Activity No. <1>	
<Hands-on Activity 1.1 Basic C++ Programming>	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** 09/09/24
Section: CPE21S4	**Date Submitted:** 09/11/24
Name(s): Magistrado, Aira Pauleen M.	**Instructor:** Maria Rizette Sayo
6. Output	

Sections	**Answer**
Header File Declaration Section	#include <iostream>
Global Declaration Section	No global variables

Class Declaration and Method Definition Section	class Triangle {
	private:
	    double totalAngle, angleA, angleB, angleC;
	
	public:
	    Triangle(double A, double B, double C);
	    void setAngles(double A, double B, double C);
	    const bool validateTriangle();
	};
	
	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);
	}

Main Function	int main() { // Create a Triangle object with angles 40, 30, and 110

	Triangle set1(40, 30, 110);
	
	    // Validate the triangle and output the result
	    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;
	}

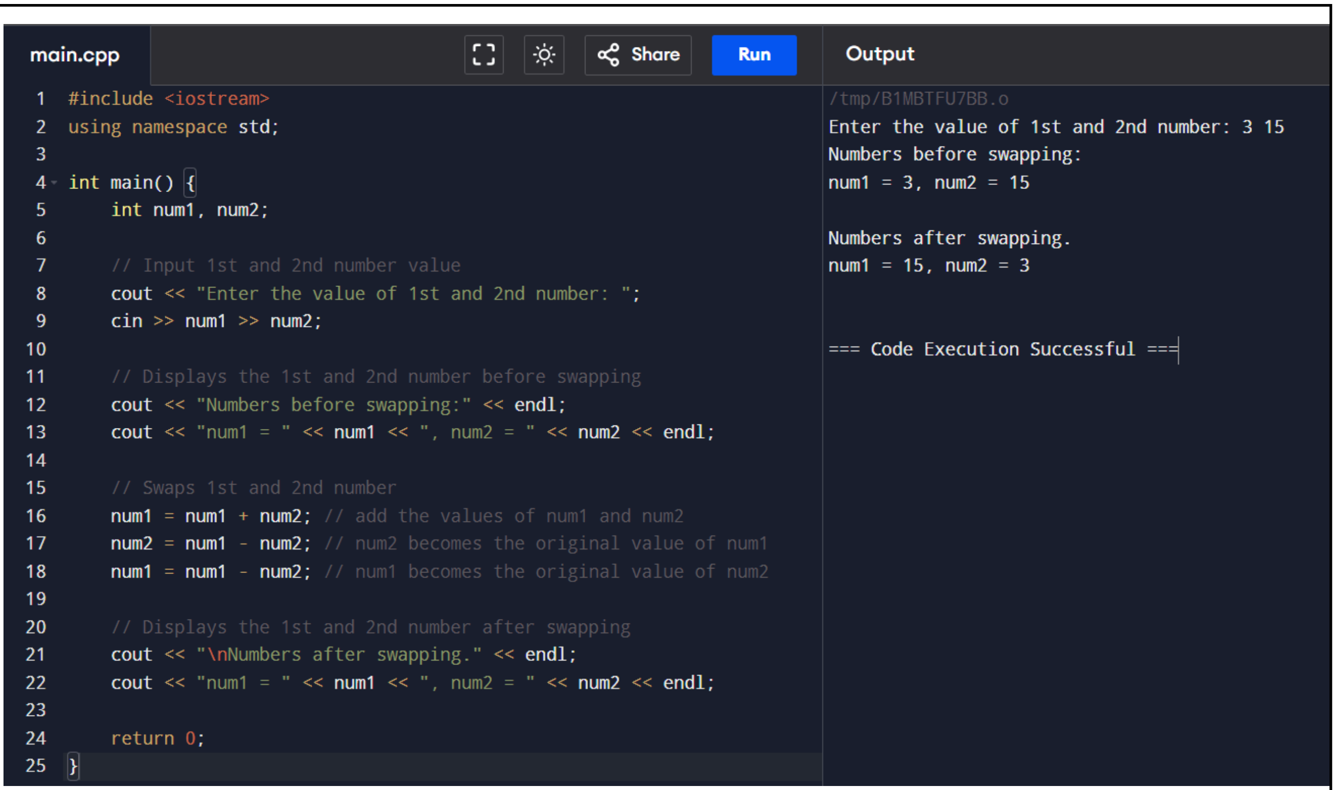
Method Definition	// Constructor definition
	Triangle::Triangle(double A, double B, double C) {
	    angleA = A;
	    angleB = B;
	    angleC = C;
	    totalAngle = A + B + C;
	}
	
