**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине: «Объектно-ориентированное проектирование и программирование»

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

Выполнил: студент гр. ИТИ-21

Беловский С.С.

Принял: -------

----- -.-.

Гомель 2025

**Цель работы:** Создание консольного приложения с интерфейсом пользователя.

**Задание:** Разработать класс, представляющий собой плоскую фигуру, согласно варианту (рисунок 1).



Рисунок 1 – Вариант задания

**Ход выполнения задания**

На рисунке 2 представлен внешний вид меню и ввод точек трапеции. Код программы приведен в приложении А.

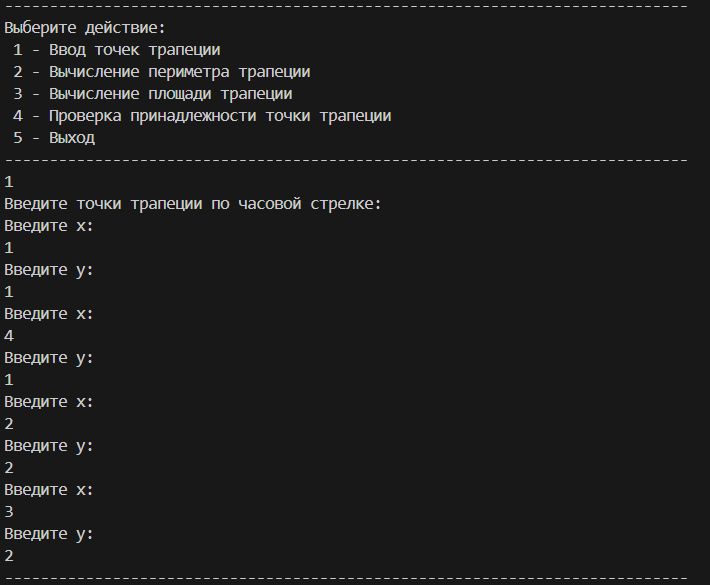


Рисунок 2 – Внешний вид меню и ввод трапеции

На рисунке 3 представлен результат выполнения тестов задания.

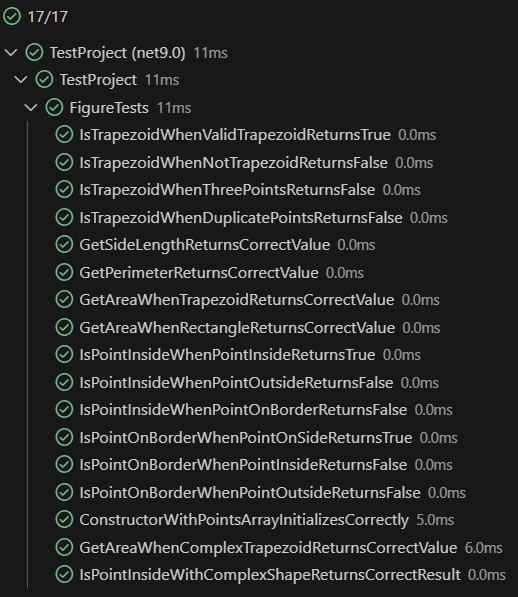


Рисунок 3 – Результат выполнения тестов

В приложении Б представлены полученные *xml* файлы.

**Вывод:** Были изучены методы создания ООП приложений на базе языка *C#*, изучены методы создания библиотек классов и их подключение к основному проекту.

**ПРИЛОЖЕНИЕ А**

(Обязательное)

**Текст программы**

**Program.cs:**

using ClassLib;

Point[] points = new Point[4];

Figure figure = new Figure(points);

bool exit = false;

bool isTrapezoid = false;

void enterPoints(Point[] points)

{

Console.WriteLine("Введите точки трапеции по часовой стрелке: ");

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

{

Console.WriteLine("Введите x: ");

double x = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Введите y: ");

double y = Convert.ToDouble(Console.ReadLine());

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

}

}

void checkPointBelongsTrapezoid()

