Programming Fundamentals

Lab Manual - Week 05

Introduction

Welcome to your favorite programming Lab. In this lab manual, we shall work together to learn and implement new programming concepts.

Skills to be learned:

- Solve problems by using User-defined Functions and Pre-defined Functions.
- Understanding and Implementing Void and Value Returning Functions.

Let's do some coding.

Skill: Solve problems by using User-defined Functions and Pre-defined Functions.

Introduction

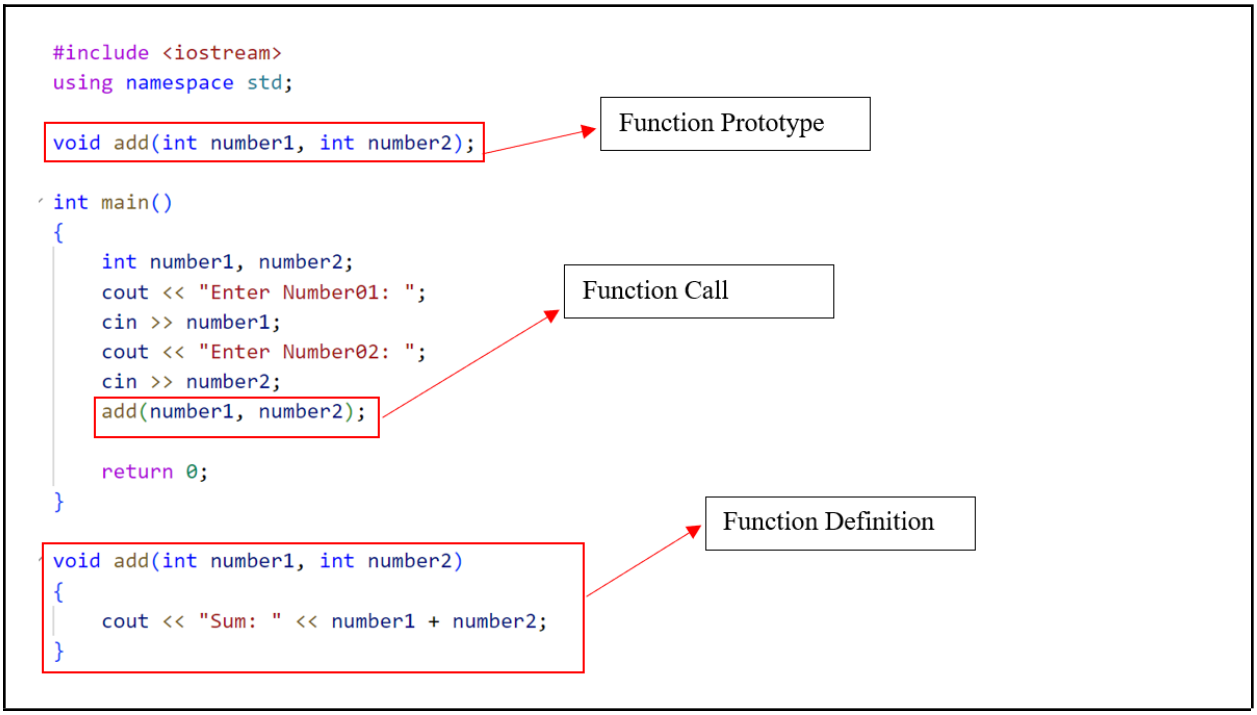
By this week, you have learned how to write a program that contains functions. The functions are the reusable pieces of code that can be executed multiple times using the function call. However, there are two types of functions that include

- User-defined functions
- Pre-defined functions

User-defined functions

The type of functions created and implemented by the user are referred to as user-defined functions. Consider the following example for understanding the user-defined functions.

Task 01(WP): Create a function that takes two numbers from the user and prints their sum on the screen.



All such functions that are defined and created by the user are referred to as user-defined functions.

Pre-defined functions

These functions are pre-defined by the Programmers of the language and are not implemented by the user himself. In most cases, the user just includes a library file that provides that Pre-Defined function.
Consider the example below for further understanding:

Task 02(WP): Write a program that halts the execution for 200 milliseconds using the Sleep function.

Skill: Solve problems by using User-defined Functions and Pre-defined Functions

```cpp
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    while (true)
    {
        cout << "Name: ";
        Sleep(200);
    }
    return 0;
}
```

It is a predefined function that halts the execution for the given time.

