УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

**“ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО”**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

по дисциплине «Объектно-ориентированное программирование»

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

Выполнил:

студент гр. ИТП-21

Клевцов А.О.

Принял: ассистент

Гуменников Е.Д.

Гомель 2022

**Цель работы:** Освоить основы синтаксиса объектно-ориентированного языка программирования на языке *c#*.

**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
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

Таблица 1

|  |  |
| --- | --- |
| **Вариант** | **Условие Задачи** |
| 11 | Криволинейная трапеция на заданном интервале, образуемая параболой и осью ОХ |

**Ход работы**

В ходе работы был разработан класс *Trapeze,* который представляет плоскую фигуру, заданную осью ОХ и параболой *y=a-b\*x^2*, где параметры a и b можно задавать при создании объекта класса. Класс помещён в отдельный файл – библиотеку классов, и подключён к основному проекту. В трапеции хранятся такие переменные как *a*, *b*, *x01*, *x02*, где иксы — это точки, в которых у=0. Так же класс содержит методы для: выяснения возможности существования данной трапеции, вычисления: площади, длины, периметра – и проверки точки на принадлежность к трапеции. Код программы будет представлен в приложении А.

Пример функций, представленных в программе

А) Трапеция с ветвями вверх и вершиной ниже оси ОХ

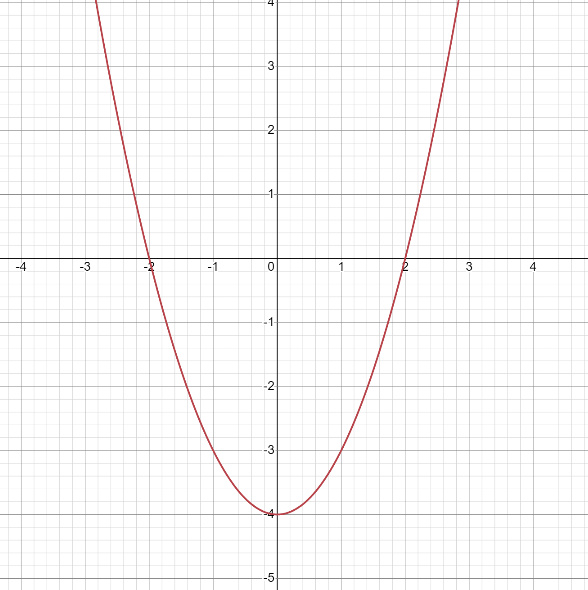


Рисунок 1 ­­– Трапеция y=-4+x^2

Б) Трапеция с ветвями вниз и вершиной выше оси ОХ

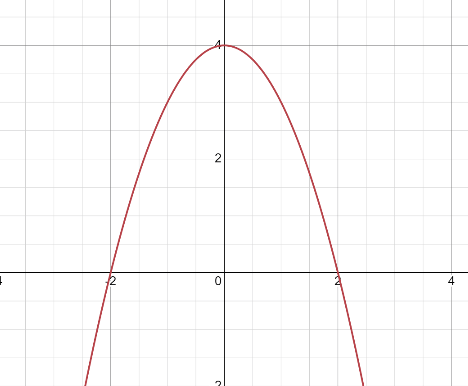


Рисунок 2 ­­– Трапеция y=4-x^2

**Вывод:** в ходе лабораторной работы были освоены основы синтаксиса объектно-ориентированного языка программирования на языке *c#*.

**Приложение А**

Листинг основной программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LABA1

{

internal class Program

{

static void Main(string[] args)

{

// Задание: Криволинейная трапеция на заданном интервале, образуемая парабалой и осью ОХ

Console.WriteLine("Криволинейная трапеция образуемая парабалой и осью ОХ \n");

Trapeze t1 = new Trapeze(); // Трапеция, у которой ветви вниз, вершины выше Ох

Trapeze t2 = new Trapeze(-4, -1); // Трапеция, у которой ветви вверх, вершины ниже Ох

Trapeze t3 = new Trapeze(4, -1); // Объет у кторого парметры не образуют тропецию

Console.WriteLine("Функция трапеции заданной по стандарту:");

Console.WriteLine($"\ty = {t1.a} - ({t1.b})\*x^2");

Console.WriteLine("\tВозможна ли тропеция?:" + t1.IsPossible());

