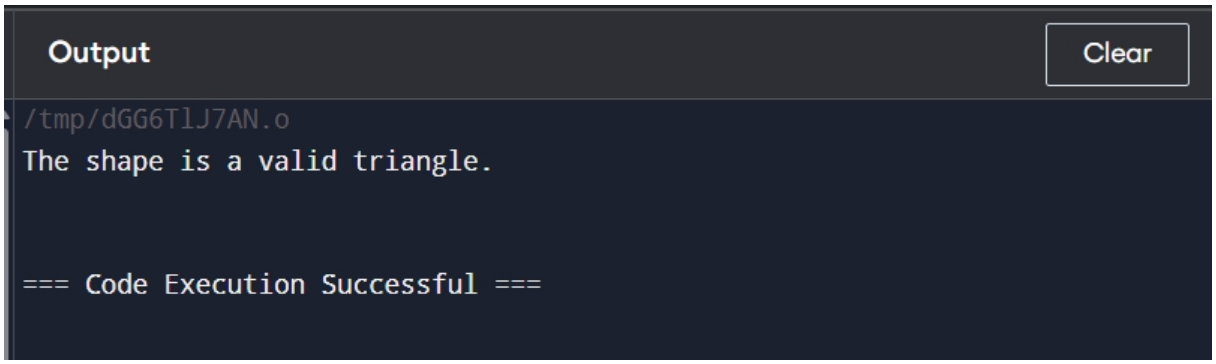
Activity No. <n>	
Hands-on Activity 1.2 Basic C++ Programming	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: 09/09/2024**
Section: CPE21s4	**Date Submitted: 09/09/2024**
Name(s): Alexzander J. Reyes	**Instructor: Mrs. Maria Sayo**

6. Output

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


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);
}

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;
}
```

TABLE 1-1. C++ STRUCTURE CODE FOR ANSWER



Logic Error in validateTriangle():

- The validateTriangle() method currently checks if the total angle is less than or equal to 180 degrees (totalAngle <= 180). This logic is incorrect because a valid triangle requires the angles to sum exactly to 180 degrees. Additionally, it doesn't check if any angle is negative or zero, which would also invalidate the triangle.

Output Inaccuracy:

- The output "The shape is a valid triangle." is incorrect because the angles (40, 30, 110) sum to exactly 180, which is a valid triangle. However, the validation logic would allow a sum less than 180, which is invalid in real-world geometry.

Recommended :

- The validateTriangle() method should be updated to check for the exact sum of 180 degrees and ensure that all angles are positive.

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

TABLE 1.2 ILO B output and comments.

Sections	**Answer**
Header File Declaration Section	#include<iostream> using namespace std;
Global Declaration Section	// no global declaration
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;
	}
	
	// Method to set angles
	void Triangle::setAngles(double A, double B, double C) {
	angleA = A;
	angleB = B;
	angleC = C;
	totalAngle = A + B + C;
	}
	
	// Method to validate the triangle
	const bool Triangle::validateTriangle() {
	return (totalAngle <= 180);
	}

Main Function	int main() {
	// Driver code
	Triangle set1(40, 30, 110); // Creating a Triangle object
	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;
	}

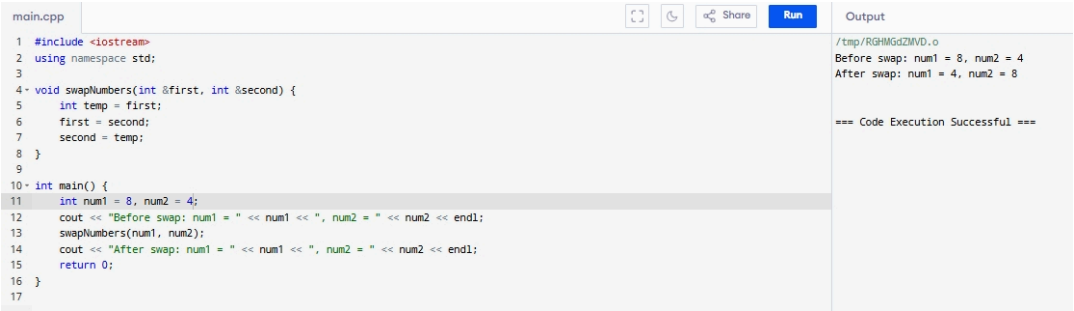
Method Definition	Triangle::Triangle(double A, double B, double C) {
	angleA = A;
	angleB = B;
	angleC = C;
	totalAngle = A + B + C;
	}
	void Triangle::setAngles(double A, double B, double C) {
	angleA = A;
	angleB = B;
	angleC = C;
	totalAngle = A + B + C;

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

7. Supplementary Activity

1.

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

void swapNumbers(int &first, int &second) {
    int temp = first;
    first = second;
    second = temp;
}

int main() {
    int num1 = 8, num2 = 4;
    cout << "Before swap: num1 = " << num1 << ", num2 = " << num2 << endl;
    swapNumbers(num1, num2);
    cout << "After swap: num1 = " << num1 << ", num2 = " << num2 << endl;
    return 0;
}
```



2.