Notice that we have included **windows.h** library file in our program and we can use the **Sleep()** function **that accepts parameters** as time in milliseconds and halts the execution of the program for that much time.
Sleep() is a pre-defined function that we have used in this example.

There is a similar library of pre-defined mathematical functions that we can use in our programs to solve different mathematical problems. Consider the below-mentioned tasks for better understanding.

Task 03(WP): Write a C++ program that takes two numbers from the user and prints the greater number on the screen.

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

int main()
{
    int number1, number2;
    cout << "Enter Number01: ";
    cin >> number1;
    cout << "Enter Number02: ";
    cin >> number2;
    cout << "Greater Number: " << max(number1, number2);
    return 0;
}
```

This function returns the greater number from the given two parameters

Skill: Solve problems by using User-defined Functions and Pre-defined Functions

Notice that we have included the library **cmath** in our program that enables us to use the **max()** function that accepts two numbers as parameters and returns the greater. By now, you should have the idea that the **max()** is a predefined function which has been defined in the **cmath** library.

Congratulations !!!! You have learned the difference between user-defined and pre-defined functions.

Task 01(OP):

Write a C++ program that takes two numbers as input from the user and prints the minimum out of two on the screen. **Hint: min**(number1,number2)

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task1.exe
Enter the first number: 34.5
Enter the second number: -76.4
The minimum of 34.5 and -76.4 is: -76.4

G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task1.exe
Enter the first number: -100
Enter the second number: 3
The minimum of -100 and 3 is: -100
```

Food for Thought: What is the datatype of parameters and the return type of the **min** function?

Task 02(OP):

Write a C++ program that takes two numbers from the user and takes the power of the first number as the second number entered by the user. **Hint: pow**(number1,number2)

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task2.exe
Enter the base number: 5
Enter the exponent: 2
5 raised to the power 2 is: 25

G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task2.exe
Enter the base number: 4.5
Enter the exponent: 2
4.5 raised to the power 2 is: 20.25
```

Skill: Solve problems by using User-defined Functions and Pre-defined Functions

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task2.exe
Enter the base number: 9
Enter the exponent: 0.5
9 raised to the power 0.5 is: 3
```

Task 03(OP): Write a C++ program that takes a number from the user as input and print its square root on the screen. **Hint: sqrt**(number)

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task3.exe
Enter a number: 196
The square root of 196 is: 14
```

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task3.exe
Enter a number: 134.6
The square root of 134.6 is: 11.6017
```

Following are a few other pre-defined in the cmath library. Try out for yourself.

Function	**Description**
cbrt(x)	Returns the cube root of x
ceil(x)	Returns the value of x rounded up to its nearest integer
floor(x)	Returns the value of x rounded down to its nearest integer
cos(x)	Returns the cosine of x (x is in radians)
sin(x)	Returns the sine of x (x is in radians)
tan(x)	Returns the tangent of an angle (x is in radians)

Task 04(CP):

Imagine you are a field researcher working on a project to measure and document the heights of trees in a forest. You are equipped with a distance measuring device and an angle measurement tool.

Skill: Solve problems by using User-defined Functions and Pre-defined Functions

Programming Fundamentals

Lab Manual - Week 05

One day, you come across a tall tree in the forest, and you want to determine its height without having to climb it. You have a distance measuring tool that can measure the horizontal distance from your position to the base of the tree, and you also have an angle measurement tool that can measure the angle of elevation from your position to the top of the tree.

To calculate the height of the tree, you need to:

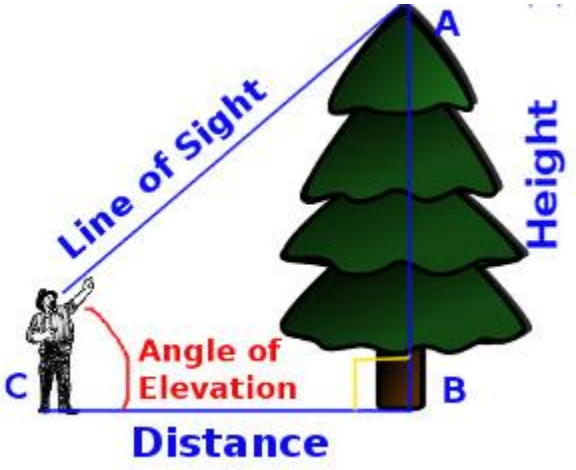
1. Measure the horizontal distance from your position to the base of the tree (in feet).
2. Measure the angle of elevation from your position to the top of the tree (in degrees).