Console.WriteLine("\tПлощадь:" + t1.Area());

Console.WriteLine("\tДлина стороны:" + t1.Long() + "\n");

Console.WriteLine("Функция трапеции заданной с помощью передоваемых параметров a = -4 & b = -1:");

Console.WriteLine($"\ty = {t2.a} - ({t2.b})\*x^2");

Console.WriteLine("\tВозможна ли тропеция?:" + t2.IsPossible());

Console.WriteLine("\tПериметр:" + t2.Perimeter());

Console.WriteLine("\tПринадлежит ли точка (-1, -1):" + t2.IsPointExists(-1, -1));

Console.WriteLine("\tПринадлежит ли точка (3, -1):" + t2.IsPointExists(3, -1) + "\n");

Console.WriteLine("Функция трапеции заданной по стандарту:");

Console.WriteLine($"\ty = {t3.a} - ({t3.b})\*x^2");

Console.WriteLine("\tВозможна ли тропеция?:" + t3.IsPossible() + "\n");

}

}

}

Листинг библиотеки классов

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LABA1

{

internal class Trapeze

{

// трапеция из оси Ox и параболы y = a - b\*x^2

public double a, b, x01 ,x02;

public Trapeze(double a, double b) // задание объкта - трапеция

{

this.a = a;

this.b = b;

this.x01 = -1 \* Math.Sqrt(a / b); //Вычисление 1 точеки пересечения с Ox

this.x02 = Math.Sqrt(a / b); //Вычисление 2 точеки пересечения с Ox

}

public Trapeze() // параметры трапеции по умолчанию

{

a = 4;

b = 1;

x01 = -1 \* Math.Sqrt(a / b);

x02 = Math.Sqrt(a / b);

}

public bool IsPossible() // проверка на правильность существования фигуры

{

return a > 0 && b > 0 || a < 0 && b < 0; // 1) либо парабола смотрит вниз и находится нод осью Ox

} // 2) либо наоборот

public double Area() // вычисление площади с помощью интегрального метода

{

double area = 0; // значение площади

if (a > 0 && b > 0) // для 1) типа тропеции

{

for (double pos = x01; pos < x02 ; pos+= 0.001 ) // идём с шагом по оси Х и вычилсяя значения

{

area += a - b \* pos \* pos; // суммируем их в площадь и в конце

}

area = area\*0.001; // и умножаем на шаг получая площадь с точность шага = 0.001

}

else if (a < 0 && b < 0) // аналогично для 2) типа

{

for (double pos = x01; pos < x02; pos += 0.001)

{

area += a - b \* pos \* pos;

}

area = -area \* 0.001 ; // умножение на минус 1 т.к. все рассматриваемые у отрицательны на отрезке

}

return area;

}

public double Long() // вычисление стороны

{

return Math.Abs(x01 - x02);

}

public double Perimeter() // вычисление периметра

{

double length = 0, y0; // длина и некая точка у0 при точке pos

double temp = a - b \* x01 \* x01; // первая точка у0

for (double pos = x01 + 0.001; pos < 0; pos += 0.001) // идём по параболе слева на право до половины с шагом = 0.001

{ // чтобы посчитать длину дуги делим её на отрезки

y0 = a - b \* pos \* pos; //считаем моментальную точку у0

length += Math.Sqrt( (y0 - temp)\* (y0 - temp) + 0.001\*0.001 ); // используя теорему пифагора добавляем к длинне маленькйи отрезок дуги

temp = y0;

}

return length\*2 + Long(); // умножаем длину половины параболы на 2 и прибовляем длину единственной строны

}

public bool IsPointExists(double x, double y) // проверка точки с координатами х и у на принадлежность трапеции

{

bool resalt = false;

if (x>= x01 && x<= x02 ) // икс находится между крайними точками трапеции?

{

if (a > 0 && b > 0) // 1 тип - ветви вниз, вершины выше Ох

{

resalt = y >= 0 && y < (a - b \* x \* x); // игрик должен быть выше Ох и меньше у функции в х

}

if (a < 0 && b < 0) // 2 тип - ветви вверх, вершина ниже Ох

{

resalt = y<=0 && y > (a - b \* x \* x); // игрик должен быть ниже Ох и больше у функции в х

}

}

return resalt;

}

}

}

Начало формы

Конец формы