**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

по дисциплине: «Объектно-ориентированное проектирование и программирование»

на тему: «Операции и перегруженные методы

класса»

Выполнил студент

группы ИТП-21

Лосев М. А.

Проверил ассистент

Карась О. В.

Гомель 2025

**Цель работы:** Создание консольного приложения с интерфейсом пользователя.

**Задание:** Разработать класс, представляющий собой плоскую фигуру, согласно варианту (рисунок 1).

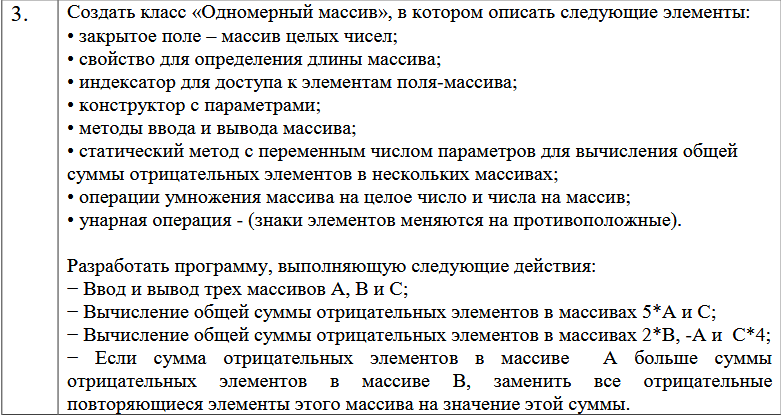


Рисунок 1 – Вариант задания

Ход выполнения задания

1. В приложении А указан код программы*.*
2. На рисунках 2 показан результат запуска программы.
3. На рисунке 3 представлены пройденные тесты

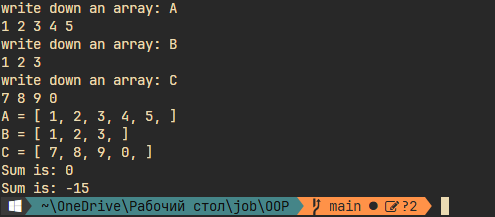


Рисунок 2 – Результат выполнения программы

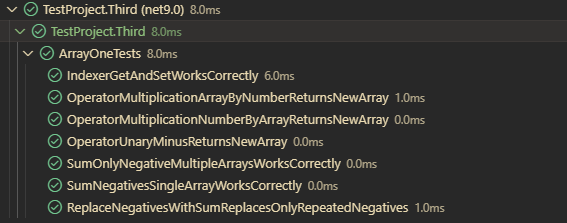


Рисунок 3 – Результат выполнения тестов

В приложении А представлен код программы. В приложении Б представлены полученные *xml* файлы.

**Вывод:** Были изучены методы создания ООП приложений на базе языка *C#* , изучены методы создания библиотек классов и их подключение к основному проекту.

**ПРИЛОЖЕНИЕ А**

(Обязательное)

**Текст программы**

***Program.cs*:**

using System.Collections;

using System.Text;

namespace ClassLib.Third;

public class ArrayOne : IEnumerable<int>

{

/// <summary>

/// inner array

/// </summary>

private int[] array;

/// <summary>

/// array length

/// </summary>

public int Length => array.Length;

/// <summary>

/// array indexer

/// </summary>

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

/// <returns></returns>

public int this[int index]

{

get => array[index];

set => array[index] = value;

}

/// <summary>

/// initialize by size without any elements

/// </summary>

/// <param name="size"></param>

public ArrayOne(int size) => array = new int[size];

/// <summary>

/// initialize from standard array

/// </summary>

/// <param name="array"></param>

public ArrayOne(int[] array) => this.array = (int[])array.Clone();

/// <summary>

/// multiplication

/// </summary>

/// <param name="number"></param>

/// <param name="array"></param>

/// <returns></returns>

public static ArrayOne operator \*(int number, ArrayOne array)

{

var result = new int[array.Length];

for (var i = 0; i < array.Length; i++)

result[i] = array[i] \* number;

return new ArrayOne(result);

}

/// <summary>

/// multiplication

/// </summary>

/// <param name="array"></param>

/// <param name="number"></param>

/// <returns></returns>

public static ArrayOne operator \*(ArrayOne array, int number)

=> number \* array;

/// <summary>

/// operator

/// </summary>

/// <param name="array"></param>

/// <returns></returns>

public static ArrayOne operator -(ArrayOne array)

{

var result = new int[array.Length];

for (var i = 0; i < array.Length; i++)

result[i] = -array[i];

return new ArrayOne(result);

}

/// <summary>

/// input array

/// </summary>

/// <param name="name"></param>

/// <returns></returns>

public static ArrayOne Input(string name)

{

Console.WriteLine($"write down an array: {name}");

var input = Console.ReadLine();

var arr = input?.Split(" ").Select(int.Parse).ToArray() ?? [];

return new ArrayOne(arr);

}

/// <summary>

/// print array

/// </summary>

/// <param name="name"></param>

public void Print(string name)

{

var stringBuilder = new StringBuilder();

stringBuilder.Append("[ ");

foreach (var element in array)

stringBuilder.AppendFormat("{0}, ", element);

stringBuilder.Append("]");

Console.WriteLine($"{name} = {stringBuilder}");

}

/// <summary>

/// static method: sum of negative elements in multiple arrays

/// </summary>

/// <param name="arrays"></param>

/// <returns></returns>

public static int SumOnlyNegative(params ArrayOne[] arrays)

{

return arrays

.Sum(a => a.Where(e => e < 0).Sum());

}

/// <summary>

/// sum of negative elements in one array

/// </summary>

/// <returns></returns>

public int SumNegatives() => array.Where(e => e < 0).Sum();

/// <summary>

/// replace repeated negative elements with given sum

/// </summary>

/// <param name="sum"></param>

public void ReplaceNegativesWithSum(int sum)

{

var seen = new HashSet<int>();

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

{

if (array[i] < 0)

{

if (seen.Contains(array[i]))

array[i] = sum;

else

seen.Add(array[i]);

}

}

}

/// <inheritdoc/>

public IEnumerator<int> GetEnumerator()

{

foreach (var element in array)

yield return element;

}

/// <inheritdoc/>

IEnumerator IEnumerable.GetEnumerator()

{

return GetEnumerator();

}

}