Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

О Т Ч Ё Т

Объектно-ориентированное программирование

Семестр 3

Лабораторная работа 9

Выполнил

студент группы РИС-22-2б

Баяндин К.С.

Проверила

доцент кафедры ИТАС

Викентьева О.Л.

Пермь, 2023

1. **Постановка задачи**

Часть 1.

1. Реализовать (в отдельном файле) определение нового класса/типа данных (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:

1) статическую функцию;

2) метод класса.

В основной функции продемонстрировать работу функции.

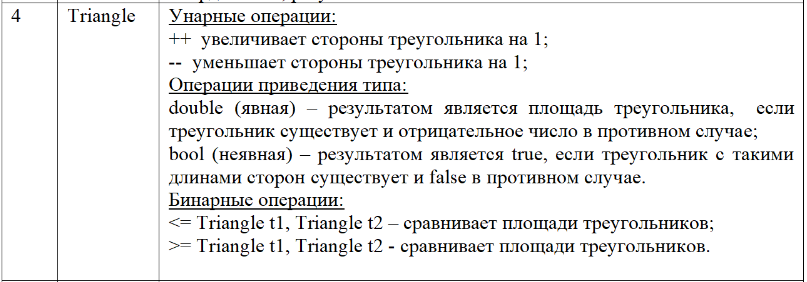
4. Используя статическую компоненту класса подсчитать количество созданных в программе объектов.



Часть 2.

1. Добавить к реализованному классу указанные в варианте перегруженные операции.

2. Написать демонстрационную программу, в которой создаются объекты пользовательских классов и выполняются указанные операции.



Часть 3

1. Реализовать класс-коллекцию (в отдельном файле), полем которого является одномерный массив (не использовать стандартные коллекции C#!) из элементов заданного в варианте типа. Например, для класса Fraction нужно создать класс FractionArray следующим образом:

class FractionArray

{

Fraction[] arr;//одномерный массив элементов типа Fraction

. . . .

}

В классе реализовать

· конструктор без параметров,

· конструктор с параметрами, заполняющий элементы случайными значениями,

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

· метод для просмотра элементов массива.

2. Реализовать индексатор для доступа к элементам коллекции. Предусмотреть проверку при выходе индекса за пределы массива.

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

4. Написать функцию в классе Program для выполнения указанного в варианте задания (использовать индексатор и, если необходимо, перегрузить нужные для выполнения задачи операции).



1. **Диаграммы классов**

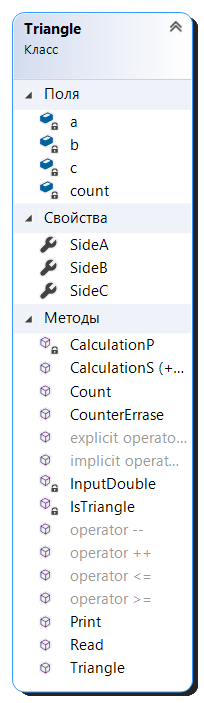


Рисунок 1 – UML-диаграмма класса Triangle

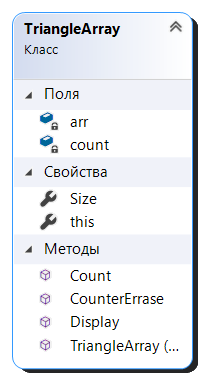


Рисунок 2 – UML-диаграмма класса TriangleArray

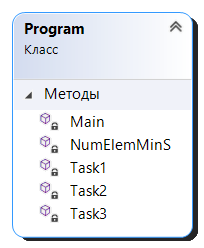


Рисунок 3 – UML-диаграмма класса Program

1. **Код программы**

**Triangle.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab\_OOP\_9

{

class Triangle

{

private double a;

private double b;

private double c;

private static int count = 0;