	// Method to set angles definition
	void Triangle::setAngles(double A, double B, double C) {
	    angleA = A;
	    angleB = B;
	    angleC = C;
	    totalAngle = A + B + C;
	}
	
	// Method to validate the triangle definition
	bool Triangle::validateTriangle() {
	    return (totalAngle == 180);
	}

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

The angles 40, 30 and 110 have the sum of 180 degrees and so they are recognized as a form of a valid triangle. The validTriangle functions as the determinator if the sum of the angles are 180 degrees.

7. Supplementary Activity

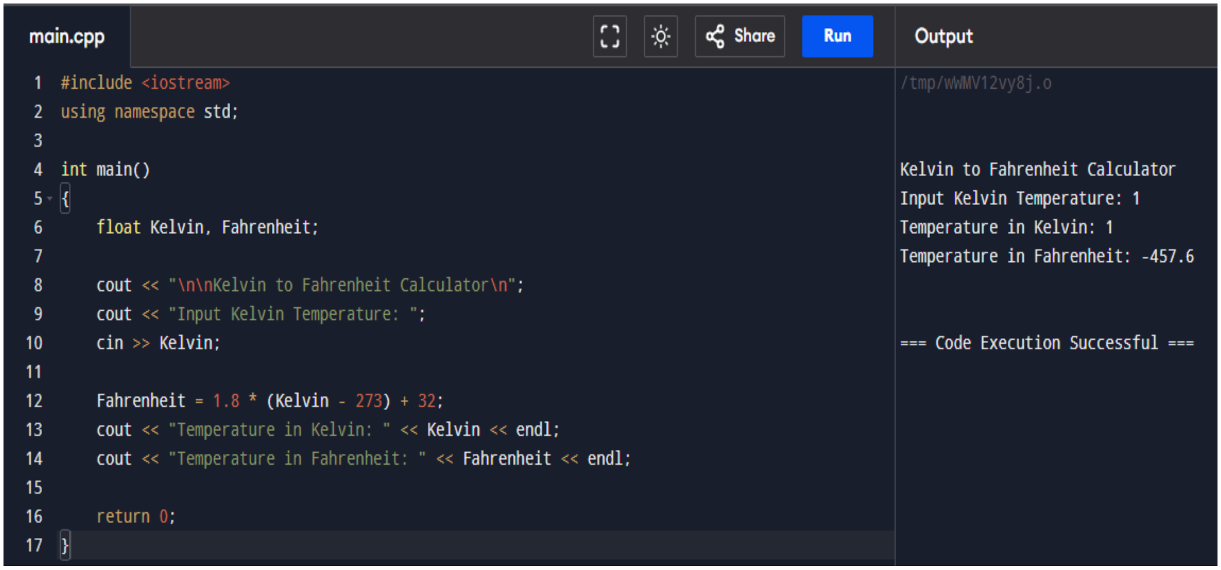
1. **Create a C++ program to swap the two numbers in different variables.**



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

int main() {
    int num1, num2;

    // Input 1st and 2nd number value
    cout << "Enter the value of 1st and 2nd number: ";
    cin >> num1 >> num2;

    // Displays the 1st and 2nd number before swapping
    cout << "Numbers before swapping:" << endl;
    cout << "num1 = " << num1 << ", num2 = " << num2 << endl;

    // Swaps 1st and 2nd number
    num1 = num1 + num2; // add the values of num1 and num2
    num2 = num1 - num2; // num2 becomes the original value of num1
    num1 = num1 - num2; // num1 becomes the original value of num2

