



# Fundamentals of programming I

Lab 5



# Functions

- ▶ A program may need to repeat the same piece of code at various places.
- ▶ It may be required to perform certain task repeatedly.
- ▶ C++ provides some pre-defined functions, such as **main()**, which is used to execute code. But you can also create your own functions to perform certain actions.
- ▶ **There are two types of functions :**
- ▶ **1.Standard Library Functions:**
  - Predefined in c++
- ▶ **2.User-defined Function:**
  - Created by users.

# Create a Function(User-defined Function)

```
➤ return-type function-name(parameters){  
    // function body  
}
```

## Syntax

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

- **myFunction()** is the name of the function.
- **void** mean that the function does not have a return value.
- **the empty parentheses** mean it doesn't have any parameters.

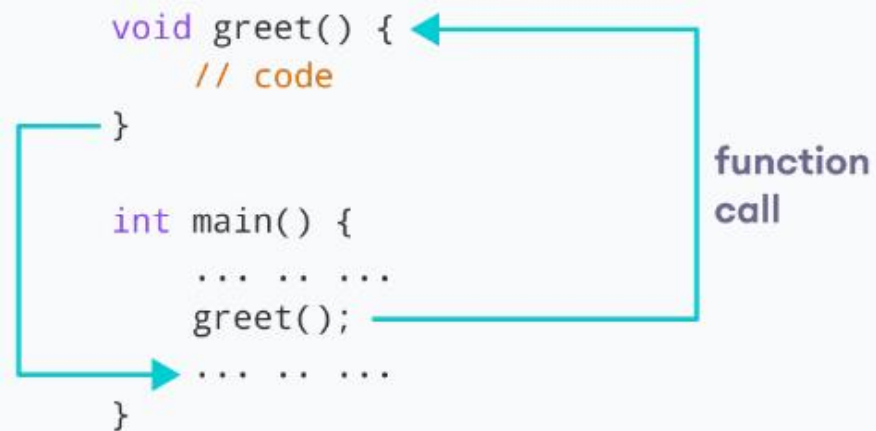
# Call a Function

- Functions invoked by a function-call-statement which consist of it's name and information it needs (arguments)
- Inside main(), call Function():

```
#include<iostream>

void greet() {
    // code
}

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



function call



# Example

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

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

// Outputs "I just got executed!"
```



# A function can be called multiple times:

```
void myFunction() {  
    cout << "I just got executed!\n";  
}  
  
int main() {  
    myFunction();  
    myFunction();  
    myFunction();  
    return 0;  
}  
  
// I just got executed!  
// I just got executed!  
// I just got executed!
```

# Parameters and Arguments

- Information can be passed to functions as a parameter. Parameters act as variables inside the function
- The term parameter refers to any declaration within the parentheses following the function name in a function declaration or definition; You can add as many parameters as you want, just separate them with a comma.
- the term argument refers to any expression within the parentheses of a function call.

## Syntax

```
void functionName(parameter1, parameter2, parameter3) {  
    // code to be executed  
}
```



# Example

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

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

```
// Liam Refsnes  
// Jenny Refsnes  
// Anja Refsnes
```



# Example

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

void display( int n )
{
    cout << "Number is " << n << endl;
}

int main() {
    int a;
    cout << "Enter number" << endl;
    cin >> a;
    display(a);

