

ACTIVITY NO. 1

REVIEW OF C++ PROGRAMMING

Course Code: CPE010	Program: Computer Engineering												
Course Title: Data Structures and Algorithms	Date Performed: 07-29-25												
Section: CPE21S4	Date Submitted: 07-29-25												
Name: Santiago, David Owen A.	Instructor: Engr. Jimlord Quejado												
1. Objective(s)													
<ul style="list-style-type: none"> • Implement basic programming and OOP in C++ 													
2. Intended Learning Outcomes (ILOs)													
After this module, the student should be able to:													
<ol style="list-style-type: none"> 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													
<p>Part A: Introduction to C++ Code</p> <p>Structure of C++ Code</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 5px;">Sections</th><th style="text-align: left; padding: 5px;">Sample Code</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;">Header File Declaration Section</td><td style="padding: 5px;"> <pre>#include<iostream> using namespace std;</pre> </td></tr> <tr> <td style="padding: 5px;">Global Declaration Section</td><td style="padding: 5px;"> <pre>int count = 0;</pre> </td></tr> <tr> <td style="padding: 5px;">Class Declaration and Method Definition Section</td><td style="padding: 5px;"> <pre>class rectangle{ private: double recLength, recWidth; public: rectangle(double L, double W); void setLength(double L); void setWidth(double W); double getPerimeter(); };</pre> </td></tr> <tr> <td style="padding: 5px;">Main Function</td><td style="padding: 5px;"> <pre>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; }</pre> </td></tr> <tr> <td style="padding: 5px;">Method Definition</td><td style="padding: 5px;"> <pre>rectangle::rectangle(double L, double W) { recLength = L; recWidth = W; count++; }</pre> </td></tr> </tbody> </table>		Sections	Sample Code	Header File Declaration Section	<pre>#include<iostream> using namespace std;</pre>	Global Declaration Section	<pre>int count = 0;</pre>	Class Declaration and Method Definition Section	<pre>class rectangle{ private: double recLength, recWidth; public: rectangle(double L, double W); void setLength(double L); void setWidth(double W); double getPerimeter(); };</pre>	Main Function	<pre>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; }</pre>	Method Definition	<pre>rectangle::rectangle(double L, double W) { recLength = L; recWidth = W; count++; }</pre>
Sections	Sample Code												
Header File Declaration Section	<pre>#include<iostream> using namespace std;</pre>												
Global Declaration Section	<pre>int count = 0;</pre>												
Class Declaration and Method Definition Section	<pre>class rectangle{ private: double recLength, recWidth; public: rectangle(double L, double W); void setLength(double L); void setWidth(double W); double getPerimeter(); };</pre>												
Main Function	<pre>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; }</pre>												
Method Definition	<pre>rectangle::rectangle(double L, double W) { recLength = L; recWidth = W; count++; }</pre>												

```

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

Sections	Answer
Header File Declaration Section	
Global Declaration Section	
Class Declaration and Method Definition Section	
Main Function	
Method Definition	

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

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

```

1 #include <iostream>
2
3 void printSum(int a, int b);
4 void compare(int x, int y);
5 bool logicOperation(bool a, bool b);
6
7 int main(){
8
9     std::cout<<"Task 1"<<std::endl;
10    printSum(5, 3);
11    std::cout<<"Task 2"<<std::endl;
12    compare(5, 5);
13    std::cout<<"Task 3"<<std::endl;
14    logicOperation(1, 0);
15    return 0;
16 }
17
18 void printSum(int a, int b){
19 // int sum = a + b;
20     std::cout<<"the sum of "<<a<<" and "<<b<<" is: "<<a + b<<std::endl;
21 }
22 void compare(int x, int y){
23 if (x > y){
24     std::cout<<x<<" is greater than "<<y<<std::endl;
25 } else if (x < y){
26     std::cout<<x<<" is less than "<<y<<std::endl;
27 }
28 else{std::cout<<x<<" and "<<y<<" are equal "<<std::endl;
29 }
30 }
31
32 bool logicOperation(bool a, bool b){
33     bool orResult = a || b;
34     bool andResult = a && b;
35     bool notResult = a != b;
36
37     std::cout<<"The result of OR is: "<<orResult<<std::endl;
38     std::cout<<"The result of AND is: "<<andResult<<std::endl;
39     std::cout<<"The result of NOT is: "<<notResult<<std::endl;
40     return true;
41 }

```

Task 1
 the sum of 5 and 3 is: 8
 Task 2
 5 and 5 are equal
 Task 3
 The result of OR is: 1
 The result of AND is: 0
 The result of NOT is: 1

Process exited after 0.008987 seconds with return value 0
 Press any key to continue . . .

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.
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.'

8. Conclusion

Provide the following:

- Summary of lessons learned
 - In this lesson, I recalled basic syntax and code structure of C++. I also practiced using functions and function prototypes that print and return different values. Functions can be used to do arithmetic operations, compare values, do logical operations, and many more. The global variable type that I found most interesting is the void data type, as it only prints an output without return values.
- Analysis of the procedure
 - The procedure in creating function prototypes is as follows:
 1. Declare the global function prototype before the main function
 2. Outside of the main function, declare the function then insert the code to be executed when it's called
 3. Call the function within the main function using arguments within the main function.
- Analysis of the supplementary activity
- Concluding statement / Feedback: How well did you think you did in this activity? What are your areas for improvement?
 - In conclusion, C++, like many other programs, provides a useful way of creating functions that can easily be called within the main function. We use function prototypes to first declare the function before it is called in the main function to reduce clutter. Once they are declared, their actual commands to be executed are specified below the main function which allows them to easily be called and accessed in the main function.

9. Assessment Rubric