    // Displays the 1st and 2nd number after swapping
    cout << "\nNumbers after swapping." << endl;
    cout << "num1 = " << num1 << ", num2 = " << num2 << endl;

    return 0;
}
```

```
/tmp/B1MBTFU7BB.o
Enter the value of 1st and 2nd number: 3 15
Numbers before swapping:
num1 = 3, num2 = 15

Numbers after swapping.
num1 = 15, num2 = 3


=== Code Execution Successful ===
```

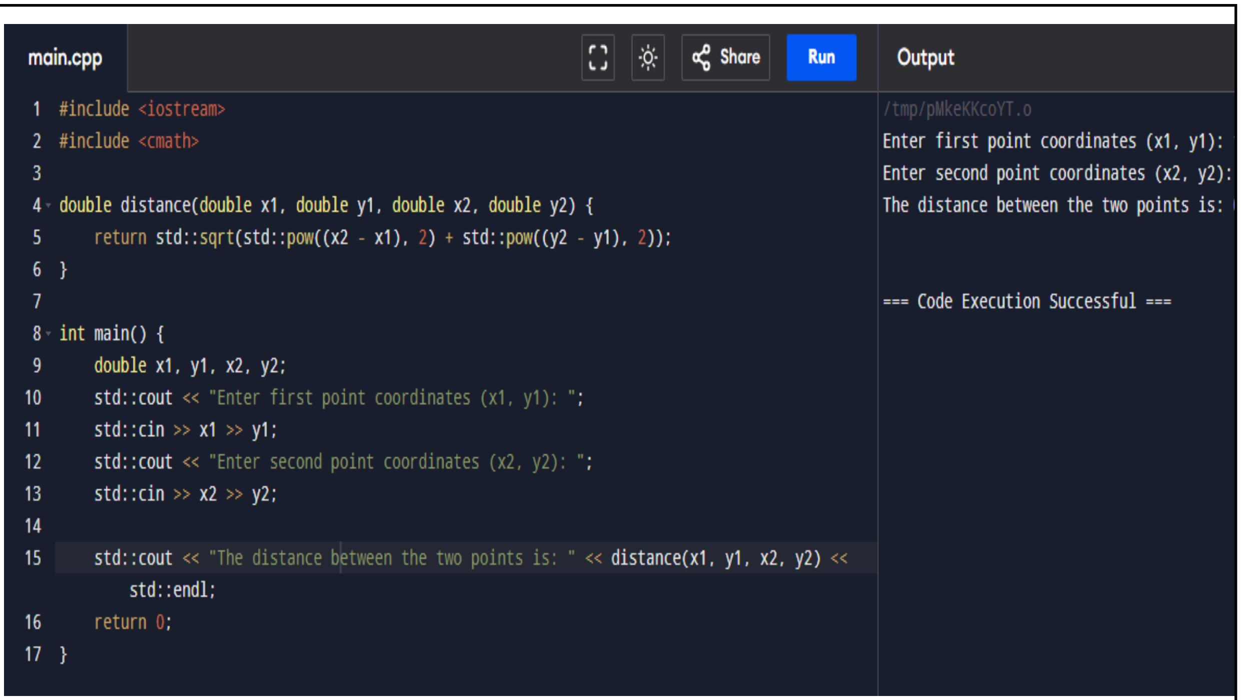
2. **Create a C++ program that has a function to convert temperature in Kelvin to Fahrenheit.**



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

int main()
{
    float Kelvin, Fahrenheit;

    cout << "\n\nKelvin to Fahrenheit Calculator\n";
    cout << "Input Kelvin Temperature: ";
    cin >> Kelvin;

    Fahrenheit = 1.8 * (Kelvin - 273) + 32;
    cout << "Temperature in Kelvin: " << Kelvin << endl;
    cout << "Temperature in Fahrenheit: " << Fahrenheit << endl;

    return 0;
}
```

```
/tmp/wWMV12vy8j.o


Kelvin to Fahrenheit Calculator
Input Kelvin Temperature: 1
Temperature in Kelvin: 1
Temperature in Fahrenheit: -457.6


=== Code Execution Successful ===
```

