Programming Fundamentals

Week 04 - Lab Manual

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:

- Write reusable code using functions.
- Write a simple conditional statement that involves a single boolean expression
- Write a while loop to run the program continuously

Let's do some coding.

Skill: Write reusable code using functions.

Introduction

In our daily life routine, we often need to perform the same tasks again and again. For example, let's consider the example of a Car.

- **Acceleration**: Accelerating a car is like pushing the gas pedal to make it go faster. It's a way of increasing speed.
- **Braking**: Braking is what you do when you need the car to stop or slow down. It's a way of reducing speed or coming to a complete stop.
- **Headlights**: Turning the headlights on and off is a function we use to control the car's lights, providing visibility in the dark and conserving energy during the day.

Just as we use these functions in a car to perform specific actions, in programming, we create functions to perform specific tasks that can be reused whenever needed. Functions simplify our code, making it easier to manage and understand. The following are the properties of functions.

- Functions are executed only when they are called
- Functions can receive different values that are known as Parameters
- Functions may or may not return a value when finished execution.

Function Prototype, function definition, and function call are the core elements of understanding a function.

Skill: Write reusable code using functions

Programming Fundamentals

Week 04 - Lab Manual

The following table lists the difference between these words.

Function	**Description**
Function Prototype	It must be defined at the start of the program. A function prototype contains the • Return Type • Functions Name • Function Parameters
Function Definition	It is the portion of the code that implements the body/logic of the reusable functionality.
Function Call	It is the part of the code that is used to call the function. **A Function is executed only when it is called.**
Consider the following programs to get a better understanding of functions.	

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

```
#include<iostream>
using namespace std;
main()
{
     int number1, number2, sum;
     cout << "Enter First Number: ";
     cin >> number1;
     cout << "Enter Second Number: ";
     cin >> number2;
     sum = number1 + number2;
     cout << "Sum: " << sum;
}
```

We have written a program to solve our problem of addition.
Now, let's assume that we want to add two more numbers in this same program.
Let's modify this code so it would first take two numbers and add them and then again takes two numbers from the user and add them too.

Skill: Write reusable code using functions

```
#include<iostream>
using namespace std;
main()
{
      int number1, number2, sum;
      cout << "Enter First Number: ";
      cin >> number1;
      cout << "Enter Second Number: ";
      cin >> number2;
      sum = number1 + number2;
      cout << "Sum: " << sum;

      int number3, number4, sum1;
      cout << "Enter First Number: ";
      cin >> number3;
      cout << "Enter Second Number: ";
      cin >> number4;
      sum1 = number3 + number4;
      cout << "Sum: " << sum1;
}
```

Look closely and see that the code is repeating.
Now, we know that functions are used for repeating structure problems.
Let's code this one out using Functions.

Task 02(WP): Write a function in C++ that takes two numbers from the user and prints their sum on the screen.

Now, we need to do the following to write a function.

1. **Provide Function Prototype**
2. **Provide Function Definition**
3. **Provide Function Call**

Skill: Write reusable code using functions

Programming Fundamentals

Week 04 - Lab Manual

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

void add();

main()
{
    add();
}

void add()
{
    int number1, number2, sum;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    sum = number1 + number2;
    cout << "Sum: " << sum;
}
```

Function Prototype. It consists of function name, return type and function parameters. In this case there is no parameter.

Function Call.

Function Definition. It consists of complete code that we want to use repeatedly in our program

See the decomposition of a function into different parts that constitute a function.

Return type **Function Name** **Function Parameters**

```
void add()
{
    int number1, number2, sum;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    sum = number1 + number2;
    cout << "Sum: " << sum;
}
```

Function Body

Now, we can just **call the function as many times as we want.**
Look at the below code and output as an example:

Skill: Write reusable code using functions

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

void add();

main()
{
	add();
	add();
}

void add()
{
	int number1, number2, sum;
	cout << "Enter First Number: ";
	cin >> number1;
	cout << "Enter Second Number: ";
	cin >> number2;
	sum = number1 + number2;
	cout << "Sum: " << sum << endl;
}
```

```
Enter First Number: 3
Enter Second Number: 4
Sum: 7
Enter First Number: 2
Enter Second Number: 1
Sum: 3
```

Great Work Guys !! You have implemented your first program using Functions in C++. **Let's learn about functions with parameters now.**

Task 03(WP): Write a function in a C++ program that **takes two numbers as parameters** and prints their sum on screen.

