Пермский филиал федерального государственного автономного образовательного учреждения высшего образования   
«Национальный исследовательский университет   
«Высшая школа экономики»

*Факультет экономики, менеджмента и бизнес-информатики*

Никитин Андрей Владимирович

**Отчет по лабораторной работе №9**

студента образовательной программы «Программная инженерия»

по направлению подготовки *09.03.04 Программная инженерия*

Руководитель

доцент кафедры

информационных

технологий в бизнесе

к.т.н. О. Л. Викентьева

Пермь, 2020 год

1. **Постановка задачи**

Общая задача:

1. Реализовать (в отдельном файле) определение нового класса (закрытые атрибуты, свойства, конструкторы,  инициализация и вывод атрибутов).
2. Для демонстрации работы с объектами написать главную функцию, в которой создаются объекты класса и выводится информация, которая содержится в атрибутах.
3. Написать функцию, реализующую указанное в варианте действие. Рассмотреть два варианта:

−        статическую функцию;

−        метод класса.

1. В основной функции продемонстрировать работу функции.
2. Используя статическую компоненту класса подсчитать количество созданных в программе объектов.
3. Добавить к реализованному классу указанные в варианте перегруженные операции.
4. Написать демонстрационную программу, в которой создаются объекты пользовательских классов и выполняются указанные операции.
5. Реализовать класс (в отдельном файле), полем которого является одномерный массив из элементов заданного в варианте типа. В классе реализовать

−        конструктор без параметров,

−        конструктор с параметрами, заполняющий элементы случайными значениями,

−        конструктор с параметрами, позволяющий заполнить массив элементами, заданными пользователем с клавиатуры,

−        индексатор (для доступа к элементам массива),

−        метод для просмотра элементов массива.

1. Написать демонстрационную программу, позволяющую создать массив разными способами и распечатать элементы массива. Подсчитать количество созданных объектов.

Вариант задачи:

|  |  |  |
| --- | --- | --- |
| 4 | double a, double b, double c (длина катетов) | Вычислить площадь треугольника. Результат должен быть типа double. |
| Triangle | Унарные операции:  ++ увеличивает стороны треугольника на 1;  -- уменьшает стороны треугольника на 1;  Операции приведения типа:  double (явная) – результатом является площадь треугольника, если треугольник существует и отрицательное число в противном случае;  bool (неявная) – результатом является true, если треугольник с такими длинами сторон существует и false в противном случае.  Бинарные операции:  <= Triangle t1, Triangle t2 – сравнивает площади треугольников;  >= Triangle t1, Triangle t2 - сравнивает площади треугольников. |
| Triangle | Номер элемента с минимальной площадью. |

1. **Диаграмма классов**

Изображение выглядит как снимок экрана

Автоматически созданное описание

1. **Листинг программы**

## Program.Cs

sing System;

namespace Lab\_9

{

class Program

{

public Triangle Triangle

{

get => default;

set

{

}

}

public TriangleArray TriangleArray

{

get => default;

set

{

}

}

static void Main(string[] args)

{

Triangle triangle = new Triangle();

triangle = new Triangle(3, 4, 5);

triangle.Show();

triangle++;

triangle.Show();

Triangle triangle1 = new Triangle(5, 6, 7);

triangle1.Show();

triangle1--;

triangle.Show();

Console.WriteLine($"Треугольник 1 >= Треугольник 2 : {triangle >= triangle1}");

Console.WriteLine($"Треугольник 1 <= Треугольник 2 : {triangle++ <= triangle1--}");

TriangleArray triangleArray = new TriangleArray();

triangleArray = new TriangleArray(5);

triangleArray.Show();

triangleArray = new TriangleArray(6, new object());

triangleArray.Show();

Console.WriteLine($"5 треугольник");

triangleArray[5].Show();

triangleArray = new TriangleArray(10);

triangleArray.Show();

int index = triangleArray.MinimumSquareIndex();

Console.WriteLine($"Индекс минимального элемента в массиве: {index}");

}

}

}

## Triangle.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab\_9

{

// Класс треугольник

public class Triangle

{

protected double a; // Первая сторона

protected double b; // Вторая сторона

protected double c; // Третья сторона

public double A

{

get { return a; }

set

{

try

{

if (value < 0)

throw new ArgumentException("Ошибка, введите положительную длину 1 cтороны треугольника!");

else

a = value;

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

public double B

{

get { return b; }

set

{

try

{

if (value < 0)

throw new ArgumentException("Ошибка, введите положительную длину 2 cтороны треугольника!");

else

b = value;

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

public double C

{

get { return c; }

set

{

try

{

if (value < 0)

throw new ArgumentException("Ошибка, введите положительную длину 3 cтороны треугольника!");

else

c = value;

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

// конструктор без параметров

public Triangle()

{

a = 0; b = 0; c = 0;

}

// конструктор с пераметрами

public Triangle(double \_a, double \_b, double \_c)

{

// вызов внутреннего метода класса из конструктора

if (Triangle.Existence(\_a, \_b, \_c))

{

a = \_a;

b = \_b;

c = \_c;

}

else

{

new Triangle();

}

}

// метод класса, определяющий можно ли по данным a, b, c построить треугольник

public bool Existence()

{

if (a < 0 && b < 0 && c < 0) return false;

if (a >= (b + c)) return false;

if (b >= (a + c)) return false;

if (c >= (a + b)) return false;

return true;

}

// статическая функция класса, определяющая можно ли по данным a, b, c построить треугольник

public static bool Existence(double \_a, double \_b, double \_c)

