Activity No. 1.2	
Hands-on Activity 1.1 Basic C++ Programming	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:Sep 9, 2024**
Section:CPE21S4	**Date Submitted:Sep 9, 2024**
Name(s):Cabilan Cyruz Peter C.	**Instructor:**

6. Output

Section	Answer
Step#1 Header File Declaration Section	#include <iostream>
Step#2 Global Declaration Section	class Triangle{ private: double totalAngle, angleA, angleB, angleC;

Step#3 Class Declaration and Method Definition Section	public:
	Triangle(double A, double B, double C);
	void setAngles(double A, double B, double C);
	const bool validateTriangle();
	};

Step#4 Main Function	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);
	}

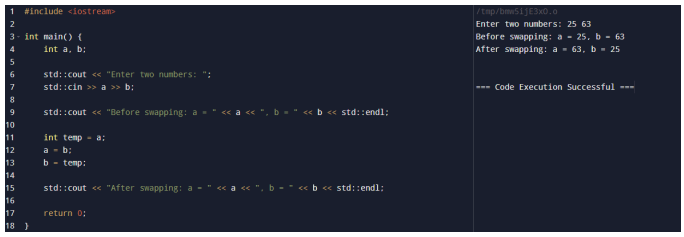
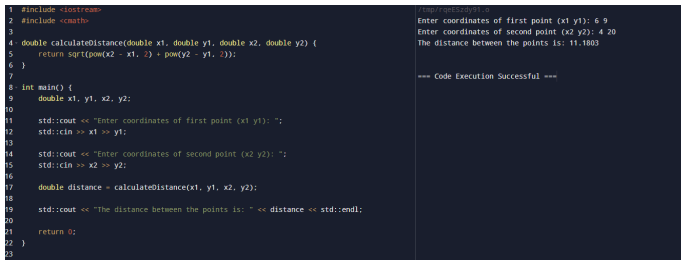
Step#5 Method Definition	int main(){
	//driver code
	Triangle set1(40, 30, 110);
	if(set1.validateTriangle()){
	std::cout << "The shape is a valid triangle.\n";
	} else {
	std::cout << "The shape is NOT a valid triangle.\n";
	}
	return 0;
	}

```cpp
#include <iostream>

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;
}
```

```
The shape is a valid triangle.
```

7. Supplementary Activity

C++ program to swap the two numbers in different variables.	
C++ program that has a function to convert temperature in Kelvin to Fahrenheit.	
C++ program that has a function that will calculate the distance between two points.	

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.'
	#include <iostream>
	#include <cmath>
	
	double kelvinToFahrenheit(double kelvin) {
	return (kelvin - 273.15) * 9/5 + 32;
	}

```
double fahrenheitToKelvin(double fahrenheit) {
	return (fahrenheit - 32) * 5/9 + 273.15;
}

double calculateTriangleArea(double base, double height)
{
	return 0.5 * base * height;
}

double calculateTrianglePerimeter(double a, double b,
double c) {
	return a + b + c;
}

std::string determineTriangleType(double a, double b,
double c) {
	double angles[3];

	angles[0] = acos((pow(b, 2) + pow(c, 2) - pow(a,
2)) / (2 * b * c)) * 180 / M_PI;
	angles[1] = acos((pow(a, 2) + pow(c, 2) - pow(b,
2)) / (2 * a * c)) * 180 / M_PI;
	angles[2] = 180 - angles[0] - angles[1];

	bool acute = angles[0] < 90 && angles[1] < 90 &&
angles[2] < 90;
	bool obtuse = angles[0] > 90 || angles[1] > 90 ||
angles[2] > 90;

	if (acute) return "Acute-angled";
	if (obtuse) return "Obtuse-angled";
	return "Right-angled or Other";
}

int main() {
	int choice;
	double temperature, result;

	std::cout << "Choose conversion:" << std::endl;
	std::cout << "1. Kelvin to Fahrenheit" << std::endl;
	std::cout << "2. Fahrenheit to Kelvin" << std::endl;
	std::cout << "Enter your choice (1 or 2): ";
	std::cin >> choice;

	if(std::cin.fail() || (choice != 1 && choice != 2)) {
	std::cerr << "Invalid choice!" << std::endl;
	return 1;
	}

	std::cout << "Enter the temperature: ";
```

```
        std::cin >> temperature;

        if(choice == 1) {
        result = kelvinToFahrenheit(temperature);
        std::cout << temperature << " K is equal to " <<
result << " F" << std::endl;
        } else if(choice == 2) {
        result = fahrenheitToKelvin(temperature);
        std::cout << temperature << " F is equal to " <<
result << " K" << std::endl;
        }

        double a, b, c, base, height;

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

        std::cout << "Enter the base and height of the
triangle (base height): ";
        std::cin >> base >> height;

        double area = calculateTriangleArea(base,
height);
        double perimeter = calculateTrianglePerimeter(a,
b, c);
        std::string type = determineTriangleType(a, b, c);

        std::cout << "The area of the triangle is: " << area
<< std::endl;
        std::cout << "The perimeter of the triangle is: " <<
perimeter << std::endl;
        std::cout << "The triangle is: " << type <<
std::endl;

        return 0;
}
```

8. Conclusion

I feel I did a good job refining the triangle validation logic and making the code more efficient. I learned a lot about ensuring that geometric conditions are checked accurately and how to use const to clarify which methods should not change the object's state. Updating TotalAngle only when needed made the code cleaner and more reliable. In the future, I want to focus on testing edge cases and making sure all parts of the code handle unexpected inputs gracefully. Overall, this exercise was a great learning experience and helped me improve my coding practices.