ENSF 614 – Fall 2021

Lab 5 – Tuesday, October 26

Student Name: Aastha Patel and Bhavyai Gupta

Submission date: October 26, 2021

# Exercise B – Source file point.h

/\*

 \* File Name:               point.h

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise B

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#ifndef POINT\_H

#define POINT\_H

class Point

{

private:

    double x;

    double y;

    int id;

    static int num\_of\_objects;

public:

    Point(double a, double b);

    // REQUIRES

    //    two arguments of type double

    // PROMISES

    //    creates Point object with arguments a and b of double type

    //    assigns appropriate id to the object created

    //    increments num\_of\_objects

    ~Point();

    // PROMISES

    //    destroys the Point object and decrements num\_of\_objects

    Point(const Point &P);

    // REQUIRES

    //    reference of Point object as argument P

    // PROMISES

    //    creates Point object with deep copy of data members of P

    //    assigns appropriate id to the object created

    //    increments num\_of\_objects

    Point &operator=(const Point &rhs);

    // REQUIRES

    //    reference of Point object on right hand side of =

    // PROMISES

    //    deep copy of data members of rhs to object being created

    //    assigns appropriate id to the object being created

    //    increments num\_of\_objects

    void display() const;

    // PROMISES

    //    prints the Point object on stdout

    double getx() const;

    // PROMISES

    //    returns the x co-ordinate of Point

    double gety() const;

    // PROMISES

    //    returns the y co-ordinate of Point

    void setx(double a);

    // REQUIRES

    //    an argument of type double

    // PROMISES

    //    sets the x co-ordinate of Point as a

    void sety(double b);

    // REQUIRES

    //    an argument of type double

    // PROMISES

    //    sets the y co-ordinate of Point as b

    int counter() const;

    // PROMISES

    //    returns the num\_of\_objects

    double distance(const Point &P) const;

    // REQUIRES

    //    reference to Point object

    // PROMISES

    //    returns the distance between this Point and P on the cartesian plane

    static double distance(const Point &P, const Point &Q);

    // REQUIRES

    //    two references to Point objects as arguments

    // PROMISES

    //    returns the distance between P and Q on the cartesian plane

};

#endif

# Exercise B – Source file point.cpp

/\*

 \* File Name:               point.cpp

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise B

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "point.h"

#include <stdio.h>

#include <math.h>

using namespace std;

int Point::num\_of\_objects = 0;

Point::Point(double x, double y)

{

    this->x = x;

    this->y = y;

    this->id = ++num\_of\_objects + 1000;

}

Point::Point(const Point &P)

{

    this->x = P.getx();

    this->y = P.gety();

    this->id = ++num\_of\_objects + 1000;

}

Point &Point::operator=(const Point &rhs)

{

    if (this != &rhs)

    {

        this->x = rhs.getx();

        this->y = rhs.gety();

        this->id = ++num\_of\_objects + 1000;

    }

    return \*this;

}

Point::~Point()

{

    --num\_of\_objects;

}

void Point::display() const

{

    printf("X-coordinate: %.2f\n", getx());

    printf("Y-coordinate: %.2f\n", gety());

}

double Point::getx() const

{

    return this->x;

}

double Point::gety() const

{

    return this->y;

}

void Point::setx(double x)

{

    this->x = x;

}

void Point::sety(double y)

{

    this->y = y;

}

int Point::counter() const

{

    return num\_of\_objects;

}

double Point::distance(const Point &P) const

{

    double dx2 = pow((this->getx() - P.getx()), 2);

    double dy2 = pow((this->gety() - P.gety()), 2);

    return sqrt(dx2 + dy2);

}

double Point::distance(const Point &P, const Point &Q)

{

    double dx2 = pow((P.getx() - Q.getx()), 2);

    double dy2 = pow((P.gety() - Q.gety()), 2);

    return sqrt(dx2 + dy2);

}

# Exercise B – Source file shape.h

/\*

 \* File Name:               shape.h

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise B

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "point.h"

#ifndef SHAPE\_H

#define SHAPE\_H

class Shape

{

protected:

    Point origin;

    char \*shapeName;

public:

    Shape(double x, double y, const char \*shapeName);

    // REQUIRES

    //    two arguments of type double and a pointer to built-in string

    // PROMISES

    //    creates Shape object with the supplied arguments

    virtual ~Shape();

    // PROMISES

    //    destroys the Shape object

    //    deallocates the memory referenced by shapeName

    Shape(const Shape &s);