Your task is to create a C++ program that takes these measurements as input from you and calculates the height of the tree using trigonometric principles. Once the program calculates the height, it will display the result on the screen.

By using this program, you can quickly estimate the height of trees in the forest without the need for climbing or specialized equipment, which is valuable information for your research project.

Remember: 1 radian = 57.2958 degrees

"Let's apply some trigonometric magic for the first time!"



Skill: Solve problems by using User-defined Functions and Pre-defined Functions

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task4.exe
Enter the distance from the base of the tree (in feet): 43
Enter the angle of elevation (in degrees): 30
The height of the tree is: 24.8261 feet
```

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task4.exe
Enter the distance from the base of the tree (in feet): 500
Enter the angle of elevation (in degrees): 2
The height of the tree is: 17.4604 feet
```

Task 05(CP):

Imagine you are a software developer working on a project for a math education program. As part of the program, you need to create a tool that helps students and teachers solve quadratic equations of the form:

```
ax^2 + bx + c = 0
```

Where:

- `a` represents the coefficient of the quadratic term.
- `b` represents the coefficient of the linear term.
- `c` represents the constant term.

Your task is to develop a C++ program that allows users to input the values of `a`, `b`, and `c`. The program will then calculate the roots of the quadratic equation using the quadratic formula and display the results.

The program should handle different cases:

1. If the discriminant (the value under the square root in the quadratic formula) is positive, there are two real and distinct solutions.
2. If the discriminant is zero, there is one real solution (a repeated root).
3. If the discriminant is negative, there are no real solutions, but there will be complex solutions.

$$\underline{\text{Determinant}} = b^2 - 4ac$$

If Determinant = 0, then one real root
If Determinant > 0, then two real roots
If Determinant < 0, then two complex root

Skill: Solve problems by using User-defined Functions and Pre-defined Functions

Programming Fundamentals

Lab Manual - Week 05

If determinant > 0,

$$root1 = \frac{-b + \sqrt{(b^2 - 4ac)}}{2a}$$

$$root2 = \frac{-b - \sqrt{(b^2 - 4ac)}}{2a}$$

If determinant = 0,

$$root1 = root2 = \frac{-b}{2a}$$

If determinant < 0,

$$root1 = \frac{-b}{2a} + i\frac{\sqrt{-(b^2 - 4ac)}}{2a}$$

$$root2 = \frac{-b}{2a} - i\frac{\sqrt{-(b^2 - 4ac)}}{2a}$$

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task5.exe
Enter the value of a: 1
Enter the value of b: 2
Enter the value of c: 3
Complex Solutions: x = -1 + 1.41421i and x = -1 - 1.41421i
```

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task5.exe
Enter the value of a: 5
Enter the value of b: 6
Enter the value of c: 1
Solutions: x = -0.2 and x = -1
```

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task5.exe
Enter the value of a: 1
Enter the value of b: 2
Enter the value of c: 1
Solution: x = -1
```

Conclusion

Functions	Description
User-defined Functions	These are the type of the functions that are defined and implemented by the user. The user has to do the following in order to use user-defined functions

Skill: Solve problems by using User-defined Functions and Pre-defined Functions

	1. Define **Function Prototype** 2. Define **Function Definition** 3. Make **Function Call** when the functionality is required.
Pre-defined Functions	These are the type of functions that are already defined and created by programmers. The user has to do the following for using these functions. 1. **Include** the library that has the definition of the function. 2. Make a **Function Call** when the functionality is required.

Skill: Solve problems by using User-defined Functions and Pre-defined Functions

Programming Fundamentals

Lab Manual - Week 05

Skill: Understanding and Implementing Void and Value Returning Functions

Introduction

Functions **may return a value** to the **calling function** depending on the **user requirement**.

Consider the previously used example mentioned below

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

void add(int number1, int number2);

int main()
{
    int number1, number2;
    cout << "Enter Number01: ";
    cin >> number1;
    cout << "Enter Number02: ";
    cin >> number2;
    add(number1, number2);

    return 0;
}

void add(int number1, int number2)
{
    cout << "Sum: " << number1 + number2;
}
```

