Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

**Кафедра прикладной математики и кибернетики**

Современные технологии программирования

Лабораторная работа №1

«Модульное тестирование программ на языке С# средствами Visual Studio» Вариант №2

Выполнил: студент 4 курса группы ИП-111

Кузьменок Денис Витальевич

Проверил преподаватель: Зайцев Михаил Георгиевич

Новосибирск, 2024 г.

**Цель:**

Сформировать практические навыки разработки модульных тестов для тестирования функций классов и выполнения модульного тестирования на языке С# с помощью средств автоматизации Visual Studio.

**Задание:**

Разработайте на языке С# класс, содержащий функции в соответствии с вариантом задания. Разработайте тестовые наборы данных по критерию С0 для тестирования функций класса. Протестируйте созданный класс с помощью средств автоматизации модульного тестирования Visual Studio. Напишите отчёт о результатах проделанной работы.

1. Функция получает два одномерных массива a, b одинаковой длины. Возвращает массив c полученный суммированием значений массивов a, b.
2. Функция получает одномерный массив вещественных переменных и целое – параметр сдвига. Функция изменяет массив циклическим сдвигом значений его элементов влево на число позиций, равное параметру сдвига.
3. Функция находит и возвращает индекс начала первого вхождения последовательности целых чисел, представленных массивом int[] seq в другую последовательность, представленную массивом int[] vec.

**Реализация**

В ходе выполнения лабораторной работы мною был реализован класс ArrayOperations в соответствии с вариантом задания. Класс содержит три метода:

public static int[] SumArrays(int[] a, int[] b) – Метод принимает два целочисленных массива. Вычисляет и возвращает новый массив, заполненный элементами, полученными в ходе суммирования соответствующих элементов массивов a и b.

public static void CyclicShifLeft(double[] a, int shift) – Метод на вход принимает массив чисел с плавающей точкой и параметр сдвига в массиве. Обрабатываются два случая:

1. Когда сдвиг положительное число, то вычисляется по модулю этот сдвиг в том случае, если сдвиг указан больше, чем длина массива;
2. Когда сдвиг отрицательный, то генерируется исключение с соответствующим сообщением.

В конечном итоге, на выходе получается массив с указанным сдвигом влево.

public static int FindSequenceStart(int[] vec, int[] seq) – Метод ищет индекс первого вхождения подмассива seq в массив vec. Если длина seq больше чем длина vec, то генерируется исключение с соответствующим сообщением. В цикле перебирается массив vec, и если находится соответствие, то возвращается индекс. В противном случае -1.

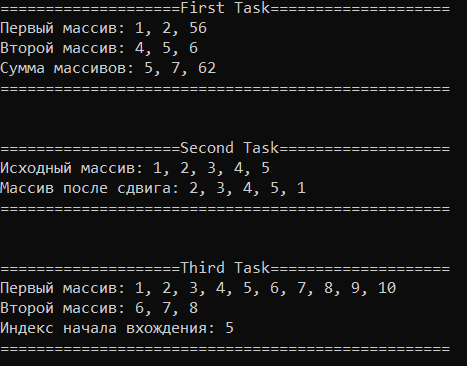
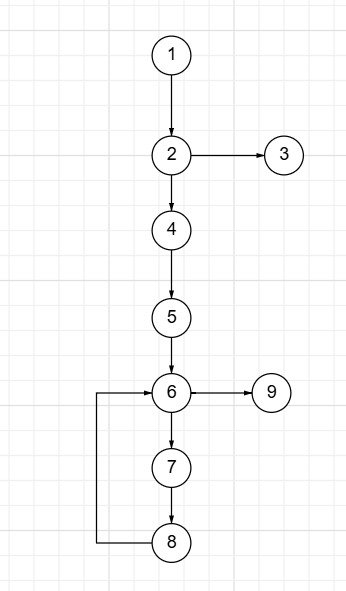