    // REQUIRES

    //    reference of Shape object as argument s

    // PROMISES

    //    creates Shape object with deep copy of data members of s

    Shape& operator=(const Shape &rhs);

    // REQUIRES

    //    reference of Shape object on right hand side of =

    // PROMISES

    //    deep copy of data members of rhs to object being created

    const Point &getOrigin() const;

    // PROMISES

    //    returns reference to origin

    const char \*getName() const;

    // PROMISES

    //    returns pointer to shapeName

    virtual void display() const;

    // PROMISES

    //    prints the Shape object on stdout

    virtual double distance(Shape &S) const;

    // REQUIRES

    //    reference to Shape object

    // PROMISES

    //    returns the distance between this Shape and S on the cartesian plane

    static double distance(Shape &S, Shape &T);

    // REQUIRES

    //    two references to Shape objects as arguments

    // PROMISES

    //    returns the distance between S and T on the cartesian plane

    virtual double area() const = 0;

    // PROMISES

    //    returns the area of the Shape

    virtual double perimeter() const = 0;

    // PROMISES

    //    returns the perimeter of the Shape

    void move (double dx, double dy);

    // PROMISES

    //    updates the origin of Shape by moving co-ordinates by (dx, dy)

};

#endif

# Exercise B – Source file shape.cpp

/\*

 \* File Name:               shape.cpp

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise B

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "shape.h"

#include "point.h"

#include <stdio.h>

#include <string.h>

using namespace std;

Shape::Shape(double x, double y, const char \*shapeName) : origin(Point(x, y))

{

    this->shapeName = new char[strlen(shapeName) + 1];

    strcpy(this->shapeName, shapeName);

}

Shape::~Shape()

{

    delete[] this->shapeName;

    this->shapeName = nullptr;

}

Shape::Shape(const Shape &s) : origin(Point(s.getOrigin().getx(), s.getOrigin().gety()))

{

    this->shapeName = new char[strlen(s.getName()) + 1];

    strcpy(this->shapeName, s.getName());

}

Shape &Shape::operator=(const Shape &rhs)

{

    if (this != &rhs)

    {

        delete[] this->shapeName;

        this->origin = Point(rhs.getOrigin().getx(), rhs.getOrigin().gety());

        this->shapeName = new char[strlen(rhs.getName()) + 1];

        strcpy(this->shapeName, rhs.getName());

    }

    return \*this;

}

void Shape::display() const

{

    printf("Shape Name  : %s\n", this->getName());

    this->getOrigin().display();

}

const Point &Shape::getOrigin() const

{

    return this->origin;

}

const char \*Shape::getName() const

{

    return this->shapeName;

}

double Shape::distance(Shape &S) const

{

    double dist = this->getOrigin().distance(S.getOrigin());

    return dist;

}

double Shape::distance(Shape &S, Shape &T)

{

    double dist = S.getOrigin().distance(S.getOrigin(), T.getOrigin());

    return dist;

}

void Shape::move(double dx, double dy)

{

    double old\_x = this->getOrigin().getx();

    double old\_y = this->getOrigin().gety();

    origin.setx(old\_x + dx);

    origin.sety(old\_y + dy);

}

# Exercise B – Source file square.h

/\*

 \* File Name:               square.h

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise B

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "shape.h"

#include "point.h"

#ifndef SQUARE\_H

#define SQUARE\_H

class Square : virtual public Shape

{

protected:

    double side\_a;

public:

    Square(double x, double y, double side, const char \*shapeName);

    // REQUIRES

    //    three arguments of type double and a pointer to built-in string

    // PROMISES

    //    creates Square object with the supplied arguments

    double area() const;

    // PROMISES

    //    returns area of the square

    double perimeter() const;

    // PROMISES

    //    returns perimeter of the square

    double get\_side\_a() const;

    // PROMISES

    //    returns side\_a of the Square

    void set\_side\_a(double side);

    // REQUIRES

    //    an argument of type double

    // PROMISES

    //    sets the side\_a of Square as side

    void display() const;

    // PROMISES

    //    prints the Square object to stdout

};

#endif

# Exercise B – Source file square.cpp

/\*

 \* File Name:               square.cpp

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise B

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "square.h"

#include "shape.h"

#include "point.h"

#include <stdio.h>

using namespace std;

Square::Square(double x, double y, double side, const char \*shapeName): Shape(x, y, shapeName)

