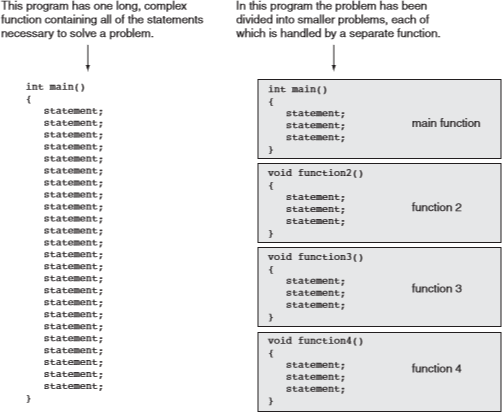
Lab 06 Functions

# (Definition, Calling, Forward Declaration)

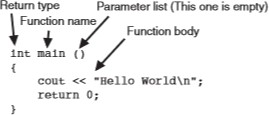
* **Functions**

A function is a collection of statements that performs a specific task. Functions are commonly used to break a problem down into small manageable pieces. Instead of writing one long function that contains all of the statements necessary to solve a problem, several small functions that each solve a specific part of the problem can be written. These small functions can then be executed in the desired order to solve the problem. This approach is sometimes called divide and conquer because a large problem is divided into several smaller problems that are easily solved. The figure 1 illustrates this idea by comparing two programs: one that uses a long complex function containing all of the statements necessary to solve a problem, and another that divides a problem into smaller problems, each of which are handled by a separate function.

# Figure 1



* **Defining a function**

A function definition contains the statements that make up the function. When creating a function, you must write its definition. All function definitions have the following parts:

* + **Return type:** A function can send a value to the part of the program that executed it. The return type is the data type of the value that is sent from the function.
  + **Name:** You should give each function a descriptive name. In general, the same rules that apply to variable names also apply to function names.
  + **Parameter list:** The program can send data into a function. The parameter list is a list of variables that hold the values being passed to the function.
  + **Body:** The body of a function is the set of statements that perform the function’s operation. They are enclosed in a set of braces.

# Void Functions

It isn’t necessary for all functions to return a value. These are called void functions. The displayMessage function, which follows, is an example.

**void displayMessage()**

**{**

**cout << "Hello from the function display Message.\n";**

**}**

# The return Statement

The return statement causes a function to end immediately. When the last statement in a void function has finished executing, the function terminates and the program returns to the statement following the function call. It’s possible, however, to force a function to return before the last statement has been executed. When the return statement is encountered, the function immediately terminates and control of the program returns to the statement that called the function. This is demonstrated in Program 1.

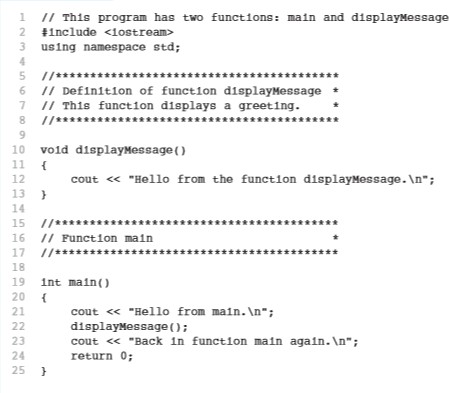
**Program 1**



* **Calling Function**

A function is executed when it is called. Function main is called automatically when a program starts, but all other functions must be executed by function call statements. When a function is called, the program branches to that function and executes the statements in its body. Let’s look at below Program 2, which contains two functions: main and displayMessage .

# Program 2



The function displayMessage is called by the following statement in line 22: displayMessage(); This statement is the function call.

* **Function Header** void displayMessage()

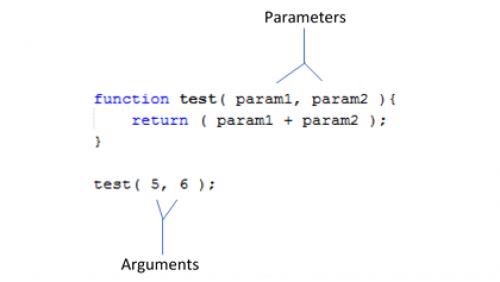
The function header is part of the function definition. It declares the function’s return type, name, and parameter list. It is not terminated with a semicolon because the definition of the function’s body follows it.

* **Function Call** displayMessage();

The function call is a statement that executes the function, so it is terminated with a semi- colon like all other C++ statements. The return type is not listed in the function call, and, if the program is not passing data into the function, the parentheses are left empty.

