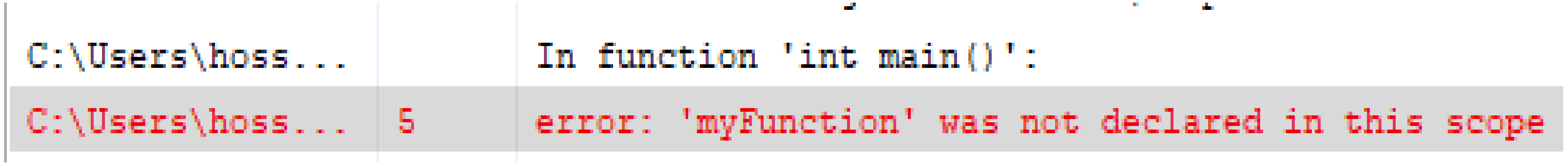
- If a user-defined function, such as myFunction() is declared after the main() function, an error will occur:



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

int main()  {
   myFunction();
   return 0;
}

void myFunction() {
   cout << "I just got executed!";
}
```



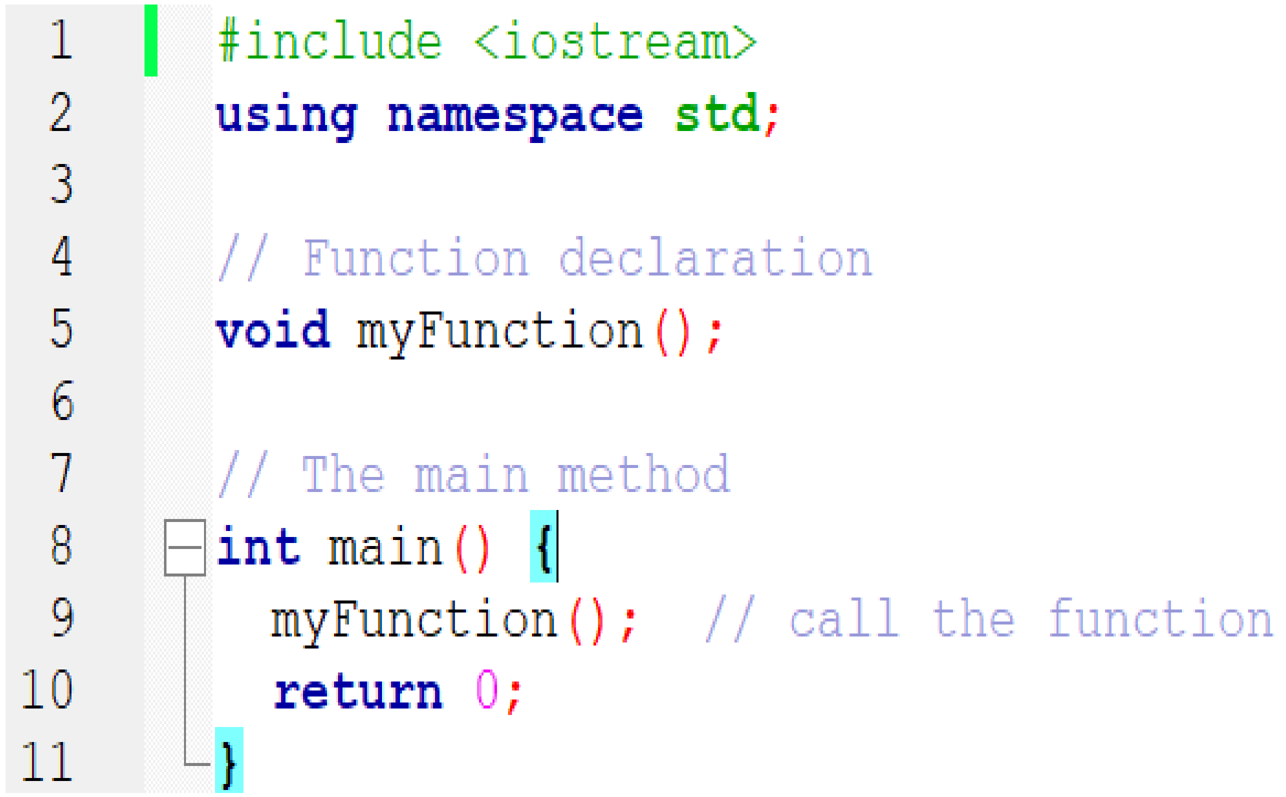
```
#include <iostream>
using namespace std;

int main()  {
   myFunction();
   return 0;
}

void myFunction()  {
   cout << "I just got executed!";
}
```



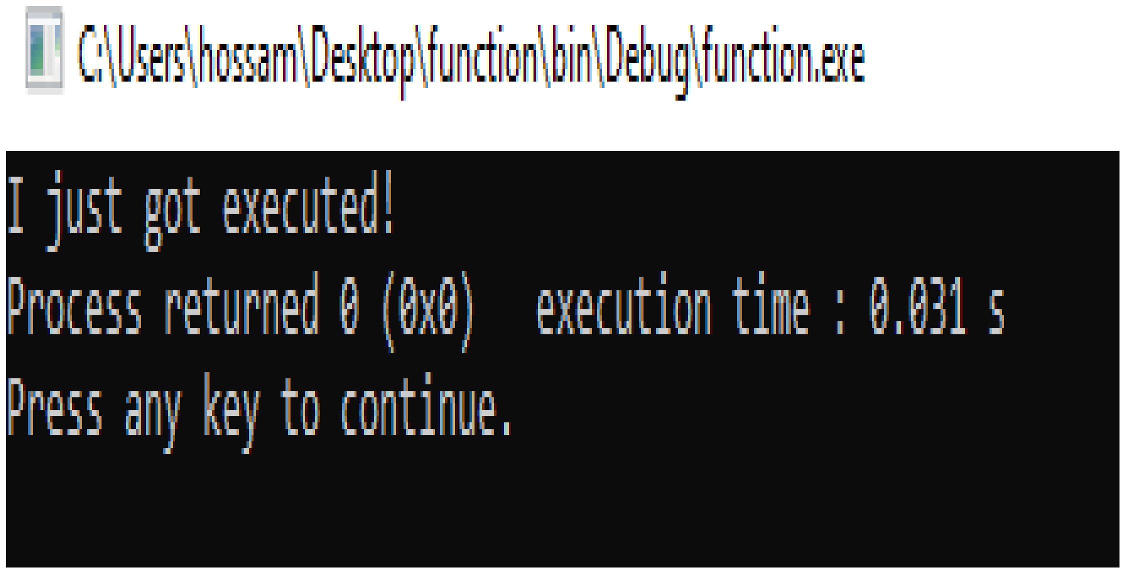
```
C:\Users\hoss...          In function 'int main()':
C:\Users\hoss...    5     error: 'myFunction' was not declared in this scope
```

CALL A FUNCTION "CONT"

- function declaration above main(), and function definition below main().
- **Example :**



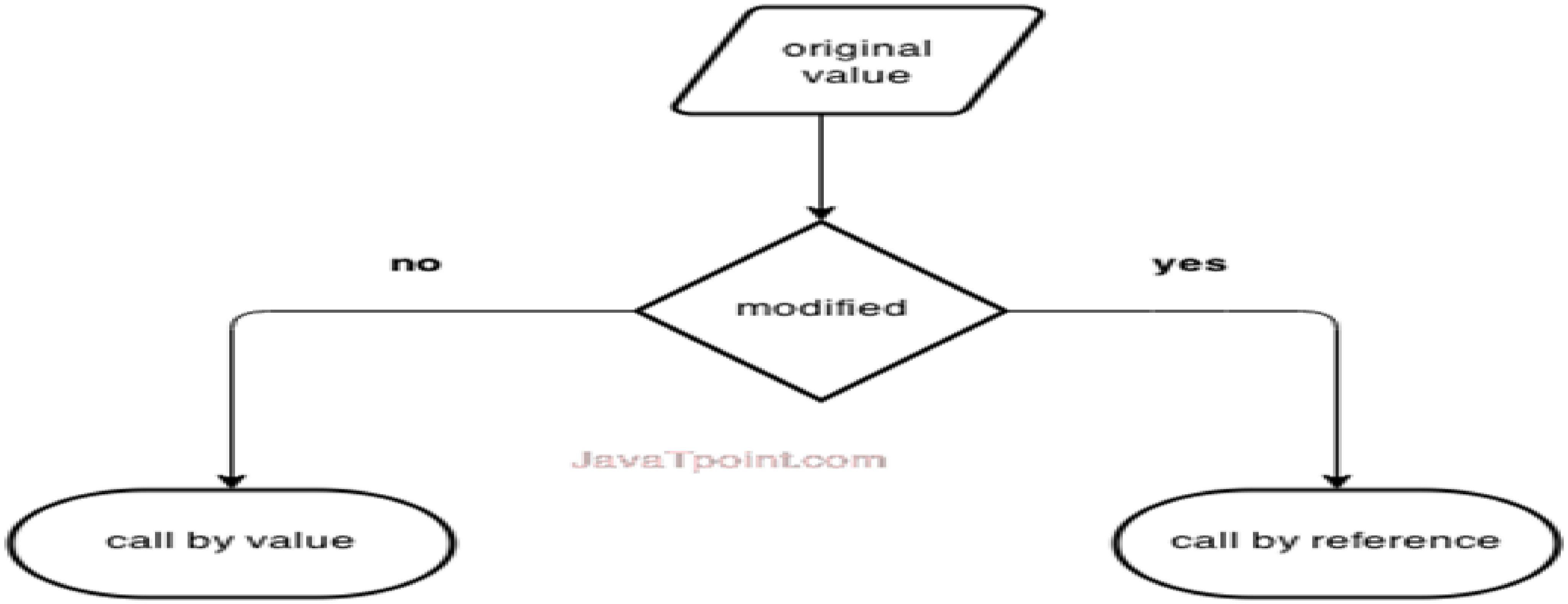
```
#include <iostream>
using namespace std;