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

// Function to convert temperature from Kelvin to Fahrenheit
double kelvinToFahrenheit(double kelvin) {
    return (kelvin - 273.15) * 9/5 + 32;
}

int main() {
    double kelvin = 200.0;
    cout << "Temperature in Fahrenheit: " << kelvinToFahrenheit(kelvin) << endl;
```

```
    return 0;
}
```

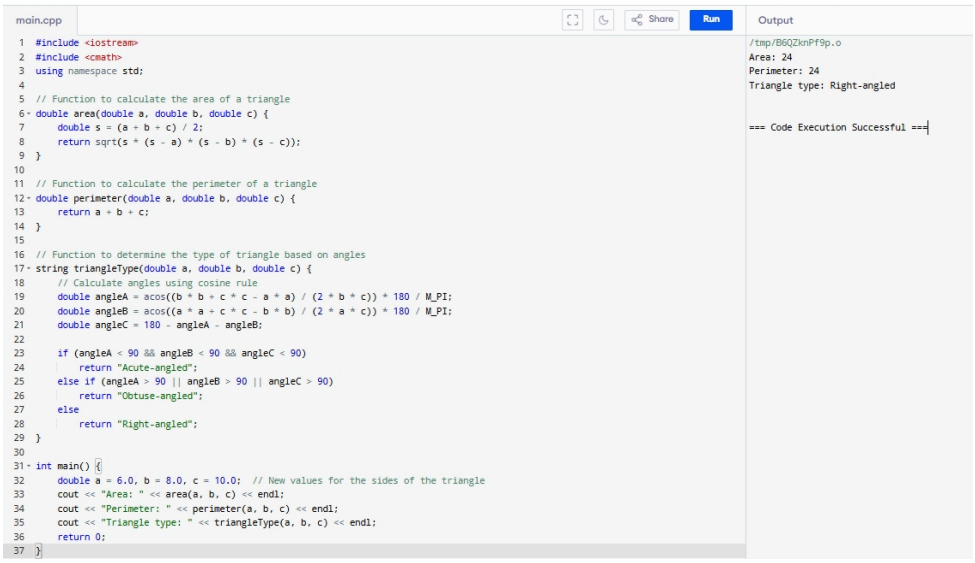
3.

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

// Function to calculate the distance between two points
double distance(double x1, double y1, double x2, double y2) {
    return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));
}

int main() {
    double x1 = 3.0, y1 = 4.0, x2 = 7.0, y2 = 8.0;  // Changed values for points
    cout << "Distance: " << distance(x1, y1, x2, y2) << endl;
    return 0;
}
```

4.

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

// Function to calculate the area of a triangle
double area(double a, double b, double c) {
    double s = (a + b + c) / 2;
    return sqrt(s * (s - a) * (s - b) * (s - c));
```

```
}

// Function to calculate the perimeter of a triangle
double perimeter(double a, double b, double c) {
    return a + b + c;
}

// Function to determine the type of triangle based on angles
string triangleType(double a, double b, double c) {
    // Calculate angles using cosine rule
    double angleA = acos((b * b + c * c - a * a) / (2 * b * c)) * 180 / M_PI;
    double angleB = acos((a * a + c * c - b * b) / (2 * a * c)) * 180 / M_PI;
    double angleC = 180 - angleA - angleB;

    if (angleA < 90 && angleB < 90 && angleC < 90)
        return "Acute-angled";
    else if (angleA > 90 || angleB > 90 || angleC > 90)
        return "Obtuse-angled";
    else
        return "Right-angled";
}

int main() {
    double a = 6.0, b = 8.0, c = 10.0;  // New values for the sides of the triangle
    cout << "Area: " << area(a, b, c) << endl;
    cout << "Perimeter: " << perimeter(a, b, c) << endl;
    cout << "Triangle type: " << triangleType(a, b, c) << endl;
    return 0;
}
```



8. Conclusion

Through this activity, we acquired the ability to utilize fundamental C++ ideas such as functions, pass-by-reference, elementary mathematical operations, and problem-solving with mathematical formulas. In addition, we worked with classes to study the fundamentals of object-oriented programming, specifically concentrating on the definition and application of functions for calculating and assessing triangle attributes.

The process offered an organized method for addressing the issues one step at a time. First, we implemented basic functions, such as temperature conversion and number swapping, to help us better grasp function prototypes and their definitions. Afterwards, we linked programming with real-world issues by employing mathematical ideas to compute distances and triangle attributes using formulas. Overall, the fundamentals of C++ were successfully reinforced by this task. I'm sure I can use C++ to solve mathematical issues and develop simple programs in the language. Practicing more complex problems, particularly those involving geometry or algorithms, might be one area for progress.

9. Assessment Rubric