ACTIVITY NO. 1

REVIEW OF C++ PROGRAMMING	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: 7/29/25**
Section: CPE 010- CPE 21S4	**Date Submitted: 7/29/25**
Name: QUIOYO, ANGELO M	**Instructor:**

1. Objective(s)

- Implement basic programming and OOP in C++

2. Intended Learning Outcomes (ILOs)

After this module, the student should be able to:

a. Create code that follows the basic C++ code structure;
b. Implement appropriate class definition and instances based on given requirements;
c. Solve different problems using the C++ programming language.

3. Discussion

Part A: Introduction to C++ Code

Structure of C++ Code

Sections	Sample Code
Header File Declaration Section	`#include<iostream>` `using namespace std;`
Global Declaration Section	`int count = 0;`

Class Declaration and Method Definition Section	`class rectangle{`
	`private:`
	`  double recLength, recWidth;`
	
	`public:`
	`  rectangle(double L, double W);`
	`  void setLength(double L);`
	`  void setWidth(double W);`
	`  double getPerimeter();`
	`};`

Main Function	`int main(){`
	`  rectangle shape1(2, 5);`
	`  std::cout << "The perimeter of the rectangle is " << shape1.getPerimeter() << ".\n";`
	`  std::cout << count << " number of objects created.";`
	`  return 0;`
	`}`

Method Definition	`rectangle::rectangle(double L, double W) {`
	`  recLength = L;`
	`  recWidth = W;`
	`  count++;`
	`}`

```
void rectangle::setLength(double L) {
    recLength = L;
}

void rectangle::setWidth(double W) {
    recWidth = W;
}

double rectangle::getPerimeter() {
    return (2*recLength) + (2*recWidth);
}
```

It is not required for all sections to have code for every use-case. However, for best practices you would prefer to have an overall structure to follow to increase code readability and reusability.

Data Types

d. Primary Data Type: int, float, char and void
e. User defined data type: structure, union, class, enumeration
f. Derived data type: array, function, pointer, reference

Local & Global Variables

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

int globalVal = 0; //Global Variable

int main(){
    int localVal = 5; //Local Variable

    std::cout << "Global Variable has value " << globalVal << ".\n";
    std::cout << "Local Variable has value " << localVal << ".\n";

    return 0;
}
```

Operators

Arithmetic	Relational	Logical
Addition +	Greater than >	AND &&
Subtraction −	Less than <	OR ||
Multiplication *	Greater than or equal >=	NOT !
Division /	Less than or equal <=	
Modulo %	Equal ==	
Increment ++	Not equal !=	
Decrement --		

Bitwise Operators

Let A = 60 and B = 13. Binary values are as follows:

```
A = 0011 1100
B = 0000 1101
```

Bitwise AND -> &	`A & B`	`0000 1100`
Bitwise OR -> |	`A | B`	`0011 1101`
Bitwise XOR -> ^	`A ^ B`	`0011 0001`
Bitwise Complement -> ~	`~A`	`1100 0011`

Assignment Operator

Assign a value to a variable. Example:

Assign the value 20 to a variable A.

```
int A = 20;
```

The assignment operator is a basic component denoted as "=".

Part B: Classes and Objects using C++

To create a class use the class keyword. Syntax is:

```
class myClass {
  public:
       int myNum;
       string myString;
};
```

`public` here is an access specifier. It indicates that the attributes and methods listed under it are accessible outside the class. A simple table is provided below to summarize the access specifiers used in c++.

Specifiers	Within same class	In derived class	Outside the class
`private`	Yes	No	No
`protected`	Yes	Yes	No
`public`	Yes	Yes	Yes

We can then create an object from this class:

```
int main(){
       //this creates the object
       myClass object1;