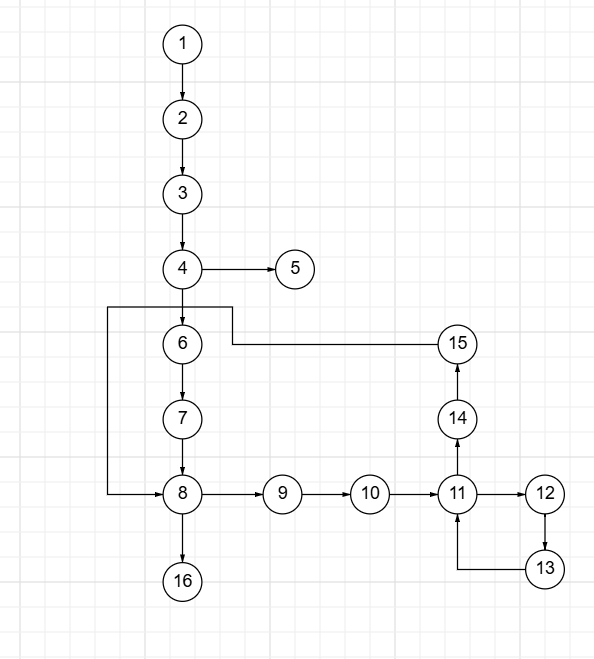
Рис. 1 – Результат запуска программы

По результатам создания методов были созданы управляющие графы для каждого из них:

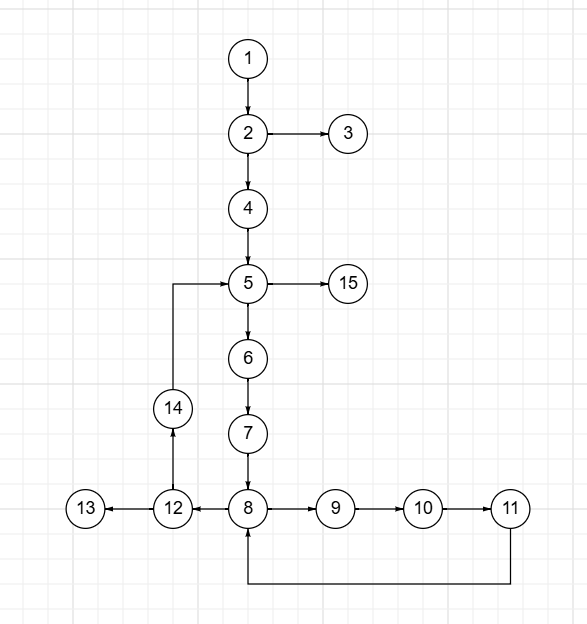
Для public static int[] SumArrays(int[] a, int[] b):



Для public static void CyclicShifLeft(double[] a, int shift):



Для public static int FindSequenceStart(int[] vec, int[] seq):



1. **SumArrays**
2. **Корректные массивы:**

* Входные данные: a = [1, 2, 3], b = [4, 5, 6]
* Ожидаемый результат: суммирование двух массивов и получение результирующего c = [5, 7, 9]

1. **Массивы разной длины:**

* Входные данные: a = [1, 2, 3, 78], b = [4, 5, 6]
* Ожидаемый результат: генерация исключения invalid\_argumanet\_2

1. **CyclicShifLeft**
2. **Корректные данные:**

* Входные данные: a = [1, 2, 3], shift = 2
* Ожидаемый результат: циклический сдвиг влево элементов массива a = [3, 1, 2]

1. **Отрицательный сдвиг:**

* Входные данные: a = [1, 2, 3], shift = -2
* Ожидаемый результат: генерация исключения invalid\_argumanet\_2

1. **CyclicShifLeft**
2. **Корректные данные:**

* Входные данные: vec = [1, 2, 3, 4, 5, 6, 7], seq = [4, 5]
* Ожидаемый результат: нахождение индекса вхождение подмассива seq в массив vec. index = 3.

1. **Длинна подмассива больше, чем длинна исходного массива:**

* Входные данные: vec = [4, 5], seq =[1, 2, 3, 4, 5, 6, 7]
* Ожидаемый результат: генерация исключения invalid\_argumanet\_2

1. **Отсутствие результата поиска:**

* Входные данные: vec = [1, 2, 3, 4, 5, 6, 7], seq = [78, 35]
* Ожидаемый результат: отсутствие индекса вхождения подмассива seq в массив vec. Возвращение -1.

Тестирование метода SumArrays: проверяет корректность вычисления суммы результирующего массива c. Включает проверку на обработку исключения в виду разной длины массивов a и b.

