**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине: «ООПиП»

на тему: **«**Основы синтаксиса объектно-ориентированного языка программирования**»**

Выполнил: студент гр. ИТИ-21

Готченя А. А.

Принял: преподаватель

Башаримов Ю. С.

Гомель 2023

**Цель работы**: познакомиться со средой разработки *Visual Studio*, изучить основы синтаксиса языка *C#* и научиться принимать решения, основываясь на методах объектно-ориентированного программирования.

**Задание:**

1. Необходимо разработать класс, представляющий собой плоскую фигуру (ромб).
2. Реализовать проверку существования данной фигуры.
3. Фигуру следует задавать координатами вершин
4. Реализовать методы вычисления длины сторон, площади и периметра
5. Реализовать методы, проверяющие принадлежность точки, заданной своими координатами на плоскости, фигуре, её границе.
6. Класс должен быть размещен в библиотеке классов.
7. Весь код должен быть снабжён элементами документирования.
8. Сгенерировать XML-файл с документацией по проекту.
9. Создать консольное приложение с интерфейсом пользователя.
10. Подключить созданную библиотеку в консольное приложение.
11. Сгенерировать исполняемый файл для последующего переноса на другие компьютеры

**Ход работы**

На рисунке 1 изображено пользовательское меню в консольном приложении.

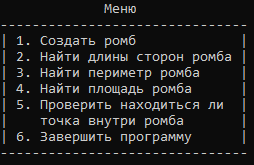


Рисунок 1 – Пользовательское меню

При выборе пункта 1 пользователь должен ввести координаты 4 вершин ромба, затем происходит проверка: является ли данная фигура ромбом? На рисунке 2 изображено успешное выполнение пункта 1 – создать ромб.

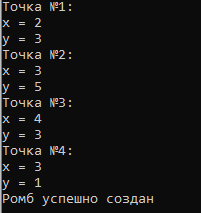


Рисунок 2 – Создание ромба

На рисунке 3 показано, что будет, если фигура, построенная на введённых координатах, окажется не ромбом.

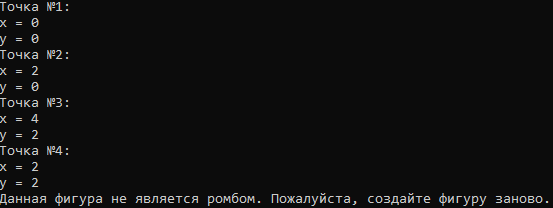


Рисунок 3 – Демонстрация создания фигуры, не являющейся ромбом

При выполнении пункта 2 происходит расчет длин сторон ромба по формуле (1.1) и их вывод.

где () и – это координаты соседних вершин ромба

На рисунке 4 изображено выполнение пункта 2 – Найти длины сторон ромба.

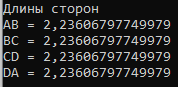


Рисунок 4 – Нахождение длин сторон ромба

При выполнении пункта 3 происходит расчет периметра ромба по формуле (1.2) и его вывод.

На рисунке 5 изображено выполнение пункта 3 – Найти периметр ромба.



Рисунок 5 – Нахождение периметра ромба

При выполнении пункта 4 происходит расчёт площади ромба по формуле (1.3) и её вывод.

На рисунке 6 изображено выполнение пункта 4 – Найти площадь ромба.

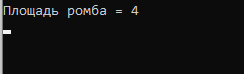


Рисунок 6 – Нахождение площади ромба

На рисунке 7 изображено выполнение пункта 5 – Проверить находиться ли точка внутри ромба.

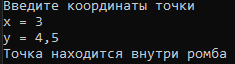


Рисунок 7 – Проверка на нахождение точки внутри ромба

В приложении А представлен код программы. В приложении B представлены сгенерированные XML-файлы документации.

