**main.cpp**

// Written by: Chandler Stevens

// Main test program

#include <iostream>

#include <string>

using namespace std;

// Import triangle package

#include "triangle.h"

/\*Purpose: Auxiliary function used to print triangle specifications

Parameters: value string and constant reference triangle

Returns: Nothing\*/

void printTriangleDetails(string label, const Triangle& tri)

{

cout << label << " triangle: [";

cout << tri.getSideA() << ", " << tri.getSideB() << ", " << tri.getSideC()

<< "]" << endl << " The area of the triangle is "

<< tri.TriangleArea() << endl;

if(tri.isRightTriangle())

cout << " Triangle is a right triangle" << endl;

else

cout << " Triangle is not a right triangle" << endl;

if (tri.isEquilateralTriangle())

cout << " Triangle is an equilateral triangle" << endl;

else

cout << " Triangle is not an equilateral triangle" << endl;

if (tri.isIsoscelesTriangle())

cout << " Triangle is an isosceles triangle" << endl;

else

cout << " Triangle is not an isosceles triangle" << endl;

cout << endl; // end of details for this triangle

}

/\*Purpose: Primary test function and used to call auxiliary functions

Parameters: None

Returns: Program end\*/

int main(void)

{

cout << "CSC2430 Triangle Lab: Written by Chandler Stevens" << endl;

Triangle t1; // Default triangle

printTriangleDetails("Default", t1);

Triangle t2(5, 4, 3); // convert constructor

printTriangleDetails("t2(5, 4, 3)", t2);

Triangle t3(4, 5, 3);

printTriangleDetails("t3(4, 5, 3)", t3);

Triangle t4(4, 6, 4);

printTriangleDetails("t4(4, 6, 4)", t4);

Triangle t5(2, 4, 2);

printTriangleDetails("t5(2, 4, 2)", t5);

t5 = t4; // Assignment of triangle objects

printTriangleDetails("Assigned t5=t4", t5);

t5 = Triangle(3, 3, 3); // Create a Triangle object "on the fly"

printTriangleDetails("On-The-Fly assigned Triangle(3, 3, 3)", t5);

t5 = Triangle(t5.getSideA(), 4, t5.getSideC()); // change 2nd side of t5

printTriangleDetails("Changed sideB", t5);

t5 = Triangle(2, t5.getSideB(), t5.getSideC()); // change 1st side of t5

printTriangleDetails("Changed sideA", t5);

t5 = Triangle(t5.getSideA(), t5.getSideB(), 4); // change 3rd side of t5

printTriangleDetails("Changed sideC", t5);

Triangle t6;

t6 = t4; // Assign new triangle to previous triangle

printTriangleDetails("Created t6 and assigned t6=t4", t6);

return(0);

}

**triangle.cpp**

//////////////////////////////////////////////////////////////////

// triangle.cpp Implementation file

#include <iostream>

#include <cmath>

using namespace std;

// Import triangle header

#include "triangle.h"

/\*Purpose: Default constructor

Parameters: None

Returns: Nothing\*/

Triangle::Triangle()

{

m\_a = 3;

m\_b = 4;

m\_c = 5;

}

/\*Purpose: Convert constructor

Parameters: 3 constant doubles

Returns: Nothing\*/

Triangle::Triangle(const double sideA,

const double sideB,

const double sideC)

{

// Confirm that sides make a triangle

if (sideA + sideB > sideC &&

sideA + sideC > sideB &&

sideB + sideC > sideA &&

sideA > 0 && sideB > 0 && sideC > 0)

{

m\_a = sideA;

m\_b = sideB;

m\_c = sideC;

}

else

{

// Convert to default triangle

cout << "Triangle constructor(" << sideA << ", " << sideB << ", "

<< sideC << ") - illegal side combination. Forcing default\n";

m\_a = 3;

m\_b = 4;

m\_c = 5;

}

}

/\*Purpose: Retrieves first side

Parameters: None

Returns: Double first side\*/

double Triangle::getSideA() const

{

return(m\_a);

}

/\*Purpose: Retrieves second side

Parameters: None

Returns: Double second side\*/

double Triangle::getSideB() const

{

return(m\_b);

}

/\*Purpose: Retrieve third side

Parameters: None

Returns: Double third side\*/

double Triangle::getSideC() const

{

return(m\_c);

}

/\*Purpose: Calculates the area of triangle using Herod's Method

Parameters: None

Returns: Triangle area\*/

double Triangle::TriangleArea() const

{

double semiperimeter = (m\_a + m\_b + m\_c) / 2,

area = sqrt((semiperimeter)\*(semiperimeter - m\_a)\*

(semiperimeter - m\_b)\*(semiperimeter - m\_c));

return area;

}

/\*Purpose: Check if triangle is pythagorean

Parameters: None

Returns: Boolean result\*/

bool Triangle::isRightTriangle() const

{

double hypotenuse = 0, result = 0;

if (m\_c > m\_a && m\_c > m\_b)

{

hypotenuse = m\_c;

result = sqrt(pow(m\_a, 2) + pow(m\_b, 2));

}

else if (m\_a > m\_b && m\_a > m\_c)

{

hypotenuse = m\_a;

result = sqrt(pow(m\_b, 2) + pow(m\_c, 2));

}

else

{

hypotenuse = m\_b;

result = sqrt(pow(m\_a, 2) + pow(m\_c, 2));

}

if (result == hypotenuse)

return true;

else

return false;

}

/\*Purpose: Check if triangle is equilateral

Parameters: None

Returns: Boolean result\*/

bool Triangle::isEquilateralTriangle() const

{

if (m\_a == m\_b && m\_b == m\_c)

return true;

else

return false;

}

/\*Purpose: Check if triangle is isosceles

Parameters: None

Returns: Boolean result\*/

bool Triangle::isIsoscelesTriangle() const

{

if ((m\_a == m\_b && m\_a != m\_c) ||

(m\_a == m\_c && m\_a != m\_b) ||

(m\_b == m\_c && m\_b != m\_a))

return true;

else

return false;

}

**triangle.h**

//////////////////////////////////////////////////////////////////

// triangle.h

// Type for representing triangles.

class Triangle {

public:

// Set of operations

// Default constructor

Triangle();

// Convert constructor

Triangle(const double, const double, const double);

// Retrieve first side

double getSideA() const;

// Retrieve second side

double getSideB() const;

// Retrieve third side

double getSideC() const;

// Calculate area of triangle using Herod's Method

double TriangleArea() const;

// Check if triangle is pythagorean

bool isRightTriangle() const;

// Check if triangle is equilateral

bool isEquilateralTriangle() const;

// Check if triangle is isosceles

bool isIsoscelesTriangle() const;

private:

// Set of values

// Triangle sides

double m\_a, m\_b, m\_c;

};