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

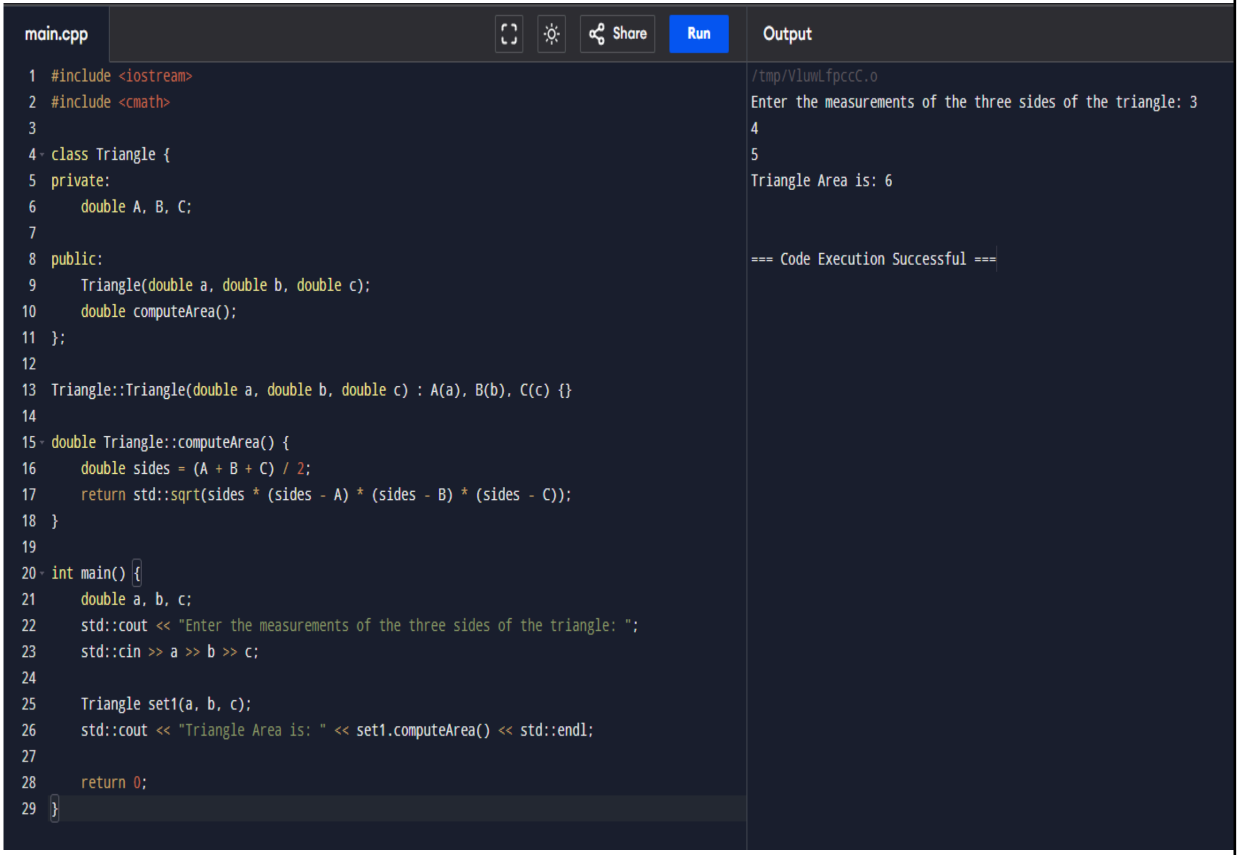


```cpp
#include <iostream>
#include <cmath>

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

int main() {
    double x1, y1, x2, y2;
    std::cout << "Enter first point coordinates (x1, y1): ";
    std::cin >> x1 >> y1;
    std::cout << "Enter second point coordinates (x2, y2): ";
    std::cin >> x2 >> y2;

    std::cout << "The distance between the two points is: " << distance(x1, y1, x2, y2) <<
        std::endl;
    return 0;
}
```

```
/tmp/pMkeKKcoYT.o
Enter first point coordinates (x1, y1):
Enter second point coordinates (x2, y2):
The distance between the two points is:


=== Code Execution Successful ===
```