{

Console.WriteLine("Введите x: ");

double x = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Введите y: ");

double y = Convert.ToDouble(Console.ReadLine());

Point point = new Point(x, y);

if (figure.isPointOnBorder(points, point))

Console.WriteLine("Точка находится на границе трапеции");

else if (figure.isPointInside(points, point))

Console.WriteLine("Точка принадлежит трапеции");

else

Console.WriteLine("Точка не принадлежит трапеции");

}

while (!exit)

{

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

Console.WriteLine("Выберите действие: \n 1 - Ввод точек трапеции \n 2 - Вычисление периметра трапеции \n 3 - Вычисление площади трапеции \n 4 - Проверка принадлежности точки трапеции \n 5 - Выход");

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

int choice = Convert.ToInt32(Console.ReadLine());

switch (choice)

{

case 1:

enterPoints(points);

isTrapezoid = figure.isTrapezoid(points);

if (!isTrapezoid) {

Console.WriteLine("Введенная фигура не является трапецией, попробуйте снова.");

break;

}

break;

case 2:

if (!isTrapezoid) {

Console.WriteLine("Введенная фигура не является трапецией, попробуйте снова.");

break;

}

Console.WriteLine("Периметр: " + Figure.getPerimeter(points));

break;

case 3:

if (!isTrapezoid) {

Console.WriteLine("Введенная фигура не является трапецией, попробуйте снова.");

break;

}

Console.WriteLine("Площадь: " + Figure.getArea(points));

break;

case 4:

if (!isTrapezoid) {

Console.WriteLine("Введенная фигура не является трапецией, попробуйте снова.");

break;

}

checkPointBelongsTrapezoid();

break;

case 5:

exit = true;

break;

default:

Console.WriteLine("Неверный выбор, попробуйте снова.");

break;

}

}

**Figure.cs:**

namespace ClassLib;

/// <summary>

/// Представляет геометрическую фигуру и предоставляет методы для работы с ней

/// </summary>

public class Figure

{

Point[] points;

/// <summary>

/// Инициализирует новую фигуру с указанными точками

/// </summary>

/// <param name="points">Массив точек, определяющих фигуру</param>

public Figure(Point[] points)

{

this.points = points;

}

/// <summary>

/// Проверяет, является ли фигура трапецией

/// </summary>

/// <param name="points">Массив точек фигуры</param>

/// <returns>True если фигура является трапецией, иначе False</returns>

public bool isTrapezoid(Point[] points)

{

if (points.Length != 4) return false;

if (points[0] == points[1] || points[1] == points[2] || points[2] == points[3] || points[3] == points[0]) return false;

if (isSidesParallel(points[0], points[1], points[2], points[3])

|| isSidesParallel(points[1], points[2], points[3], points[0])

|| isSidesParallel(points[2], points[3], points[0], points[1])) return true;

return false;

}

/// <summary>

/// Проверяет параллельность двух сторон

/// </summary>

/// <param name="p1">Первая точка первой стороны</param>

/// <param name="p2">Вторая точка первой стороны</param>

/// <param name="p3">Первая точка второй стороны</param>

/// <param name="p4">Вторая точка второй стороны</param>

/// <returns>True если стороны параллельны</returns>

private static bool isSidesParallel(Point p1, Point p2, Point p3, Point p4)

{

return (p1.x - p2.x) \* (p3.y - p4.y) == (p1.y - p2.y) \* (p3.x - p4.x);

}

/// <summary>

/// Вычисляет длину отрезка между двумя точками

/// </summary>

/// <param name="p1">Первая точка</param>

/// <param name="p2">Вторая точка</param>

/// <returns>Длина отрезка</returns>

public static double getSideLength(Point p1, Point p2)

{

return Math.Sqrt((p1.x - p2.x) \* (p1.x - p2.x) + (p1.y - p2.y) \* (p1.y - p2.y));

}

/// <summary>

/// Вычисляет периметр фигуры

/// </summary>

/// <param name="points">Массив точек фигуры</param>

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

public static double getPerimeter(Point[] points)

