

Activity No. <n>	
Hands-on Activity 1.1 Basic C++ Programming	
Course Code: CPE010	Program: Computer Engineering
Course Title: Data Structures and Algorithms	Date Performed: 09/09/2024
Section: CPE21s4	Date Submitted: 09/10/2024
Name(s): Anna Marie Zolina	Instructor: Ms. Maria Rizette Sayo
6. Output	
Sections	Answer
Header File Declaration Section	<pre>#include <iostream> using namespace std;</pre>
Global Declaration Section	
Class Declaration and Method Definition Section	<pre>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(); };</pre>
Main Function	<pre>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; }</pre>
Method Definition	<pre>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;</pre>

```

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.

```

#include <iostream>
using namespace std;

int main()
{
    int num1, num2, temp;
    cout << "Enter the value for number1" << endl;
    cin >> num1;
    cout << "Enter the value for number2" << endl;
    cin >> num2;

    cout << "Before swapping number_One: " << endl << num1 << " number_Two = " << num2 << endl;

    temp = num1;
    num1 = num2;
    num2 = temp;

    cout << "After the swapping number_One: " << endl << num1 << " number2_Two = " << num2 << endl;
    return 0;
}

```

```

/tmp/bDpFGAAb2F.o
Enter the value for number1
121
Enter the value for number2
203
Before swapping number_One:
121 number_Two = 203
After the swapping number_One:
203 number2_Two = 121

```

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

```

#include <iostream>

```

```

using namespace std;

int main()
{
    float Kelvin;
    float fahrenheit;
    cout << "Enter the temperature in Kelvin:" << endl;
    cin >> Kelvin;

    //convert Kelvin to Fahrenheit;
    fahrenheit = (Kelvin - 273.15) * 9/5 + 32;
    cout << "Temperature in fahrenheit: " << fahrenheit;
    return 0;
}

```

```

/tmp/r8C87R8bD6.o
Enter the temperature in Kelvin:
5
Temperature in fahrenheit: -450.67

=== Code Execution Successful ===

```

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

```

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

int main() {
    double distance, x1, x2, y1, y2;
    cout << "Enter the coordinates of the first point of x: ";
    cin >> x1;
    cout << "Enter the coordinates of the first point of y: ";
    cin >> y1;

    cout << "Enter the coordinates of the second point of x: ";
    cin >> x2;
    cout << "Enter the coordinates of the second point of y: ";
    cin >> y2;

    distance = sqrt(pow((x2 - x1), 2) + pow((y2 - y1), 2));
    cout << "The distance of the given two points is: " << distance << endl;

    return 0;
}

```

```
/tmp/7ZtGmd0ksg.o
```

```
Enter the coordinates of the first point of x: 2
Enter the coordinates of the first point of y: 2
Enter the coordinates of the second point of x: 2
Enter the coordinates of the second point of y: 2
The distance of the given two points is: 0
```

```
=== 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**
- 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>
using namespace std;

class Triangle {
private:
    double angleA, angleB, angleC, sideA, sideB, sideC;

public:
    // Constructor to initialize angles and sides of the triangle
    Triangle(double A, double B, double C, double a, double b, double c)
        : angleA(A), angleB(B), angleC(C), sideA(a), sideB(b), sideC(c) {}

    // Function to validate if the sum of angles forms a valid triangle
    bool validateTriangle() const {
        return (angleA + angleB + angleC) <= 180;
    }

    // Function to compute the area of the triangle using Heron's formula
    double computeArea() const {
        double s = (sideA + sideB + sideC) / 2; // Semi-perimeter
        return sqrt(s * (s - sideA) * (s - sideB) * (s - sideC));
    }

    // Function to compute the perimeter of the triangle
    double computePerimeter() const {
```

```

    return sideA + sideB + sideC;
}

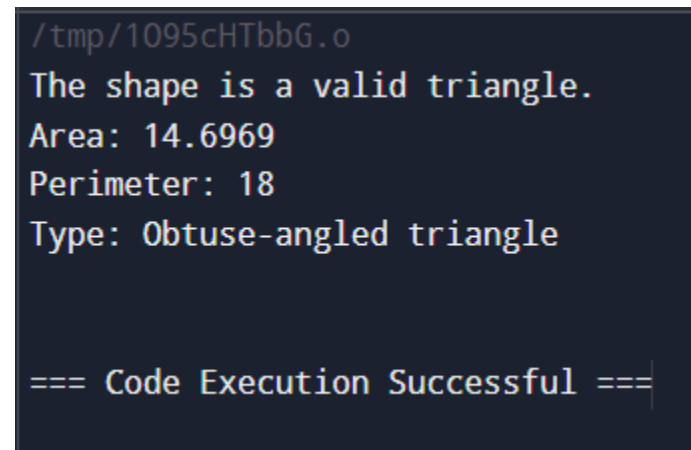
// Function to determine if the triangle is acute-angled, obtuse-angled, or other
string determineType() const {
    if (angleA < 90 && angleB < 90 && angleC < 90) return "Acute-angled triangle";
    if (angleA > 90 || angleB > 90 || angleC > 90) return "Obtuse-angled triangle";
    return "Other"; // Equilateral or right-angled cases
}
};

int main() {
    // Creating a triangle with specific angles and sides
    Triangle set1(40, 30, 110, 5, 6, 7);

    // Checking if the triangle is valid and printing its properties
    if (set1.validateTriangle()) {
        cout << "The shape is a valid triangle.\n";
        cout << "Area: " << set1.computeArea() << "\n";        // Print area of the triangle
        cout << "Perimeter: " << set1.computePerimeter() << "\n"; // Print perimeter of the triangle
        cout << "Type: " << set1.determineType() << "\n";      // Print type of triangle based on angles
    } else {
        cout << "The shape is NOT a valid triangle.\n";
    }

    return 0;
}

```



```

/tmp/1095cHTbbG.o
The shape is a valid triangle.
Area: 14.6969
Perimeter: 18
Type: Obtuse-angled triangle

=== Code Execution Successful ===

```

8. Conclusion

I've discovered that breaking apart the code's structure makes it easier for both the developers developing it and the people evaluating it to comprehend. This method of organising a C++ program is dividing it up into smaller, more manageable components like classes, functions, and headers. The readability, maintainability, and reusability are all improved by this modular structure. Developers can avoid becoming overwhelmed by the complexity of the entire codebase by defining the roles and relationships of various components clearly. This allows them to concentrate on specific areas. Moreover, clean code facilitates debugging and future updates, enabling troubleshooting and effective change implementation.

9. Assessment Rubric