//свойства

public double SideA

{

set { a = value; }

get { return a; }

}

public double SideB

{

set { b = value; }

get { return b; }

}

public double SideC

{

set { c = value; }

get { return c; }

}

//конструктор

public Triangle(double a = 0, double b = 0, double c = 0)

{

this.a = a;

this.b = b;

this.c = c;

count++;

}

// число созданных объектов

public static int Count() => count;

public static void CounterErrase() => count = 0;

//ввод double значений

static double InputDouble(string message)

{

double number;

Console.Write(message);

if (!double.TryParse(Console.ReadLine(), out number))

{

do

{

Console.WriteLine("Ошибка!");

Console.WriteLine("Введите число типа double");

Console.Write(message);

} while (!double.TryParse(Console.ReadLine(), out number));

}

return number;

}

//существует треугольник или нет

private static bool IsTriangle(Triangle trian)

{

if ((trian.a > trian.b + trian.c) || (trian.b > trian.a + trian.c) || (trian.c > trian.a + trian.b))

return false;

else

return true;

}

public void Read()

{

a = InputDouble("Введите 1 сторону: ");

b = InputDouble("Введите 2 сторону: ");

c = InputDouble("Введите 3 сторону: ");

}

public void Print()

{

Console.WriteLine($"a: {a}\nb: {b}\nc: {c}\n");

}

//возвращает полупериметр

private static double CalculationP(Triangle side)

{

return (side.a + side.b + side.c) / 2;

}

//вычисляет площадь

public double CalculationS()

{

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

}

public static double CalculationS(Triangle side)

{

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

}

//перегрузка операций

public static Triangle operator ++(Triangle trian)

{

return new Triangle(++trian.a, ++trian.b, ++trian.c);

}

public static Triangle operator --(Triangle trian)

{

return new Triangle(--trian.a, --trian.b, --trian.c);

}

//приведение к double (явная)

public static explicit operator double(Triangle trian)

{

if (IsTriangle(trian))

return CalculationS(trian);

else

return (-1.0);

}

//приведение к bool (неявное)

public static implicit operator bool(Triangle trian)

{

return IsTriangle(trian);

}

//бинарные операции

public static bool operator <=(Triangle t1, Triangle t2)

{

return CalculationS(t1) <= CalculationS(t2);

}

public static bool operator >=(Triangle t1, Triangle t2)

{

return CalculationS(t1) >= CalculationS(t2);

}

}

}

**TriangleArray.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab\_OOP\_9

{

class TriangleArray

{

Triangle[] arr;

static private int count = 0;

public int Size => arr.Length;

public static int Count() => count;

public static void CounterErrase() => count = 0;

//конструктор без параметров

public TriangleArray() : this(0) { }

//Конструктор с параметрами с рандомным заполнением

public TriangleArray(int size)

{

if (size < 0)

{

Console.WriteLine("Размер не может быть меньше 0");

size = 0;

}

arr = new Triangle[size];

var random = new Random();

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

{

double a = random.Next(0, 100);

double b = random.Next(0, 100);

double c = random.Next(0, 100);

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

}

count++;

}

//конструктор с параметрами с вводом с клавиатуры

public TriangleArray(bool console)

{

Console.WriteLine("Введите новый размер массива: ");

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

arr = new Triangle[newSize];

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

{

arr[i] = new Triangle();

arr[i].Read();

}

count++;

}

//заранее известные элементы массива

public TriangleArray(params Triangle[] Triangles)

{

arr = new Triangle[Triangles.Length];

for (int i = 0; i < Triangles.Length; i++)

{

arr[i] = Triangles[i];

}

}

//вывод элементов массива

public void Display()

{

if (arr.Length == 0)

{

Console.WriteLine("Массив пуст.");

}

foreach (Triangle Triangle in arr)

{

Triangle.Print();

}

}

//индексатор

public Triangle this[int index]

{

get

{

if (index >= 0 && index < arr.Length)

{

return arr[index];

}

else

{

throw new IndexOutOfRangeException("Выход за границы.");