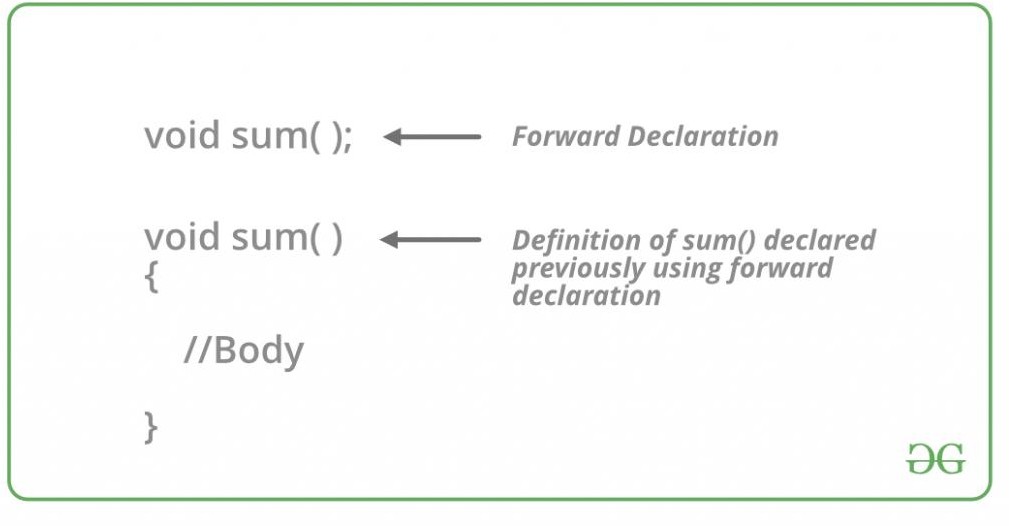
* **Argument**

An **argument**is referred to the values that are passed within a function when the function is called. These values are generally the source of the function that require the arguments during the process of execution. These values are assigned to the variables in the definition of the function that is called. The type of the values passed in the function is the same as that of the variables defined in the function definition. These are also called **Actual arguments** or **Actual Parameters**.



# Forward Declaration of Function

A forward declaration allows us to tell the compiler about the existence of an identifier before actually defining the identifier. In the case of functions, this allows us to tell the compiler about the existence of a function before we define the function’s body.



# Program 3

#include <iostream> Using namespace std;

int add(int x, int y);

// forward declaration of add() (using a function prototype)

int main()

{

cout << "The sum of 3 and 4 is: " << add(3, 4) << '\n';

// this works because we forward declared add() above return 0;

}

int add(int x, int y)

// even though the body of add() isn't defined until here

{

return x + y;

}

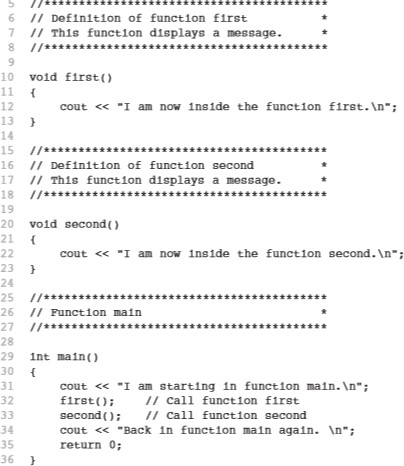
Now when the compiler reaches the call to add in main, it will know what add looks like (a function that takes two integer parameters and returns an integer), and it won’t complain.

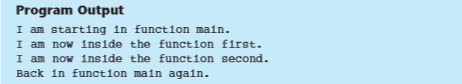
It is worth noting that function prototypes do not need to specify the names of the parameters. In the above code, you can also forward declare your function like this:

int add(int, int); // valid function prototype

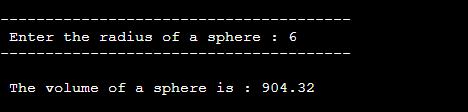
Function prototypes are usually placed near the top of a program so the compiler will encounter them before any function calls.

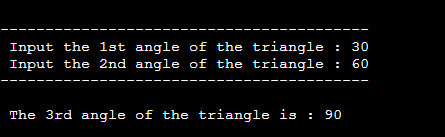
**Program 4**





**Lab Tasks**

1. Write a program in C++ that will take two numbers from user and perform arithmetic operation on it. Make separate function for each arithmetic operation.
2. Write a program in C++ that will take radius of a sphere from the user. Calculate the Volume of sphere in a Function and display its value in main function. The formula for the volume of a sphere is **V = 4/3 πr³**, where r is the radius of the sphere and 𝑃𝐼 𝑖𝑠 𝑎 𝑐𝑜𝑛𝑠𝑡𝑎𝑛𝑡 𝑣𝑎𝑙𝑢𝑒 𝑜𝑓 3.1415
   * The program should take radius from user.
   * The function should take radius as an input parameter
   * The function should be called from main.
   * The function should return value of volume to main.
   * The program prints volume in main.

1. Write a program in C++ that contains a function to calculate third angle of triangle.
   * The program should take two angles from user.
   * The function should take two angles as input parameters.
   * The function will calculate the third angle and return it to main.
   * The program should print value of third angle in main.

**Submission Instructions**

1. Create a new folder/directory with name NAME\_ROLLNO\_SECTION.
2. Move all of your .cpp files and screenshot of each question to this newly created directory and compress it into .zip file.
3. Now you have to submit this zipped file on Google Classroom