       //this accesses the public attributes
       object1.myNum = 5;
       object1.myString = "Sample";

       return 0;
}
```

4. Materials and Equipment

Personal Computer with C++ IDE

Recommended IDE:

- CLion (must use TIP email to download)
- DevC++ (use the embarcadero fork or configure to C++17)

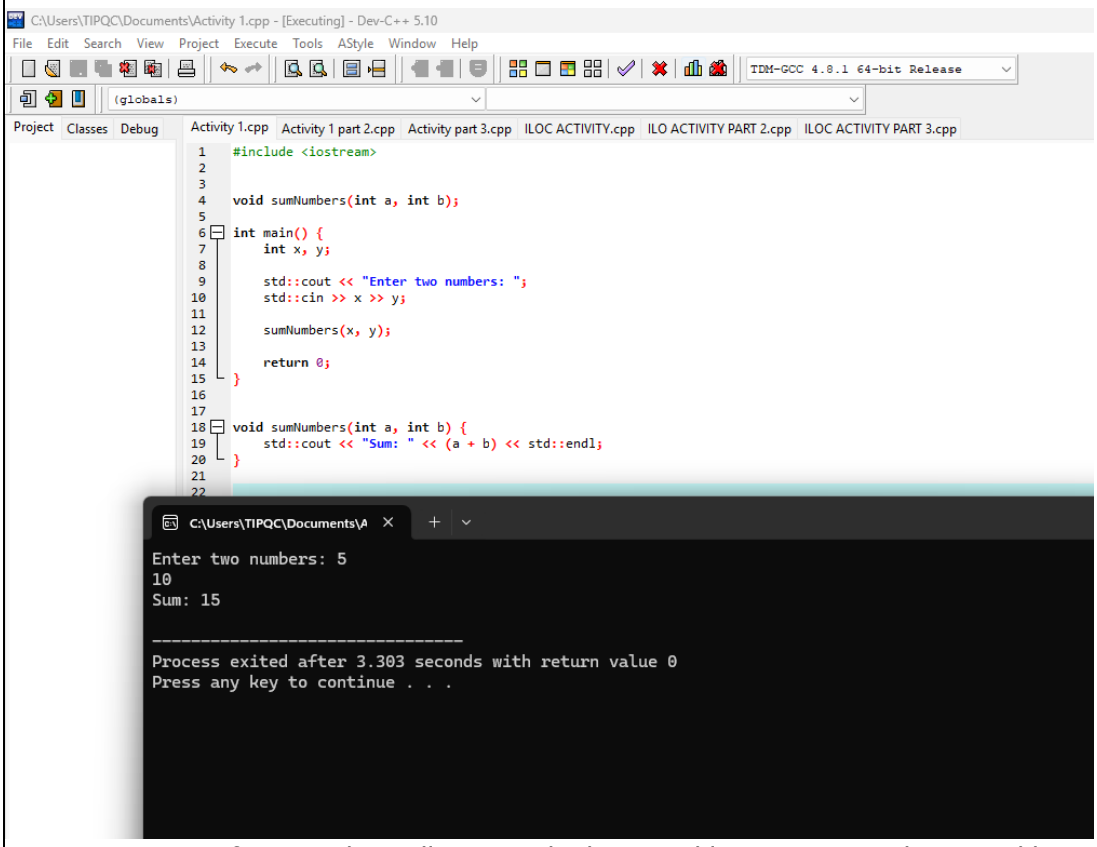
5. Procedure

ILO A: Create Code That Follows the Basic C++ Code Structure

For this activity, you have to demonstrate the use of a **function prototype**. The section on class declaration and method definition will be used for the function prototype and the function will be defined in the follow method definition section after the main function.

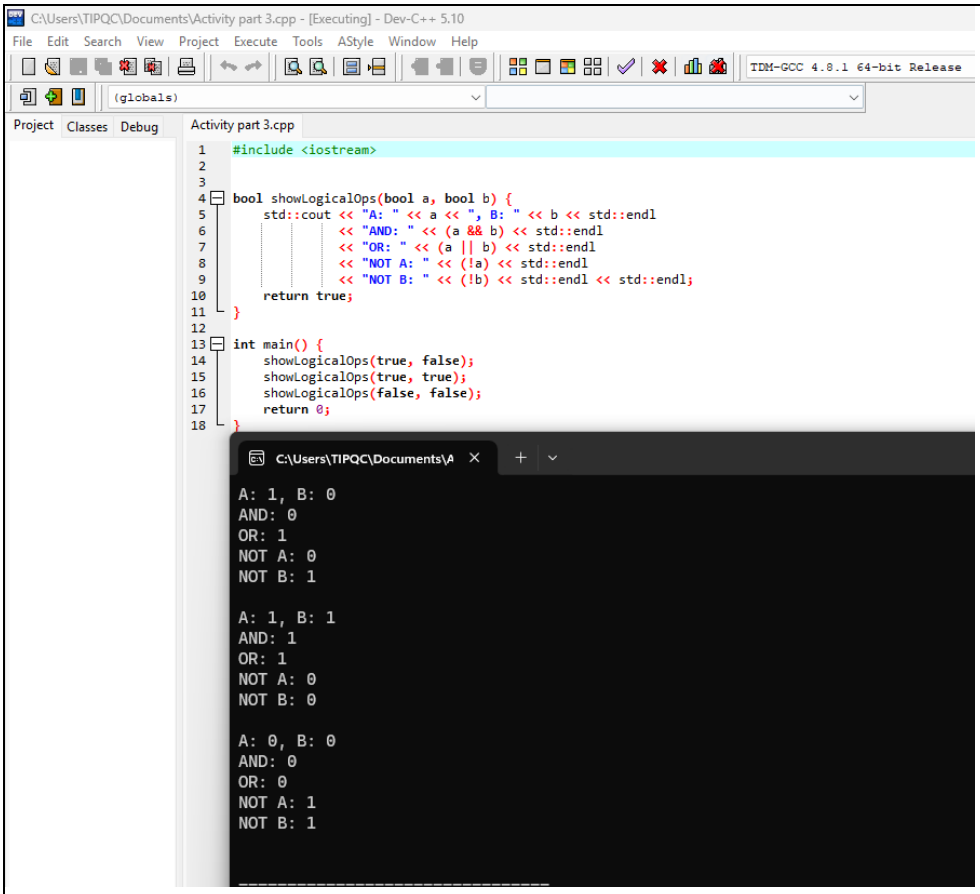
A function prototype in c++ is a declaration of the name, parameters and return type of the function before its definition. Write a C++ code the satisfies the following:

- Create a function that will take two numbers and display the sum.



- Create a function that will return whether variable A is greater than variable B.

- Create a function that will take two Boolean values and display the result of all logical operations then return true if it was a success.



Note:

- The driver program must call each function.
- The definitions must be after the main function.

ILO B: Implement Appropriate Class Definition and Instances Based on Given Requirements

In this section, the initial implementation for a class **triangle** will be implemented. The step-by-step procedure is shown below:

Step 1. Include the necessary header files. For this one, we only need `#include <iostream>`

Step 2. Create the triangle class. Assign it with private variables: totalAngle, angleA, angleB, and angleC.