}

}

set

{

if (index >= 0 && index < arr.Length)

{

arr[index] = value;

}

else

{

throw new IndexOutOfRangeException("Выход за границы.");

}

}

}

}

}

**Program.cs**

using Lab\_OOP\_9;

using System;

namespace Lab\_OOP\_9

{

class Program

{

static void Task1()

{

Console.WriteLine("Задание 1");

Triangle triangle = new Triangle();

triangle.Read();

triangle.Print();

Console.WriteLine("Метод класса. Площадь = " + triangle.CalculationS());

Console.WriteLine("Статическая функция. Площадь = " + Triangle.CalculationS(triangle));

Console.WriteLine("Кол-во созданных объектов: " + Triangle.Count());

}

static void Task2()

{

Console.WriteLine("Задание 2");

Triangle triangle1 = new Triangle();

triangle1.Read();

triangle1.Print();

Triangle triangle2 = new Triangle();

triangle2.Read();

triangle2.Print();

Console.WriteLine("trianle1++: ");

triangle1++.Print();

Console.WriteLine("trianle2--: ");

triangle2--.Print();

double square = (double)triangle1;

bool isTriangle = triangle2;

Console.WriteLine("double(явное) triangle1: " + square);

Console.WriteLine("bool(неявное) triangle2: " + isTriangle);

Console.WriteLine("triangle1 <= triangle2: " + (triangle1 <= triangle2));

Console.WriteLine("triangle1 >= triangle2: " + (triangle1 >= triangle2));

}

static void Task3()

{

Console.WriteLine("Задание 3");

var array1 = new TriangleArray();

Console.WriteLine("1 массив:");

array1.Display();

Console.WriteLine();

var array2 = new TriangleArray(3);

Console.WriteLine("2 массив:");

array2.Display();

Console.WriteLine();

var array3 = new TriangleArray(

new Triangle(3, 4, 5),

new Triangle(12, 13, 5),

new Triangle(6, 8, 10)

);

Console.WriteLine("3 массив:");

array3.Display();

Console.WriteLine();

Console.WriteLine("Кол-во созданных объектов: " + TriangleArray.Count());

Console.WriteLine("Элемент под 3 номером: ");

array3[2].Print();

Console.WriteLine();

array3[2] = new Triangle(15, 17, 8);

Console.WriteLine("Изменение на 15, 17 и 18 под 3 номером:");

array3.Display();

Console.WriteLine("Номер треугольника с самой маленькой площадью: " + NumElemMinS(array3));

}

static int NumElemMinS(TriangleArray arr)

{

int elemMinS = 0;

if (arr.Size == 0)

{

Console.WriteLine("Пусто!");

return 0;

}

double MinS = Triangle.CalculationS(arr[0]);

elemMinS = 1;

for(int i = 1; i < arr.Size; i++)

{

if (MinS > Triangle.CalculationS(arr[i]))

{

MinS = Triangle.CalculationS(arr[i]);

elemMinS = i + 1;

}

}

return elemMinS;

}

static void Main(string[] args)

{

Task1();

Console.WriteLine();

Task2();

Console.WriteLine();

Task3();

}

}

}

1. **Код юнит-тестов**

**TriangleTests.cs**

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace Lab\_OOP\_9.Tests

{

[TestClass]

public class TriangleTests

{

[TestMethod]

public void CalculationS\_ValidTriangle\_ReturnsCorrectArea()

{

// Arrange

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

double expectedArea = 6;

// Act

double actualArea = triangle.CalculationS();

// Assert

Assert.AreEqual(expectedArea, actualArea);

}

[TestMethod]

public void CalculationS\_InvalidTriangle\_ReturnsNegativeArea()

{

// Arrange

Triangle triangle = new Triangle(1, 2, 3);

double expectedArea = -1;

// Act

double actualArea = triangle.CalculationS();

// Assert

Assert.AreEqual(expectedArea, actualArea);