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

void add(int number1, int number2);

main()
{
	add(4, 5);
}

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

```
G:\Semesters\Programming Fundamentals
Sum: 9
```

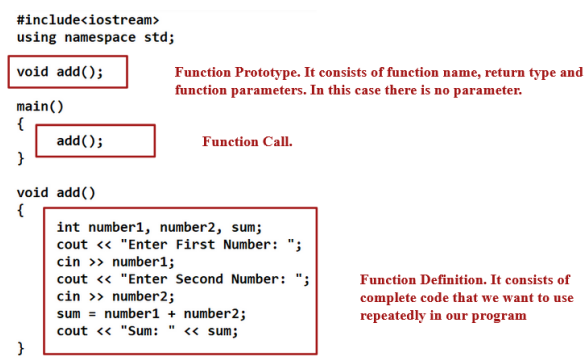
Now, the above-mentioned screenshots define a function that **takes two integer values as parameters and prints their sum on the screen.**

Skill: Write reusable code using functions

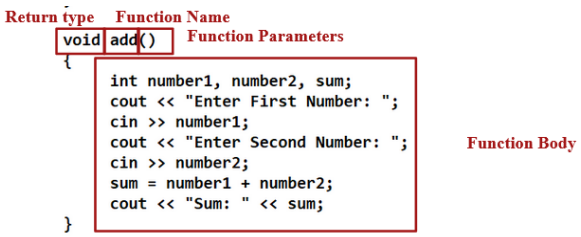
Notice that most of the code is the same, we have just provided the function two values at the time of the function call instead of taking two values from the user.

Conclusion:

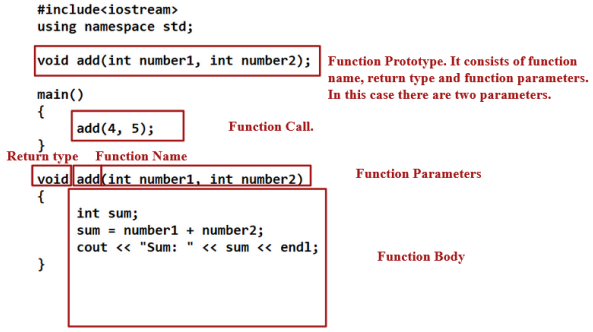
Functions are the pieces of code that are used to solve repeating structure problems. To implement functions, you need a **function prototype, function definition, and function call**



Function prototype is repeated in the function definition and consists of three elements: return type, function name, and parameters



Functions can take values with them to perform different operations on them. These are called parameters.



Well done Students, You have just learned another skill.

Skill: Write reusable code using functions

Programming Fundamentals

Week 04 - Lab Manual

Task01 (OP): Let's Fuel Up!

A vehicle needs 10 times the amount of fuel than the distance it travels. Create a function which takes distance as input in parameter and calculates the amount of fuel it needs and prints on the screen. Take the input of the distance from the user.

Examples

calculateFuel(15) ➞ 150
calculateFuel(23.5) ➞ 235

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task01.exe
Enter the distance: 34.5
Fuel needed: 345
```

Task02(OP): Inches to Feet

Create a function in C++ that accepts a measurement value in inches as a parameter and prints the equivalent of the measurement value in feet.
Take the measurement as input from the user.

Examples

inchesToFeet(324) ➞ 27
inchesToFeet(12) ➞ 1
inchesToFeet(36) ➞ 3

Notes

- 12 inches = 1 foot.

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task02.exe
Enter the measurement in inches: 3.4
Equivalent in feet: 0.283333
```

Task03(OP): Rubik's Cube

Given a Rubik's Cube with a side length of `n` given as a parameter to the function named howManyStickers, print the number of individual stickers that are needed to cover the whole cube.