4. Modify the code given in ILO B and add the following functions:

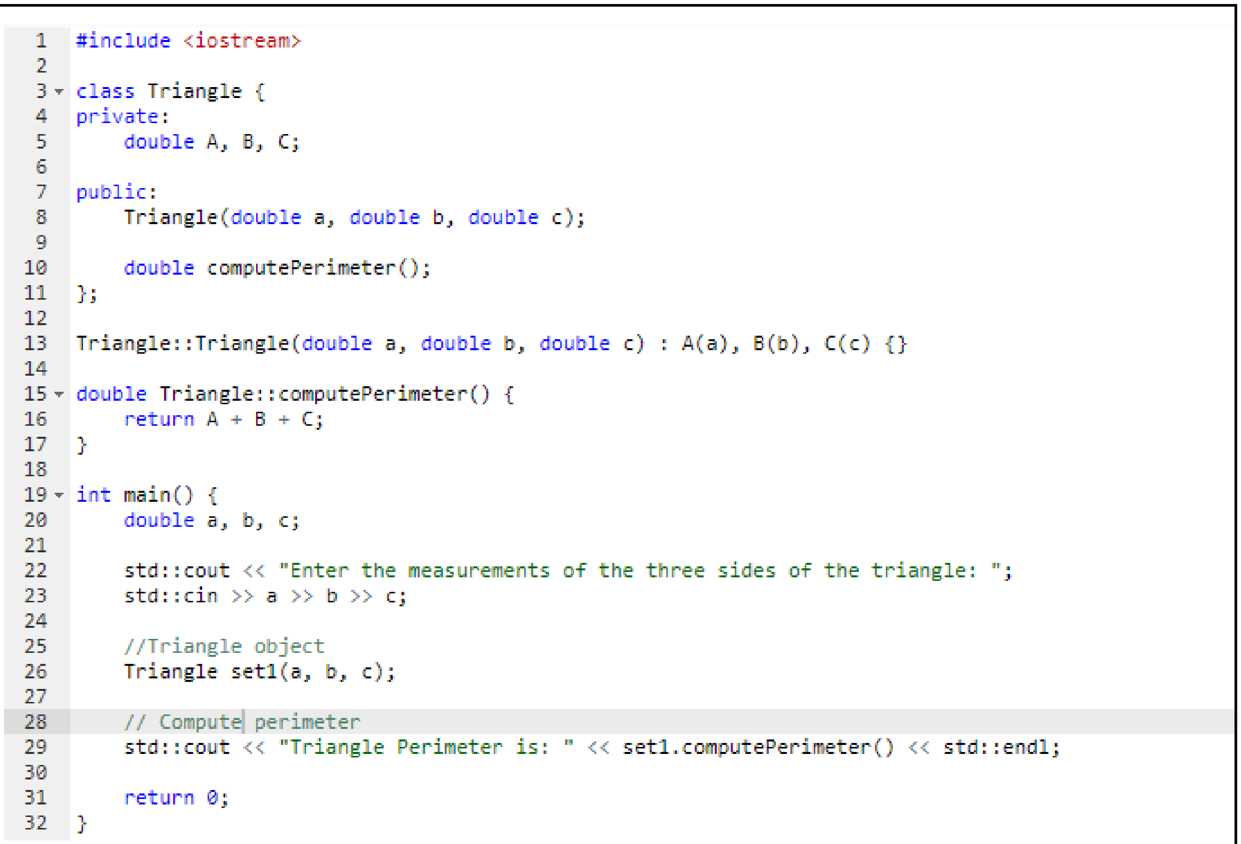
a. A function to compute for the area of a triangle



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

class Triangle {
private:
    double A, B, C;

public:
    Triangle(double a, double b, double c);
    double computeArea();
};

Triangle::Triangle(double a, double b, double c) : A(a), B(b), C(c) {}

double Triangle::computeArea() {
    double sides = (A + B + C) / 2;
    return std::sqrt(sides * (sides - A) * (sides - B) * (sides - C));
}

int main() {
    double a, b, c;
    std::cout << "Enter the measurements of the three sides of the triangle: ";
    std::cin >> a >> b >> c;