// Function declaration
void myFunction();

// The main method
int main() {
  myFunction();  // call the function
  return 0;
}
```



```
C:\Users\hossam\Desktop\function\bin\Debug\function.exe

I just got executed!
Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

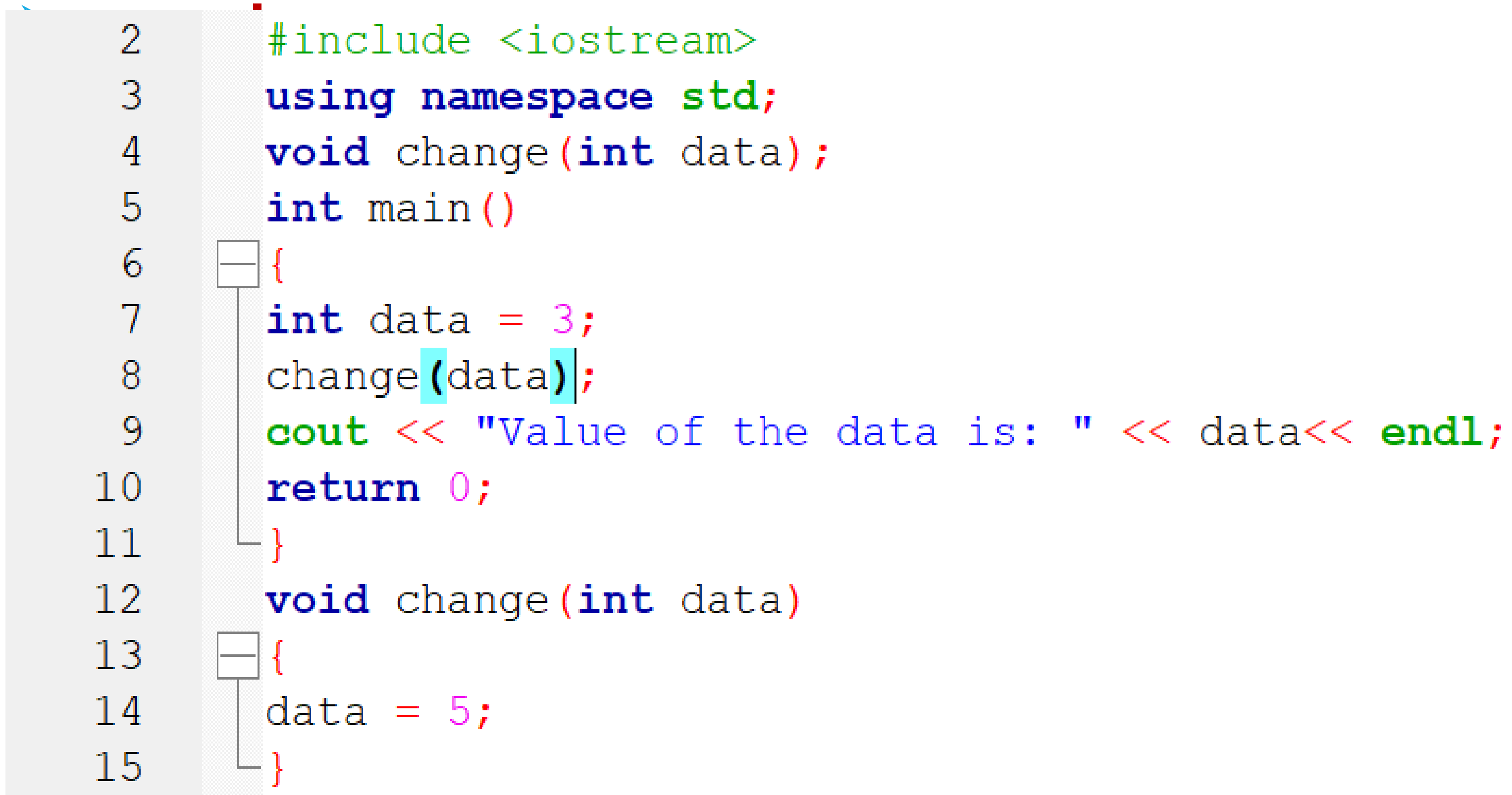
CALL BY VALUE & CALL BY REFERENCE IN C++

❖There are two ways to pass value or data to function in C language: call by value and call by reference.

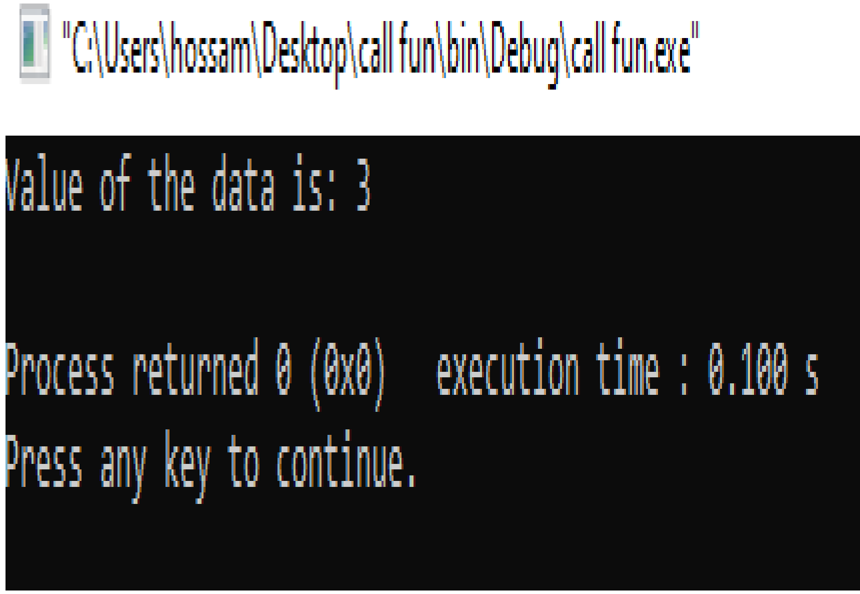


CALL BY VALUE IN C++

- In call by value, original value is not modified.
- If you change the value of function parameter, it is changed for the current function only.
  It will not change the value of variable inside the caller method such as main().



```cpp
#include <iostream>
using namespace std;
void change(int data);
int main()
{
int data = 3;
change(data);
cout << "Value of the data is: " << data<< endl;
return 0;
}
void change(int data)
{
data = 5;
}
```

