**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»

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

Выполнил: студент гр. ИТП-21

Король В.Н

Принял: преподаватель-стажер

Гуменников Е.Д

Гомель 2022

**Цель работы:** изучить основы языка программирования *C*#. Научится создавать классы их основные возможности. Изучить такие понятия как поля класса, конструктор класса и методы класса.

**Задание**

1. Необходимо разработать класс, представляющий собой плоскую фигуру,

согласно варианта (таблица 1).

2. При именовании класса, полей и методов руководствоваться соглашением о

наименовании кода (https://docs.microsoft.com/en-

us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions)

3. Фигуру следует задавать координатами вершин.

4. Реализовать метод проверки возможности существования данной фигуры

5. Реализовать методы вычисление длин сторон, площади и периметра

6. Реализовать методы, проверяющие принадлежность точки, заданной своими

координатами на плоскости, фигуре, её границе.

7. Класс должен быть размещён в библиотеке классов.

8. Весь код должен быть снабжён элементами документирования

9. Сгенерировать XML-файл с документацией по проекту

10. Создать консольное приложение с интерфейсом пользователя

11. Подключить созданную библиотеку в консольное приложение

12. Верифицировать разработанное приложение

13. Сгенерировать исполняемый файл для последующего переноса на другие компьютеры

**Вариант 14**

Криволинейная трапеция на заданном интервале, образуемая

гиперболой и осью OX.

**Ход Работы**

Для создания класса в *C*# необходимо ввести ключевое слово *class* после этого указывается название класса, в моей программе класс имеет название *Figure.* После создания класса необходимо указать его поля. Для моей задачи мне нужны только два поля это *leftBorder* и *rightBorder*. Так же класс может содержать конструктор который позволяет указывать поля класса при его объявлении. Класс *Figure* указан в листинге программы.

Для более удобной работы с методами класса был создан публичный метод *Work* который позволяет вызывать все другие приватные методы класса. Пример работы метода *Work* указан на рисунке 1.

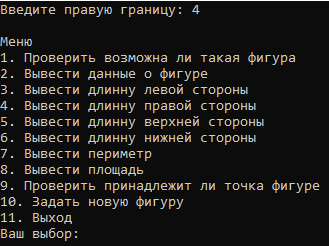


Рисунок 1 - Пример работы метода *Work*

Для проверки возможна ли такая фигура был создан метод *IsPosible* который возвращает *bool* значение в зависимости от того существует ли фигура на заданном интервале или нет. Пример работы метода *IsPosible* указан на рисунке 2.



Рисунок 2 - Пример работы метода *IsPosible*

Для вывода информации о фигуре был создан метод *ShowInfo* который выводит левую границу, правую границу и функцию гиперболы. Пример работы метода *ShowInfo* указан на рисунке 3.



Рисунок 3 - Пример работы метода ShowInfo

Для получения длин сторон были созданы методы GetLeftSide, GetRightSide, GetDownSide и GetUpSide. Пример работы этих методов указан на рисунке 4.









Рисунок 4 - Пример работы методов GetLeftSide, GetRightSide, GetDownSide и GetUpSide

Для нахождения периметра был создан метод *GetPerimeter* которая возвращает сумму значение методов GetLeftSide, GetRightSide, GetDownSide и GetUpSide. Пример работы метода *GetPerimeter* указан на рисунке 5.



Рисунок 5 - Пример работы метода GetPerimeter

Для нахождения площади фигуры был создан метод *GetArea* который возвращает значение определенного интеграла на заданном промежутке. Пример работы метода *GetArea* указан на рисунке 6.



Рисунок 6 - Пример работы метода GetArea

Для определения принадлежит ли точка фигуре или нет был создан метод *ThisPointIsExist* который возвращает *bool* значение в зависимости от того принадлежит ли точка или нет. Пример работы метода *ThisPointIsExist* указана на рисунке 7.





Рисунок 7 - Пример работы метода *ThisPointIsExist*

**Вывод:** в ходе выполнения лабораторной работы были изучены основы языка программирования *C*#. Научился создавать классы. Изучил такие понятия как поля класса, конструктор класса и методы класса. Изучил основные методы класса *Math.* Написал программу которая при помощи класса позволяет получить и изменять данные о заданной фигуре.