Skill: Write reusable code using functions

Programming Fundamentals

Week 04 - Lab Manual

- The Rubik's cube of side length 1 has **6 stickers**.
- The Rubik's cube of side length 2 has **24 stickers**.
- The Rubik's cube of side length 3 has **54 stickers**.

3x3=9

9x6=54

Examples

howManyStickers(1) ➞ 6
howManyStickers(2) ➞ 24
howManyStickers(3) ➞ 54

Notes

- Keep in mind there are 6 faces to keep track of.
- Take the side length as input from the user and then pass it to the function

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task03.exe
Enter the side length of the Rubik's Cube: 4
Number of stickers needed: 96
```

Skill: Write reusable code using functions

Programming Fundamentals

Week 04 - Lab Manual

Skill: Write a simple conditional statement that involves a single boolean expression

Introduction

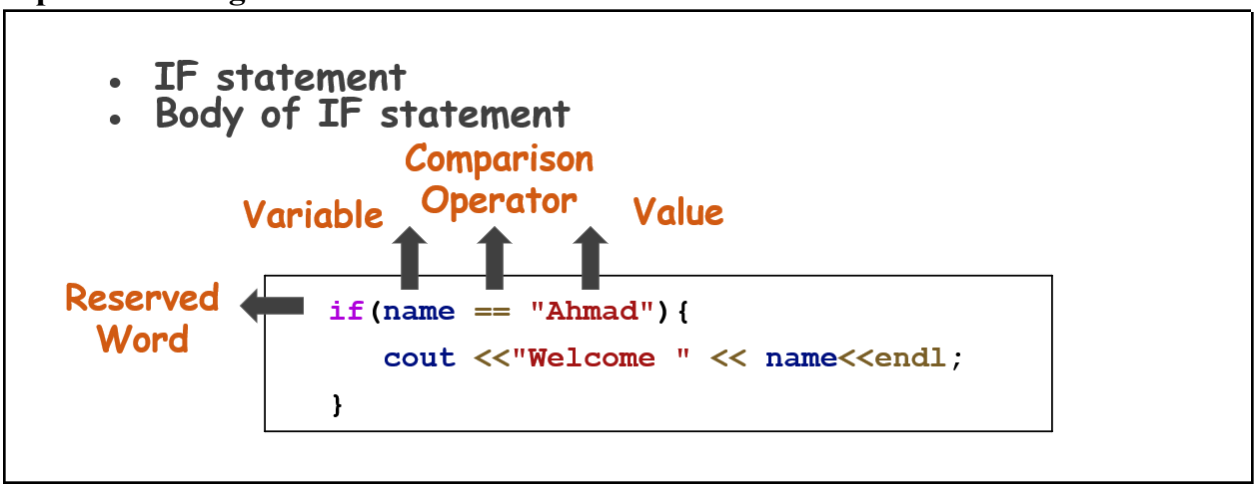
Conditional Statements are used to execute the code depending on some given conditions. In our daily life, we encounter many situations where we have to decide from multiple options. For example, whether to buy a dress or not? Or Whether to go to university today or not?

Even simple cases present us with these potential solutions that can be chosen depending on different conditions. For example, IF the cost of the dress is less than 5000 then you will buy that dress. OR IF your friends are going to university then you will go to the university😀.

