МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Гомельский государственный технический

университет имени П. О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Лабораторная работа №3

По дисциплине «Объектно-ориентированное программирование»

**«Операции и перегруженные методы класса»**

Выполнил: студент группы ИТИ-21

Говядкова П. Ю.

Принял: преподаватель

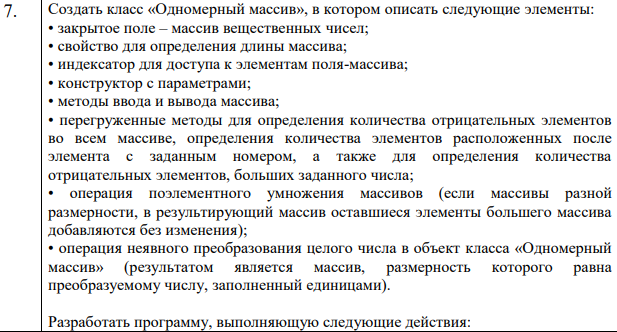
Карабчикова Е. А.

Гомель 2020

**Цель работы:** изучить основы синтаксиса объектно-ориентированного языка программирования, реализацию свойств, методов класса.

**Задание:**

1. Необходимо решить задачу, согласно варианту (Рисунок 1).
2. При создании классов руководствоваться Code Convention.
3. Весь код должен содержать элементами документирования.
4. В реализованных классах должны быть конструкторы с параметрами и по умолчанию, а также необходимые свойства и методы.
5. Для сравнения объектов использовать перегруженный оператор отношения.
6. Разработать модульные тесты для верификации созданных классов.
7. Все классы должны быть размещены в библиотеке классов.
8. Модульные тесты – в отдельном проекте.
9. В отдельном проекте реализовать консольный интерфейс.



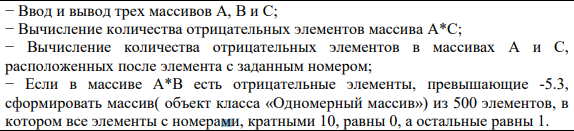


Рисунок 1 – Вариант задания

На рисунке 2 изображена структура решения.

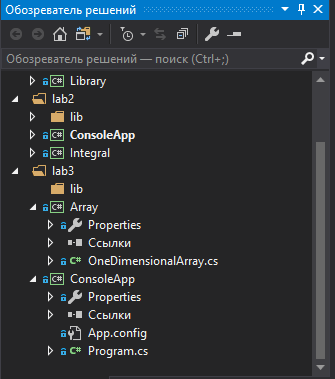


Рисунок 2 – Структура решения

На Рисунке 3 изображено начало выполнения программы.

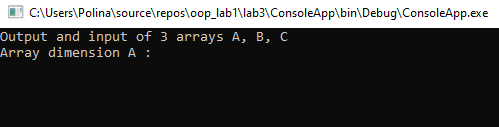


Рисунок 3 – Начало выполнения программы

На Рисунке 4 изображено начало выполнения программы: ввод массива и вывод массива.

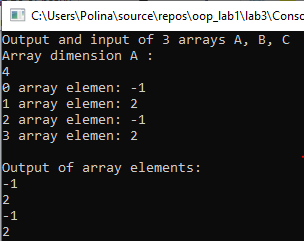


Рисунок 4 – Ввод и вывод массива А

На Рисунке 5 изображено выполнение следующего пункта задания: ввод номера элемента массива и вычисление количества отрицательных элементов после него.

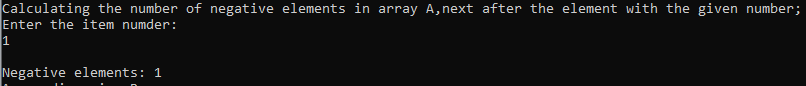


Рисунок 5 – Вычисление количества отрицательных элементов после заданного номера элемента массива

На Рисунке 6 изображено выполнение следующего пункта задания: ввод и вывод массива В.

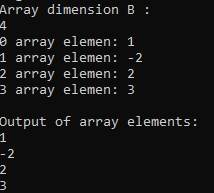


Рисунок 6 – Ввод и вывод массива В

На Рисунке 7 изображено выполнение следующего пункта задания: ввод и вывод массива C.

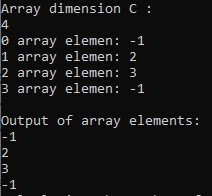


Рисунок 7 – Ввод и вывод массива С

На Рисунке 8 изображено выполнение следующего пункта задания: ввод номера элемента массива и вычисление количества отрицательных элементов после него.