}

[TestMethod]

public void OperatorPlusPlus\_IncreasesAllSidesByOne()

{

// Arrange

Triangle triangle = new Triangle(2, 3, 4);

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

// Act

Triangle actualTriangle = ++triangle;

// Assert

Assert.AreEqual(expectedTriangle.SideA, actualTriangle.SideA);

Assert.AreEqual(expectedTriangle.SideB, actualTriangle.SideB);

Assert.AreEqual(expectedTriangle.SideC, actualTriangle.SideC);

}

[TestMethod]

public void OperatorMinusMinus\_DecreasesAllSidesByOne()

{

// Arrange

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

Triangle expectedTriangle = new Triangle(2, 3, 4);

// Act

Triangle actualTriangle = --triangle;

// Assert

Assert.AreEqual(expectedTriangle.SideA, actualTriangle.SideA);

Assert.AreEqual(expectedTriangle.SideB, actualTriangle.SideB);

Assert.AreEqual(expectedTriangle.SideC, actualTriangle.SideC);

}

[TestMethod]

public void ExplicitOperatorTest()

{

// Arrange

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

double expectedDouble = 6;

// Act

double actualDouble = (double)triangle;

// Assert

Assert.AreEqual(expectedDouble, actualDouble);

}

[TestMethod]

public void ImplicitOperatorTest()

{

// Arrange

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

bool expectedBool = true;

// Act

bool actualBool = triangle;

// Assert

Assert.AreEqual(expectedBool, actualBool);

}

[TestMethod]

public void OperatorLessThanOrEqual\_TwoTrianglesWithEqualAreas\_ReturnsTrue()

{

// Arrange

Triangle triangle1 = new Triangle(3, 4, 5);

Triangle triangle2 = new Triangle(6, 8, 10);

// Act

bool actualResult = triangle1 <= triangle2;

// Assert

Assert.IsTrue(actualResult);

}

[TestMethod]

public void OperatorGreaterThanOrEqual\_TwoTrianglesWithEqualAreas\_ReturnsTrue()

{

// Arrange

Triangle triangle1 = new Triangle(3, 4, 5);

Triangle triangle2 = new Triangle(6, 8, 10);

// Act

bool actualResult = triangle1 >= triangle2;

// Assert

Assert.IsFalse(actualResult);

}

}

}

**TriangleArrayTests.cs**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.IO;

namespace Lab\_OOP\_9.Tests

{

[TestClass]

public class TriangleArrayTests

{

[TestMethod]

public void Constructor\_DefaultSize\_Zero()

{

// Arrange

int expectedSize = 0;

// Act

TriangleArray array = new TriangleArray();

// Assert

Assert.AreEqual(expectedSize, array.Size);

}

[TestMethod]

public void Constructor\_PositiveSize\_ArrayWithCorrectSize()

{

// Arrange

int expectedSize = 5;

// Act

TriangleArray array = new TriangleArray(expectedSize);

// Assert

Assert.AreEqual(expectedSize, array.Size);

}

[TestMethod]

[ExpectedException(typeof(IndexOutOfRangeException))]

public void Indexer\_OutOfRange\_ExceptionThrown()

{

// Arrange

TriangleArray array = new TriangleArray(3);

// Act

Triangle triangle = array[-1]; // доступ к индексу -1 должен вызвать исключение

// Assert

// Ожидается исключение типа IndexOutOfRangeException

}

[TestMethod]

public void Display\_EmptyArray\_PrintsEmptyMessage()

{

// Arrange

TriangleArray array = new TriangleArray();

// Act

string consoleOutput = CaptureConsoleOutput(() =>

{

array.Display();

});

// Assert

Assert.AreEqual("Массив пуст.", consoleOutput.Trim());

}

private string CaptureConsoleOutput(Action action)

{

using (var consoleOutput = new StringWriter())

{

Console.SetOut(consoleOutput);

action();

return consoleOutput.ToString();

}

}

}

}