| **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:** 09/09/2024 |
| **Name(s):** Anna Marie Zolina | **Instructor:** Ms. Ma zette Sayo |
| **6. Output** | |
| | **Sections** | **Answer** | | --- | --- | | **Header File Declaration Section** | #include <iostream>  using namespace std; | | **Global Declaration Section** |  | | **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();  }; | | **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.**  **#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;**  **}**    **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;**  **}**    **3. Create a C++ program that has a function that will calculate the distance between two points.**  **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.’** | |
| **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** | |
|  | |