**Вывод:** в результате выполнения лабораторной познакомились со средой разработки *Visual Studio*, изучили основы синтаксиса языка *C#* и научились принимать решения, основываясь на методах объектно-ориентированного программирования.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст программы**

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Laba1\_library;

namespace laba1

{

internal class Program

{

static void Main(string[] args)

{

bool isrunning = true;

int choose;

Rhombus rhombus = new Rhombus();

Point point;

while (isrunning)

{

Console.Clear();

Console.WriteLine(" Меню");

Console.WriteLine("-------------------------------");

Console.WriteLine("| 1. Создать ромб |");

Console.WriteLine("| 2. Найти длины сторон ромба |");

Console.WriteLine("| 3. Найти периметр ромба |");

Console.WriteLine("| 4. Найти площадь ромба |");

Console.WriteLine("| 5. Проверить находиться ли |");

Console.WriteLine("| точка внутри ромба |");

Console.WriteLine("| 6. Завершить программу |");

Console.WriteLine("-------------------------------");

choose = int.Parse((Console.ReadLine() ?? "0"));

switch (choose)

{

case 1:

Point[] points = new Point[4];

for (int i = 0; i < 4; i++)

{

Console.WriteLine($"Точка №{i + 1}: ");

Console.Write("x = ");

double x = double.Parse(Console.ReadLine() ?? "0");

Console.Write("y = ");

double y = double.Parse(Console.ReadLine() ?? "0");

points[i] = new Point(x, y);

}

rhombus = new Rhombus(points[0], points[1], points[2], points[3]);

if (rhombus.IsExist())

{

Console.WriteLine("Ромб успешно создан");

Console.ReadLine();

}

break;

case 2:

if (rhombus.IsExist())

{

Console.WriteLine($"Длины сторон \nAB = {Point.GetLen(rhombus.A, rhombus.B)} \n" +

$"BC = {Point.GetLen(rhombus.B, rhombus.C)} \n" +

$"CD = {Point.GetLen(rhombus.C, rhombus.D)} \n" +

$"DA = {Point.GetLen(rhombus.D, rhombus.A)}");

Console.ReadLine();

}

break;

case 3:

if (rhombus.IsExist())

{

Console.WriteLine($"Периметр ромба = {rhombus.Perimeter()}");

Console.ReadLine();

}

break;

case 4:

if (rhombus.IsExist())

{

Console.WriteLine($"Площадь ромба = {rhombus.Square()}");

Console.ReadLine();

}

break;

case 5:

if (rhombus.IsExist())

{

Console.Write("Введите координаты точки \nx = ");

double x = double.Parse(Console.ReadLine() ?? "0");

Console.Write("y = ");

double y = double.Parse(Console.ReadLine() ?? "0");

point = new Point(x, y);

if (rhombus.IsPointInsideRhombus(point))

Console.WriteLine("Точка находится внутри ромба");

else

Console.WriteLine("Точка находится вне ромба");

Console.ReadLine();

}

break;

case 6:

isrunning = false;

break;

default:

Console.WriteLine("Нет такого пункта меню");

Console.ReadLine();

break;

}

}

}

}

}

**Figure.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Laba1\_library

{

/// <summary>

/// Абстрактный класс для описания фигур

/// </summary>

public abstract class Figure

{

/// <summary>

/// Нахождение площади

/// </summary>

/// <returns>Площадь фигуры</returns>

public abstract double Square();

/// <summary>

/// Нахождение периметра

/// </summary>

/// <returns>Периметр фигуры</returns>

abstract public double Perimeter();

}

}

**Point.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Laba1\_library

{

/// <summary>

/// Класс, описывающий точку на координатной плоскости

/// </summary>

public class Point

{

/// <summary>

/// Координата по оси Ox

/// </summary>

public double X;

/// <summary>

/// Координата по оси Oy

/// </summary>

public double Y;

/// <summary>

/// Конструктор точки

/// </summary>

/// <param name="x">Координата x</param>

/// <param name="y">Координата y</param>

public Point(double x, double y)

{

X = x;

Y = y;

}

/// <summary>

/// Находит расстояние между двумя точками

/// </summary>

/// <param name="a">Точка a</param>

/// <param name="b">Точка b</param>

/// <returns>Расстояние между 2-мя точками</returns>

public static double GetLen(Point a, Point b)

{

return Math.Sqrt(Math.Pow((a.X - b.X), 2) + Math.Pow((a.Y - b.Y), 2));

}

}

}

**Line.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Laba1\_library

{

/// <summary>

/// Класс, описывающий линию на плоскости

/// </summary>

public class Line

{

/// <summary>

/// Точка, принадлежащая линии

/// </summary>

public Point A { get; set; }

/// <summary>

/// Угловой коэффициент

/// </summary>

public double K { get; set; }

/// <summary>

/// Смещение относительно оси Oy

/// </summary>

public double C { get; set; }

/// <summary>

/// Базовый конструктор линии

/// </summary>

/// <param name="p1">Любая точка, принадлежащая нашей прямой</param>

/// <param name="p2">Любая точка, принадлежащая нашей прямой, но p1 != p2</param>

public Line(Point p1, Point p2)

{

A = new Point(p1.X, p1.Y);

K = (p2.Y - p1.Y) / (p2.X - p1.X);

C = p1.Y - K \* p1.X;

}

/// <summary>

/// Находит расположение точки относительно линии.