Similarly, we are faced with similar problems in programming as well. Where we, as programmers, have to write programming code to solve such problems. Therefore, for such problems, we make use of the IF Statement.

IF Statement

We learned about solving such issues in the last class through various examples. For example, consider the below-mentioned cell where we are **evaluating a boolean expression using an IF Statement.**



Food for thought: **What will be the output of the following programs?**

Skill: Write reusable code using functions

`` #include<iostream> using namespace std; main() { cout << (3 > 2); } ```	
`` #include<iostream> using namespace std; main() { cout << (2 > 3); } ```	
`` #include<iostream> using namespace std; main() { if(1) cout << "Its Working"; } ```	
`` #include<iostream> using namespace std; main() { if(5) cout << "Its Working"; } ```	

Skill: Write reusable code using functions

`` #include<iostream> using namespace std; main() { if(0) cout << "Its Working"; } ```	
`` #include<iostream> using namespace std; main() { int a = 4, b = 4; if(b == a) { cout << "Its Working"; } } ```	
`` main() { int a = 5, b = 4; if(a + b == 9) { cout << "Its Working"; } } ```	

Most Important 👏

- A Boolean Statement evaluates to either **True (1) or False (0)**. If it's **true** the body of the IF Block is **executed** however, if the boolean expression evaluates to **False**, the code written inside the curly braces is ignored and **not executed** by the compiler.
- If there is only one statement in the body of the if statement then we can ignore the brackets if we like.