{

    this->set\_side\_a(side);

}

double Square::area() const

{

    return this->get\_side\_a() \* this->get\_side\_a();

}

double Square::perimeter() const

{

    return this->get\_side\_a() \* 4;

}

double Square::get\_side\_a() const

{

    return this->side\_a;

}

void Square::set\_side\_a(double side)

{

    this->side\_a = side;

}

void Square::display() const

{

    printf("Square Name: %s\n", this->getName());

    this->getOrigin().display();

    printf("Side a: %.2f\n", this->get\_side\_a());

    printf("Area: %.2f\n", this->area());

    printf("Perimeter: %.2f\n", this->perimeter());

}

# Exercise B – Source file rectangle.h

/\*

 \* File Name:               rectangle.h

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise B

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "square.h"

#ifndef RECTANGLE\_H

#define RECTANGLE\_H

class Rectangle : public Square

{

protected:

    double side\_b;

public:

    Rectangle(double x, double y, double a, double b, const char \*shapeName);

    // REQUIRES

    //    four arguments of type double and a pointer to built-in string

    // PROMISES

    //    creates Rectangle object with the supplied arguments

    double area() const;

    // PROMISES

    //    returns area of the rectangle

    double perimeter() const;

    // PROMISES

    //    returns perimeter of the rectangle

    double get\_side\_b() const;

    // PROMISES

    //    returns side\_b of the Rectangle

    void set\_side\_b(double side);

    // REQUIRES

    //    an argument of type double

    // PROMISES

    //    sets the side\_b of Rectangle as side

    void display() const;

    // PROMISES

    //    prints the Rectangle object to stdout

};

#endif

# Exercise B – Source file rectangle.cpp

/\*

 \* File Name:               rectangle.cpp

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise B

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "rectangle.h"

#include "square.h"

#include "shape.h"

#include "point.h"

#include <stdio.h>

using namespace std;

Rectangle::Rectangle(double x, double y, double a, double b, const char \*shapeName): Shape(x, y, shapeName), Square(x, y, a, shapeName)

{

    this->set\_side\_b(b);

}

double Rectangle::area() const

{

    return (this->get\_side\_a() \* this->get\_side\_b());

}

double Rectangle::perimeter() const

{

    return (2\*(this->get\_side\_a() + this->get\_side\_b()));

}

double Rectangle::get\_side\_b() const

{

    return this->side\_b;

}

void Rectangle::set\_side\_b(double side)

{

    this->side\_b = side;

}

void Rectangle::display() const

{

    printf("Rectangle Name: %s\n", this->getName());

    this->getOrigin().display();

    printf("Side a: %.2f\n", this->get\_side\_a());

    printf("Side b: %.2f\n", this->get\_side\_b());

    printf("Area: %.2f\n", this->area());

    printf("Perimeter: %.2f\n", this->perimeter());

}

# Exercise B – Source file graphicsWorld.h

/\*

 \* File Name:               graphicsWorld.h

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise B

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#ifndef GRAPHICSWORLD\_H

#define GRAPHICSWORLD\_H

class GraphicsWorld

{

public:

    static void run();

    // PROMISES

    //    tests various functionalities implemented and print results on stdout

};

#endif

# Exercise B – Source file graphicsWorld.cpp

/\*

 \* File Name:               graphicsWorld.cpp

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise B

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "graphicsWorld.h"

#include "rectangle.h"

#include "square.h"

#include "shape.h"

#include "point.h"

#include <iostream>

using namespace std;

void GraphicsWorld::run()

{

    cout << "Authors: Aastha Patel, Bhavyai Gupta" << endl;

    // #if 0  // Change 0 to 1 to test Point

    Point m(6, 8);

    Point n(6, 8);

    n.setx(9);

    cout << "\nExpected to display the distance between m and n is: 3";

    cout << "\nThe distance between m and n is: " << m.distance(n);

    cout << "\nExpected second version of the distance function also print: 3";

    cout << "\nThe distance between m and n is again: " << Point::distance(m, n);

    // #endif // end of block to test Point

    // #if 0  // Change 0 to 1 to test Square

    cout << "\n\nTesting Functions in class Square:" << endl;

    Square s(5, 7, 12, "SQUARE - S");

    s.display();

    // #endif // end of block to test Square

    // #if 0  // Change 0 to 1 to test Rectangle

    cout << "\nTesting Functions in class Rectangle:" << endl;