Calling Function:
A function that has function call to some other function

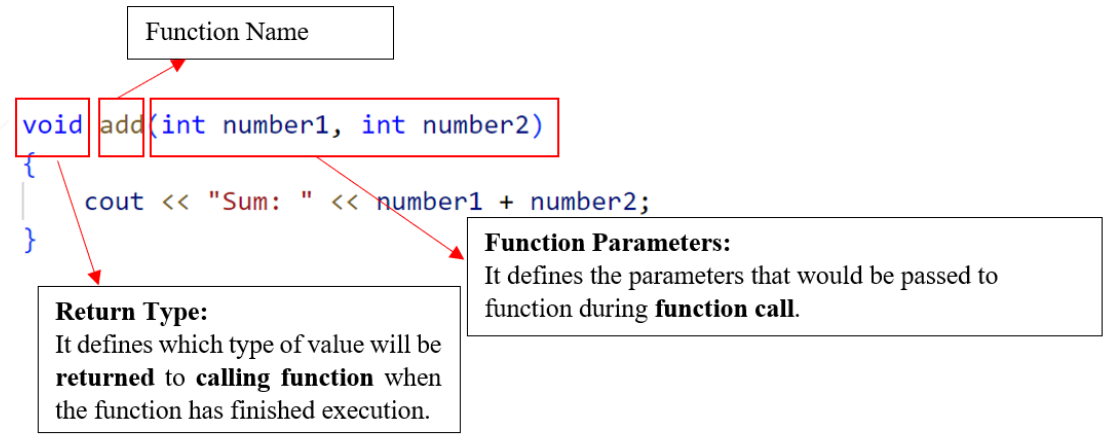
Function Call

In this example, the **calling function** is **main()** that has an **add() function call**.

So, how do we know if a function is returning any value or not? Recall this concept again

Programming Fundamentals

Lab Manual - Week 05



Function Return Type defines the type of value that will be returned to the calling function when the function has finished execution.

Following are the type of values that can be returned by a function

Datatype	**Description**
void	This datatype means that the **function will not return** any value
bool	This datatype means that the function will return **a boolean type value or variable**. True/False
int	This means that the function will return **an integer type value or variable**.
float	This means that the function will return a **float type value or variable**.
char	This means that the function will return a **character type value or variable**.
string	This means that the function will return a **string type value or variable**.

Task 01(WP): Write a function that takes a number from the user and returns it after multiplying it with 5.

Skill: Understanding and Implementing Void and Value Returning Functions

Programming Fundamentals

Lab Manual - Week 05

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

int myFunction(int number);

int main()
{
    int number, result;
    cout << "Enter Number: ";
    cin >> number;

    result = myFunction(number);

    return 0;
}

int myFunction(int number)
{
    int total;
    total = number * 5;
    return total;
}
```

Function Prototype:
The **int** return type defines that the function will return an integer value or int type variable.

Now, we know that this **function will return an integer** value so we can store it into int type variable for later use.

return is used to return the desired value to the calling function

1. The **int** return type in the function prototype defines that the function will return an integer variable.
2. Notice, that the function is returning an integer type variable **total**
3. Now, in the calling function, the **returned value** is stored in the **result** so it may be used in the future.

Great Work Students !! You have understood the various return types in the functions. Add this one to your skill set. 👏

Conclusion

Functions have a return type that defines **which kind of data will be sent back to the calling function** once the function has completed execution.

Task 06(CP):

Write a function named **checkAlphabetCase** for checking whether the alphabet entered by the user is in small case or in capital case (Suppose user will only enter 'A' or 'a'). Make a function that takes 1 Character as input, does processing according to the input

Skill: Understanding and Implementing Void and Value Returning Functions

and then returns the string. String is “You have entered Capital A” if the user enters ‘A’, otherwise “You have entered small A”.

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task6.exe
Enter a character (A/a): A
You have entered Capital A
```

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task6.exe
Enter a character (A/a): a
You have entered small a
```

Task 07(CP): Create a function that takes a number as an argument and returns true or false depending on whether the number is symmetrical or not. A number is symmetrical when it is the same as its reverse. (**The user will enter five digit number only**)

Examples:

IsSymmetrical(12567) → false
IsSymmetrical(12321) → True

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task7.exe
Enter a five-digit number: 12321
The number is symmetrical.
```

