Full Name		Student ID	

IP Address		PHY/MAC Address	

Lab 6 | Functions

By the end of this section, the students should be able to:

- Write the definition, declaration, and call for a user-defined function.
- Apply a few user-defined functions in their program.

6.0 Introduction

In C++, a program is made of one or more functions, whereby only one of them must be named `main`. A function is a collection of statements that performs a specific task. The execution of a program always starts with function `main`. A function can call other functions in order to execute those functions doing their tasks as part of the program's job. A function in C++ can return a value, produce side effects, or both. A function can be called for its returned value, its side effect, or both.

6.1 | Theoretical Background

6.1.1 | User-defined Function

- In any C++ program, you can define your own function. This type of function is called a user-defined function.
- There are three important elements in a program that uses a user-defined function:

- **Function declaration** (also known as function prototype): Consists of a statement similar to the function header. Is a way to notify the compiler about the user-defined function in the program.
- **Function definition**: Contains the C++ codes required to complete the task of the function.
- **Function call**: A statement to execute the function.

A. Function declaration of prototype

- Consists of three parts: the return type, function name, and the parameter list (same as in function header).
- Terminated with a semicolon (;).
- Placed in the global area of the program.
- Format:

```
return_type function_name(parameter_list);
```

B. Function definition

A. All function definitions have two parts: *function header* and *function body*.

B. *Function header* consists of:

   a. `return_type`: The return type is the data type of a value that is returned (sent) from the function.

   b. `function_name`: The same rules that apply to variable names also apply to function names.

   c. `parameter_list`: Is a list of variable declarations that will hold the values that are passed (given) to the function (if any).

   d. *Function_body* consists of local declarations and a set of statements (coding) that will perform the function's task.

e. Format:

```
Return_type function_name (parameter_list)
{
      Local declarations
      Statements
}
```

C. Function call

- Contains the function name followed by () and a semicolon.
- Might have a returned value (from the called function) that can be:
    - assigned to a variable, or
    - used by `cout`, or
    - used in arithmetic expressions.
- Practically, to prepare a user-defined function in a program, it is firstly defined (once only), secondly declared (once only) and lastly called (can be more than once, and from any function).

6.2 | Step by Step Examples

6.2.1 | Example 1: User-defined Function

```
1   /*We describe the development of two functions that are
2   called from main. These functions calculate the potential
3   and kinetic energy of groups of kilogram-sized blocks.*/
4   #include <iostream>
5   using namespace std;
6   void potential_energy(); //function declaration (prototype)
7   void kinetic_energy(int, double); //function declaration
8
9   int main() //function definition
10  {
11        int mass = 15;
12        double velocity = 308.24;
13
14        cout<<"The value of mass in main is mass= "
15        <<mass<<endl;
16
17        potential_energy(); //function call
18        kinetic_energy(mass, velocity); //function call
19
20        cout<<"The value of mass in main is still mass= "
21            <<mass<<endl;
22        return 0;
23  }
24
25  void potential_energy() //function definition
26  {
27        int mass = 6;
28        double pe, height = 5.2;
29        const double g = 9.81;
30
31        pe = mass * g * height;
32        cout<<"Potential energy= "<<pe<<endl;
33  }
34
35  void kinetic_energy(int m, double v) //function definition
36  {
37        double ke;
38
39        ke = 0.5 * m * v * v;
40        cout<<"Kinetic energy = "<<ke<<endl;
41  }
```

```
Output:
The value of mass in main is mass=15
Potential energy = 306.072
Kinetic energy = 712589
The value of mass in main is still mass=15
```

Line	Description

6
7
	`void potential_energy();`
	`void kinetic_energy(int, double);`
	
	● Lines 6 and 7 are function declarations (or prototypes). Function declaration indicates the function name, return type and the types of value passed to the function.
	● Line 6: function `potential_energy` returns no value because it is of type `void`. It has no parameters. This means that `potential_energy` is not passed (or will not receive) any value when it is called.
	● Line 7: function `kinetic_energy` returns no value because it is of type `void`. It has two parameters. The first parameter in an int and the second parameter is a `double`. This means that `kinetic_energy` is passed (or will receive) an `int` value and a `double` value when it is called.