    Rectangle a(5, 7, 12, 15, "RECTANGLE A");

    a.display();

    Rectangle b(16, 7, 8, 9, "RECTANGLE B");

    b.display();

    double d = a.distance(b);

    cout << "\nDistance between square a, and b is: " << d << endl;

    Rectangle rec1 = a;

    rec1.display();

    cout << "\nTesting assignment operator in class Rectangle:" << endl;

    Rectangle rec2(3, 4, 11, 7, "RECTANGLE rec2");

    rec2.display();

    rec2 = a;

    a.set\_side\_b(200);

    a.set\_side\_a(100);

    cout << "\nExpected to display the following values for objec rec2: " << endl;

    cout << "Rectangle Name: RECTANGLE A\n"

         << "X-coordinate: 5\n"

         << "Y-coordinate: 7\n"

         << "Side a: 12\n"

         << "Side b: 15\n"

         << "Area: 180\n"

         << "Perimeter: 54\n";

    cout << "\nIf it doesn't there is a problem with your assignment operator.\n"

         << endl;

    rec2.display();

    cout << "\nTesting copy constructor in class Rectangle:" << endl;

    Rectangle rec3(a);

    rec3.display();

    a.set\_side\_b(300);

    a.set\_side\_a(400);

    cout << "\nExpected to display the following values for objec rec2: " << endl;

    cout << "Rectangle Name: RECTANGLE A\n"

         << "X-coordinate: 5\n"

         << "Y-coordinate: 7\n"

         << "Side a: 100\n"

         << "Side b: 200\n"

         << "Area: 20000\n"

         << "Perimeter: 600\n";

    cout << "\nIf it doesn't there is a problem with your assignment operator.\n"

         << endl;

    rec3.display();

    // #endif // end of block to test Rectangle

    // #if 0  // Change 0 to 1 to test using array of pointer and polymorphism

    cout << "\nTesting array of pointers and polymorphism:" << endl;

    Shape \*sh[4];

    sh[0] = &s;

    sh[1] = &b;

    sh[2] = &rec1;

    sh[3] = &rec3;

    sh[0]->display();

    sh[1]->display();

    sh[2]->display();

    sh[3]->display();

    // #endif // end of block to test array of pointer and polymorphism

}

# Exercise B – Source file lab5ExB.cpp

/\*

 \* File Name:               lab5ExB.cpp

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise B

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "graphicsWorld.h"

using namespace std;

int main(int argc, char const \*argv[])

{

    GraphicsWorld::run();

    return 0;

}

# Exercise B – Program Output

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

# Exercise C – Source file circle.h

/\*

 \* File Name:               circle.h

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise C

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "rectangle.h"

#include "shape.h"

#ifndef CIRCLE\_H

#define CIRCLE\_H

class Circle : virtual public Shape

{

protected:

    double radius;

public:

    Circle(double x, double y, double r, const char \*shapeName);

    // REQUIRES

    //    three args x,y for origin r for radius and a char pointer for name

    // PROMISES

    //    create Circle object from the given args

    double area() const;

    // PROMISES

    //    calculate & return the area of circle

    double perimeter() const;

    // PROMISES

    //    calculate and returns the perimeter of circle

    double get\_radius() const;

    // PROMISES

    //    return the radius of circle

    void set\_radius(double radius);

    // REQUIRES

    //    double arg for radius

    // PROMISES

    //    set the radius of circle

    void display() const;

    // PROMISES

    //    display Circle's origin, area, perimeter and name

};

#endif

# Exercise C – Source file circle.cpp

/\*

 \* File Name:               circle.cpp

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise C

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "circle.h"

#include "shape.h"

#include "point.h"

#include <stdio.h>

#include <math.h>

using namespace std;

Circle::Circle(double x, double y, double r, const char \*shapeName) : Shape(x, y, shapeName)

{

    this->set\_radius(r);

}

double Circle::area() const

{

    return (M\_PI \* pow(this->get\_radius(), 2));

}

double Circle::perimeter() const

{

    return (2 \* M\_PI \* this->get\_radius());

}

double Circle::get\_radius() const

{

    return this->radius;

}

void Circle::set\_radius(double r)

{

    this->radius = r;

}

void Circle::display() const

{

    printf("Circle Name: %s\n", this->getName());

    this->getOrigin().display();

    printf("Radius: %.2f\n", this->get\_radius());

    printf("Area: %.2f\n", this->area());

    printf("Perimeter: %.2f\n", this->perimeter());