/// </summary>

/// <param name="a">Точка, расположение которой мы ищем</param>

/// <returns>Если (C - c1) > 0, точка находится выше линии;

/// если (C - c1) = 0, то точка находится на линии; если (C - c1) < 0, то точка находится ниже линии</returns>

public double FindLocationOfPoint(Point a)

{

double c1 = a.Y - this.K \* a.X;

return (C - c1);

}

}

}

**Rhombus.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace Laba1\_library

{

/// <summary>

/// Класс, описывающий ромб

/// </summary>

public class Rhombus : Figure

{

/// <summary>

/// Вершина A

/// </summary>

public Point A;

/// <summary>

/// Вершина B

/// </summary>

public Point B;

/// <summary>

/// Вершина C

/// </summary>

public Point C;

/// <summary>

/// Вершина D

/// </summary>

public Point D;

/// <summary>

/// Конструктор ромба

/// </summary>

/// <param name="a">Вершина А</param>

/// <param name="b">Вершина B</param>

/// <param name="c">Вершина C</param>

/// <param name="d">Вершина D</param>

public Rhombus(Point a, Point b, Point c, Point d)

{

A = a;

B = b;

C = c;

D = d;

}

/// <summary>

/// Базовый конструктор для несозданного ромба

/// </summary>

public Rhombus()

{

A = B = C = D = null;

}

/// <summary>

/// Проверка на существование Ромба

/// </summary>

/// <returns></returns>

public bool IsExist()

{

// Если ромб ещё не создан

if (A != null && B != null && C != null && D != null)

{

// Если все стороны равны и 1 из углов между диагоналями равен 90 градусов

if ((Point.GetLen(A, B) == Point.GetLen(B, C))

&& (Point.GetLen(B, C) == Point.GetLen(C, D))

&& (Point.GetLen(C, D) == Point.GetLen(D, A))

&& (Math.Floor((Math.Pow((Point.GetLen(A, C) / 2), 2) + Math.Pow((Point.GetLen(B, D) / 2), 2)))

== Math.Floor(Math.Pow((Point.GetLen(A, B)), 2))))

{

return true;

}

else

{

Console.WriteLine("Данная фигура не является ромбом. Пожалуйста, создайте фигуру заново.");

Console.ReadLine();

return false;

}

}

else

{

Console.WriteLine("Вы ещё не создали фигуру");

Console.ReadLine();

return false;

}

}

/// <summary>

/// Вычисляет периметр ромба

/// </summary>

/// <returns></returns>

public override double Perimeter()

{

return (4 \* Point.GetLen(this.A, this.B));

}

/// <summary>

/// Вычисляет площадь ромба

/// </summary>

/// <returns></returns>

public override double Square()

{

return (Point.GetLen(this.A, this.C) \* Point.GetLen(this.B, this.D) / 2);

}

/// <summary>

/// Проверяет находится ли точка внутри ромба или нет

/// </summary>

/// <param name="point"></param>

/// <returns></returns>

public bool IsPointInsideRhombus(Point point)

{

Line[] sides = new Line[4]

{

new Line(A, B),

new Line(B, C),

new Line(C, D),

new Line(D, A),

};

if ((sides[0].FindLocationOfPoint(point) >= 0 && sides[2].FindLocationOfPoint(point) <= 0)

|| (sides[0].FindLocationOfPoint(point) <= 0 && sides[2].FindLocationOfPoint(point) >= 0))

{

if ((sides[1].FindLocationOfPoint(point) >= 0 && sides[3].FindLocationOfPoint(point) <= 0)

|| (sides[1].FindLocationOfPoint(point) <= 0 && sides[3].FindLocationOfPoint(point) >= 0))

{

return true;

}

}

return false;

