Activity No. <1.2>	
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:09/11/2024**
Name(s): Danica T. Guariño	**Instructor: Maria Rizette Sayo**

6. Output

Section	Answer
Header File Declaration Section	#include <iostream>
Global Declaration Section	int count = 0;

Class Declaration and
Method Definition Section
	class Triangle {
	private:
	double totalAngle, angleA, angleB, angleC;
	
	public:
	// Step 3: Constructor and methods
	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() { // 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;
	}
	``

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);
	}
	``

7. Supplementary Activity

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

main.cpp

```cpp
// Create a C++ program to swap the two numbers in different variables.

#include <iostream>

int main() {
    int a, b, temp;

    std::cout << "Enter the value of a: ";
    std::cin >> a;
    std::cout << "Enter the value of b: ";
    std::cin >> b;

    std::cout << "Before swapping: a = " << a << ", b = " << b << std::endl;

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


    std::cout << "After swapping: a = " << a << ", b = " << b << std::endl;

    return 0;
}
```

Console

```
Run  Temperature in Fahrenheit: -441.67
Run  Enter the coordinates of the first poin
Run  The distance between the points is: 9.2
Run
Enter the value of a: 15
Enter the value of b: 50
Before swapping: a = 15, b = 50
After swapping: a = 50, b = 15
```

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

main.cpp

```cpp
// Create a C++ program that has a function to convert temperature in Kelvin to Fahrenheit.

#include <iostream>
using namespace std;

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

int main() {
    double kelvin;

    cout << "Enter temperature in Kelvin: ";
    cin >> kelvin;

    double fahrenheit = kelvinToFahrenheit(kelvin);

    cout << "Temperature in Fahrenheit: " << fahrenheit << endl;

    return 0;
}
```

Console

```
Run
Enter temperature in Kelvin: 10
Temperature in Fahrenheit: -441.67
 Enter temperature in Kelvin: 10
Temperature in Fahrenheit: -441.67
```

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

```cpp
// Create a C++ program that has a function that will calculate the distance between two points.

#include <iostream>
#include <cmath>

double calculateDistance(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 the coordinates of the first point (x1 y1): ";
    std::cin >> x1 >> y1;

    std::cout << "Enter the coordinates of the second point (x2 y2): ";
    std::cin >> x2 >> y2;

    double distance = calculateDistance(x1, y1, x2, y2);
    std::cout << "The distance between the points is: " << distance << std::endl;

    return 0;
}
```

```
Run  Temperature in Fahrenheit: -441.67
Run  Enter the coordinates of the first point (x1 y1):
Run
Enter the coordinates of the first point (x1 y1): 2
4
Enter the coordinates of the second point (x2 y2): 9
10
The distance between the points is: 9.21954
```

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.'

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

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

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

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

    bool validateTriangle() {
        if (totalAngle == 180 && angleA > 0 && angleB > 0 && angleC > 0) {
            return true;
        } else {
            return false;
        }
    }

    double computeArea(double sideA, double sideB, double sideC) {
        double s = (sideA + sideB + sideC) / 2;
        return sqrt(s * (s - sideA) * (s - sideB) * (s - sideC));
    }

    double computePerimeter(double sideA, double sideB, double sideC) {
        return sideA + sideB + sideC;
```

```
Triangle 1 is valid: 1
Triangle 2 is valid: 1
Triangle 1 Area: 10.8253
Triangle 1 Perimeter: 15
Triangle 1 Type: Acute-angled
```

```cpp
    double computePerimeter(double sideA, double sideB, double sideC) {
        return sideA + sideB + sideC;
    }

    std::string determineTriangleType() {
        if (angleA < 90 && angleB < 90 && angleC < 90) {
            return "Acute-angled";
        } else if (angleA > 90 || angleB > 90 || angleC > 90) {
            return "Obtuse-angled";
        } else if (angleA == 90 || angleB == 90 || angleC == 90) {
            return "Right-angled";
        } else {
            return "Invalid angles";
        }
    }
};

int main() {
    Triangle triangle1(60, 60, 60);
    Triangle triangle2(90, 45, 45);

    std::cout << "Triangle 1 is valid: " << triangle1.validateTriangle() << std::endl;
    std::cout << "Triangle 2 is valid: " << triangle2.validateTriangle() << std::endl;

    double sideA = 5, sideB = 5, sideC = 5;
    std::cout << "Triangle 1 Area: " << triangle1.computeArea(sideA, sideB, sideC) << std::endl;
    std::cout << "Triangle 1 Perimeter: " << triangle1.computePerimeter(sideA, sideB, sideC) << std::endl;
    std::cout << "Triangle 1 Type: " << triangle1.determineTriangleType() << std::endl;

    return 0;
}
```

```
Triangle 1 is valid: 1
Triangle 2 is valid: 1
Triangle 1 Area: 10.8253
Triangle 1 Perimeter: 15
Triangle 1 Type: Acute-angled
```

8. Conclusion

Overall, this C++ activity was quite informative. I utilized C++ to demonstrate its applicability in various scenarios, including converting Kelvin to Fahrenheit, swapping two variables, and calculating the distance between two points. These exercises helped me understand more of C++'s practical applications and how to use its variables efficiently. Furthermore, the structured exercises strengthened my understanding of key programming topics such as object-oriented programming. This practice not only reinforced my theoretical understanding but also helped me improve my practical skills. While I excelled at learning and implementing complicated topics, I recognized the need to enhance my code optimization and debugging abilities. Continuous practice and additional research will be crucial in overcoming these issues and improving my C++ skills.

9. Assessment Rubric