    Triangle set1(a, b, c);
    std::cout << "Triangle Area is: " << set1.computeArea() << std::endl;

    return 0;
}
```

```
/tmp/VluwLfpccC.o
Enter the measurements of the three sides of the triangle: 3
4
5
Triangle Area is: 6


=== Code Execution Successful ===
```

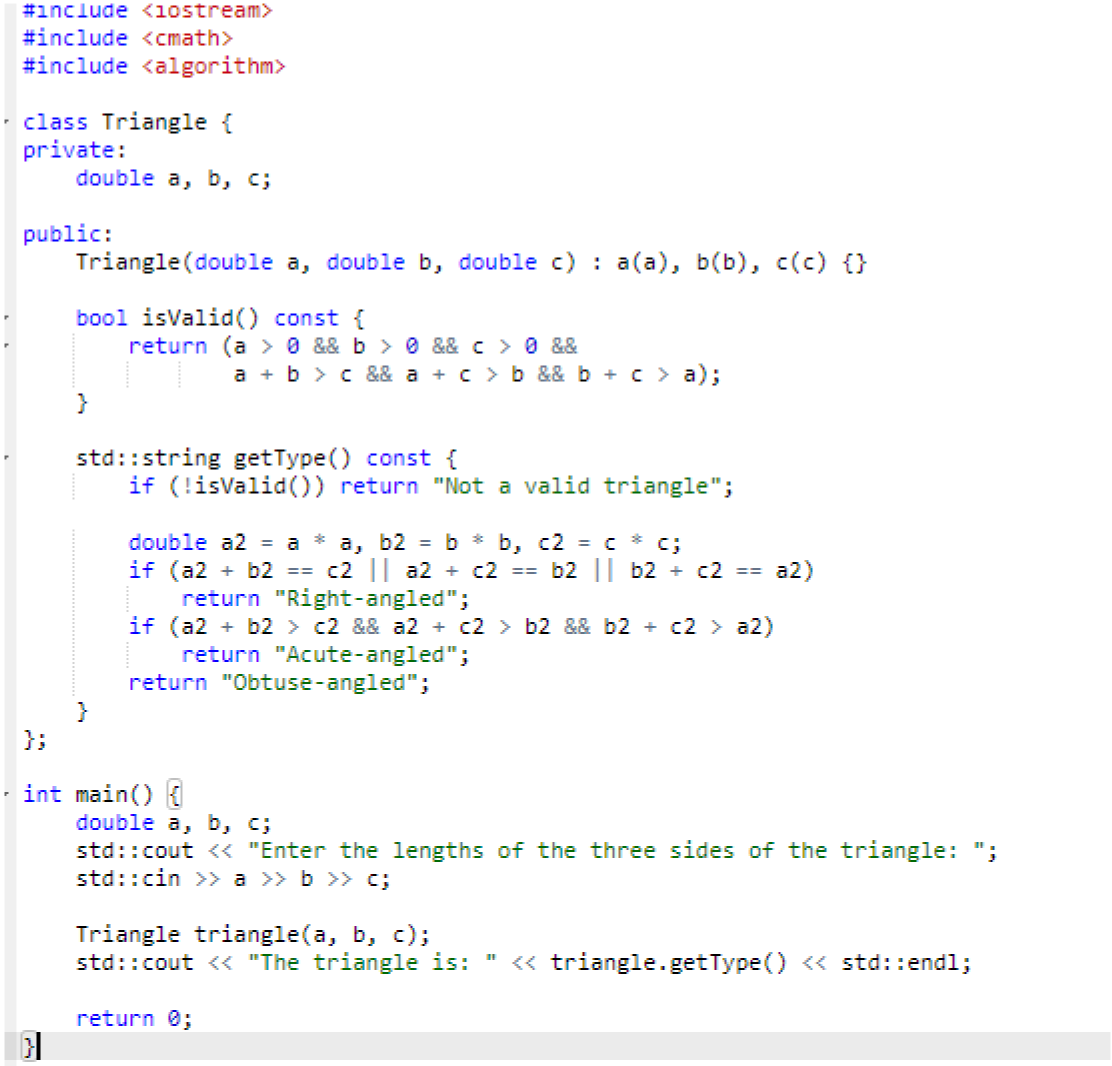
b. A function to compute for the perimeter of a triangle