}

}

}

**ПРИЛОЖЕНИЕ B**

(обязательное)

**XML-файлы документации**

**Shapes.xml**

<?xml version="1.0"?>

<doc>

<assembly>

<name>laba1\_library</name>

</assembly>

<members>

<member name="T:Laba1\_library.Figure">

<summary>

Абстрактный класс для описания фигур

</summary>

</member>

<member name="M:Laba1\_library.Figure.Square">

<summary>

Нахождение площади

</summary>

<returns>Площадь фигуры</returns>

</member>

<member name="M:Laba1\_library.Figure.Perimeter">

<summary>

Нахождение периметра

</summary>

<returns>Периметр фигуры</returns>

</member>

<member name="T:Laba1\_library.Line">

<summary>

Класс, описывающий линию на плоскости

</summary>

</member>

<member name="P:Laba1\_library.Line.A">

<summary>

Точка, принадлежащая линии

</summary>

</member>

<member name="P:Laba1\_library.Line.K">

<summary>

Угловой коэффициент

</summary>

</member>

<member name="P:Laba1\_library.Line.C">

<summary>

Смещение относительно оси Oy

</summary>

</member>

<member name="M:Laba1\_library.Line.#ctor(Laba1\_library.Point,Laba1\_library.Point)">

<summary>

Базовый конструктор линии

</summary>

<param name="p1">Любая точка, принадлежащая нашей прямой</param>

<param name="p2">Любая точка, принадлежащая нашей прямой, но p1 != p2</param>

</member>

<!-- Badly formed XML comment ignored for member "M:Laba1\_library.Line.FindLocationOfPoint(Laba1\_library.Point)" -->

<member name="T:Laba1\_library.Point">

<summary>

Класс, описывающий точку на координатной плоскости

</summary>

</member>

<member name="F:Laba1\_library.Point.X">

<summary>

Координата по оси Ox

</summary>

</member>

<member name="F:Laba1\_library.Point.Y">

<summary>

Координата по оси Oy

</summary>

</member>

<member name="M:Laba1\_library.Point.#ctor(System.Double,System.Double)">

<summary>

Конструктор точки

</summary>

<param name="x">Координата x</param>

<param name="y">Координата y</param>

</member>

<member name="M:Laba1\_library.Point.GetLen(Laba1\_library.Point,Laba1\_library.Point)">

<summary>

Находит расстояние между двумя точками

</summary>

<param name="a">Точка a</param>

<param name="b">Точка b</param>

<returns>Расстояние между 2-мя точками</returns>

</member>

<member name="T:Laba1\_library.Rhombus">

<summary>

Класс, описывающий ромб

</summary>

</member>

<member name="F:Laba1\_library.Rhombus.A">

<summary>

Вершина A

</summary>

</member>

<member name="F:Laba1\_library.Rhombus.B">

<summary>

Вершина B

</summary>

</member>

<member name="F:Laba1\_library.Rhombus.C">

<summary>

Вершина C

</summary>

</member>

<member name="F:Laba1\_library.Rhombus.D">

<summary>

Вершина D

</summary>

</member>

<member name="M:Laba1\_library.Rhombus.#ctor(Laba1\_library.Point,Laba1\_library.Point,Laba1\_library.Point,Laba1\_library.Point)">

<summary>

Конструктор ромба

</summary>

<param name="a">Вершина А</param>

<param name="b">Вершина B</param>

<param name="c">Вершина C</param>

<param name="d">Вершина D</param>

</member>

<member name="M:Laba1\_library.Rhombus.#ctor">

<summary>

Базовый конструктор для несозданного ромба

</summary>

</member>

<member name="M:Laba1\_library.Rhombus.IsExist">

<summary>

Проверка на существование Ромба

</summary>

<returns></returns>

</member>

<member name="M:Laba1\_library.Rhombus.Perimeter">

<summary>

Вычисляет периметр ромба

</summary>

<returns></returns>

</member>

<member name="M:Laba1\_library.Rhombus.Square">

<summary>

Вычисляет площадь ромба

</summary>

<returns></returns>

</member>

<member name="M:Laba1\_library.Rhombus.IsPointInsideRhombus(Laba1\_library.Point)">

<summary>

Проверяет находится ли точка внутри ромба или нет

</summary>

<param name="point"></param>

<returns></returns>

</member>

</members>

</doc>