}

# Exercise C – Source file curvecut.h

/\*

 \* File Name:               curvecut.h

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise C

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "circle.h"

#include "rectangle.h"

#ifndef CURVECUT\_H

#define CURVECUT\_H

class CurveCut : public Circle, public Rectangle

{

protected:

    double width;

public:

    CurveCut(double x, double y, double a, double w, double r, const char \*shapeName);

    // REQUIRES

    //    five args x, y for origin a, w for rectangle r for circle and a char pointer for name

    // PROMISES

    //    create CurveCut object from the given args

    double area() const;

    // PROMISES

    //    calculate & return the area of curvecut

    double perimeter() const;

    // PROMISES

    //    calculate & return the perimeter of curvecut

    void display() const;

    // PROMISES

    //    displays CurveCut's origin, length, width, radius and name

};

#endif

# Exercise C – Source file curvecut.cpp

/\*

 \* File Name:               curvecut.cpp

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise C

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "curvecut.h"

#include "circle.h"

#include "shape.h"

#include "point.h"

#include <stdio.h>

#include <math.h>

using namespace std;

CurveCut::CurveCut(double x, double y, double w, double l, double r, const char \*shapeName) : Shape(x, y, shapeName), Circle(x, y, r, shapeName), Rectangle(x, y, w, l, shapeName)

{

    double minLength = w < l ? w : l;

    if (r > minLength)

    {

        fprintf(stderr, "\n[FAIL] The radius of the circle must be always less than or equal the smaller of the width and length. Exit!\n");

        exit(1);

    }

}

double CurveCut::area() const

{

    return (Rectangle::area() - (Circle::area() / 4));

}

double CurveCut::perimeter() const

{

    return Rectangle::perimeter() - (2 \* this->get\_radius()) + (Circle::perimeter() / 4);

}

void CurveCut::display() const

{

    printf("CurveCut Name: %s\n", this->getName());

    this->getOrigin().display();

    printf("Width: %.2f\n", this->get\_side\_a());

    printf("Length: %.2f\n", this->get\_side\_b());

    printf("Radius of the cut: %.2f\n", this->get\_radius());

}

# Exercise C – Updated source file graphicsWorld.cpp

/\*

 \* File Name:               graphicsWorld.cpp

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise B and C

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "graphicsWorld.h"

#include "curvecut.h"

#include "circle.h"

#include "rectangle.h"

#include "square.h"

#include "shape.h"

#include "point.h"

#include <iostream>

using namespace std;

void GraphicsWorld::run()

{

    cout << "Authors: Aastha Patel, Bhavyai Gupta" << endl;

    cout << "\n\n+----------------+\n";

    cout << "|   EXERCISE B   |\n";

    cout << "+----------------+\n\n";

    // #if 0  // Change 0 to 1 to test Point

    Point m(6, 8);

    Point n(6, 8);

    n.setx(9);

    cout << "\nExpected to display the distance between m and n is: 3";

    cout << "\nThe distance between m and n is: " << m.distance(n);

    cout << "\nExpected second version of the distance function also print: 3";

    cout << "\nThe distance between m and n is again: " << Point::distance(m, n);

    // #endif // end of block to test Point

    // #if 0  // Change 0 to 1 to test Square

    cout << "\n\nTesting Functions in class Square:" << endl;

    Square s(5, 7, 12, "SQUARE - S");

    s.display();

    // #endif // end of block to test Square

    // #if 0  // Change 0 to 1 to test Rectangle

    cout << "\nTesting Functions in class Rectangle:" << endl;

    Rectangle a(5, 7, 12, 15, "RECTANGLE A");

    a.display();

    Rectangle b(16, 7, 8, 9, "RECTANGLE B");

    b.display();

    double d = a.distance(b);

    cout << "\nDistance between square a, and b is: " << d << endl;

    Rectangle rec1 = a;

    rec1.display();

    cout << "\nTesting assignment operator in class Rectangle:" << endl;

    Rectangle rec2(3, 4, 11, 7, "RECTANGLE rec2");

    rec2.display();

    rec2 = a;

    a.set\_side\_b(200);

    a.set\_side\_a(100);

    cout << "\nExpected to display the following values for objec rec2: " << endl;

    cout << "Rectangle Name: RECTANGLE A\n"

         << "X-coordinate: 5\n"

         << "Y-coordinate: 7\n"

         << "Side a: 12\n"