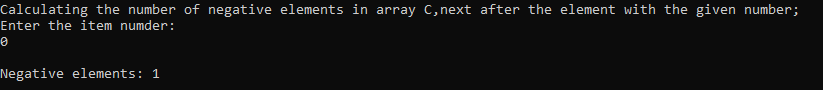


Рисунок 8 – Вычисление количества отрицательных элементов после заданного номера элемента массива

На Рисунке 9 изображено выполнение следующего пункта задания: вычисление количества отрицательных элементов в массиве АС и вывод массива.

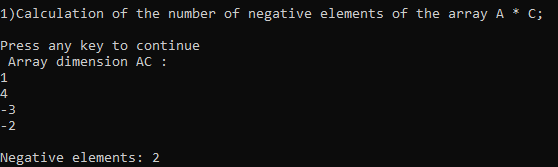


Рисунок 9 – Вывод массива АС и вычисление количества отрицательных элементов в нем.

**Проверка для получения массива АС:**

|  |  |  |
| --- | --- | --- |
| **Массив А** | **Массив С** | **Массив АС** |
| **-1** | **-1** | **1** |
| **2** | **2** | **4** |
| **-1** | **3** | **-3** |
| **2** | **-1** | **-2** |

На Рисунке 10 изображено выполнение следующего пункта задания: если в массиве А\*В есть отрицательные элементы, превышающие -5.3,сформировать массив из 500 элементов, в котором все элементы с номерами, кратными 10, равны 0, а остальные равны 1.

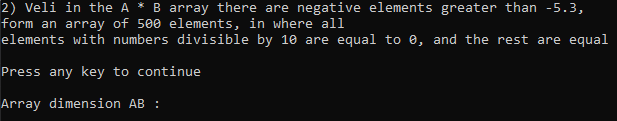


Рисунок 10 – Массив не создается, т.к. нет отрицательных элементов, превышающих -5.3.

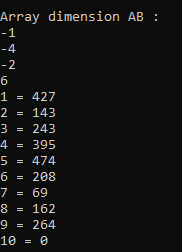


Рисунок 9 – Вывод массива AB

**Проверка для получения массива АВ:**

|  |  |  |
| --- | --- | --- |
| **Массив А** | **Массив В** | **Массив АВ** |
| **-1** | **1** | **-1** |
| **2** | **-2** | **-4** |
| **-1** | **2** | **-2** |
| **2** | **3** | **6** |

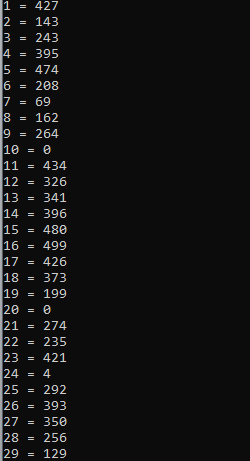


Рисунок 10 – Вывод нового массива из 500 элементов

На Рисунке 11 изображён выполнение программы, когда в массиве А\*В нет отрицательных элементов, превышающих -5.3,чтобы сформировать массив из 500 элементов, в котором все элементы с номерами, кратными 10, равны 0, а остальные равны 1.

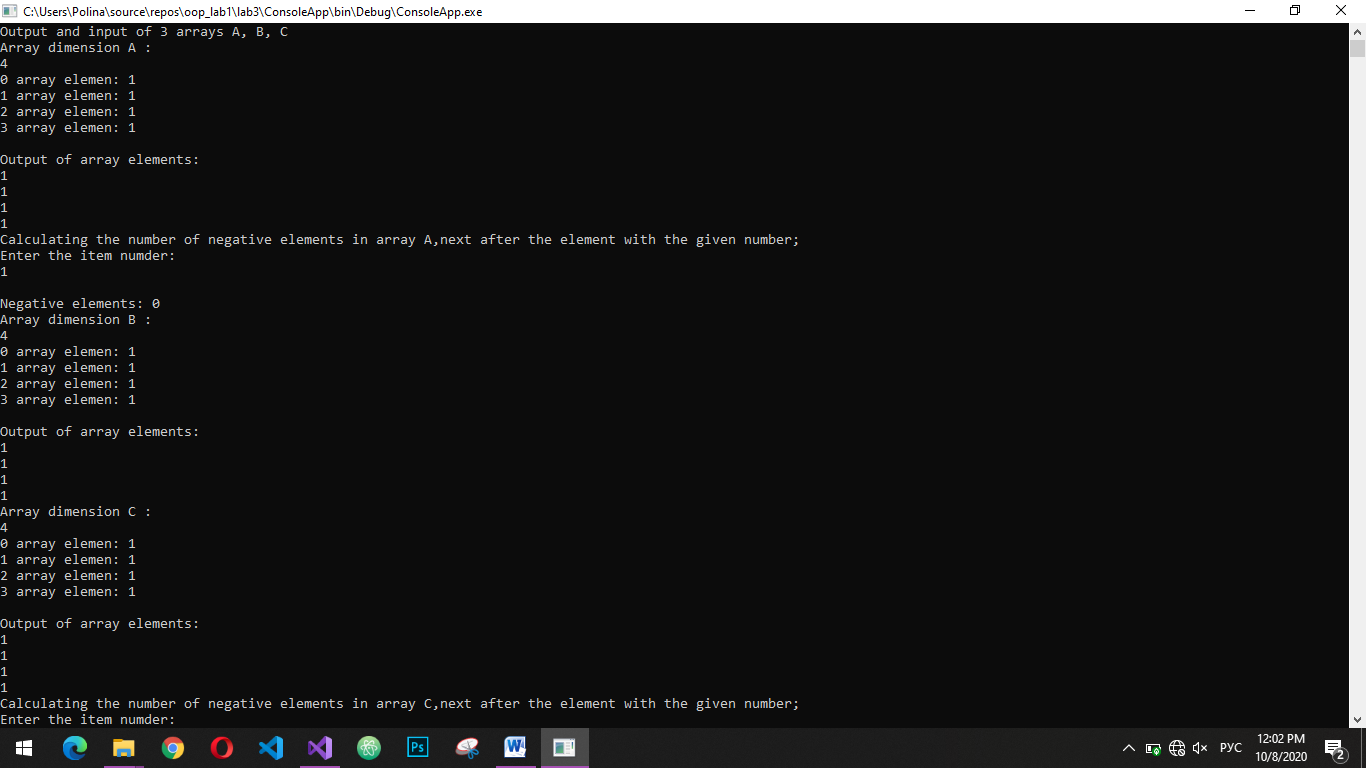
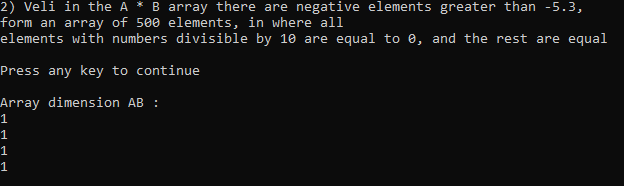


