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 программирование»

**«Основы синтаксиса объектно-ориентированного языка программирования»**

Выполнил: студент группы ИТИ-21

Говядкова П. Ю.

Принял: преподаватель

Карабчикова Е. А.

Гомель 2020

**Цель работы:** Изучить основы синтаксиса объектно-ориентированного языка программирования.

**Задание:**

1. Разработать класс представляющий собой плоскую фигуру, согласно варианта.



Рисунок 1 – Вариант задания

2. Реализовать метод проверки существования данной фигуры.

3. Реализовать методы вычисление длин сторон, площади и периметра.

4. Реализовать метод, проверяющий принадлежность точки, заданной своими координатами на плоскости, фигуре, её границе.

5. Класс разместить в библиотеке классов.

6.Снабдить код элементами документирования.

7. Создать консольное приложение с интерфейсом пользователя.

8. Подключить созданную библиотеку в консольное приложение.

9. Верифицировать разработанное приложение.

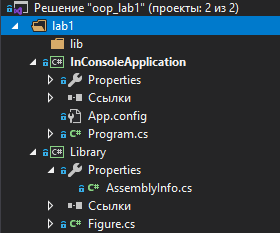


Рисунок 2 - Структура C# решения

**Выполнение программы:**

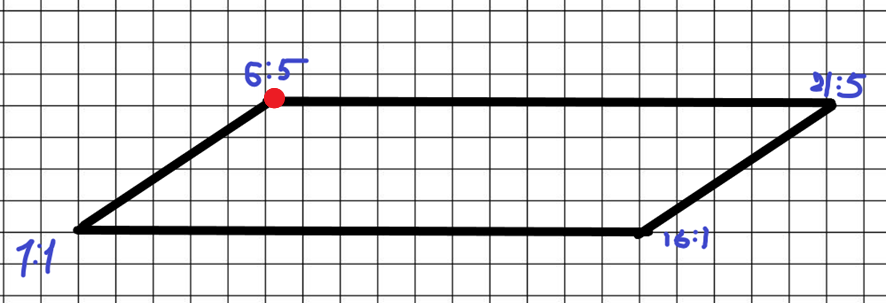


Рисунок 3 – Задаваемая фигура

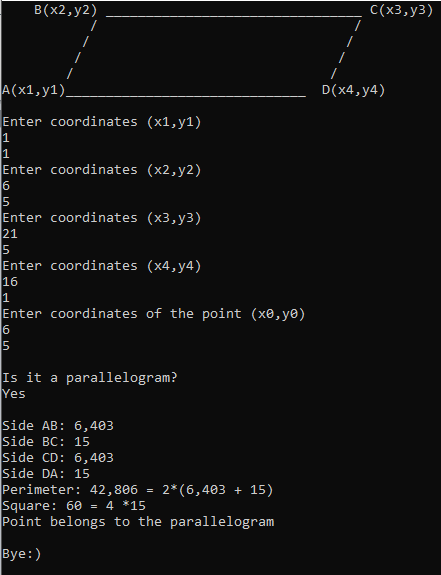


Рисунок 4 – Вывод результата выполнения программы для заданной фигуры

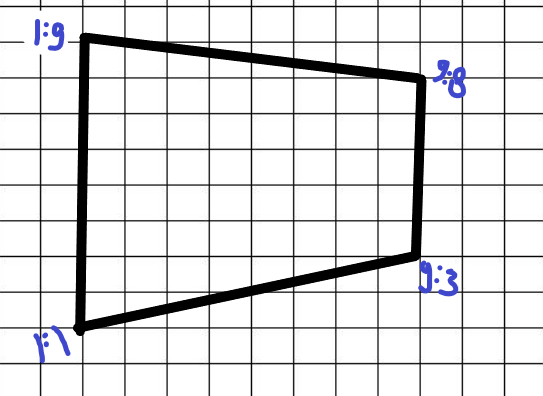


Рисунок 5 – Задаваемая фигура

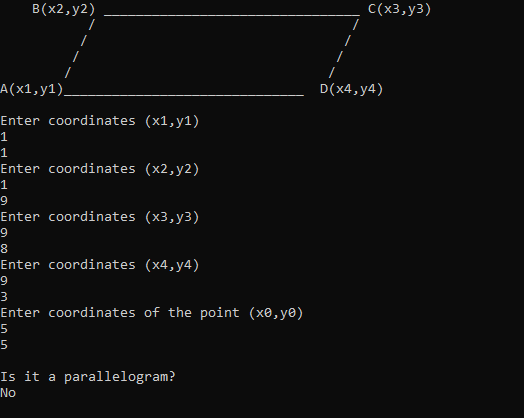


Рисунок 6 – Вывод результата выполнения программы для заданной фигуры

**Вывод:** в ходе выполнения лабораторной работы былиизучены основы синтаксиса объектно-ориентированного языка программирования. Разработан класс, представляющий собой параллелограмм. Реализованы методы: проверка существования данной фигуры, вычисление длин сторон, площади и периметра, проверка принадлежности точки с координатами x, y данной фигуре. Класс помещён в библиотеку классов. Создано консольное приложение с интерфейсом пользователя.

**Приложение А**

**Листинг программы**

**Класс Figure:**

using System;

namespace InConsoleApplication

{