```
class Triangle{
private:
    double totalAngle, angleA, angleB, angleC;
```

Step 3. We then create public methods. The constructor must allow for creation of the object with 3 initial angles to be stored in our previously defined variables `angleA, angleB` and `angleC`. Another method has to be made if the user wants to change the initial values, this will also accept 3 arguments to change the values in `angleA, angleB` and `angleC`. Lastly, a function to validate whether the given values make our shape an actual triangle.

```
public:
    Triangle(double A, double B, double C);
    void setAngles(double A, double B, double C);
```

```
    const bool validateTriangle();
};
```

Step 4. Define the methods.

```
Triangle::Triangle(double A, double B, double C) {
    angleA = A;
```

```
        angleB = B;
        angleC = C;
        totalAngle = A+B+C;
    }

    void Triangle::setAngles(double A, double B, double C) {
        angleA = A;
        angleB = B;
        angleC = C;
        totalAngle = A+B+C;
    }

    const bool Triangle::validateTriangle() {
        return (totalAngle <= 180);
    }
```

Step 5. Create the driver code.

```
    int main(){
        //driver code
        Triangle set1(40, 30, 110);
        if(set1.validateTriangle()){
            std::cout << "The shape is a valid triangle.\n";
        } else {
            std::cout << "The shape is NOT a valid triangle.\n";
        }

