Jeremy Scheuerman

COSC 220

Dr. Wang

2/4/2021

Lab_2.1

Src

```
//Jeremy Scheuerman
//Lab 2
//Dr. Wang
//Find average of 2darray
//
#include <iostream>
#include <ctime>
#include <iomanip>
using namespace std;
const int rows=5;
const int col=10;
void findAverage(int arr[][col],int rows,float avg[])
{
   //function to find average of rows in 2d array
   for (int i=0; i<rows; i++)
   {
      float sum=0;
```

```
        for (int j=0; j<3; j++)
        {
            sum+= arr[i][j];
        }
        avg[i]=sum/3;
        cout<<"The average of row "<<i+1<<" is "<<setprecision(4)<<avg[i]<<endl;
    }
}

int main()
{
    srand(time(NULL));
    //declare 2d array values
    int two_d_arr[rows][col];
    for (int i=0; i<rows; i++)
    {
        cout<<"Row "<<i+1<<": ";
        for (int j=0; j<col; j++)
        {
            two_d_arr[i][j]=rand() %80+20;
            cout<<two_d_arr[i][j]<<" ";
        }
        cout<<endl;
```

```
    }
    cout<<endl;



    float avg[5];
    //declare average array
    findAverage(two_d_arr,rows,avg);
    return 0;
}
Lab_2.2
//Jeremy Scheuerman
//COSC 220
//Lab 2
#include <iostream>
#include <cmath>
#include "Point_class.cpp"
using namespace std;



int main()
{
    Point p1(45,55);
    Point p2(76,43);
```

```
    float distance;
        //define points
        distance=p1+p2;
        cout<<"The distance between the 2 points is "<<distance<<endl;

    return 0;
}
```

Lab_2.3

```
#include <iostream>
#include <iomanip>
#include "Line_class.cpp"
using namespace std;



Line *MakeLineArray(int size)
{

    //constructs and returns an array of Line objects
    Line *line_arr=new Line [size];
    return line_arr;
}
Line Shortest (Line *ptr,int size)
{
```

```
    int shortest=0;
    for (int i=1; i<size; i++)
    {
        if (ptr[shortest].getLength()>ptr[1].getLength())
        {
            shortest=1;


        }


    }
    return ptr[shortest];


}
void sortLine(Line * ptr, int size)
{
    //use bubble sort to sort arrays
    for (int i=0; i<size; i++)
    {


        if (ptr[0].getLength()>ptr[i].getLength())
        {
            swap(ptr[0],ptr[i]);
        }
```

```
    }
}

int main()
{
    Line * ptr,longest;
    float x1,y1;
    float x2,y2;
    float distance;
    int size=100;
    ptr=MakeLineArray(size);

    for (int i=0; i<size; i++)
    {
        Point begin_pt;
        Point end_pt;
        begin_pt.setX(rand()%100+1);
        begin_pt.setY(rand()%100+1);
        end_pt.setX(rand()%100+1);
        end_pt.setY(rand()%100+1);

        ptr[i].setBegin(begin_pt);
```

```
        ptr[i].setEnd(end_pt);


    }
    Shortest(ptr,size);


    Shortest(ptr,size).showCoordinate();
    return 0;
}
```

Line_Class

```
#include "Point_class.cpp"
#include <iostream>
using namespace std;
class Line
{
public:
    //constructors
    Line();
    Line(Point b, Point e);
    //set the beginning point of the Line
    void setBegin(Point b);
    //set the end point of the line
    void setEnd(Point e);
    //return the Length of the line segment
```

```
    double getLength() const;
    //display the x and y coordinate of begin and end point
    void showCoordinate() const;
private://Point is the class defined in Question4
    Point begin;
    Point end;
};
Line::Line()
{
    begin = Point();
    end = Point();
}

Line::Line(Point b, Point e)
{
    begin = b;
    end = e;
}

void Line::setBegin(Point b)
{
    begin = b;
}
```

```
void Line::setEnd(Point e)
{
    end = e;
}

double Line::getLength() const
{
    return begin + end;
}

void Line::showCoordinate() const
{
    cout << "Begin coordinate: " << endl
         << "x: " << begin.getX()
         <<endl<< "y: " << begin.getY() << endl << endl;

    cout << "End coordinate: " << endl
         << "x: " << end.getX()<<endl
         << "y: " << end.getY() <<endl;
}
```

Point class

```
#include <cmath>
```

```
class Point
{
private:
   float x,y;
public:
   Point();
   Point(float xc,float yc);
   void setX(float xc);
   void setY(float yc);
   float getX()const;
   float getY()const;

};

Point::Point()
{
   x=0.0;
   y=0.0;

}
Point::Point(float xc,float yc)
{
   x=xc;
```

```
    y=yc;


}
void Point::setX(float xc)
{
    x=xc;
}
void Point::setY(float yc)
{
    y=yc;
}
float Point::getX()const
{
    return x;
}
float Point::getY()const
{
    return y;
}
float operator+(Point lhs,Point rhs)
{
    float dist;
```

```
    dist =sqrt(pow((lhs.getX()-rhs.getX()), 2)+pow ((lhs.getY()-rhs.getY()),2));
    return dist;

}
```