public class Figure

{

private double \_x1, \_x2, \_x3, \_x4, \_y1, \_y2, \_y3, \_y4;

public Figure(double x1, double x2, double x3, double x4, double y1, double y2, double y3, double y4)

{

\_x1 = x1;

\_x2 = x2;

\_x3 = x3;

\_x4 = x4;

\_y1 = y1;

\_y2 = y2;

\_y3 = y3;

\_y4 = y4;

}

public double Height

{

get

{

return (\_y2 - \_y1);

}

}

public double LeftSide

{

get

{

return Math.Round(Math.Sqrt(Math.Pow(\_x2 - \_x1, 2) + Math.Pow(Height, 2)),3);

}

}

public double SinAlpha

{

get

{

return Math.Sin(Height/ LeftSide);

}

}

public double UpperSide

{

get

{

return Math.Round((\_x3 - \_x2),3);

}

}

public double RightSide

{

get

{

return Math.Round(Math.Sqrt(Math.Pow(\_x3 - \_x4, 2) + Math.Pow(Height, 2)),3);

}

}

public double CosBetta

{

get

{

return Math.Cos(Math.PI - SinAlpha);

}

}

public double FirstDiagonal

{

get

{

return (Math.Sqrt(Math.Pow(LeftSide, 2) + Math.Pow(UpperSide, 2) - 2 \* LeftSide \* UpperSide \* CosBetta));

}

}

public double SecondDiagonal

{

get

{

return (Math.Sqrt(Math.Pow(LeftSide, 2) + Math.Pow(UpperSide, 2) + 2 \* LeftSide \* UpperSide \* CosBetta));

}

}

public double UnderSide

{

get

{

return Math.Round(Math.Sqrt((Math.Pow(FirstDiagonal,2)+Math.Pow(SecondDiagonal,2)-2\*LeftSide\*LeftSide)/2),3);

}

}

public double Perimeter

{

get

{

return (2 \* (LeftSide + UpperSide));

}

}

public double Square

{

get

{

return (Height \* UnderSide);

}

}

public static bool IsParallelogram(double x1, double x2, double x3, double x4, double y1, double y2, double y3, double y4)

{

double coefficient\_for\_LeftSide = (y2 - y1) / (x2 - x1);

double coefficient\_for\_RightSide = (y4 - y3) / (x4 - x3);

double coefficient\_for\_UpperSide = (y3 - y2) / (x3 - x2);

double coefficient\_for\_UnderSide = (y4 - y1) / (x4 - x1);

return ((coefficient\_for\_LeftSide == coefficient\_for\_RightSide && coefficient\_for\_UpperSide == coefficient\_for\_UnderSide) && coefficient\_for\_LeftSide != coefficient\_for\_UpperSide);