Skill: Write reusable code using functions

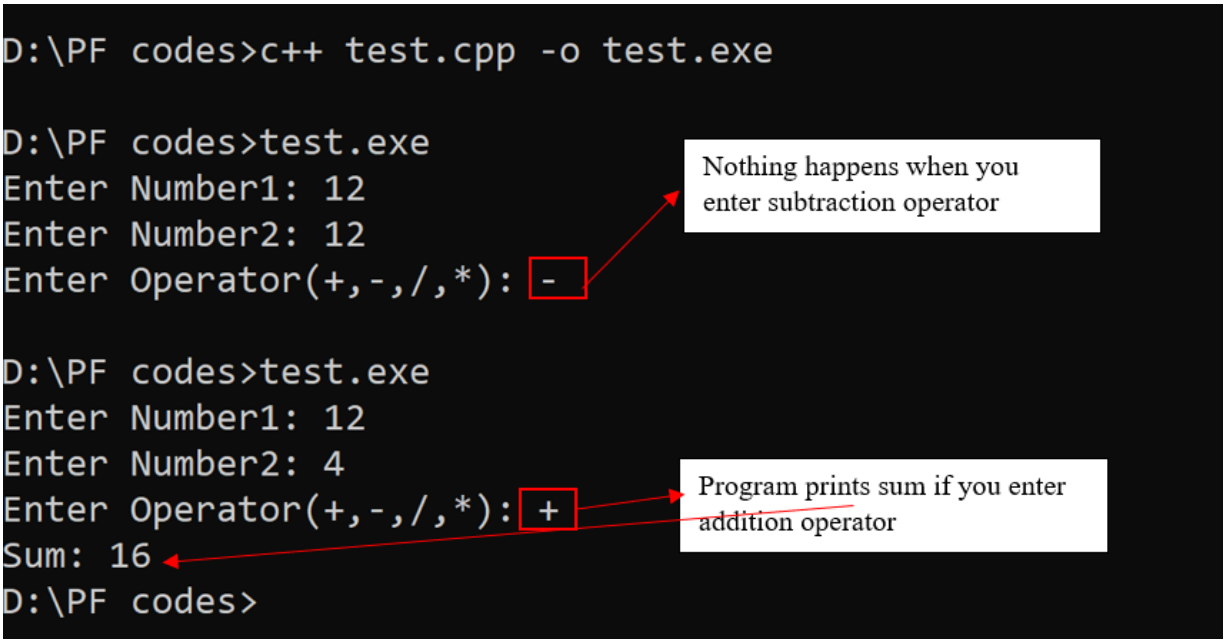
- If we do not write the brackets of the if statement then the program only considers the next line as the body of the if statement.

Let's code it out !

Consider the following task.

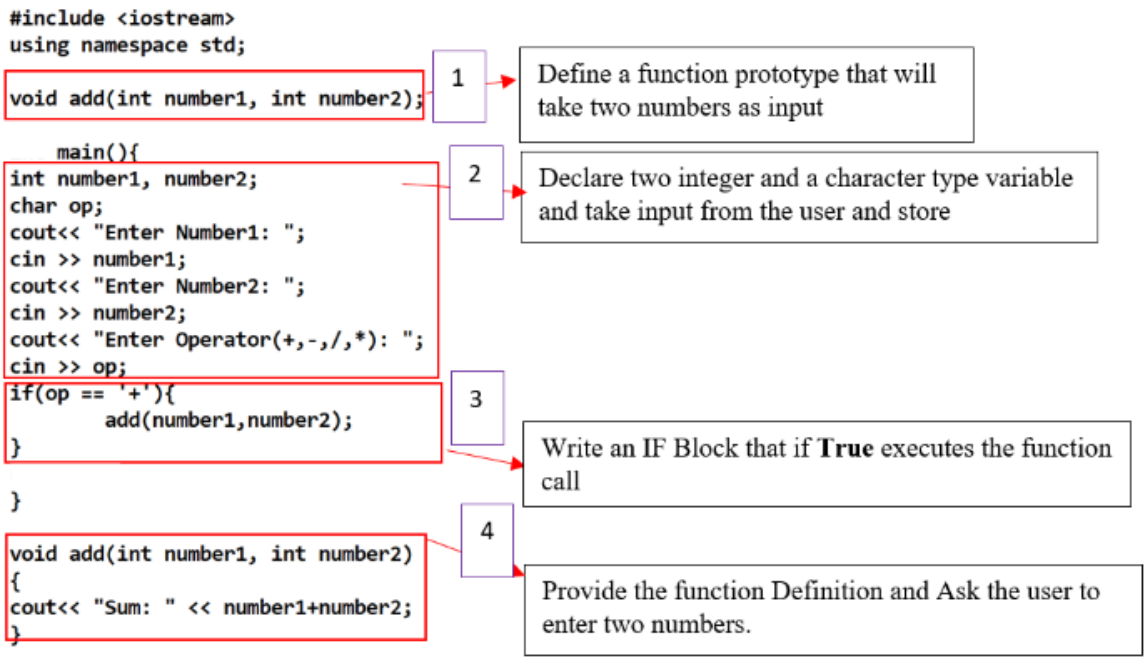
Task 01(WP): Write a function that **asks the user to enter two numbers,** and then also takes a **mathematical operator from the user**. If the operator is an **addition operator (+)** the function should **add the numbers and print their sum on the screen** otherwise nothing happens.

Sample output:



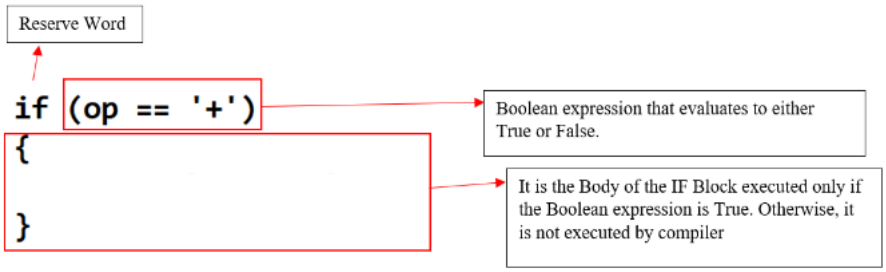
Now, take a minute and **think about how it can be done by using a boolean expression inside an IF Block**.

Skill: Write reusable code using functions



You are doing great so far. Great Work People!

Conclusion

Statement	Description
IF Block	

Task 04(OP): Basic Calculator

Write a C++ program for a basic calculator that can perform four operations: addition, subtraction, multiplication, and division. The program should ask the user to enter two

Skill: Write reusable code using functions

numbers and an operator (+, -, *, /), and then it should display the result of the operation.

Hint: You will need to write multiple IF Blocks. Also write separate functions for each calculation.

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task.exe
Enter 1st number: 4
Enter 2nd number: 8
Enter an operator (+, -, *, /): *
Multiplication: 32
```