{

return getSideLength(points[0], points[1]) + getSideLength(points[1], points[2]) + getSideLength(points[2], points[3]) + getSideLength(points[3], points[0]);

}

/// <summary>

/// Вычисляет площадь фигуры (трапеции)

/// </summary>

/// <param name="points">Массив точек фигуры</param>

/// <returns>Площадь фигуры</returns>

public static double getArea(Point[] points)

{

if (isSidesParallel(points[0], points[1], points[2], points[3]))

{

return (getSideLength(points[0], points[1]) + getSideLength(points[2], points[3])) / 2 \* getHeight(points[0], points[1], points[2], points[3]);

}

if (isSidesParallel(points[1], points[2], points[3], points[0]))

{

return (getSideLength(points[1], points[2]) + getSideLength(points[3], points[0])) / 2 \* getHeight(points[1], points[2], points[3], points[0]);

}

return (getSideLength(points[2], points[3]) + getSideLength(points[0], points[1])) / 2 \* getHeight(points[2], points[3], points[0], points[1]);

}

/// <summary>

/// Вычисляет высоту трапеции

/// </summary>

/// <param name="A">Первая точка основания</param>

/// <param name="B">Вторая точка основания</param>

/// <param name="C">Первая точка противоположной стороны</param>

/// <param name="D">Вторая точка противоположной стороны</param>

/// <returns>Высота трапеции</returns>

private static double getHeight(Point A, Point B, Point C, Point D)

{

double dirX = B.x - A.x;

double dirY = B.y - A.y;

double acX = C.x - A.x;

double acY = C.y - A.y;

double area = Math.Abs(dirX \* acY - dirY \* acX);

double baseLength = Math.Sqrt(dirX \* dirX + dirY \* dirY);

return area / baseLength;

}

/// <summary>

/// Проверяет, находится ли точка на границе фигуры

/// </summary>

/// <param name="points">Массив точек фигуры</param>

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

/// <returns>True если точка находится на границе</returns>

public bool isPointOnBorder(Point[] points, Point point)

{

return isPointOnLine(points[0], points[1], point) ||

isPointOnLine(points[1], points[2], point) ||

isPointOnLine(points[2], points[3], point) ||

isPointOnLine(points[3], points[0], point);

}

/// <summary>

/// Проверяет, находится ли точка внутри фигуры

/// </summary>

/// <param name="points">Массив точек фигуры</param>

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

/// <returns>True если точка находится внутри фигуры</returns>

public bool isPointInside(Point[] points, Point point)

{

int count = 0;

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

{

var a = points[i];

var b = points[(i + 1) % 4];

if ((a.y > point.y) != (b.y > point.y) && point.x < (b.x - a.x) \* (point.y - a.y) / (b.y - a.y + 1e-10) + a.x)

{

count++;

}

}

return count % 2 == 1;

}

/// <summary>

/// Проверяет, находится ли точка на отрезке между двумя точками

/// </summary>

/// <param name="p1">Первая точка отрезка</param>

/// <param name="p2">Вторая точка отрезка</param>

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

/// <returns>True если точка находится на отрезке</returns>

private bool isPointOnLine(Point p1, Point p2, Point point)

{

double cross = (point.y - p1.y) \* (p2.x - p1.x) - (point.x - p1.x) \* (p2.y - p1.y);

if (cross != 0) return false;

double dot = (point.x - p1.x) \* (p2.x - p1.x) + (point.y - p1.y) \* (p2.y - p1.y);

if (dot < 0) return false;

double lenSq = (p2.x - p1.x) \* (p2.x - p1.x) + (p2.y - p1.y) \* (p2.y - p1.y);

return dot <= lenSq;

}

}

**Point.cs:**

namespace ClassLib;

/// <summary>

/// Представляет точку в двумерном пространстве

/// </summary>

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

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

public record struct Point(double x, double y);

**ПРИЛОЖЕНИЕ Б**

(Обязательное)