        return 0;
    }
```

Include the output of running this code in section 6. Note your observations and comments.

6. Output:

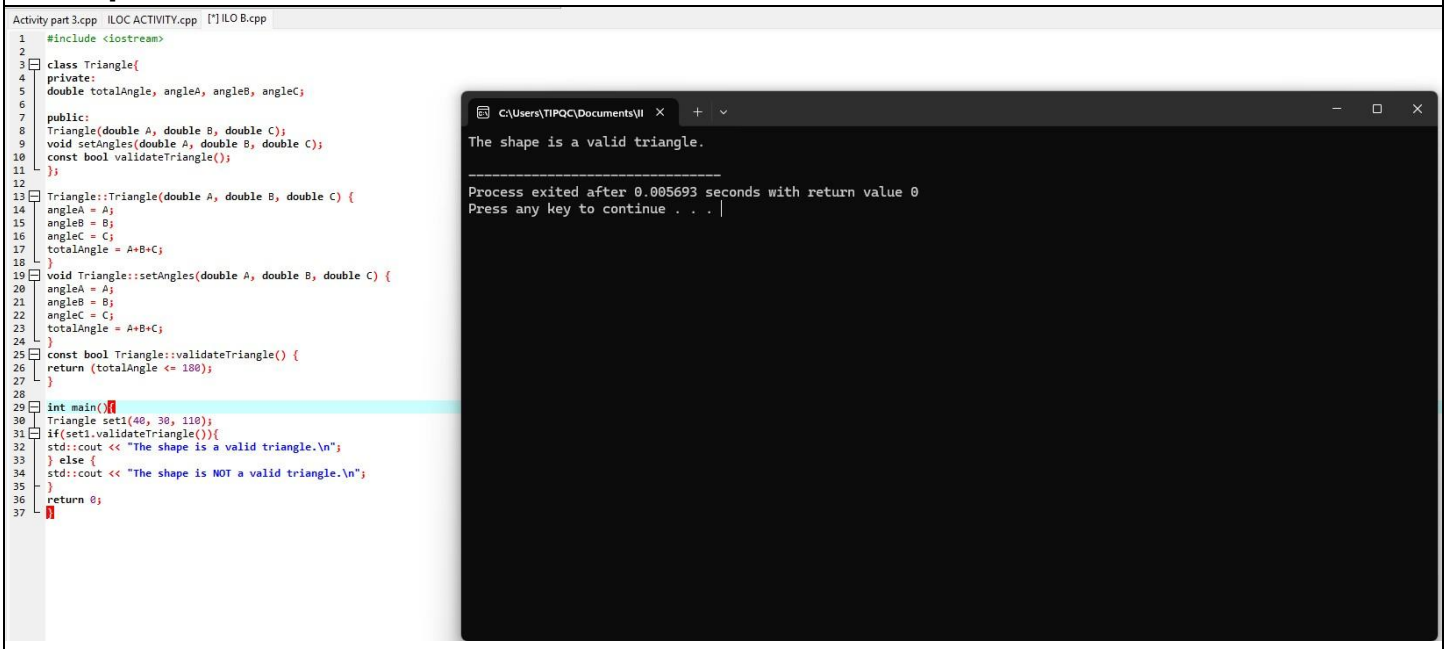


Table 1-1. C++ Structure Code for Answer

Table 1-2. ILO B output observations and comments.

7. Supplementary Activity

ILO C: Solve Different Problems using the C++ Programming Language

The supplementary activities are meant to gauge your ability in using C++. The problems below range from easy to intermediate to advanced problems. Note your difficulties after answering the problems below.

1. Create a C++ program to swap the two numbers in different variables.

ILOC ACTIVITY.cpp

```cpp
#include <iostream>
#include <algorithm>

int main() {
    int num1 = 5;
    int num2 = 10;

    std::cout << "Original values: num1 = " << num1 << ", num2 = " << num2 << std::endl;


    int temp = num1;
    num1 = num2;
    num2 = temp;

    std::cout << "After swap (Method 1): num1 = " << num1 << ", num2 = " << num2 << std::endl;


    num1 = 20;
    num2 = 30;
    std::cout << "\nOriginal values for Method 2: num1 = " << num1 << ", num2 = " << num2 << std::endl;


    std::swap(num1, num2);

    std::cout << "After swap (Method 2): num1 = " << num1 << ", num2 = " << num2 << std::endl;

    return 0;
}
```

```
C:\Users\TIPQC\Documents\Il
Original values: num1 = 5, num2 = 10
After swap (Method 1): num1 = 10, num2 = 5

Original values for Method 2: num1 = 20, num2 = 30
After swap (Method 2): num1 = 30, num2 = 20

