ex4

Sheharyar Alam Khan

February 9, 2019

Contents

1 main.cpp

The Purpose
    The purpose of this exercise is exactly the same as exercise 3 except instead of using points we're using arrays. This exercise shows us how much easier and cleaner structures make our code since we dont have to deal with different variable names or arrays.
    The Process
    To do this we get the x and y coordinates of all three points. For this I used a for loop in main.

```
#include "triangle.hpp"
#include <iostream>

using namespace std;
```

This program takes in the x and y coordinates of all three points of the triangle and stores them into an either an x array or a y array. it then passes them down to the triangle constructor to make a triangle structure. It then calls the primeter function then prints out the perimeter onto the console.

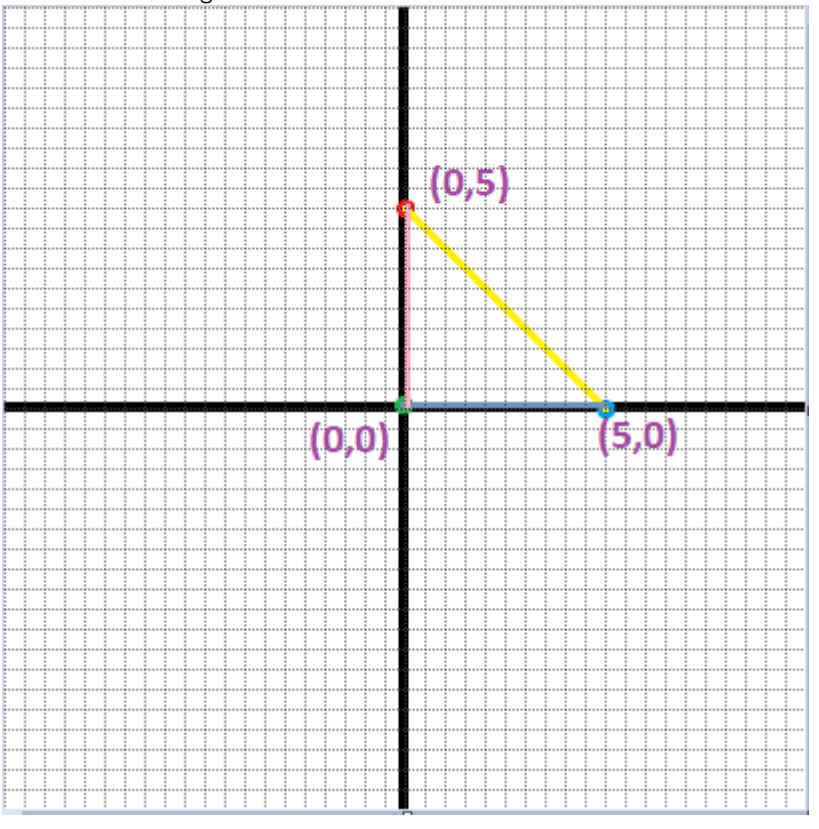
```
int main(void)
{
        float coords[6];
        //makes the user enter x coords first
        bool x = true;

        cout << "Enter Coords:" << endl;
        for (int i = 0; i < 6; i++)
        {
                //prints out X or Y depending on x
                if (x)
                {
```

```
                        cout << "X";
                        //makes it print out Y next cycle
                        x = false;
                }
                else
                {
                        cout << "Y";
                        //makes it print out X next cycle
                        x = true;
                }
                //prints i/2 makes it print out 0,0,1,1,2,2
                cout << i/2 <<" :" << endl;
                cin >> coords[i];
        }

        Triangle tri(coords[0],coords[1],coords[2],coords[3],coords[4],
            coords[5]);
        cout << "Perimeter: " << perimeter(tri) << endl;
}
```

This is what the output looks like:

```
sheharyarak@aDELL MINGW64 /c/Projects/CS124/lab1
$ ./ex4.exe
Enter Coords:
X0 :
0
Y0 :
0
X1 :
0
Y1 :
5
X2 :
5
Y2 :
0
Perimeter: 17.0711
```

This is what the diagram looks like:



2 triangle.h

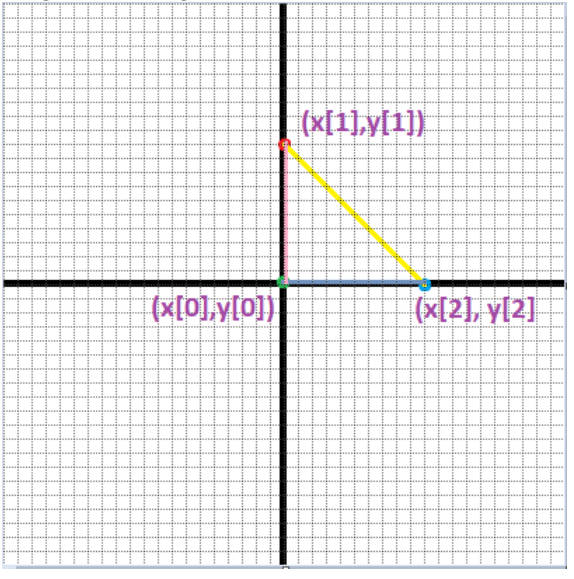
```
#ifndef TRIANGLE_H
#define TRIANGLE_H


struct  Triangle
{
        float x[3];
        float y[3];

        Triangle(float x1, float y1, float x2, float y2, float x3, float y3)
            ;
};

float   perimeter(Triangle tri);

#endif
```

3 triangle.cpp

```
#include "triangle.hpp"
#include <math.h>

Triangle::Triangle (float x1, float y1, float x2, float y2, float x3, float y3)
    This constructor takes in the x and y coords and puts them in either the x array or the
    y array in the triangle structure.
Triangle::Triangle (float x1, float y1, float x2, float y2, float x3, float
    y3)
{
        x[0] = x1;
        y[0] = y1;

        x[1] = x2;
        y[1] = y2;

        x[2] = x3;
        y[2] = y3;
}
```

`float perimeter(Triangle tri)`

This function calculates the perimeter using the distance formula. Its exactly the same as the previous function but instead of using points this function uses the x an y coordinates from the arrays inside the triangle structure and uses the distance formula to get the lenght of each of the sides then returns their sum.




```
float    perimeter(Triangle tri)
{
        float ab;
        float bc;
        float ca;

        ab = sqrt((tri.x[1] - tri.x[0]) * (tri.x[1] - tri.x[0]) + (tri.y[1]
            - tri.y[0]) * (tri.y[1] - tri.y[0]));
```

```
        bc = sqrt((tri.x[2] - tri.x[1]) * (tri.x[2] - tri.x[1]) + (tri.y[2]
            - tri.y[1]) * (tri.y[2] - tri.y[1]));
        ca = sqrt((tri.x[0] - tri.x[2]) * (tri.x[0] - tri.x[2]) + (tri.y[1]
            - tri.y[2]) * (tri.y[0] - tri.y[2]));

        return (ab + bc + ca);
}
```