**Структура программы**

**ClassLib.xml**

<?xml version="1.0"?>

<doc>

<assembly>

<name>ClassLib</name>

</assembly>

<members>

<member name="T:ClassLib.Figure">

<summary>

Представляет геометрическую фигуру и предоставляет методы для работы с ней

</summary>

</member>

<member name="M:ClassLib.Figure.#ctor(ClassLib.Point[])">

<summary>

Инициализирует новую фигуру с указанными точками

</summary>

<param name="points">Массив точек, определяющих фигуру</param>

</member>

<member name="M:ClassLib.Figure.isTrapezoid(ClassLib.Point[])">

<summary>

Проверяет, является ли фигура трапецией

</summary>

<param name="points">Массив точек фигуры</param>

<returns>True если фигура является трапецией, иначе False</returns>

</member>

<member name="M:ClassLib.Figure.isSidesParallel(ClassLib.Point,ClassLib.Point,ClassLib.Point,ClassLib.Point)">

<summary>

Проверяет параллельность двух сторон

</summary>

<param name="p1">Первая точка первой стороны</param>

<param name="p2">Вторая точка первой стороны</param>

<param name="p3">Первая точка второй стороны</param>

<param name="p4">Вторая точка второй стороны</param>

<returns>True если стороны параллельны</returns>

</member>

<member name="M:ClassLib.Figure.getSideLength(ClassLib.Point,ClassLib.Point)">

<summary>

Вычисляет длину отрезка между двумя точками

</summary>

<param name="p1">Первая точка</param>

<param name="p2">Вторая точка</param>

<returns>Длина отрезка</returns>

</member>

<member name="M:ClassLib.Figure.getPerimeter(ClassLib.Point[])">

<summary>

Вычисляет периметр фигуры

</summary>

<param name="points">Массив точек фигуры</param>

<returns>Периметр фигуры</returns>

</member>

<member name="M:ClassLib.Figure.getArea(ClassLib.Point[])">

<summary>

Вычисляет площадь фигуры (трапеции)

</summary>

<param name="points">Массив точек фигуры</param>

<returns>Площадь фигуры</returns>

</member>

<member name="M:ClassLib.Figure.getHeight(ClassLib.Point,ClassLib.Point,ClassLib.Point,ClassLib.Point)">

<summary>

Вычисляет высоту трапеции

</summary>

<param name="A">Первая точка основания</param>

<param name="B">Вторая точка основания</param>

<param name="C">Первая точка противоположной стороны</param>

<param name="D">Вторая точка противоположной стороны</param>

<returns>Высота трапеции</returns>

</member>

<member name="M:ClassLib.Figure.isPointOnBorder(ClassLib.Point[],ClassLib.Point)">

<summary>

Проверяет, находится ли точка на границе фигуры

</summary>

<param name="points">Массив точек фигуры</param>

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

<returns>True если точка находится на границе</returns>

</member>

<member name="M:ClassLib.Figure.isPointInside(ClassLib.Point[],ClassLib.Point)">

<summary>

Проверяет, находится ли точка внутри фигуры

</summary>

<param name="points">Массив точек фигуры</param>

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

<returns>True если точка находится внутри фигуры</returns>

</member>

<member name="M:ClassLib.Figure.isPointOnLine(ClassLib.Point,ClassLib.Point,ClassLib.Point)">

<summary>

Проверяет, находится ли точка на отрезке между двумя точками

</summary>

<param name="p1">Первая точка отрезка</param>

<param name="p2">Вторая точка отрезка</param>

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

<returns>True если точка находится на отрезке</returns>

</member>

<member name="T:ClassLib.Point">

<summary>

Представляет точку в двумерном пространстве

</summary>

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

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

</member>

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

<summary>

Представляет точку в двумерном пространстве

</summary>

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

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

</member>

<member name="P:ClassLib.Point.x">

<summary>Координата X точки</summary>

</member>

<member name="P:ClassLib.Point.y">

<summary>Координата Y точки</summary>

</member>

</members>

</doc>