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task.exe
Enter 1st number: 4
Enter 2nd number: 3
Enter an operator (+, -, *, /): -
Subtraction: 1
```

Task 05(OP): Vote

A person is eligible to vote if his/her age is greater than or equal to 18. Define a function that prints if he/she is eligible to vote. Function name should be Vote.

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task05.exe
Enter your age: 18
You are eligible to vote.
```

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task05.exe
Enter your age: 17
You are not eligible to vote.
```

Task 06(OP): Pass or Fail

Write a C++ Function that helps determine whether a student passes or fails a test. The program should ask the user to enter their test score as an integer. Then it should pass this score to the function and if the score is greater than 50, then the function should display "Pass," indicating that the student has passed the test. Otherwise, it should display "Fail," indicating that the student has not passed the test.

Skill: Write reusable code using functions

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task06.exe
Enter your score: 60
Pass
```

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task06.exe
Enter your score: 50
Fail
```

Task 07(OP): Even or Odd

Given an integer as a parameter to the function, determine whether the number is even or odd. You have to take the number as input from the user.

Examples

evenOrOdd(0) ➞ "even"
evenOrOdd(1) ➞ "odd"
evenOrOdd(8) ➞ "even"

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task07.exe
Enter a number: 4
Number 4 is even
```

Task 08(OP): Shopping Discount

A store offers a 10% discount on the total purchase amount every Sunday. Write a C++ program that helps customers calculate their payable amount based on the day of the week and the total purchase amount. The program should ask the user to enter the day of purchase and the total purchase amount. If the purchase is made on Sunday, the program should apply a 10% discount; otherwise, no discount is applied.

You have to write a function that takes Day and purchase amount as input in parameters and prints the total payable amount.

Examples:

calculatePayableAmount("Sunday", 1000) ➞ "Payable Amount: $900"
calculatePayableAmount("Monday", 1000) ➞ "Payable Amount: $1000"

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task08.exe
Enter the day of purchase: Sunday
Enter the total purchase amount: $1000
Payable Amount: $900
```

Skill: Write reusable code using functions

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task08.exe
Enter the day of purchase: Monday
Enter the total purchase amount: $1000
Payable Amount: $1000
```

Task09 (OP): Let's Fuel Up Again!

A vehicle needs 10 times the amount of fuel than the distance it travels. However, it must always carry a minimum of 100 fuel before setting off.

Create a function which takes distance as input in parameter and calculates the amount of fuel it needs and prints on the screen. Take the input of the distance from the user.

Examples

calculateFuel(15) ⟶ 150
calculateFuel(23.5) ⟶ 235
calculateFuel(3) ⟶ 100

Notes

- Return 100 if the calculated fuel turns out to be less than 100.

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task01.exe
Enter the distance: 34.5
Fuel needed: 345
```

```
G:\Semesters\Programming Fundamentals (Fall 2023)\Week 4\Lab Tasks>Task09.exe
Enter the distance: 3
Fuel needed: 100
```

Skill: Write reusable code using functions

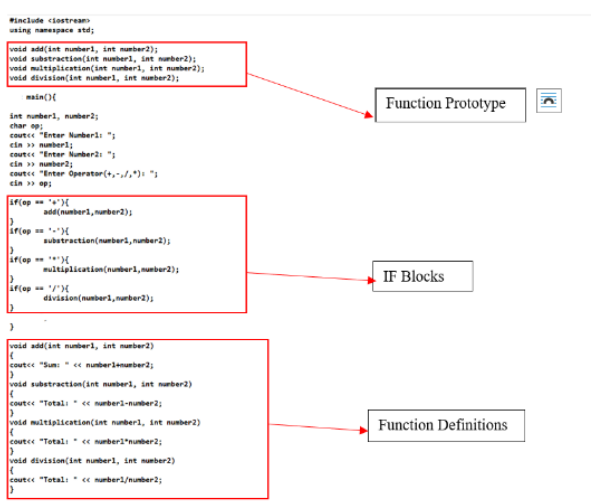
Programming Fundamentals

Week 04 - Lab Manual

Skill: Write a while loop to run the program continuously

Introduction

Let's just look at the progress of our program that we have been able to implement from the start of this Lab.




