ACTIVITY NO. 1

REVIEW OF C++ PROGRAMMING	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** July 30, 2025
Section: CPE21S4	**Date Submitted:** July 30,2025
Name: Bautista, Mariela S.	**Instructor: Engr.**Jimlord Quejado

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{ priva
te:
    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.

Sections	**Answer**
Header File Declaration Section	#include <iostream>

Global Declaration Section	using namespace std;
	void displaySum(int, int);
	bool isGreater(int, int);
	bool logicOperations(bool, bool);

Class Declaration and Method Definition Section	
Main Function	int main() { ... } Calls all three functions: displaySum(), isGreater(), and logicOperations()
Method Definition	void displaySum(...), bool isGreater(...), bool logicOperations(...)

7. Supplementary Activity

6. Output

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

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

The Triangle class worked as expected. I tried two sets of angles: one valid and one not, and the program correctly identified which one forms a triangle. The validateTriangle() method checks if the total angle is 180 and all angles are greater than zero, which is the right logic.

Overall, the class setup using constructor, setter method, and validation made it easy to test different values. It helped me understand how to build and use classes properly in C++. I also saw how important it is to organize methods well and check inputs.

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.
2. Create a C++ program that has a function to convert temperature in Kelvin to Fahrenheit.
3. Create a C++ program that has a function that will calculate the distance between two points.
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.'

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

int main() {
    int a = 10, b = 20, temp;
    cout << "Before Swap: a = " << a << ", b = " << b << endl;

    temp = a;
    a = b;
    b = temp;

    cout << "After Swap: a = " << a << ", b = " << b << endl;
    return 0;
}
```

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

     float kelvinToFahrenheit(float kelvin) {
     return (kelvin - 273.15) * 9/5 + 32;
 }

  int main() {
      float kelvin = 310.15;
      cout << "Kelvin to Fahrenheit: " << kelvinToFahrenheit(kelvin) << "°F" << endl;
       return 0;
}
```

```
3. #include <iostream>
 #include <cmath>
  using namespace std;

   float distance(float x1, float y1, float x2, float y2) {
   return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));
  }

     int main() {
   float x1 = 1, y1 = 2, x2 = 4, y2 = 6;
    cout << "Distance: " << distance(x1, y1, x2, y2) << endl;
  return 0;
  }

  4.#include <iostream>
  #include <cmath>
  using namespace std;

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

  public:
      Triangle(double A, double B, double C, double a, double b, double c);
      void setAngles(double A, double B, double C);
      bool validateTriangle();
      double area();
      double perimeter();
      string triangleType();
  };

  Triangle::Triangle(double A, double B, double C, double a, double b, double c) {
      angleA = A; angleB = B; angleC = C;
      totalAngle = A + B + C;
      sideA = a; sideB = b; sideC = c;
  }

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

  bool Triangle::validateTriangle() {
      return totalAngle == 180 && angleA > 0 && angleB > 0 && angleC > 0;
  }
```

```
double Triangle::perimeter() {
    return sideA + sideB + sideC;
}

double Triangle::area() {
    double s = perimeter() / 2;
    return sqrt(s * (s - sideA) * (s - sideB) * (s - sideC));
}

string Triangle::triangleType() {
    if (angleA < 90 && angleB < 90 && angleC < 90) return "Acute-angled";
    if (angleA > 90 || angleB > 90 || angleC > 90) return "Obtuse-angled";
    return "Others";
}

int main() {
    Triangle tri(60, 60, 60, 5, 5, 5);

    if (tri.validateTriangle()) {
        cout << "Triangle is valid." << endl;
        cout << "Area: " << tri.area() << endl;
        cout << "Perimeter: " << tri.perimeter() << endl;
        cout << "Type: " << tri.triangleType() << endl;
    } else {
        cout << "Triangle is not valid." << endl;
    }

    return 0;
}
```

8. Conclusion

Provide the following:

- **Summary of lessons learned**

I learned how to use my knowledge previously in our first python class and how properly structure C++ programs using function prototypes, function definitions, and class implementations. I also learned how to apply logic and math through code using basic geometry and programming concepts.

- **Analysis of the procedure**

The procedures for ILO A and B helped me understand the basics of how C++ code is organized and how functions and classes work together. Writing the prototype before the function and putting method definitions after main() made me more aware of proper program structure.

- **Analysis of the supplementary activity**

This part helped me apply what I learned to real problems. Swapping numbers and converting temperature were simple, but modifying the triangle class made me think more logically. It really tested how well I could apply what I learned from earlier lessons into something more complete and useful.

- **Concluding statement / Feedback: How well did you think you did in this activity? What are your areas for improvement?**

I think I did okay overall. I was able to complete each task and make sure the code worked. I feel more confident using functions and creating classes now. One thing I want to improve is writing cleaner code faster and solving problems without always needing to look up formulas. Practice and repetition will definitely help with that.

9. Assessment Rubric