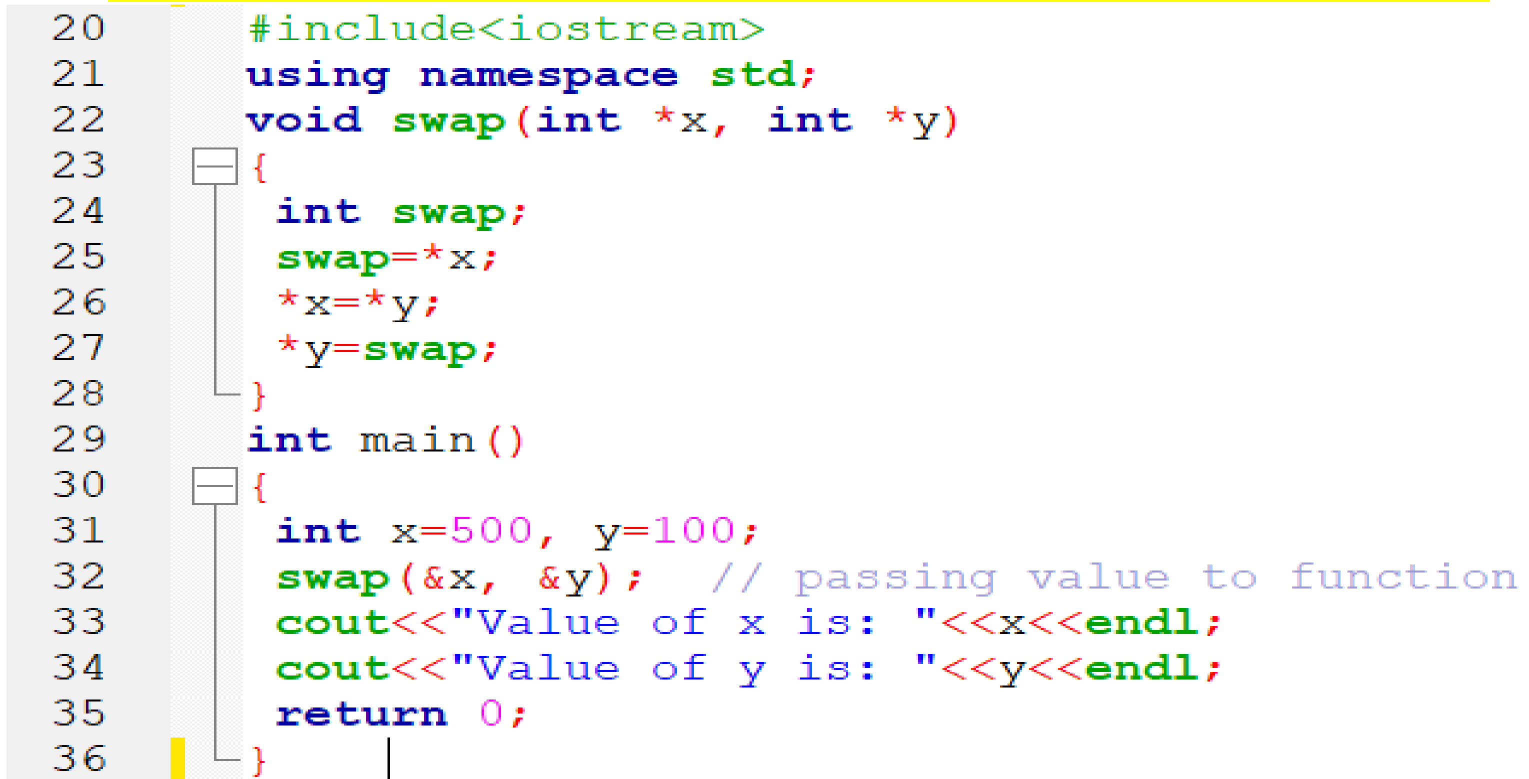


```
"C:\Users\hossam\Desktop\call fun\bin\Debug\call fun.exe"

Value of the data is: 3

Process returned 0 (0x0)   execution time : 0.100 s
Press any key to continue.
```

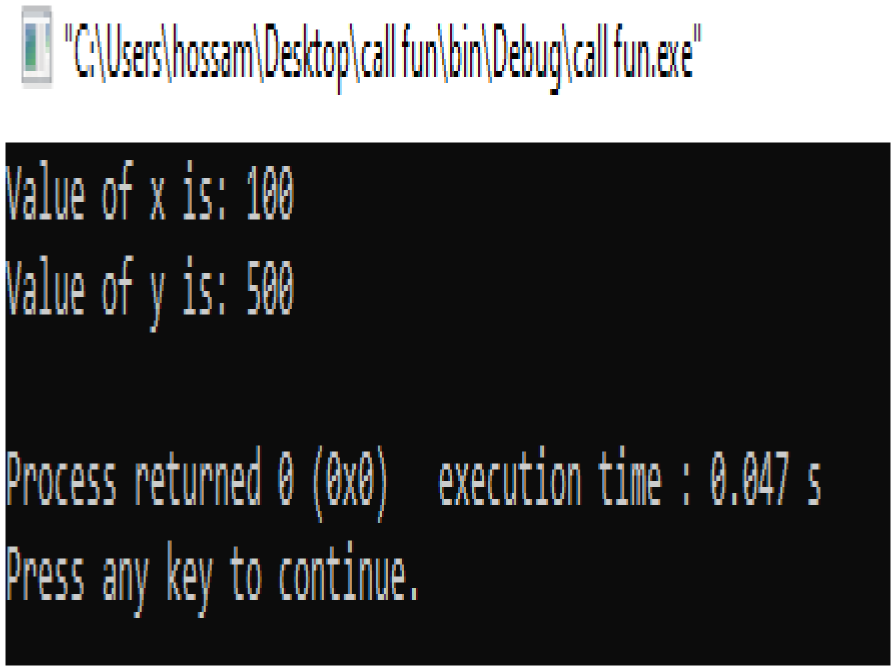
CALL BY REFERENCE IN C++

- In call by reference, original value is modified because we pass reference (address).
- address of the value is passed in the function, Hence, value changed inside the function,

