

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	
<pre> #include&lt;iostream&gt; 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 &lt;= 180); }  int main(){ //driver code     Triangle set1(40, 30, 110); if(set1.validateTriangle()){         std::cout &lt;&lt; "The shape is a valid triangle.\n";     } else {         std::cout &lt;&lt; "The shape is NOT a valid triangle.\n";     }  return 0; } </pre>	

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

Output
Clear

```

/tmp/dGG6TlJ7AN.o
The shape is a valid triangle.

=== Code Execution Successful ===

```

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);

	<pre> 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 &lt;= 180); } </pre>
Main Function	<pre> int main() {     // Driver code     Triangle set1(40, 30, 110); // Creating a Triangle object     if (set1.validateTriangle()) {         std::cout &lt;&lt; "The shape is a valid triangle.\n";     } else {         std::cout &lt;&lt; "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;     angleC = C;     totalAngle = A + B + C; } </pre>

```

}

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;
}

```



```

main.cpp
1 #include <iostream>
2 using namespace std;
3
4 void swapNumbers(int &first, int &second) {
5     int temp = first;
6     first = second;
7     second = temp;
8 }
9
10 int main() {
11     int num1 = 8, num2 = 4;
12     cout << "Before swap: num1 = " << num1 << ", num2 = " << num2 << endl;
13     swapNumbers(num1, num2);
14     cout << "After swap: num1 = " << num1 << ", num2 = " << num2 << endl;
15     return 0;
16 }
17
Output
/tmp/RGHWGdQWV0.o
Before swap: num1 = 8, num2 = 4
After swap: num1 = 4, num2 = 8

=== Code Execution Successful ===

```

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;
}
```



```
main.cpp
1 #include <iostream>
2 using namespace std;
3
4 // Function to convert temperature from Kelvin to Fahrenheit
5 double kelvinToFahrenheit(double kelvin) {
6     return (kelvin - 273.15) * 9/5 + 32;
7 }
8
9 int main() {
10     double kelvin = 200.0;
11     cout << "Temperature in Fahrenheit: " << kelvinToFahrenheit(kelvin) << endl;
12     return 0;
13 }
14
```

Output

```
/tmp/36HsGzBKxy.o
Temperature in Fahrenheit: -99.67

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

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;
}
```



```
main.cpp
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4
5 // Function to calculate the distance between two points
6 double distance(double x1, double y1, double x2, double y2) {
7     return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));
8 }
9
10 int main() {
11     double x1 = 3.0, y1 = 4.0, x2 = 7.0, y2 = 8.0; // Changed values for points
12     cout << "Distance: " << distance(x1, y1, x2, y2) << endl;
13     return 0;
14 }
15
```

Output

```
/tmp/0WigLDVlMmVp.o
Distance: 5.65685

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

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;
}

```

The screenshot shows a C++ code editor with a file named 'main.cpp'. The code implements three functions: `area`, `perimeter`, and `triangleType`. The `area` function uses Heron's formula. The `perimeter` function simply sums the sides. The `triangleType` function uses the cosine rule to calculate the angles and then determines the triangle type based on whether all angles are acute, one is obtuse, or one is a right angle. The `main` function tests these functions with sides `a=6.0`, `b=8.0`, and `c=10.0`. The output window on the right shows the results: Area: 24, Perimeter: 24, and Triangle type: Right-angled. A status message at the bottom of the output window reads '=== Code Execution Successful ==='.

```

main.cpp
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4
5 // Function to calculate the area of a triangle
6 double area(double a, double b, double c) {
7     double s = (a + b + c) / 2;
8     return sqrt(s * (s - a) * (s - b) * (s - c));
9 }
10
11 // Function to calculate the perimeter of a triangle
12 double perimeter(double a, double b, double c) {
13     return a + b + c;
14 }
15
16 // Function to determine the type of triangle based on angles
17 string triangleType(double a, double b, double c) {
18     // Calculate angles using cosine rule
19     double angleA = acos((b * b + c * c - a * a) / (2 * b * c)) * 180 / M_PI;
20     double angleB = acos((a * a + c * c - b * b) / (2 * a * c)) * 180 / M_PI;
21     double angleC = 180 - angleA - angleB;
22
23     if (angleA < 90 && angleB < 90 && angleC < 90)
24         return "Acute-angled";
25     else if (angleA > 90 || angleB > 90 || angleC > 90)
26         return "Obtuse-angled";
27     else
28         return "Right-angled";
29 }
30
31 int main() {
32     double a = 6.0, b = 8.0, c = 10.0; // New values for the sides of the triangle
33     cout << "Area: " << area(a, b, c) << endl;
34     cout << "Perimeter: " << perimeter(a, b, c) << endl;
35     cout << "Triangle type: " << triangleType(a, b, c) << endl;
36     return 0;
37 }

```

Output

```

/tmp/B6QZknP9p.o
Area: 24
Perimeter: 24
Triangle type: Right-angled

=== Code Execution Successful ===

```

## 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**