Тестирование метода CyclicShifLeft: проверяет корректность циклического сдвига влево массива на заданное количество позиций. Включает проверку на отрицательного сдвига (генерируется исключение).

Тестирование метода FindSequenceStart: проверяет нахождение подмассива seq в массиве vec, и в случае успеха возвращает индекс позиции. Включает проверки на корректный размер двух массивов (массив seq должен быть меньше, чем массив vec, иначе генерируется исключение), возвращение -1, если позиция не была найдена.

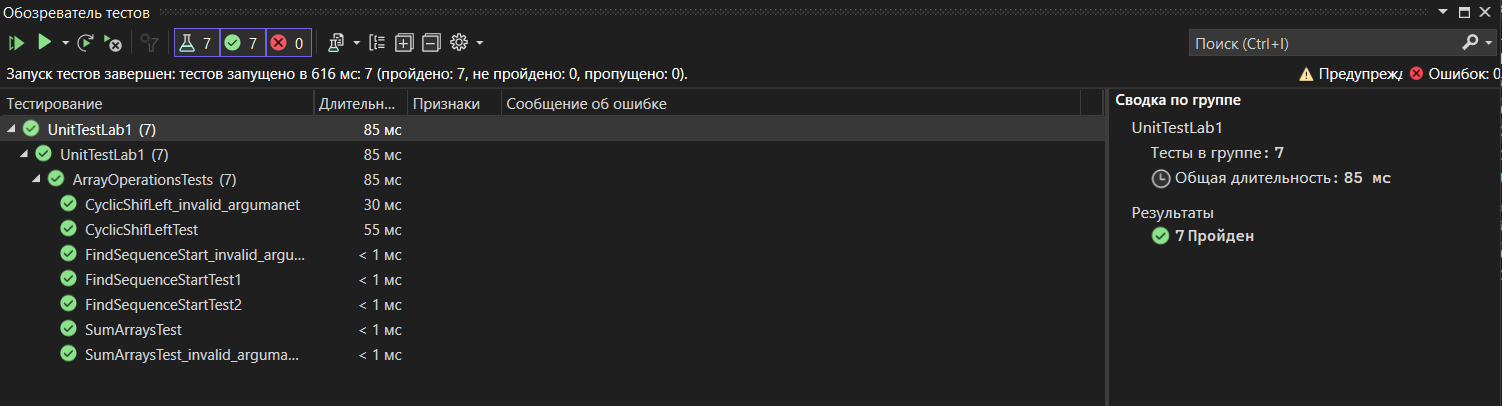


Рис. 2 – Результат выполнения модульных тестов.

**Вывод:**

В ходе выполнения лабораторной работы я приобрел практический опыт в:

1. Разработке методов классов на языке C#: Создание и реализация методов внутри классов C#.
2. Разработке модульных тестов: Написание тестов для отдельных функций или методов классов.
3. Выполнении модульного тестирования на C# с помощью Visual Studio: Использование инструментов автоматизации Visual Studio для запуска и анализа модульных тестов.

**Листинг программы:**

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab1

{

class Program

{

static void Main(string[] args)

{

//ArrayOperations arrayOperations = new ArrayOperations();

//Первое задание

Console.WriteLine("====================First Task====================");

int[] first = { 1, 2, 56 };

int[] second = { 4, 5, 6 };

int[] sumArray = ArrayOperations.SumArrays(first, second);

Console.WriteLine("Первый массив: " + string.Join(", ", first));

Console.WriteLine("Второй массив: " + string.Join(", ", second));

Console.WriteLine("Сумма массивов: " + string.Join(", ", sumArray));

Console.WriteLine("==================================================\n");

//Второе задание

Console.WriteLine("\n====================Second Task===================");

double[] array = { 1.0, 2.0, 3.0, 4.0, 5.0 };

int shift = 1;

Console.WriteLine("Исходный массив: " + string.Join(", ", array));

ArrayOperations.CyclicShifLeft(array, shift);

Console.WriteLine("Массив после сдвига: " + string.Join(", ", array));

Console.WriteLine("==================================================\n");

//Третье задание

Console.WriteLine("\n====================Third Task====================");

int[] vec = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

int[] seq = { 6, 7, 8 };