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task7.exe
Enter a five-digit number: 12345
The number is not symmetrical.
```

Task 08(CP): Create a function that determines whether a number is Oddish or Evenish. A number is Oddish if the sum of all of its digits is odd, and a number is Evenish if the sum of all of its digits is even. If a number is Oddish, return "Oddish". Otherwise, return "Evenish". (**The user will enter five digit number only**)

OddishOrEvenish(12345) → "Oddish"
// 1 + 2 + 3 + 4 + 5 = 15
// 15 % 2 = 1

OddishOrEvenish(12348) → "Evenish"
// 1 + 2 + 3 + 4 + 8 = 18

Skill: Understanding and Implementing Void and Value Returning Functions

```
// 18 % 2 = 0
```

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task8.exe
Enter a five-digit number: 12345
Oddish
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task8.exe
Enter a five-digit number: 12348
Evenish
```

Task 09(CP): Imagine you have a magical device that allows you to time travel, but there's a twist! This device only lets you skip 15 minutes into the future at a time. Your mission is to create a program that helps you calculate the exact time you'll arrive after using this time-traveling device.

You need to develop a program that takes the current time in hours and minutes (in the 24-hour format) and calculates the precise time you'll land in after time-traveling 15 minutes into the future. Your program should then display the future time in the "hh:mm" format.

Constraints:

Hours are always between 0 and 23.

Minutes are always between 0 and 59.

Create a function named **timeTravel** that returns the future time as a string in "hh:mm" format, making it easier for time travelers to plan their adventures.

Input	**Output**
1 46	2:1
0 01	0:16
23 59	0:14
11 08	11:23

Skill: Understanding and Implementing Void and Value Returning Functions

12 49	13:4

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task9.exe
Enter Hours: 12
Enter Minutes: 49
13:4
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task9.exe
Enter Hours: 0
Enter Minutes: 1
0:16
```

Task 10(CP): Write a C++ function that converts a number in the range of [1 ... 99] into text (in English).

Input	Output
25	TwentyFive
42	FortyTwo
6	Six

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task10.exe
Enter a number (1-99): 21
TwentyOne
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task10.exe
Enter a number (1-99): 1
One
```

Task 11(CP):

A pool with volume V fills up via two pipes. Each pipe has a certain flow rate (the liters of water, flowing through a pipe for an hour). A worker starts the pipes simultaneously and goes out for N hours. Write a program that finds the state of the pool the moment the worker comes back. Create a function named **calculatePoolState** that takes V, P1, P2 and H as input and returns the string with the calculated answer.

Input data:
V – the volume of the pool in liters - an integer in the range of [1 … 10000].

Skill: Understanding and Implementing Void and Value Returning Functions

P1 – the flow rate of the first pipe per hour – an integer in the range of [1 … 5000].
P2 – the flow rate of the second pipe per hour – an integer in the range of [1 … 5000].
H – the hours that the worker is absent – a floating-point number in the range of [1.0 … 24.00].

Output data:

- To what extent the pool has filled up and how much percent has each pipe contributed. All percent values must be formatted to an integer (without rounding).
    - "The pool is [x]% full. Pipe 1: [y]%. Pipe 2: [z]%."
- If the pool has overflown – with how many liters it has overflown for the given time – a floating-point number.
    - "For [x] hours the pool overflows with [y] liters."

Input	Output	Input	Output
1000 100 120 3	The pool is 66% full. Pipe 1: 45%. Pipe2: 54%.	100 100 100 2.5	For 2.5 hours the pool overflows with 400 liters.

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task11.exe
Enter volume of the pool in liters: 1000
Enter flow rate of the first pipe per hour: 100
Enter flow rate of the second pipe per hour: 120
Enter hours that the worker is absent: 3
The pool is 66% full. Pipe 1: 45%. Pipe 2: 54%.
```

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 5\Lab Tasks>Task11.exe
Enter volume of the pool in liters: 100
Enter flow rate of the first pipe per hour: 100
Enter flow rate of the second pipe per hour: 100
Enter hours that the worker is absent: 2.5
For 2.5 hours, the pool overflows with 400 liters.
```

Good Luck and Best Wishes !!

Happy Coding ahead :)

Skill: Understanding and Implementing Void and Value Returning Functions