17	`potential_energy();` ● This is a function call. A function call transfers program execution control to the called function. ● After executing line 17, control goes to function `potential_energy` and the next lines of code executed are lines from 24-32 which are within the body of `potential_energy`. ● After `potetial_energy` has finished executing, control goes back to the location of the call to `potential_energy` (which is line 17 in function main). The next line executed is line 18.
18	`kinetic_energy(mass, velocity);` ● A call to function `kinetic_energy`. This call causes the values of mass and velocity to be passed from main to kinetic energy and causes execution control to be passed over to function `kinetic_energy`.

25
26
27
28
	`void potential_energy()`
	`{`
	`  int mass = 6;`
	`  double pe, height = 5.2;`

```
29        const double g = 9.81;
30
31        pe = mass * g * height;
32        cout<<"Potential energy= "<<pe<<endl;
33  }
```

- A function definition, which includes the function's executable statements. In this example, `potential_energy` calculates $E_p=mgh$, and prints it.

```
35  void kinetic_energy(int m, double v)
36  {
37        double ke;
38
39        ke = 0.5 * m * v * v;
40        cout<<"Kinetic energy = "<<ke<<endl;
41  }
```

- A function definition, which includes the function's executable statements. In this example, `kinetic_energy` calculates $E_k=0.5\,mv^2$, and prints it.

6.3 | Exercises

1. Write a program that has a function to print your name. The function would be called from the `main` function.

2. Write a program that has a definition of a function named `swap` with 2 parameters of type `float`, and its purpose is to display the swapped value. In function `main`, display the original value and then call function `swap` to do its task before the program ends.

3. A litre is 0.264179 gallons. Write a program with the `main` function that will read in the number of litres of petrol consumed by the user's car and the number of kilometres travelled by the car and will then output the number of kilometres per gallon the car delivered. Define a function to compute the number of kilometres per gallon. Your program should use a globally defined constant for the number of litres per gallon.

6.4 | Self-Review Questions

1) State whether the following statements are **TRUE** or **FALSE**.

a)	Functions should be given names that reflect their purpose.	
b)	The execution of a program always starts with function `main`.	
c)	Function headers are terminated with a semicolon.	
d)	A function body must be enclosed within a pair of braces.	
e)	When a function terminates, it always branches back to function `main`, regardless of where it was called from.	
f)	Arguments are passed to the function parameters in the order they appear in the function call.	
g)	In a function prototype, the names of the parameter variables may be left out.	

2) Indicate which of the following is the function prototype, the function header, and the function call:

a) `void display(double num)`

b) `void display(double)`

c) `display(12.34);`

Answer:

3)

a) Write a header for a function named `disCal`. The function should return a `double` and have two `double` parameters: `time` and `speed`.

b) Write the body of the `disCal` function that calculates **distance = time X speed**.

c) Write an example of the call to `disCal` function that passes the value 2.5 and 60.5 and assigns its return value to variable `result`.

Answer:

4) What is the result of the program's execution if the input is 5.6?

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

const double PI = 3.142;

float getRadius();
double circumference(float);
void printCircumference(float rad);

int main()
{
     float radius;
     radius = getRadius();
     printCircumference(radius);
     cout<<endl;
     return 0;
}

float getRadius()
{
     float r;
     cout<<"Enter radius of the circle: ";
     cin>>r;
     return r;
}

double circumference (float r)
{
     return (2 * PI * r);
}

void printCircumference(float rad)
{
     cout<<"The circumference of the circle with radius = "
         << rad <<" is ";
     cout<<circumference(rad);
}
```

Answer:

5) Find the errors, if any, in the following function declarations:

a)	`void (function1) void;`
b)	`void function2 (void)`
c)	`void function (n, x, a, b);`
d)	`void function1 (int, double float, long int, char);`
e)	`void function1 (int a, double b, float, c);`
Answer:	

6) Each of the following functions has errors. Identify the errors as many as you can.

a)	`void sum (int sum1, sum2, sum3)`
	`{`
	`  return sum1 + sum2 + sum3;`
	`}`

b)	`double average(int value1, int value2, int value3)`
	`{`
	`  double average;`
	`  average = value1 + value2 + value3 / 3;`
	`}`

c)	`void area(int length = 30; int width)`
	`{`
	`  return length * width;`
	`}`

Answer:	