Рисунок 11 – Ввод трёх массивов

Так как нет отрицательных элементов, новый массив не создан.



**Вывод:** в ходе лабораторнойработыизучены основы синтаксиса объектно-ориентированного языка программирования. Реализован класс “Одномерный массив” с полями, свойствами и методами для взаимодействия с объектом данного класса. Также были реализованы задания для работы с данным классом.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

*OneDimensionalArray:*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Linq.Expressions;

using System.Net;

using System.Text;

using System.Threading.Tasks;

namespace Array

{

/// <summary>

/// OneDimensionalArray

/// </summary>

public class OneDimensionalArray

{

/// <summary>

/// The array

/// </summary>

public double[] array;

/// <summary>

/// The array name

/// </summary>

public string array\_name;

/// <summary>

/// Gets the length of the array.

/// </summary>

/// <value>

/// The length of the array.

/// </value>

public int GetLength()

{

return array.Length;

}

/// <summary>

/// Gets or sets the <see cref="System.Double"/> at the specified index.

/// </summary>

/// <value>

/// The <see cref="System.Double"/>.

/// </value>

/// <param name="index">The index.</param>

/// <returns></returns>

public double this[int index]

{

get

{

return array[index];

}

set

{

array[index] = value;

}

}

/// <summary>

/// Initializes a new instance of the <see cref="OneDimensionalArray"/> class.

/// </summary>

/// <param name="array\_name">Name of the array.</param>

public OneDimensionalArray(string array\_name)

{

this.array\_name = array\_name;

array = new double[] { };

//Для работоспособоности теста.

array = new double[] { -1, -2, 4, -3 };

}

/// <summary>

/// Gets the dimensional array.

/// </summary>

public void GetDimensionalArray()

{

int array\_size = 0;

bool flag = true;

while (flag)

{

try

{

array\_size = int.Parse(Console.ReadLine());

flag = false;

}

catch

{}

}

array = new double[array\_size];

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

{

flag = true;

while (flag)

{

try

{

array[i] = double.Parse(Console.ReadLine());

flag = false;

}

catch

{}

}

}

}

/// <summary>

/// Negativ elements of the arrays A and C.

/// </summary>

public int NegativElements()

{

int count = 0;

try

{

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

{

if (array[i] < 0)

{

count++;

}

}

}

catch

{

count = 0;

}

return count;

}

/// <summary>Negativs the elements.</summary>

/// <param name="item">The item.</param>

/// <returns>

/// <br />

/// </returns>

public int NegativElements(double item)

{

int count = 0;

try

{

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

{

if (i > item && array[i] < 0)

{

count++;

}

}

}

catch

{

count = 0;

}

return count;

}

/// <summary>

/// Implements the operator \*.

/// </summary>

/// <param name="ArrayA">The array a.</param>

/// <param name="ArrayC">The array c.</param>

/// <returns>

/// The result of the operator.