}

public static bool IsBelong( double x0, double y0, double x1, double x2, double x3, double x4, double y1, double y2, double y3, double y4)

{

bool Flag;

double point\_for\_LeftSide = (x1 - x0) \* (y2 - y1) - (x2 - x1) \* (y1 - y0);

double point\_for\_UpperSide = (x2 - x0) \* (y3 - y2) - (x3 - x2) \* (y2 - y0);

double point\_for\_RightSide = (x3 - x0) \* (y4 - y3) - (x4 - x3) \* (y3 - y0);

double point\_for\_UnderSide = (x4 - x0) \* (y1 - y4) - (x1 - x4) \* (y4 - y0);

if (point\_for\_LeftSide <= 0 && point\_for\_UpperSide <= 0 && point\_for\_RightSide <= 0 && point\_for\_UnderSide <= 0)

{

Flag = true;

}

else

{

Flag = false;

}

return Flag;

}

}

}

**Класс Program:**

using System;

using System.Diagnostics;

namespace InConsoleApplication

{

class Program

{

static void Main(string[] args)

{

double x0, x1, x2, x3, x4, y0, y1, y2, y3, y4;

x0 = 0;

y0 = 0;

Console.WriteLine(" B(x2,y2) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ C(x3,y3)");

Console.WriteLine(" / /");

Console.WriteLine(" / /");

Console.WriteLine(" / /");

Console.WriteLine(" / /");

Console.WriteLine("A(x1,y1)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ D(x4,y4)\n");

Console.WriteLine("Enter coordinates (x1,y1)");

x1 = Convert.ToDouble(Console.ReadLine());

y1 = Convert.ToDouble(Console.ReadLine());

do

{

Console.WriteLine("Enter coordinates (x2,y2)");

x2 = Convert.ToDouble(Console.ReadLine());

y2 = Convert.ToDouble(Console.ReadLine());

} while (y1 >= y2);

Console.WriteLine("Enter coordinates (x3,y3)");

x3 = Convert.ToDouble(Console.ReadLine());

y3 = Convert.ToDouble(Console.ReadLine());

do

{

Console.WriteLine("Enter coordinates (x4,y4)");

x4 = Convert.ToDouble(Console.ReadLine());

y4 = Convert.ToDouble(Console.ReadLine());

} while (y4>=y3);

Console.WriteLine("Enter coordinates of the point (x0,y0)");

x0 = Convert.ToDouble(Console.ReadLine());

y0 = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("\nIs it a parallelogram?");

if (Figure.IsParallelogram(x1, x2, x3, x4, y1, y2, y3, y4) == true)

{

Console.WriteLine("Yes\n");

}

else

{

Console.WriteLine("No\n");

Console.ReadKey();

Process.GetCurrentProcess().Kill();

}

Figure Parallelogram = new Figure( x1, x2, x3, x4, y1, y2, y3, y4);

Console.WriteLine($"Side AB: { Parallelogram.LeftSide}");

Console.WriteLine($"Side BC: { Parallelogram.UpperSide}");

Console.WriteLine($"Side CD: { Parallelogram.RightSide}");

Console.WriteLine($"Side DA: { Parallelogram.UnderSide}");

Console.WriteLine($"Perimeter: { Parallelogram.Perimeter} = 2\*({ Parallelogram.LeftSide} + { Parallelogram.UpperSide})");

Console.WriteLine($"Square: { Parallelogram.Square} = { Parallelogram.Height} \*{ Parallelogram.UnderSide} ");

if (Figure.IsBelong(x0, y0, x1, x2, x3, x4, y1, y2, y3, y4) == true)

{

Console.WriteLine("Point belongs to the parallelogram\n");

}

else

Console.WriteLine("Point dosen't belong to the parallelogram\n");

Console.WriteLine("Bye:)");

Console.ReadKey();

}

}

}