{

if (\_a < 0 && \_b < 0 && \_c < 0) return false;

if (\_a >= (\_b + \_c)) return false;

if (\_b >= (\_a + \_c)) return false;

if (\_c >= (\_a + \_b)) return false;

return true;

}

// метод класса, определяющий периметр треугольника

public double Perimetr()

{

// вызов внутреннего метода из метода Perimetr(), проверка

if (Existence())

{

return a + b + c;

}

else

{

return 0.0;

}

}

// статическая функция класса, определяющая периметр треугольника

public static double Perimetr(Triangle triangle)

{

// вызов внутреннего метода из метода Perimetr(), проверка

if (triangle.Existence())

{

return triangle.a + triangle.b + triangle.c;

}

else

{

return 0.0;

}

}

// метод класса, определяющий площадь треугольника

public double Square()

{

// вызов внутреннего метода из метода Square(), проверка

if (Existence())

{

double p = (a + b + c) / 2;

return Math.Sqrt(p \* (p - a) \* (p - b) \* (p - c));

}

else

return 0.0;

}

// статическая функция класса, определяющ площадь треугольника

public static double Square(Triangle triangle)

{

if (triangle.Existence())

{

double p = (triangle.a + triangle.b + triangle.c) / 2;

return Math.Sqrt(p \* (p - triangle.a) \* (p - triangle.b) \* (p - triangle.c));

}

else

return 0.0;

}

// метод класса, выводящий информацию об объекте

public void Show()

{

Console.WriteLine($"Треуголник со сторонами {a}, {b}, {c} c площадью {Square()} квадратных едениц");

}

// метод класса, выводящий создающий информацию об объекте

public void Create(Triangle triangle)

{

Console.Write("Enter first kathetus of {1 triangle:");

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

Console.Write("Enter second kathetus of triangle:");

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

Console.Write("Enter third kathetus of triangle:");

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

triangle = new Triangle(A, B, C);

}

// унарная операция, увеличивающеая координаты стороны треугольника на 1

public static Triangle operator ++ (Triangle tr1)

{

tr1.a += 1.0; tr1.b += 1.0; tr1.c += 1.0;

return tr1;

}

// унарная операция, увеличивающая координаты стороны треугольника на 1

public static Triangle operator -- (Triangle tr1)

{

tr1.A -= 1.0; tr1.A -= 1.0; tr1.A -= 1.0;

if (tr1.A > 0 && tr1.A > 0 && tr1.A > 0 && tr1.Existence())

return tr1;

else

return new Triangle();

}

// бинарная операция, сравнивающая площадь двух треугольников

public static bool operator <= (Triangle tr1, Triangle tr2)

{

return tr1.Square() <= tr2.Square();

}

// бинарная операция, сравнивающая площадь двух треугольников

public static bool operator >= (Triangle tr1, Triangle tr2)

{

return tr1.Square() >= tr2.Square();

}

// операция явного приведения типа double: "Если треугольник существует, то присвоить площадь треугольника, иначе -1"

public static explicit operator double (Triangle tr1)

{

if (tr1.Existence())

{

return tr1.Square();

}

else

return -1;

}

// операция неявного приведения типа bool: "Если треугольник существует с такими сторонами, то true, иначе false"

public static implicit operator bool (Triangle tr1)

{

if (tr1.Existence())

{

return true;

}

else

{

return false;

}

}

}

}

## TriangleArray.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab\_9

{

public class TriangleArray

{

public const int MaxSize = 100;

public const int MaxValue = 100;

public Triangle[] arr;

public int size;

// конструктор без параметров

public TriangleArray ()

{

arr = null;

size = 0;

}

// конструктор с пераметрами для случайного заполнения

public TriangleArray (int \_size)

{

Random random = new Random();

Triangle[] arrNew = new Triangle[\_size];

for(int i=0; i < \_size; i++)

{

double a, b, c;

do

{

a = random.Next(1, MaxValue);

b = random.Next(1, MaxValue);

c = random.Next(1, MaxValue);

} while (Triangle.Existence(a, b, c));

arrNew[i] = new Triangle(a, b, c);

}

arr = arrNew;

size = \_size;

}

// конструктор с пераметрами для ручного заполнения

public TriangleArray(int \_size, object keyboard)

{

Triangle[] arrNew = new Triangle[\_size];

for(int i = 0; i < \_size; i++)

{

double a, b, c;

do

{

a = InputNumber();

b = InputNumber();

c = InputNumber();

} while (Triangle.Existence(a, b, c));

arrNew[i] = new Triangle(a, b, c);

}

arr = arrNew;

size = \_size;

}

// индексатор (для доступа к элеменам массива)

public Triangle this[int index]

{

get

{

return arr[index];

}

set

{

arr[index] = value;

}

}

public Triangle Triangle

{

get => default;

set

{

}

}

// функция класса, находящая номер элемента с минимальной площадью

public int MinimumSquareIndex()

{

Triangle minimumTriangle = arr[0];

int minimumIndex = 0;

for(int i = 1; i < size; i++)

{

if(arr[i].Square() < minimumTriangle.Square())

{

minimumIndex = i;

minimumTriangle = arr[i];

}

}

return minimumIndex;

}

// Функция класса, выводящая информацию о массиве

public void Show()

{

Console.WriteLine("Массив треугольников");

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

{

Console.Write($"{i + 1} ");

arr[i].Show();

}

}

// приватная функция для ручного ввода числа

protected double InputNumber()

{

double number;

while (!double.TryParse(Console.ReadLine(), out number) || number < 0)

{

Console.WriteLine("Ошибка. Введите снова длину треугольника.");

}

return number;

}

}

}

1. **Тестирование**

Изображение выглядит как снимок экрана

Автоматически созданное описание