```
D:\PF codes>c++ test.cpp -o test.exe

D:\PF codes>test.exe
Enter Number1: 12
Enter Number2: 32
Enter Operator(+,-,/,*): +
Sum: 44
D:\PF codes>test.exe
Enter Number1: 2
Enter Number2: 3
Enter Operator(+,-,/,*): *
Total: 6
D:\PF codes>test.exe
Enter Number1: 32
Enter Number2: 12
Enter Operator(+,-,/,*): -
Total: 20
D:\PF codes>test.exe
Enter Number1: 6
Enter Number2: 2
Enter Operator(+,-,/,*): /
Total: 3
D:\PF codes>
```

We have a working simple calculator but the problem is that after each step **we need to execute the program again** in order to perform another operation.

If only the program could execute itself again on its own.

Thinking about Solution ?

Loops are used to execute the program repeatedly under a given condition.

Any code written **inside these curly braces will be executed an infinite number of times**.	`while(true)` `{` `}`

Let's add this functionality into the above mentioned code to make it repeat itself.
Question !
Which code do you think that we need to put inside these curly braces so it will be executed again.
Think again ! You may be surprised.
Hint: After printing the result on screen we want the program to again ask for input

Skill: Write a while loop to run the program continuously

and then execute the function call depending on the entered operator.

Task 01(WP): Write a program using functions and if statements that **repeatedly** acts like a **calculator** and **never stops** unless we **forcefully** close the window.

```
void add(int number1, int number2);
void substraction(int number1, int number2);
void multiplication(int number1, int number2);
void division(int number1, int number2);

   main(){

int number1, number2;
char op;
while(true)
{
cout<< "Enter Number1: ";
cin >> number1;
cout<< "Enter Number2: ";
cin >> number2;
cout<< "Enter Operator(+,-,/,*): ";
cin >> op;

if(op == '+'){
        add(number1,number2);
}
if(op == '-'){
        substraction(number1,number2);
}
if(op == '*'){
        multiplication(number1,number2);
}
if(op == '/'){
        division(number1,number2);
}
}

}

void add(int number1, int number2)
{
cout<< "Sum: " << number1+number2<<endl;
}
void substraction(int number1, int number2)
{
cout<< "Total: " << number1-number2<<endl;
}
void multiplication(int number1, int number2)
{
cout<< "Total: " << number1*number2<<endl;
}
void division(int number1, int number2)
{
cout<< "Total: " << number1/number2<<endl;
}
```

All the code written inside the curly braces of while(true) will be executed indefinite times.

Skill: Write a while loop to run the program continuously

Programming Fundamentals

Week 04 - Lab Manual

```
D:\PF codes>c++ test.cpp -o test.exe

D:\PF codes>test.exe
Enter Number1: 12
Enter Number2: 3
Enter Operator(+,-,/,*): *
Total: 36
Enter Number1: 12
Enter Number2: 12
Enter Operator(+,-,/,*): +
Sum: 24
Enter Number1:
```

Task 10(OP): Write a program that keeps **printing your name** until closed **forcefully**.

Task 11(OP): Write a program that keeps printing the **name entered by the user**.
Instruction: Use a **function** that accepts a **string parameter** name and prints that name

Task 12(CP): A Store has announced to give a 10% discount on the total purchase amount of every Sunday and 5% on every other day. Write a program that takes Day, total purchase amount and outputs the payable. Now, After outputting the payable amount on screen, the program should again ask for the same details of some other customer and go on until closed forcefully.

You have learnt a lot today. Keep practicing these codes and we will see you on programming day with new skills to learn.

Good Luck and Best Wishes !!

Happy Coding ahead :)

Skill: Write a while loop to run the program continuously