it is reflected inside as well as outside the function.



```
#include<iostream>
using namespace std;
void swap(int *x, int *y)
{
 int swap;
 swap=*x;
 *x=*y;
 *y=swap;
}
int main()
{
 int x=500, y=100;
 swap(&x, &y);   // passing value to function
 cout<<"Value of x is: "<<x<<endl;
 cout<<"Value of y is: "<<y<<endl;
 return 0;
}
```



```
"C:\Users\hossam\Desktop\call fun\bin\Debug\call fun.exe"

Value of x is: 100
Value of y is: 500


Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

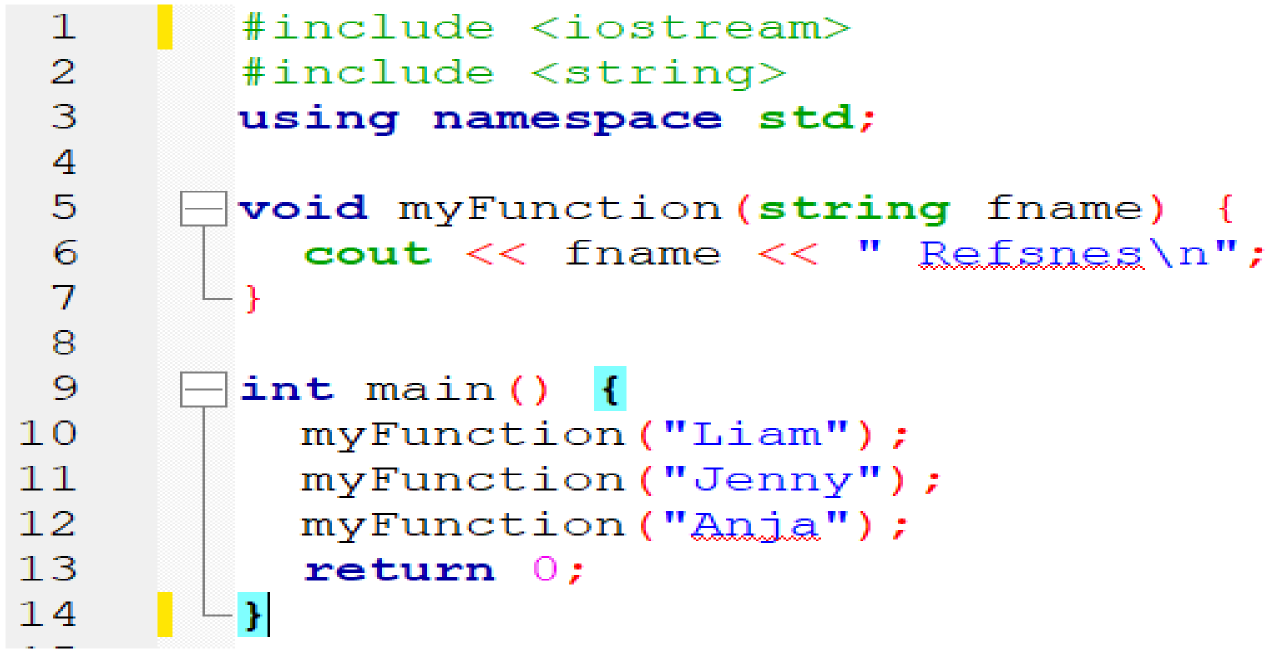
C++ FUNCTION PARAMETERS & ARGUMENTS

- Information can be passed to functions as a parameter.
- Parameters act as variables inside the function.
- Syntax :

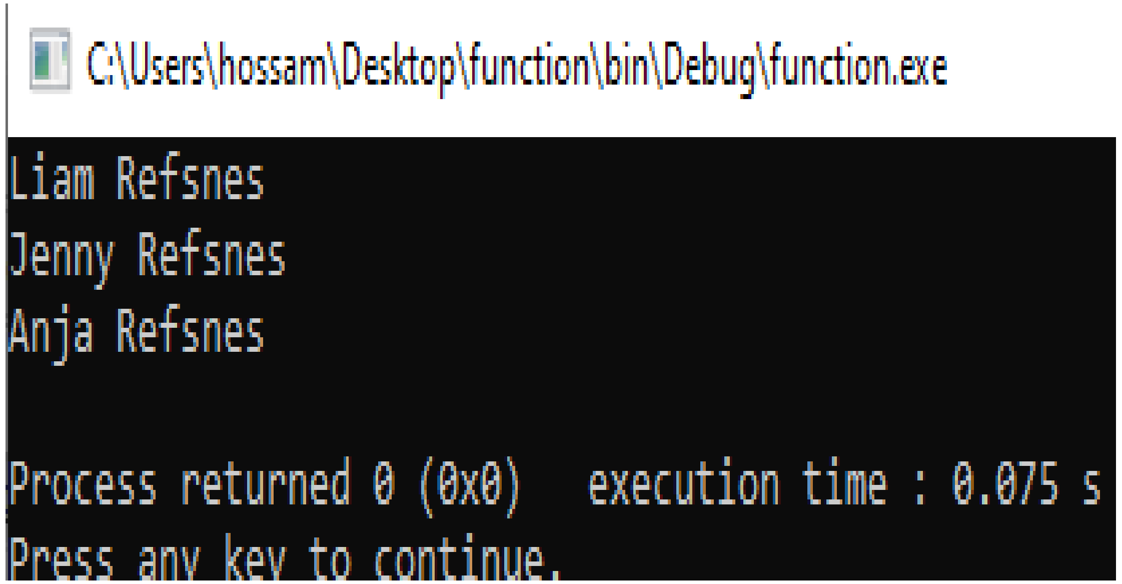
```
returnType functionName (parameter1,
parameter2,...) {
    // function body
}
```

C++ FUNCTION PARAMETERS & ARGUMENTS "CONT"

➢ **Example :**



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

void myFunction(string fname) {
   cout << fname << " Refsnes\n";
}

int main() {
   myFunction("Liam");
   myFunction("Jenny");
   myFunction("Anja");
   return 0;
}
```



```
C:\Users\hossam\Desktop\function\bin\Debug\function.exe
Liam Refsnes
Jenny Refsnes
Anja Refsnes

Process returned 0 (0x0)   execution time : 0.075 s
Press any key to continue.
```

➢ **a function that takes a string called fname as parameter. When the function is called,
we pass along a first name, which is used inside the function to print the full name:**

➢ **When a parameter is passed to the function, it is called an argument.
So, from the example above: fname is a parameter, while Liam, Jenny and Anja**

C++ FUNCTION OVERLOADING

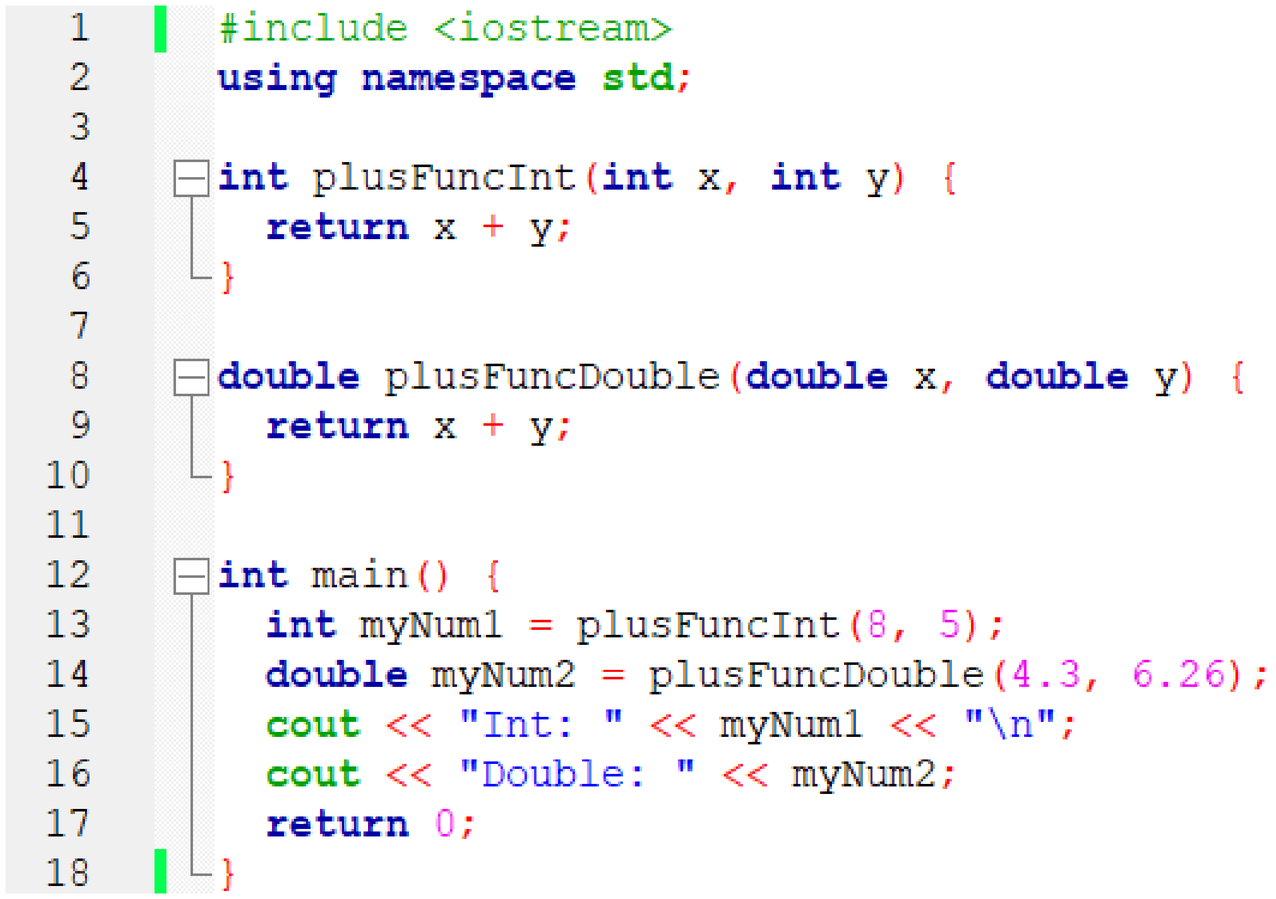
- function overloading, multiple functions can have the same name with different type of parameters :

- Example :

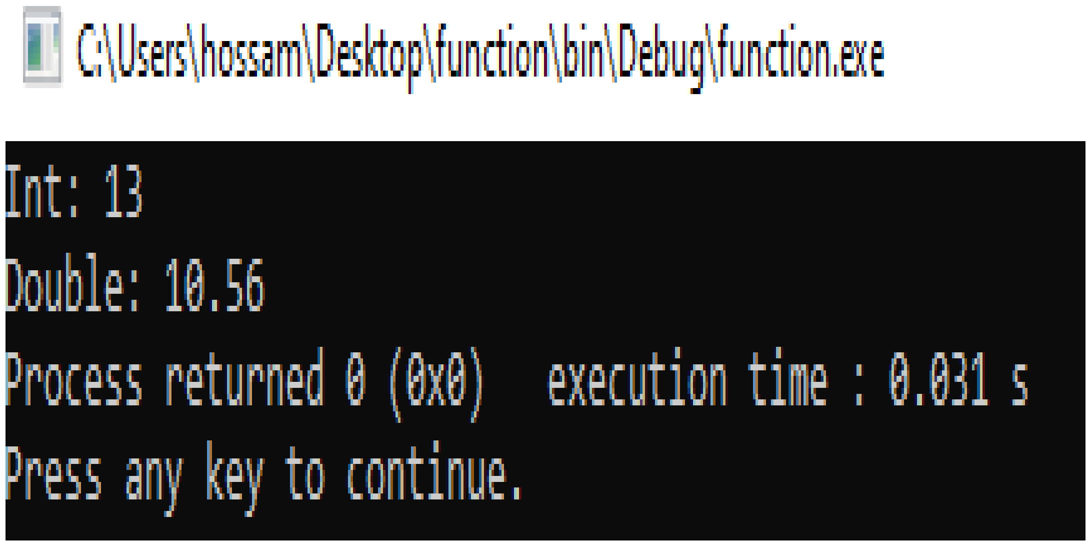
```
int myFunction(int x)
float myFunction(float x)
double myFunction(double x, double y)
```

C++ FUNCTION OVERLOADING "CONT"

➢ **Example : two functions that add numbers of different type:**



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

int plusFuncInt(int x, int y) {
  return x + y;
}

double plusFuncDouble(double x, double y) {
  return x + y;
}

int main() {
  int myNum1 = plusFuncInt(8, 5);
  double myNum2 = plusFuncDouble(4.3, 6.26);
  cout << "Int: " << myNum1 << "\n";
  cout << "Double: " << myNum2;
  return 0;
}
```

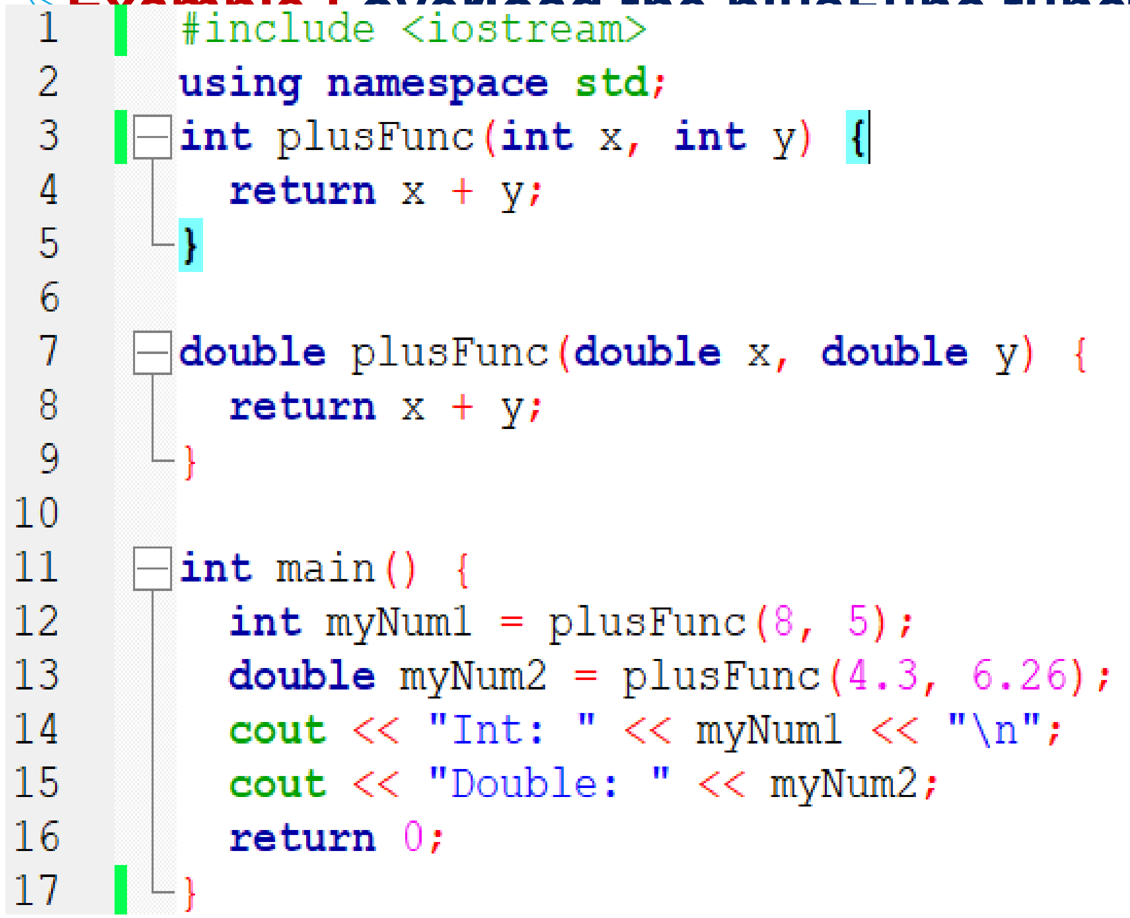


C:\Users\hossam\Desktop\function\bin\Debug\function.exe

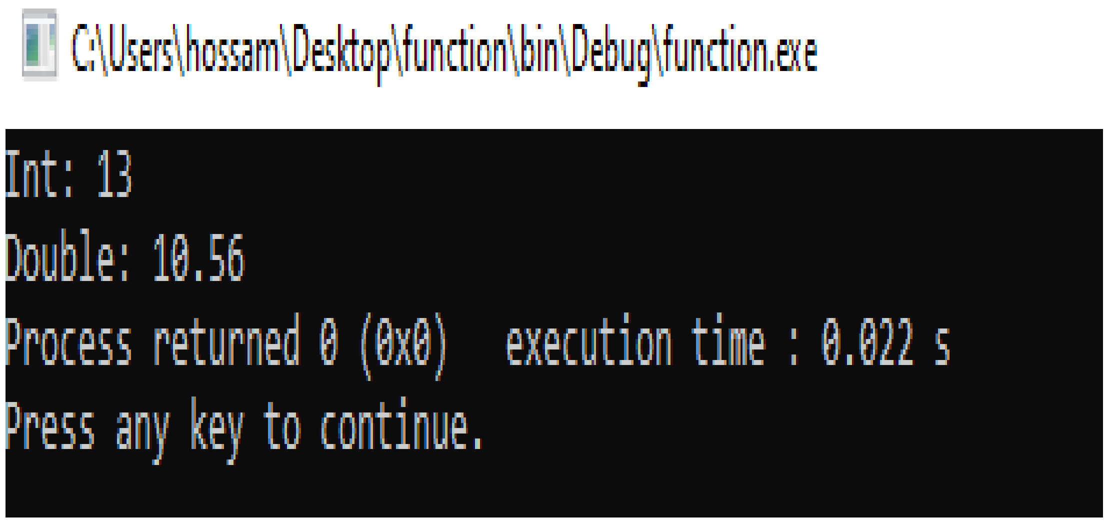
```
Int: 13
Double: 10.56
Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

C++ FUNCTION OVERLOADING “CONT”