    return 0;
}
```

## Output

Enter number  
4  
Number is 4

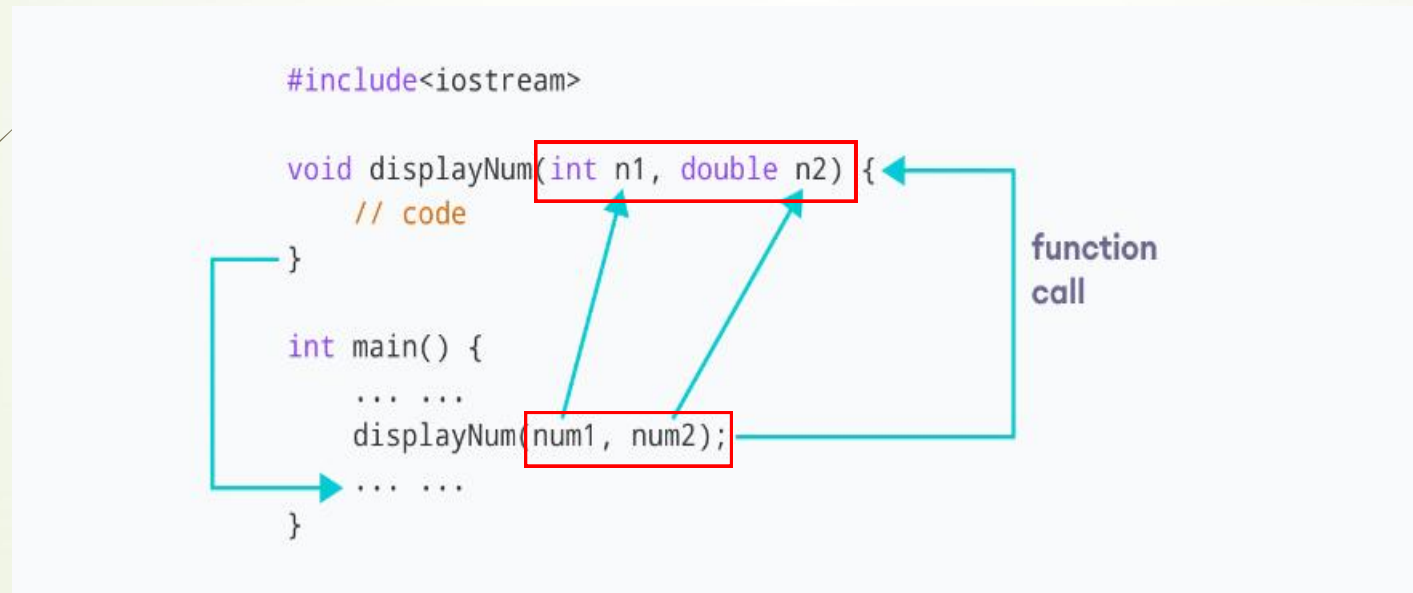
# Default Parameter Value

- You can also use a default parameter value, by using the equals sign (=).
- If we call the function without an argument, it uses the default value.

```
void myFunction(string country = "Norway") {  
    cout << country << "\n";  
}  
  
int main() {  
    myFunction("Sweden");  
    myFunction("India");  
    myFunction();  
    myFunction("USA");  
    return 0;  
}  
  
// Sweden  
// India  
// Norway  
// USA
```

# Multiple Parameters

- Inside the function, you can add as many parameters as you want:



- **Note that:** when you are working with multiple parameters, the function call must have the **same number of arguments** as there are parameters, and the arguments must be **passed in the same order**.



# Example

```
void myFunction(string fname, int age) {  
    cout << fname << " Refsnes. " << age << " years old. \n";  
}  
  
int main() {  
    myFunction("Liam", 3);  
    myFunction("Jenny", 14);  
    myFunction("Anja", 30);  
    return 0;  
}  
  
// Liam Refsnes. 3 years old.  
// Jenny Refsnes. 14 years old.  
// Anja Refsnes. 30 years old.
```

# Example

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

void displayNum(int n1, float n2) {
    cout << "The int number is " << n1<<endl;
    cout << "The double number is " << n2;
}

int main() {

    int num1 = 5;
    double num2 = 5.5;

    // calling the function
    displayNum(num1, num2);

    return 0;
}
```

output

The int number is 5  
The double number is 5.5

# Return Values

- The **void** keyword, used in the previous examples, indicates that the function should not return a value. If you want the function to return a value, you can use a **data type (such as int, string, etc.)** instead of void, and use the return keyword inside the function:

```
#include<iostream>

int add(int a, int b) {
    return (a + b);
}

int main() {
    int sum;
    sum = add(100, 78);
    ... ..
}
```

The diagram illustrates a function call. A teal arrow originates from the expression `add(100, 78)` in the `main` function and points to the `return (a + b);` statement in the `add` function. Another teal arrow starts from the closing brace of the `main` function and points to the ellipsis (`... ..`) below the assignment statement. A teal rectangular box encloses the `return` statement in the `add` function and the `sum = add(100, 78);` line in the `main` function. To the right of this box, the text "function call" is written.

# Example

```
// program to add two numbers using a function

#include <iostream>

using namespace std;

int add(int a, int b) {
    return (a + b);
}

int main() {
    int sum;

    sum = add(100, 78);

    cout << "    Sum = " << sum << endl;

    return 0;
}
```



# The different types of functions

- Function may have no return datatype, and may have no list of input parameters; write void at the beginning of header.
- Function may have no return datatype, and may have list of input parameters; write void at the beginning of header and write the type and name for each input parameter.
- Function may have return datatype, and may have no list of input parameters; write the return data type in the header of function –either primitive or complex.
- Function may have return datatype, and may have list of input parameters; write the return data type in the header of function –either primitive or complex- and write the type and name for each input parameter.





Thank You !