We have a point class wich represents a (x,y) coordinates.

Member variables:

double xCord which stores the x-coordinates of the point.

double yCord which stores the y-coordinates of the point.

Constructors:

Point(double x,double y) this initalizes a point with a specifix coordinates.

Point() this is a default constructor that initializes a point to (0,0).

Member functions;

double getX(), getY() this returns the x and y coordinates.

void setX(double x), void setY(double y) this sets x and y cords to a new value x and y.

void output which displays the points. The if and else is so that it can match the given example output.

operator overloads:

friend ostream& operator<< overloads the <<(output) operator to output the point in the format (x, y).

what i did here was ostream is os and its similar to cout so its the same as cout << etc <<; and i return os.

Point& p allows us to access the objects data

friend istream& operator>> overloads the >>(input) operator to input values into the point.

This is similar to the previous operator overload<< but this is a input operator instead of a output operator.

The only thing different here is "is" inputstream where it prompts you input x and y and then returns x and y as the output.

friend bool operator== this operator checks if two points are equal to each other based on thier coordinates.

In this operator we are just returning true or false to if the points x and y 1 match x and y 2.

So i did return if x1 and x2 equal and y1 and y2 if they dont equal it returns false.

Point operator+ this operator adds two points together and returns a new Point.

In here what i did was i made 2 new variables and set those two new variables = to x1+x2 and y1+y2 and i return the 2 new variables.

Point operator+(int intV) this operator does the same as the previous operator+ but instead of two points its a point and a integer.

In here what i did was similar to what i did in the other operator but instead of adding two points it add a point and a integer.

Point& operator+= this operator adds another point to the current point and then updates it.

This operator is simlar to the other operator but instead of creating a new resault without chaging the original this operator adds a value to an existing variable, updating it directly.

So instead of making 2 new doubles we can just add two points together and return \*this

Point& operator+=(int intV) this operator is the same as the previous operator+= but instead it adds a integer to both coordinates of the point and updates it.

Similar to the past operator but instead we are adding a point and a int instead if two points.

the difference between a + and += is that the + operator returns a resault without changing the original variable while += adds and then reassigns the resault to the original variable modifying it directly.

double& operaor[] allows us to access the coordinates using a index 0 for x and 1 for y and if we enter a out of bounds number it will halt with an assertion error.

what i did here was i used assert so if we enter a out of bounds number it will be a error and also added <cassert> to make this work.

After that i used a if else satement so depending on what index we are going for 0 or 1 it returns 0 for x and y for 1.

Point& operator++() this is a prefix increment operator that increase both coordinates by 1.

What i did here was a prefix incrment operator which increases x and y by 1 and returns \*this

Point operator++(int) this is a postfix operator which increments both coords by 1 then returns the original value.

This is similar to the previous operator but instead of prefix this is a postfix incrment.

What i did here was created a temp point to store the original xnad y and then increment the coordinates and then return the temp with the original vlues.

Rectangle class;

this class represents a rectangle that is defined by its width, height, and lower left corner point.

Member variables:

double height this stores the hieght of the rectangle.

double width this stores the widht of the rectangle.

Point Llcornor this represents the lower left corner of the rectangle. This uses our already made class point wich represents (x,y).

Constructors;

Rectangle(): this is a default constructor that initalizes the rectangle with width and height of 0 and llcorner at (0,0).

Rectangle(double w, doubleh, Pointc): this initializes a rectangle with specific width, height, and cornor point.

Member functions;

void output() this displays the rectangle height and width and the cornor points.

double circumference() this returns the perimeter of the rectangle which the formula is 2\*(width+height).

double area() this returns the area of the rectangles area wich the formula is width\*height.

operaor overloads:

double operator[] this operator allows acces to the rectangles width, height, cornor x and y by using an index 0 for width, 1 for height, 2 for cornor x, 3 for cornor y and like in our previous operator[] in point if we eneter a number out of bounds it dosent work.

What i did here is the same as what i did in the previous class but just different numbers and instead of just a if else its if and else if and else cause the range of our index increased.

friend ostream& operator<< this overloads the << operator to output the rectangles width height and cornor points.

Same operator as the one in Point class it just outputs different variables that belong to the rectangle class.

friend istream& operator>> this overloads the >> operator to input the rectangles diemensions and cornor points.

what i did here is similar to the input operator in point class its for width height and a extra input for the cornor point

and then i updated the values and returned is.

None-member function:

we have a non-member function called distance that uses the distance formula to get the distance.

distance formula is d=(x1-x2)^2+(y1-y2)^2

what i did here was i made two variables tx and ty and set them equal to x1-x2 and y1-y2 then i returned tx\*tx-ty\*ty.

Usage Instructions;

compilation;

you compile the program using the provided Makefile by navigating to the project directory the command is cd <your path< after your in your project path you run the make command and if everything works and runing you then run the command ./prog which should run and if you make changes to any file yoou do make clean which removes all files and you run make again to remake eberything clean.