- Instead of defining two functions that should do the same thing, it is better to overload one.
- Example : overload the plusFunc function to work for both int and double:



```cpp
#include <iostream>
using namespace std;
int plusFunc(int x, int y) {
  return x + y;
}

double plusFunc(double x, double y) {
  return x + y;
}

int main() {
  int myNum1 = plusFunc(8, 5);
  double myNum2 = plusFunc(4.3, 6.26);
  cout << "Int: " << myNum1 << "\n";
  cout << "Double: " << myNum2;
  return 0;
}
```

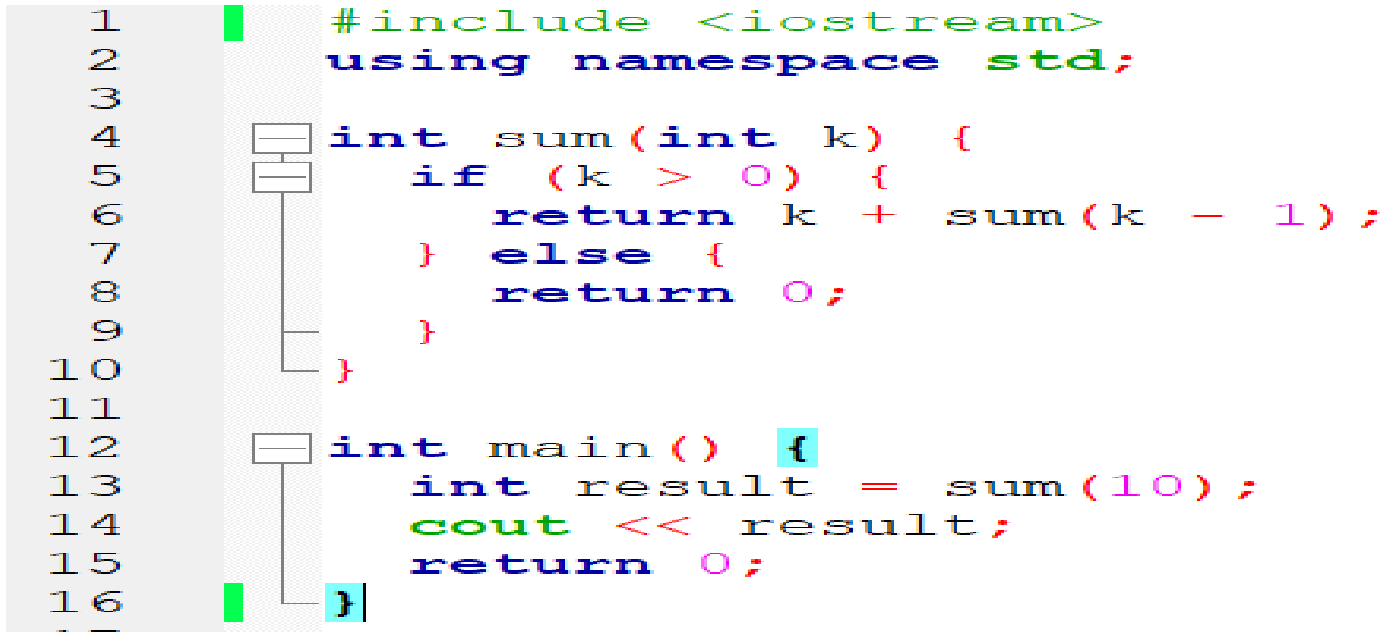


```
C:\Users\hossam\Desktop\function\bin\Debug\function.exe

Int: 13
Double: 10.56
Process returned 0 (0x0)   execution time : 0.022 s
Press any key to continue.
```

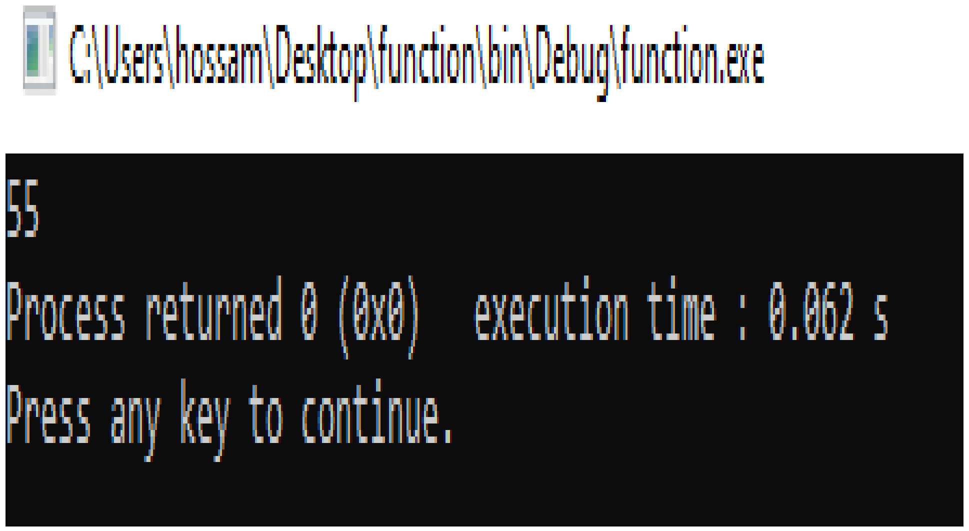
C++ Recursion

❖**Recursion** is the technique of making **a function call itself.**

❖**Example** : recursion is used to add a range of numbers together **by breaking it down into the simple task** of adding two numbers:



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

int sum(int k)  {
  if  (k > 0)  {
    return k + sum(k - 1);
  } else  {
    return 0;
  }
}

int main()  {
  int result = sum(10);
  cout << result;
  return 0;
}
```



C:\Users\hossam\Desktop\function\bin\Debug\function.exe