         << "Side b: 15\n"

         << "Area: 180\n"

         << "Perimeter: 54\n";

    cout << "\nIf it doesn't there is a problem with your assignment operator.\n"

         << endl;

    rec2.display();

    cout << "\nTesting copy constructor in class Rectangle:" << endl;

    Rectangle rec3(a);

    rec3.display();

    a.set\_side\_b(300);

    a.set\_side\_a(400);

    cout << "\nExpected to display the following values for objec rec2: " << endl;

    cout << "Rectangle Name: RECTANGLE A\n"

         << "X-coordinate: 5\n"

         << "Y-coordinate: 7\n"

         << "Side a: 100\n"

         << "Side b: 200\n"

         << "Area: 20000\n"

         << "Perimeter: 600\n";

    cout << "\nIf it doesn't there is a problem with your assignment operator.\n"

         << endl;

    rec3.display();

    // #endif // end of block to test Rectangle

    // #if 0  // Change 0 to 1 to test using array of pointer and polymorphism

    cout << "\nTesting array of pointers and polymorphism:" << endl;

    Shape \*sh[4];

    sh[0] = &s;

    sh[1] = &b;

    sh[2] = &rec1;

    sh[3] = &rec3;

    sh[0]->display();

    sh[1]->display();

    sh[2]->display();

    sh[3]->display();

    // #endif // end of block to test array of pointer and polymorphism

    cout << "\n\n+----------------+\n";

    cout << "|   EXERCISE C   |\n";

    cout << "+----------------+\n\n";

    // #if 0

    cout << "\nTesting Functions in class Circle:" << endl;

    Circle c(3, 5, 9, "CIRCLE C");

    c.display();

    cout << "the area of " << c.getName() << " is: " << c.area() << endl;

    cout << "the perimeter of " << c.getName() << " is: " << c.perimeter() << endl;

    d = a.distance(c);

    cout << "\nThe distance between rectangle a and circle c is: " << d << endl;

    CurveCut rc(6, 5, 10, 12, 9, "CurveCut rc");

    rc.display();

    cout << "the area of " << rc.getName() << " is: " << rc.area() << endl;

    cout << "the perimeter of " << rc.getName() << " is: " << rc.perimeter();

    d = rc.distance(c);

    cout << "\nThe distance between rc and c is: " << d << endl;

    // Using array of Shape pointers:

    // Shape \*sh[4];

    sh[0] = &s;

    sh[1] = &a;

    sh[2] = &c;

    sh[3] = &rc;

    sh[0]->display();

    cout << "The area of " << sh[0]->getName() << " is: " << sh[0]->area();

    cout << "\nthe perimeter of " << sh[0]->getName() << " is: " << sh[0]->perimeter() << endl << endl;

    sh[1]->display();

    cout << "\nThe area of " << sh[1]->getName() << " is: " << sh[1]->area();

    cout << "\nthe perimeter of " << sh[0]->getName() << " is: " << sh[1]->perimeter() << endl << endl;

    sh[2]->display();

    cout << "\nThe area of " << sh[2]->getName() << " is: " << sh[2]->area();

    cout << "\nthe circumference of " << sh[2]->getName() << " is: " << sh[2]->perimeter() << endl << endl;

    sh[3]->display();

    cout << "\nThe area of " << sh[3]->getName() << " is: " << sh[3]->area();

    cout << "\nthe perimeter of " << sh[3]->getName() << " is: " << sh[3]->perimeter() << endl << endl;

    cout << "\nTesting copy constructor in class CurveCut:" << endl;

    CurveCut cc = rc;

    cc.display();

    cout << "\nTesting assignment operator in class CurveCut:" << endl;

    CurveCut cc2(2, 5, 100, 12, 9, "CurveCut cc2");

    cc2.display();

    cc2 = cc;

    cc2.display();

    // #endif

}

# Exercise C – Source file lab5ExC.cpp

/\*

 \* File Name:               lab5ExC.cpp

 \* Course:                  ENSF 614 - Fall 2021

 \* Lab # and Assignment #:  Lab 5 Exercise C

 \* Lab section:             B01

 \* Completed by:            Aastha Patel, Bhavyai Gupta

 \* Submission Date:         October 26, 2021

 \*/

#include "graphicsWorld.h"

using namespace std;

int main(int argc, char const \*argv[])

{

    GraphicsWorld::run();

    return 0;

}

# Exercise C – Program Output

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated