Nathan Roe
OOP with C++
EN.605.604 Section 81
1/31/2021

Module 1 Assignment: Triangle Test

Design

Project requirements state that the code must accept 3 integer values, and provide the type of triangle based on sides. Doing so will require two steps:

1. Determine whether the sides provided can create a valid polygon
    a. To check whether a triangle is valid, I can use the Triangle Inequality Theorem. The theorem states that each pair of sides must add to be greater than the third side.
2. If valid, determine the type of triangle based on relationships between side lengths
    a. For this piece, there are three possible cases:
        i. All Three sides are equal - Equilateral
        ii. Two of three sides are equal - Isosceles
        iii. All sides are different lengths - Scalene

Two do this, I will create one class, Triangle, with functions required to conduct these two checks. The main loop will then prompt user inputs and pass these two the Triangle, which will store side lengths as member variables for access within the functions.

```cpp
// Accepts user input for integer Triangle side lengths
// and returns the side-type of the given triangle

#include "Triangle/triangle.hpp"
#include <iostream>

using namespace std;

// Function main begins triangle evaluation loop
int main()
{
    // User input variable declarations
    int sideLength1;
    int sideLength2;
    int sideLength3;

    // Print to screen and store user input values
    cout << "Triangle Side Type Calculator" << endl;
    cout << "Enter First Side Length as an Integer:" << endl;
    cin >> sideLength1;

    if (!cin.fail())
    {
        cout << "Enter Second Side Length as an Integer:" << endl;
        cin >> sideLength2;
    }

    if (!cin.fail())
    {
        cout << "Enter Third Side Length as an Integer:" << endl;
        cin >> sideLength3;
    }

    // Return error if given value is not an integer
    if (cin.fail())
    {
        cout << "Side Lengths must be Integers" << endl;
        return 1;
    }

    // Create Triangle
    Triangle *triangle = new Triangle(sideLength1,
                                      sideLength2,
                                      sideLength3);
    // Determine if Triangle is valid
    if (triangle->IsValidTriangle())
    {
        // If triangle is valid, determing side-type
        cout << "This is a(n) <" + triangle->GetTriangleTypeString() + "> Triangle" <<
endl;
        return 0;
    }

    // If triangle is invalid, print error to console
    cout << "This is not a valid triangle" << endl;
    return 1;
} // End function main
```

```cpp
#include <iostream>
using namespace std;

// Creates Triangle object with three sides as provided
// to the contstructor. Upon instantiation, the Triangle
// determines its side-type, and the user can retrieve
// a string of the type, and whether the triangle is valid.
class Triangle
{
public:
    // Object constructor, with integer side lengths
    // as inputs
    Triangle(int sideLength1, int sideLength2, int sideLength3);

    // Evaluate triangle side lengths to determine
    // whether it is a valid shape
    bool IsValidTriangle();

    // Returns a string contatining the side-type
    // of this Triangle in Title Case
    std::string GetTriangleTypeString();

private:
    // Enum of possible triangle side types
    enum class TriangleType
    {
        Invalid,
        Equilateral,
        Isoceles,
        Scalene
    };

    // Triangle side lengths
    int mSideLength1 = 0;
    int mSideLength2 = 0;
    int mSideLength3 = 0;

    TriangleType triangleType = TriangleType::Invalid;

    // Evaluates side lengths to determine
    // the side type of the triangle
    void DetermineTriangleType();
};
```

```cpp
#include "triangle.hpp"

// Triangle constructor; stores side lengths
// and determines triangle type
Triangle::Triangle(int sideLength1,
                   int sideLength2,
                   int sideLength3)
{
    mSideLength1 = sideLength1;
    mSideLength2 = sideLength2;
    mSideLength3 = sideLength3;

    this->DetermineTriangleType();
} // End constructor

// Function to determine side type based on
// comparing triangle side lengths
void Triangle::DetermineTriangleType()
{
    // Equilateral if all sides are equal
    if (mSideLength1 == mSideLength2 && mSideLength2 == mSideLength3)
    {
        this->triangleType = TriangleType::Equilateral;
    }
    // Isoceles if only two sides are equal
    else if (mSideLength1 == mSideLength2 || mSideLength2 == mSideLength3 ||
mSideLength1 == mSideLength3)
    {
        this->triangleType = TriangleType::Isoceles;
    }
    // Scalene if no sides are equal
    else if (this->IsValidTriangle())
    {
        this->triangleType = TriangleType::Scalene;
    }
    // Invalid, if the sides cannot form a triangle
    else
    {
        this->triangleType = TriangleType::Invalid;
    }
} // End function DetermineTriangleType

// Function determines whether a valid triangle can be formed
// with the given side lengths, based on the Triangle
// Inequality Theorem: https://en.wikipedia.org/wiki/Triangle_inequality
bool Triangle::IsValidTriangle()
{
    if (mSideLength1 + mSideLength2 <= mSideLength3 ||
        mSideLength2 + mSideLength3 <= mSideLength1 ||
        mSideLength1 + mSideLength3 <= mSideLength2)
    {
        return false;
    }
    else
    {
        return true;
    }
```

```
57 } // End function IsValidTriangle
58
59 // Function returns string containing triangle side type
60 std::string Triangle::GetTriangleTypeString()
61 {
62     switch (this->triangleType)
63     {
64     case TriangleType::Equilateral:
65         return "Equilateral";
66     case TriangleType::Isoceles:
67         return "Isoceles";
68     case TriangleType::Scalene:
69         return "Scalene";
70     default:
71         return "Invalid";
72     }
73 } // End function GetTriangleTypeString
```

Tests

Equilateral

```
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
5
Enter Second Side Length as an Integer:
5
Enter Third Side Length as an Integer:
5
This is a(n) <Equilateral> Triangle
nateroe63@penguin:~/GradSchool/TriangleTest$
```

Isosceles

```
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
4
Enter Second Side Length as an Integer:
4
Enter Third Side Length as an Integer:
5
This is a(n) <Isoceles> Triangle
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
4
Enter Second Side Length as an Integer:
5
Enter Third Side Length as an Integer:
4
This is a(n) <Isoceles> Triangle
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
5
Enter Second Side Length as an Integer:
4
Enter Third Side Length as an Integer:
4
This is a(n) <Isoceles> Triangle
nateroe63@penguin:~/GradSchool/TriangleTest$
```

Scalene

```
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
3
Enter Second Side Length as an Integer:
4
Enter Third Side Length as an Integer:
5
This is a(n) <Scalene> Triangle
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
5
Enter Second Side Length as an Integer:
3
Enter Third Side Length as an Integer:
4
This is a(n) <Scalene> Triangle
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
4
Enter Second Side Length as an Integer:
5
Enter Third Side Length as an Integer:
3
This is a(n) <Scalene> Triangle
nateroe63@penguin:~/GradSchool/TriangleTest$
```

Invalid

```
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
1
Enter Second Side Length as an Integer:
2
Enter Third Side Length as an Integer:
3
This is not a valid triangle
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
2
Enter Second Side Length as an Integer:
7
Enter Third Side Length as an Integer:
50
This is not a valid triangle
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
50
Enter Second Side Length as an Integer:
50
Enter Third Side Length as an Integer:
100
This is not a valid triangle
nateroe63@penguin:~/GradSchool/TriangleTest$
```

Error Handling

```
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
@
Side Lengths must be Integers
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
1
Enter Second Side Length as an Integer:
3
Enter Third Side Length as an Integer:
@
Side Lengths must be Integers
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
this is not an int
Side Lengths must be Integers
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
3.5
Enter Second Side Length as an Integer:
Side Lengths must be Integers
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
1
Enter Second Side Length as an Integer:
4.0
Enter Third Side Length as an Integer:
Side Lengths must be Integers
nateroe63@penguin:~/GradSchool/TriangleTest$ ./TriangleTest
Triangle Side Type Calculator
Enter First Side Length as an Integer:
33300000000000000000000000000000000000000000000000000000000000000000000
Side Lengths must be Integers
nateroe63@penguin:~/GradSchool/TriangleTest$
```