**Листинг программы**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab1

{

internal class Figure

{

private float \_leftBorder, \_rightBorder;

public Figure(float leftBorder, float rightBorder)

{

\_leftBorder = leftBorder;

\_rightBorder = rightBorder;

}

public Figure()

{

\_leftBorder = 1.0f;

\_rightBorder = 2.0f;

}

private void ShowInfo()

{

Console.WriteLine($"Левая граница - {\_leftBorder}. Правая граница - {\_rightBorder}. Формула гиперболы - y = 1/x.");

}

private bool IsPosible()

{

return \_rightBorder > \_leftBorder && (\_leftBorder < 0 && \_rightBorder < 0 || \_leftBorder > 0 && \_rightBorder > 0);

}

public float GetDownSide()

{

return (float)Math.Abs(\_leftBorder - \_rightBorder);

}

private float GetUpSide()

{

float upLenght = 0f;

for (float i = \_leftBorder; i < \_rightBorder; i += 0.0001f)

{

upLenght += (float)(Math.Sqrt(1 + 1/(Math.Pow(i, 4))) \* 0.0001f);

}

return upLenght;

}

private float GetLeftSide()

{

return (float)Math.Sqrt(Math.Pow((1 / \_leftBorder), 2));

}

private float GetRightSide()

{

return (float)Math.Sqrt(Math.Pow((1/\_rightBorder), 2));

}

private float GetArea()

{

return (float)(Math.Log(\_rightBorder) - Math.Log(\_leftBorder));

}

private float GetPerimeter()

{

return GetDownSide() + GetUpSide() + GetRightSide() + GetLeftSide();

}

private bool ThisPointPosible(float x, float y)

{

if(\_leftBorder < 0 && \_rightBorder < 0)

return x >= \_leftBorder && x <= \_rightBorder && y >= 1/x && y <= 0;

else if(\_leftBorder > 0 && \_rightBorder > 0)

return x >= \_leftBorder && x <= \_rightBorder && y <= 1 / x && y >= 0;

return false;

}

public void Work(Figure figure)

{

int choose;

bool menuIsWork = true;

while (menuIsWork)

{

Console.WriteLine("Меню");

Console.WriteLine("1. Проверить возможна ли такая фигура");

Console.WriteLine("2. Вывести данные о фигуре");

Console.WriteLine("3. Вывести длинну левой стороны");

Console.WriteLine("4. Вывести длинну правой стороны");

Console.WriteLine("5. Вывести длинну верхней стороны");

Console.WriteLine("6. Вывести длинну нижней стороны");

Console.WriteLine("7. Вывести периметр");

Console.WriteLine("8. Вывести площадь");

Console.WriteLine("9. Проверить принадлежит ли точка фигуре");

Console.WriteLine("10. Задать новую фигуру");

Console.WriteLine("11. Выход");

Console.Write("Ваш выбор: ");

choose = Convert.ToInt32(Console.ReadLine());

Console.Clear();

switch (choose)

{

case 1:

if (figure.IsPosible())

Console.WriteLine("Такая фигура существует");

else

Console.WriteLine("Такая фигура не существует");

break;

case 2:

figure.ShowInfo();

break;

case 3:

Console.WriteLine($"Длинна левой стороны - {figure.GetLeftSide()}");

break;

case 4:

Console.WriteLine($"Длинна правой стороны - {figure.GetRightSide()}");

break;

case 5:

Console.WriteLine($"Длинна верхней стороны - {figure.GetUpSide()}");

break;

case 6:

Console.WriteLine($"Длинна нижней стороны - {figure.GetDownSide()}");

break;

case 7:

Console.WriteLine($"Периметр фигуры - {figure.GetPerimeter()}");

break;

case 8:

Console.WriteLine($"Площадь фигуры - {figure.GetArea()}");

break;

case 9:

Console.Write("Введите координату точки по X: ");

float x = Convert.ToSingle(Console.ReadLine());

Console.Write("Введите координату точки по Y: ");

float y = Convert.ToSingle(Console.ReadLine());

if (figure.ThisPointPosible(x, y))

Console.WriteLine("Данная точка лежит на искомой фигуре");

else

Console.WriteLine("Данная точка не лежит на искомой фигуре");

break;

case 10:

Console.Write("Введите левую границу: ");

float leftBorder = Convert.ToSingle(Console.ReadLine());

Console.Write("Введите правую границу: ");

float RightBorder = Convert.ToSingle(Console.ReadLine());

figure = new Figure(leftBorder, RightBorder);

break;

case 11:

menuIsWork = false;

break;

}

Console.ReadKey();

}

}

}

}