```cpp
#include <iostream>

class Triangle {
private:
    double A, B, C;

public:
    Triangle(double a, double b, double c);

    double computePerimeter();
};

Triangle::Triangle(double a, double b, double c) : A(a), B(b), C(c) {}

double Triangle::computePerimeter() {
    return A + B + C;
}

int main() {
    double a, b, c;

    std::cout << "Enter the measurements of the three sides of the triangle: ";
    std::cin >> a >> b >> c;

    //Triangle object
    Triangle set1(a, b, c);

    // Compute perimeter
    std::cout << "Triangle Perimeter is: " << set1.computePerimeter() << std::endl;

    return 0;
}
```

```cpp
#include <iostream>

class Triangle {
private:
    double A, B, C;

public:
    Triangle(double a, double b, double c);

    double computePerimeter();
};

Triangle::Triangle(double a, double b, double c) : A(a), B(b), C(c) {}

double Triangle::computePerimeter() {
    return A + B + C;
}

int main() {
    double a, b, c;

    std::cout << "Enter the measurements of the three sides of the triangle: ";
    std::cin >> a >> b >> c;

    //Triangle object
    Triangle set1(a, b, c);

    // Compute perimeter
    std::cout << "Triangle Perimeter is: " << set1.computePerimeter() << std::endl;

    return 0;
}
```

c. A function that determines whether the triangle is acute-angled, obtuse-angled or ‘others.’



```
#include <iostream>
#include <cmath>
#include <algorithm>

class Triangle {
private:
    double a, b, c;

public:
    Triangle(double a, double b, double c) : a(a), b(b), c(c) {}

    bool isValid() const {
        return (a > 0 && b > 0 && c > 0 &&
                a + b > c && a + c > b && b + c > a);
    }

    std::string getType() const {
        if (!isValid()) return "Not a valid triangle";

        double a2 = a * a, b2 = b * b, c2 = c * c;
        if (a2 + b2 == c2 || a2 + c2 == b2 || b2 + c2 == a2)
            return "Right-angled";
        if (a2 + b2 > c2 && a2 + c2 > b2 && b2 + c2 > a2)
            return "Acute-angled";
        return "Obtuse-angled";
    }
};

int main() {
    double a, b, c;
    std::cout << "Enter the lengths of the three sides of the triangle: ";
    std::cin >> a >> b >> c;

    Triangle triangle(a, b, c);
    std::cout << "The triangle is: " << triangle.getType() << std::endl;

    return 0;
}
```

```
#include <iostream>
#include <cmath>
#include <algorithm>

class Triangle {
private:
    double a, b, c;

public:
    Triangle(double a, double b, double c) : a(a), b(b), c(c) {}

    bool isValid() const {
        return (a > 0 && b > 0 && c > 0 &&
                a + b > c && a + c > b && b + c > a);
    }
    std::string getType() const {
        if (!isValid()) return "Not a valid triangle";

        double a2 = a * a, b2 = b * b, c2 = c * c;
        if (a2 + b2 == c2 || a2 + c2 == b2 || b2 + c2 == a2)
            return "Right-angled";
        if (a2 + b2 > c2 && a2 + c2 > b2 && b2 + c2 > a2)
            return "Acute-angled";
        return "Obtuse-angled";
    }
};

int main() {
    double a, b, c;
    std::cout << "Enter the lengths of the three sides of the triangle: ";
    std::cin >> a >> b >> c;

    Triangle triangle(a, b, c);
    std::cout << "The triangle is: " << triangle.getType() << std::endl;

    return 0;
}
```

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?

The activity was about the summary of c++ concepts such as its structures like headers and functions, data types, operators, and classes and objects. The activity demonstrated the use of constructors, and object validation of classes. The supplementary activity applied c++ concepts or functions that can be used in practical applications/problems. They also apply the use of mathematical methods. Overall the activity has provided me an insightful look back at the basic c++ concepts and applications. Areas for improvement would be to practice more on code optimization to further solidify my understanding and proficiency in the use of c++

9. Assessment Rubric