--------------------------------
Process exited after 0.007827 seconds with return value 0
Press any key to continue . . .
```

2. Create a C++ program that has a function to convert temperature in Kelvin to Fahrenheit.

ILO ACTIVITY PART 2.cpp

```cpp
#include <iostream>


double kelvinToFahrenheit(double kelvin) {

    return (kelvin - 273.15) * 1.8 + 32;
}

int main() {
    double kelvinTemp;

    std::cout << "Enter temperature in Kelvin: ";
    std::cin >> kelvinTemp;

    if (kelvinTemp < 0) {
        std::cout << "Error: Temperature cannot be below 0 Kelvin.\n";
    } else {
        double fahrenheitTemp = kelvinToFahrenheit(kelvinTemp);
        std::cout << kelvinTemp << " Kelvin is " << fahrenheitTemp << " Fahrenheit.\n";
    }

    return 0;
}
```

```
C:\Users\TIPQC\Documents\Il
Enter temperature in Kelvin: 69
69 Kelvin is -335.47 Fahrenheit.

--------------------------------
Process exited after 4.563 seconds with return value 0
Press any key to continue . . .
```

3. Create a C++ program that has a function that will calculate the distance between two points.

[*] ILO C 4.cpp | ILO C 3.cpp

```cpp
#include <iostream>
#include <cmath>


double calculateDistance(double x1, double y1, double x2, double y2) {
    double dx = x2 - x1;
    double dy = y2 - y1;
    return std::sqrt(dx * dx + dy * dy);
}

int main() {

    double point1_x = 1.0;
    double point1_y = 2.0;

    double point2_x = 4.0;
    double point2_y = 6.0;


    double distance = calculateDistance(point1_x, point1_y, point2_x, point2_y);


    std::cout << "The distance between (" << point1_x << ", " << point1_y
              << ") and (" << point2_x << ", " << point2_y << ") is: "
              << distance << std::endl;

    return 0;
}
```

```
C:\Users\TIPQC\Documents\II
The distance between (1, 2) and (4, 6) is: 5

--------------------------------
Process exited after 0.006966 seconds with return value 0
Press any key to continue . . .
```

4. Modify the code given in ILO B and add the following functions:
    a. A function to compute for the area of a triangle
    b. A function to compute for the perimeter of a triangle
    c. A function that determines whether the triangle is acute-angled, obtuse-angled or 'others.'

[*] ILO C 4.cpp

```cpp
#include <iostream>
#include <cmath>

class Triangle {
private:
    double angleA, angleB, angleC;
    double sideA, sideB, sideC;

public:

    Triangle(double ang1, double ang2, double ang3, double s1, double s2, double s3) :
        angleA(ang1), angleB(ang2), angleC(ang3), sideA(s1), sideB(s2), sideC(s3) {}


    bool validateAngles() const {
        return std::abs((angleA + angleB + angleC) - 180.0) < 0.001;
    }


    double getArea() const {
        double s = (sideA + sideB + sideC) / 2.0;
        return std::sqrt(s * (s - sideA) * (s - sideB) * (s - sideC));
    }


    double getPerimeter() const {
        return sideA + sideB + sideC;
    }


    std::string getType() const {
        if (!validateAngles()) return "Invalid Angle Triangle";
        if (angleA == 90 || angleB == 90 || angleC == 90) return "Right-angled";
        if (angleA > 90 || angleB > 90 || angleC > 90) return "Obtuse-angled";
        return "Acute-angled";
    }
};

int main() {

    Triangle myTriangle(90, 53.13, 36.87, 3, 4, 5);


    if (myTriangle.validateAngles()) {
        std::cout << "Area: " << myTriangle.getArea() << "\n";
        std::cout << "Perimeter: " << myTriangle.getPerimeter() << "\n";
```

```
C:\Users\TIPQC\Documents\II
Area: 6
Perimeter: 12
Type: Right-angled

--------------------------------
Process exited after 0.01475 seconds with return value 0
Press any key to continue . . .
```

8. Conclusion
Provide the following: • Summary of lessons learned • Analysis of the procedure • Analysis of the supplementary activity • Concluding statement / Feedback: How well did you think you did in this activity? What are your areas for improvement?
9. Assessment Rubric