/// </returns>

public static OneDimensionalArray operator \*(OneDimensionalArray ArrayA, OneDimensionalArray ArrayC)

{

OneDimensionalArray NewArray = new OneDimensionalArray(ArrayA.array\_name + "\*" + ArrayC.array\_name);

if (ArrayA.GetLength() == ArrayC.GetLength())

{

NewArray.array = new double[ArrayA.GetLength()];

for( int i = 0; i < NewArray.GetLength(); i++)

{

NewArray[i] = ArrayA[i] \* ArrayC[i];

}

}

return NewArray;

}

/// <summary>Arrays for ab.</summary>

/// <returns>

/// New array[500]

/// </returns>

public double[] ArrayAB()

{

double[] newarray = new double[501];

bool flag = false;

try

{

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

{

if (array[i] < 0 && array[i] > -5.3)

{

flag = true;

break;

}

else flag = false;

}

if(flag == true)

{

Random rand = new Random();

for (int j = 1; j <= 500; j++)

{

newarray[j] = rand.Next(500);

if (j % 10 == 0)

{

newarray[j] = 0;

}

}

}

}

catch

{}

return newarray;

}

}

}

Program:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Array;

namespace ConsoleApp

{

/// <summary>

/// Program

/// </summary>

class Program

{

/// <summary>

/// Defines the entry point of the application.

/// </summary>

/// <param name="args">The arguments.</param>

static void Main(string[] args)

{

int item;

OneDimensionalArray ArrayA = new OneDimensionalArray("A");

OneDimensionalArray ArrayB = new OneDimensionalArray("B");

OneDimensionalArray ArrayC = new OneDimensionalArray("C");

OneDimensionalArray MultiplicatoinAandC = new OneDimensionalArray("A\*C");

OneDimensionalArray MultiplicatoinAandB = new OneDimensionalArray("A\*B");

Console.WriteLine("Output and input of 3 arrays A, B, C");

Console.Write("Array dimension A :\n");

ArrayA.GetDimensionalArray();

Console.Write("\nOutput of array elements: \n");

for (int i = 0; i < ArrayA.GetLength(); i++)

{

Console.WriteLine(ArrayA.array[i]);

}

Console.WriteLine("Calculating the number of negative elements in array A,next after the element with the given number; ");

Console.WriteLine("Enter the item numder:");

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

Console.Write($"\nNegative elements: {ArrayA.NegativElements(item)}\n");

Console.Write("Array dimension B :\n");

ArrayB.GetDimensionalArray();

Console.Write("\nOutput of array elements: \n");

for (int i = 0; i < ArrayB.GetLength(); i++)

{

Console.WriteLine(ArrayB.array[i]);

}

Console.Write("Array dimension C :\n");

ArrayC.GetDimensionalArray();

Console.Write("\nOutput of array elements: \n");

for (int i = 0; i < ArrayC.GetLength(); i++)

{

Console.WriteLine(ArrayC.array[i]);

}

Console.WriteLine("Calculating the number of negative elements in array C,next after the element with the given number; ");

Console.WriteLine("Enter the item numder:");

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

Console.Write($"\nNegative elements: {ArrayC.NegativElements(item)}\n");

MultiplicatoinAandC = ArrayA \* ArrayC;

MultiplicatoinAandB = ArrayA \* ArrayB;

Console.WriteLine("\n1)Calculation of the number of negative elements of the array A \* C;");

Console.WriteLine("\nPress any key to continue");

Console.ReadKey();

Console.Write("Array dimension AC :\n");

if (MultiplicatoinAandC.GetLength() == 0)

{

Console.WriteLine("\nIMPOSSIBLE");

}

else

{

for (int i = 0; i < MultiplicatoinAandC.GetLength(); i++)

{

Console.WriteLine(MultiplicatoinAandC.array[i]);

}

Console.Write($"\nNegative elements: {MultiplicatoinAandC.NegativElements()}\n");

}

Console.WriteLine("2) Veli in the A \* B array there are negative elements greater than -5.3,");

Console.WriteLine("form an array of 500 elements, in where all ");

Console.WriteLine("elements with numbers divisible by 10 are equal to 0, and the rest are equal");

Console.WriteLine("\nPress any key to continue");

Console.ReadKey();

Console.Write("\nArray dimension AB :\n");

if (MultiplicatoinAandB.GetLength() == 0)

{

Console.WriteLine("\nIMPOSSIBLE");

}

else

{

for (int i = 0; i < MultiplicatoinAandB.GetLength(); i++)

{

Console.WriteLine(MultiplicatoinAandB.array[i]);

}

double[] array = MultiplicatoinAandB.ArrayAB();

for (int i = 0; i <= 500; i++)

{

Console.Write(i +" = "+array[i]);

Console.Write('\n');

}

}

Console.ReadKey();

}

}

}