Console.WriteLine("Первый массив: " + string.Join(", ", vec));

Console.WriteLine("Второй массив: " + string.Join(", ", seq));

int index = ArrayOperations.FindSequenceStart(vec, seq);

Console.WriteLine($"Индекс начала вхождения: {index}");

Console.WriteLine("==================================================\n");

}

}

}

**ArrayOperations.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab1

{

public class ArrayOperations

{

public class invalid\_argumanet\_2: ArgumentException

{

public invalid\_argumanet\_2(string message) : base(message) { }

}

public static int[] SumArrays(int[] a, int[] b)

{

if (a.Length != b.Length) throw new invalid\_argumanet\_2("Массивы должны быть одинаковой длины!");

int[] result = new int[a.Length];

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

{

result[i] = a[i] + b[i];

}

return result;

}

public static void CyclicShifLeft(double[] a, int shift)

{

int n = a.Length;

shift = shift % n;

if (shift < 0) throw new invalid\_argumanet\_2("Сдвиг не может быть отрицательным!");

if (shift > 0)

{

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

{

double temp = a[0];

for (int j = 0; j < n - 1; j++)

{

a[j] = a[j + 1];

}

a[n - 1] = temp;

}

}

}

public static int FindSequenceStart(int[] vec, int[] seq)

{

if (seq.Length > vec.Length) throw new invalid\_argumanet\_2("Подстрока не может быть больше, чем исходная строка!");

for (int i = 0; i <= vec.Length - seq.Length; i++)

{

bool correct = true;

for (int j = 0; j < seq.Length; j++)

{

if (vec[i + j] != seq[j])

{

correct = false;

break;

}

}

if (correct)

{

return i;

}

}

return -1;

}

}

}

**UnitTest1.cs**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using lab1;

namespace UnitTestLab1

{

[TestClass]

public class ArrayOperationsTests

{

[TestMethod]

public void SumArraysTest()

{

int[] first = { 1, 2 };

int[] second = { 3, 4 };

int[] expected = { 4, 6 };

int[] result = ArrayOperations.SumArrays(first, second);

CollectionAssert.AreEqual(expected, result);

}

[TestMethod]

[ExpectedException(typeof(ArrayOperations.invalid\_argumanet\_2))]

public void SumArraysTest\_invalid\_argumanet()

{

int[] first = { 1 };

int[] second = { 3, 4 };

ArrayOperations.SumArrays(first, second);

}

[TestMethod]

public void CyclicShifLeftTest()

{

double[] result = { 1.0, 2.0, 3.0, 4.0, 5.0 };

int shift = 4;

double[] expected = { 5.0, 1.0, 2.0, 3.0, 4.0 };

ArrayOperations.CyclicShifLeft(result, shift);

CollectionAssert.AreEqual(expected, result);

}

[TestMethod]

[ExpectedException(typeof(ArrayOperations.invalid\_argumanet\_2))]

public void CyclicShifLeft\_invalid\_argumanet()

{

double[] result = { 1.0, 2.0, 3.0, 4.0, 5.0 };

int shift = -2;

ArrayOperations.CyclicShifLeft(result, shift);

}

[TestMethod]

public void FindSequenceStartTest1()

{

int[] vec = { 4, 67, 32, 5, 997, 52, 0, -53, 63, 841, 11, 8 };

int[] seq = { 0, -53 };

int expected = 6;

int result = ArrayOperations.FindSequenceStart(vec, seq);

Assert.AreEqual(expected, result);

}

[TestMethod]

public void FindSequenceStartTest2()

{

int[] vec = { 4, 67, 32, 5, 997, 52, 0, -53, 63, 841, 11, 8 };

int[] seq = { 999, -53 };

int expected = -1;

int result = ArrayOperations.FindSequenceStart(vec, seq);

Assert.AreEqual(expected, result);

}

[TestMethod]

[ExpectedException(typeof(ArrayOperations.invalid\_argumanet\_2))]

public void FindSequenceStart\_invalid\_argumanet()

{

int[] vec = { 999, -53 };

int[] seq = { 4, 67, 32, 5, 997, 52, 0, -53, 63, 841, 11, 8 };

ArrayOperations.FindSequenceStart(vec, seq);

}

}

}