```
55
Process returned 0 (0x0)   execution time : 0.062 s
Press any key to continue.
```

❖**Example Explained :**

❖When the sum() function is called, it adds parameter k to the sum of all numbers smaller than k and returns the result. When k becomes 0, the function just returns 0.

C++ RECURSION "CONT"

- **Solve of example : When running, the program follows these steps:**
  - **10 + sum(9)**
  - **10 + (9 + sum(8))**
  - **10 + (9 + (8 + sum(7)))**
  - **10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + sum(0)**
  - **10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0**
- **Since the function does not call itself when k is 0, the program stops there and returns the result.**

PASSING ARRAY TO A FUNCTION IN C++

❖In C++, we can pass arrays as an argument to a function. also we can return arrays from a function.

➢ Syntax for Passing Arrays as Function Parameters is :

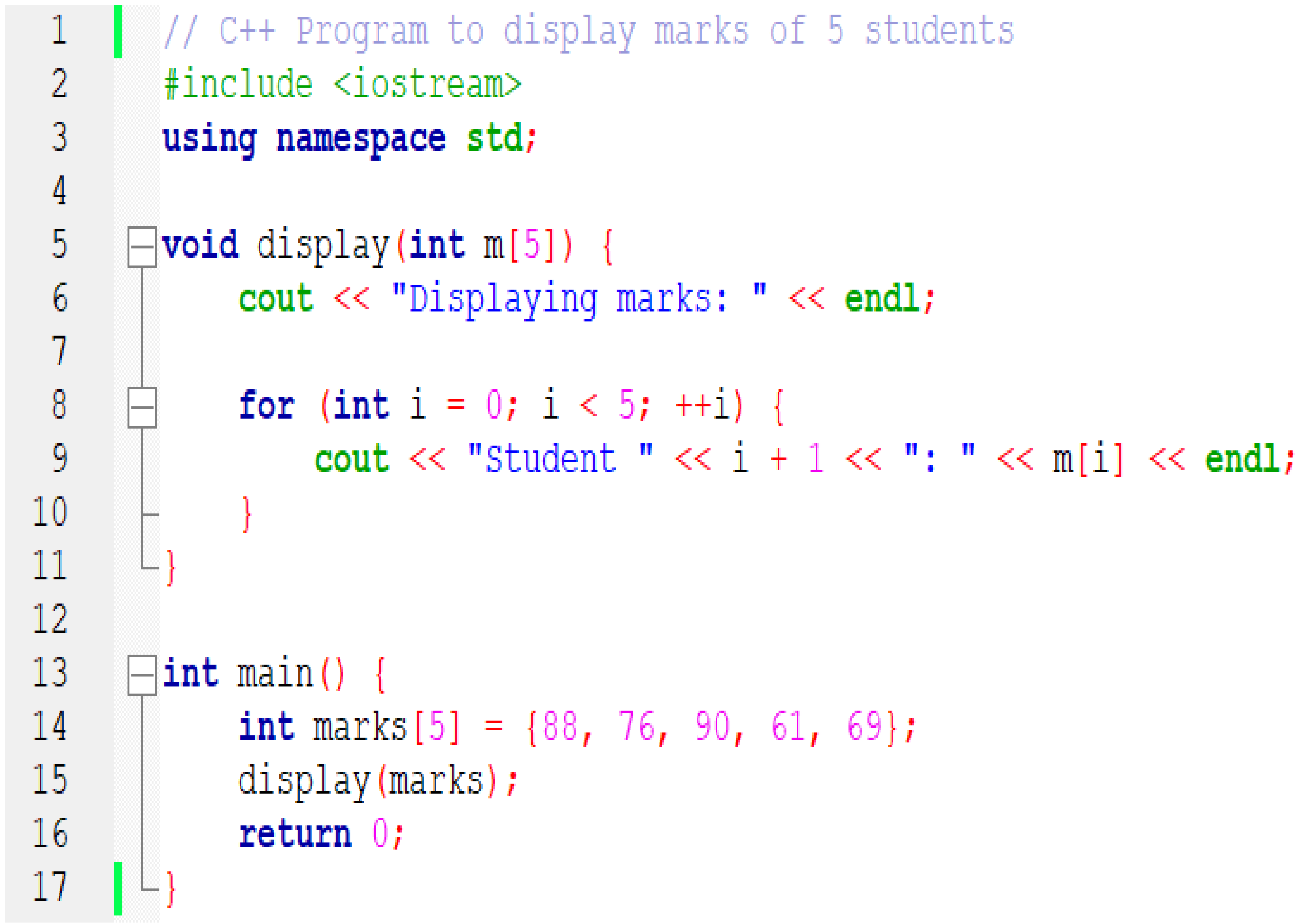
```
returnType functionName(dataType
arrayName[arraySize]) {
    // code
}
```

➢Example :

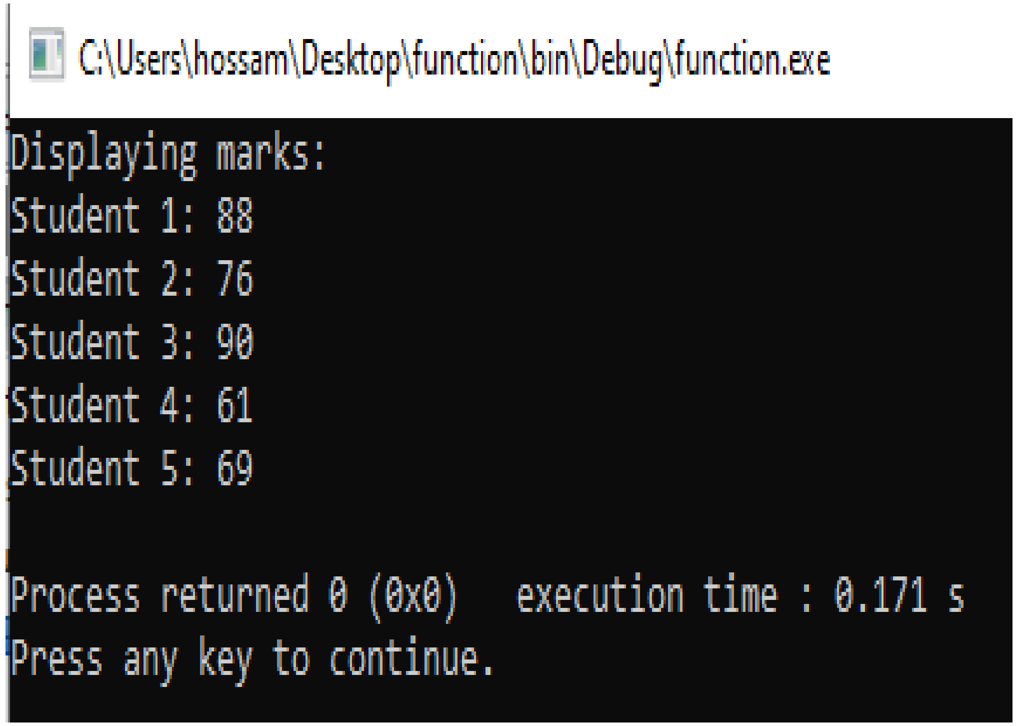
```
int total (int marks[5]) {
    // code
}
```

PASSING ARRAY TO A FUNCTION IN C++ “CONT”

➢ **Example : Passing One-dimensional Array to a Function .**



```cpp
// C++ Program to display marks of 5 students
#include <iostream>
using namespace std;

void display(int m[5]) {
    cout << "Displaying marks: " << endl;

    for (int i = 0; i < 5; ++i) {
        cout << "Student " << i + 1 << ": " << m[i] << endl;
    }
}

int main() {
    int marks[5] = {88, 76, 90, 61, 69};
    display(marks);
    return 0;
}
```

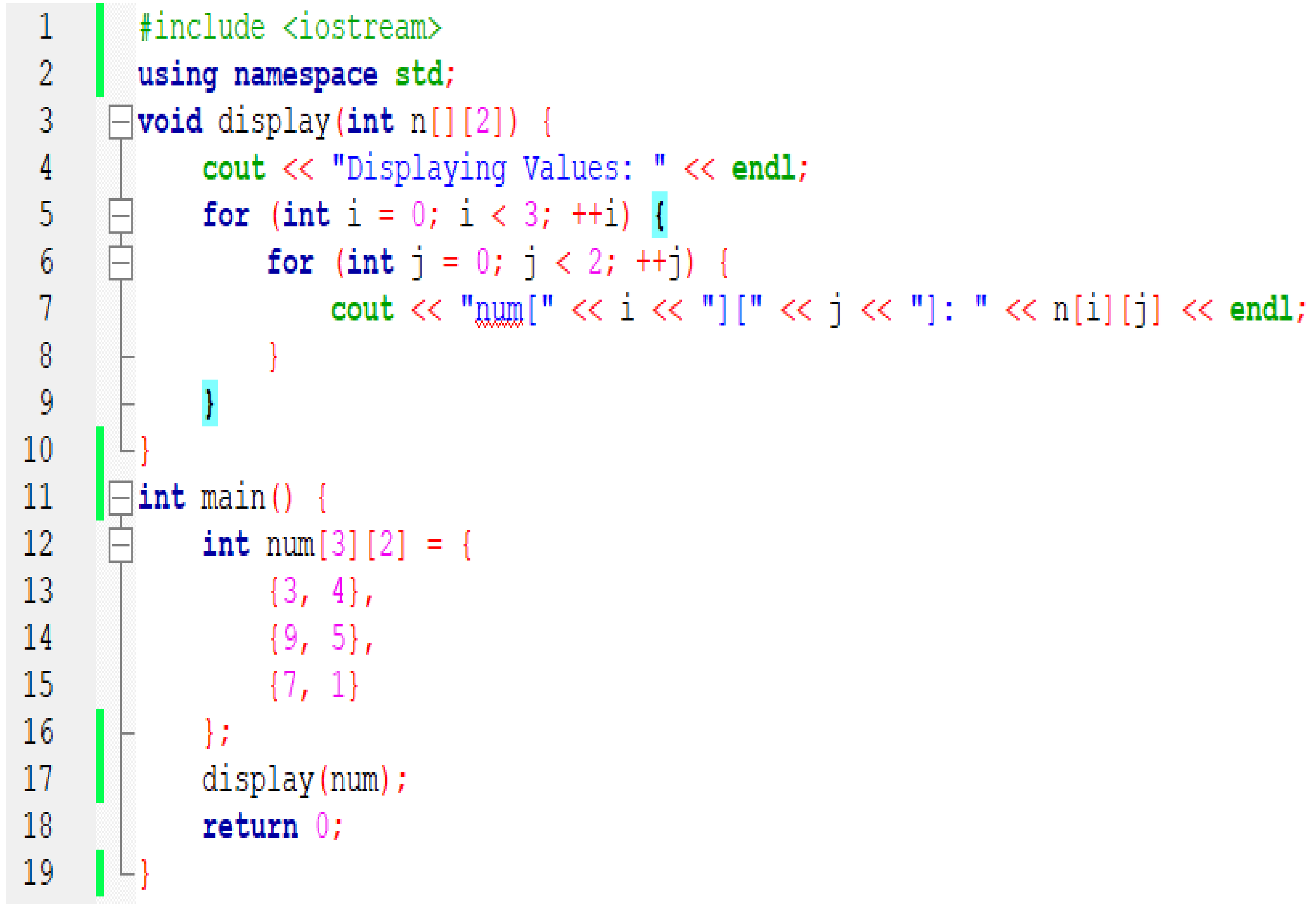


C:\Users\hossam\Desktop\function\bin\Debug\function.exe

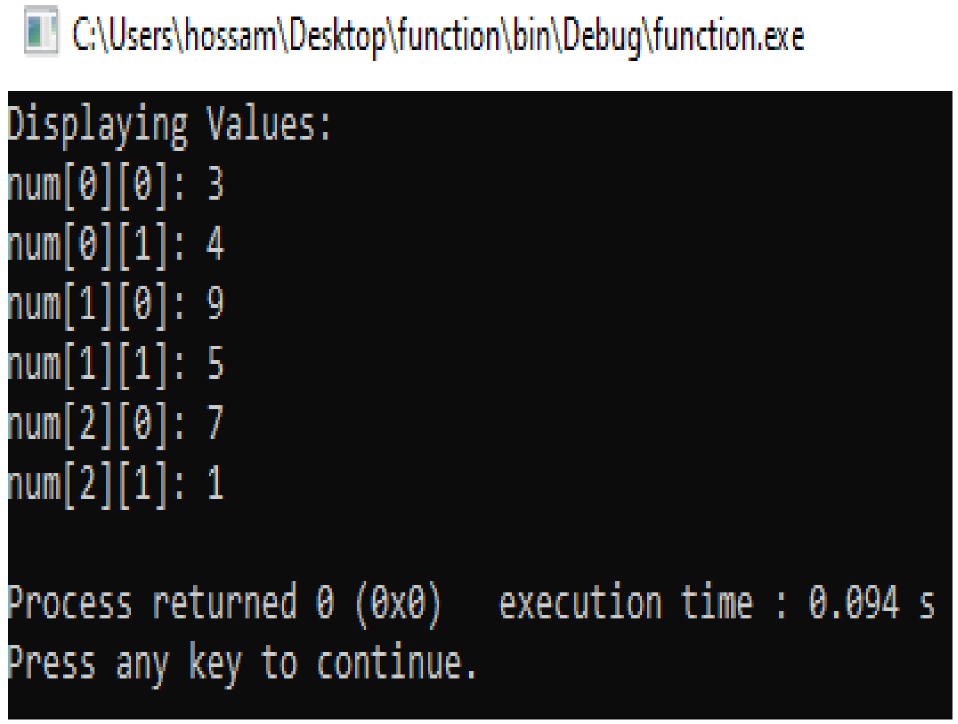
```
Displaying marks:
Student 1: 88
Student 2: 76
Student 3: 90
Student 4: 61
Student 5: 69


Process returned 0 (0x0)   execution time : 0.171 s
Press any key to continue.
```

PASSING ARRAY TO A FUNCTION IN C++ “CONT”

➤ **Example : Passing Multidimensional Array to a Function .**



```cpp
#include <iostream>
using namespace std;
void display(int n[][2]) {
    cout << "Displaying Values: " << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 2; ++j) {
            cout << "num[" << i << "][" << j << "]: " << n[i][j] << endl;
        }
    }
}
int main() {
    int num[3][2] = {
        {3, 4},
        {9, 5},
        {7, 1}
    };
    display(num);
    return 0;
}
```



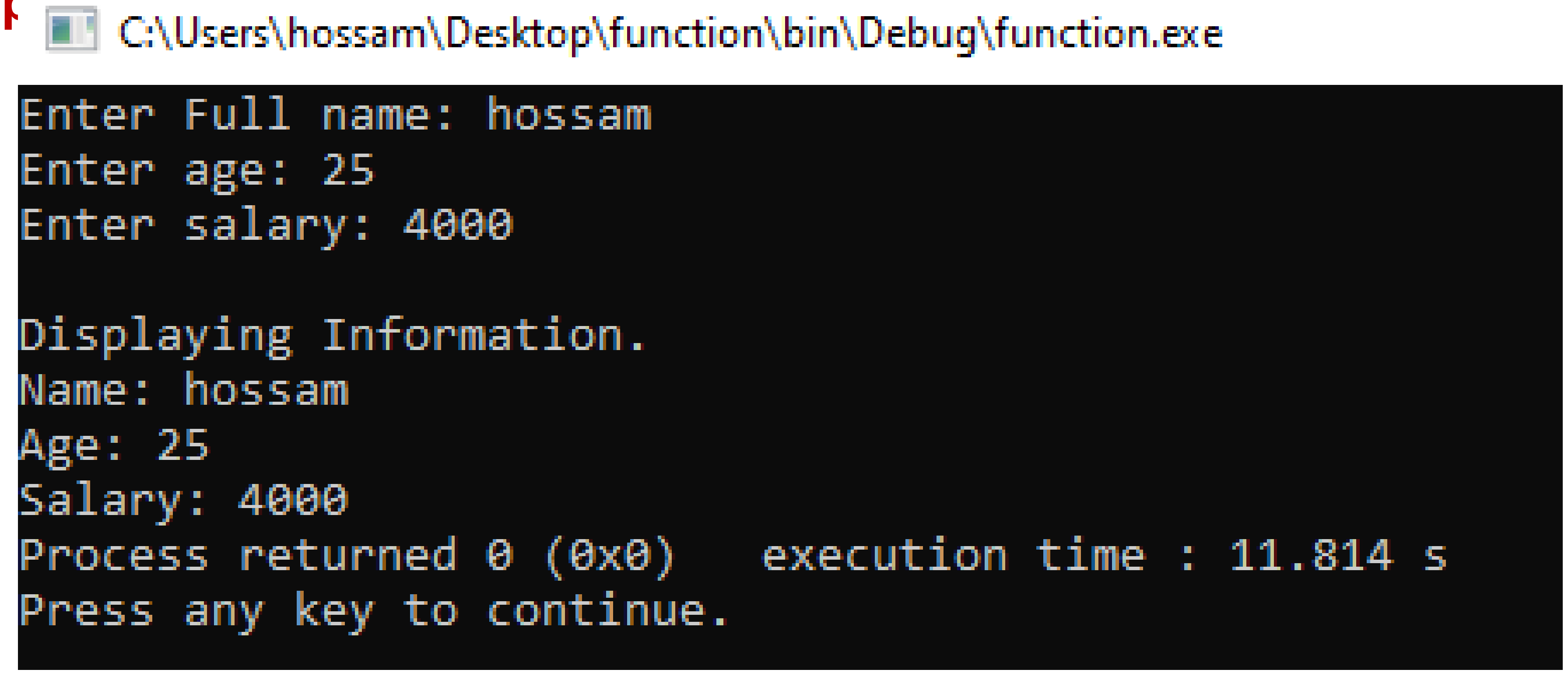
C:\Users\hossam\Desktop\function\bin\Debug\function.exe

```
Displaying Values:
num[0][0]: 3
num[0][1]: 4
num[1][0]: 9
num[1][1]: 5
num[2][0]: 7
num[2][1]: 1

Process returned 0 (0x0)   execution time : 0.094 s
Press any key to continue.
```

PASSING STRUCTURE TO FUNCTION IN C++

❖ A structure variable can be passed to a function in similar way as normal argument.

➢ Examp


C:\Users\hossam\Desktop\function\bin\Debug\function.exe

```
Enter Full name: hossam
Enter age: 25
Enter salary: 4000

Displaying Information.
Name: hossam
Age: 25
Salary: 4000
Process returned 0 (0x0)   execution time : 11.814 s
Press any key to continue.
```

THANKS

Dr/Ghada Maher
Eng/ Hossam Medhat