**Лабораторная работа № 15-16**

Разработка динамической библиотеки по работе с массивами с использованием перегрузки операций

**Цель работы:** получить практические навыки разработки динамической библиотеки, её тестирования, подключения и использования в разных проектах.

**Ход работы:**

1. Для того чтобы разработать собственную динамическую библиотеку, создайте новый проект и выберите вид проекта: **библиотека классов**. Опишите структуры данных, методы и операции класса по заданию. Обязательно добавьте XML-комментарии перед заголовком каждого метода.
2. На выполнение библиотечный модуль не запускается. Необходимо сохранить проект и произвести сборку на диск (построение).
3. Чтобы проверить работу библиотечного модуля, разработайте **юнит-тесты** и протестируйте основные методы и операции библиотеки. Для подключения собственной библиотеки к любому проекту, необходимо добавить ссылку на библиотечный модуль в проект (ссылку на файл с расширением .dll), а также указать имя модуля в списке используемых модулей (в разделе using).
4. Разработайте **консольное приложение**, в главной функции которого организуйте создание объектов-массивов и вызов методов и операций класса.
5. В отдельном проекте разработайте **оконное** **приложение**, и тоже подключите к нему вашу библиотеку. Пользовательский интерфейс может быть оформлен в виде системы меню или панелей с кнопками.

Приложение должно обрабатывать 2 массива разной размерности. Размер каждого массива запрашивается у пользователя. Для работы с массивами надо предусмотреть на форме 2 таблицы, размеры которых задаются в процессе выполнения программы. Заполнение каждого массива может происходить при помощи генератора случайных чисел или из текстового файла (в последнем случае размер массива должен определяться по количеству чисел в файле). Для выбора режима заполнения массива используйте переключатели «зависимые кнопки выбора».

***Задание 1*** Разработка собственной библиотеки.

**В библиотечный модуль необходимо включить следующие методы:**

* Конструкторы класса;
* Методы заполнения массива произвольного размера случайными числами;
* Загрузка данных массива из строки (числа разделены одним пробелом);
* Загрузка данных массива из текстового файла;
* Формирование строки для вывода массива — перегрузка метода ToString();
* Метод сравнения двух объектов класса — перегрузка метода Equals() и связанного с ним метода GetHashCode();
* Вывод массива произвольного размера на консоль;
* Вывод массива произвольного размера в компонент таблица (перегрузка предыдущего метода);
* Перегрузка унарных операций ++ и - - для массива;
* Перегрузка операций сложения и вычитания для двух массивов разного размера;
* Перегрузка операций сложения и вычитания массива с заданным целым числом, число прибавляется/вычитается к каждому элементу массива;
* Метод по вашему индивидуальному заданию.

**Вариант 1**

Заполнить одномерный массив целыми числами. Вычислить количество перемен знака с минуса на плюс или с плюса на минус. Нуль знака не имеет.

с первым элементом.

using System;

using System.IO;

using System.Reflection.Emit;

namespace ClassLibrary1

{

    public class MyException : Exception

    {

        public const string ArgumentError = "Массив не обработан!";

        public const string SizeError = "Выход за границы массива!";

        public int Value { get; }

        public MyException(string message)

            : base(message)

        {

        }

        public MyException(string message, int value)

            : base(message)

        {

            Value = value;

        }

    }

    public class Arr

    {

        private int[] a;

        private int size;

        private static Random rnd = new Random();

        public Arr()

        {

            size = 0;

            a = null;

        }

        public Arr(int n)

        {

            a = new int[n];

            size = n;

        }

        public Arr(params int[] x)

        {

            if (x == null) throw new MyException(MyException.ArgumentError);

            size = x.Length;

            a = new int[size];

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

                a[i] = x[i];

        }

        public Arr(Arr B)

        {

            if (B.a == null)

                throw new MyException(MyException.ArgumentError);

            size = B.size;

            a = new int[size];

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

                a[i] = B.a[i];

        }

        public override bool Equals(object obj)

        {

            if (obj == null || obj as Arr == null)

                return false;

            if (size != ((Arr)obj).size)

                return false;

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

                if (a[i] != ((Arr)obj).a[i])

                    return false;

            return true;

        }

        public override int GetHashCode()

        {

            return base.GetHashCode();

        }

        public int Size

        {

            get { return size; }

        }

        public void RndInput()

        {

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

                a[i] = rnd.Next(-100, 101);

        }

        public void RndInput(int n1)

        {

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

                a[i] = rnd.Next(n1 + 1);

        }

        public static Arr operator --(Arr arr)

        {

            Arr temp = new Arr(arr.size);

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

                temp.a[i] = arr.a[i] - 1;

            return temp;

        }

        public static Arr operator ++(Arr arr)

        {

            Arr temp = new Arr(arr.size);

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

                temp.a[i] = arr.a[i] + 1;

            return temp;

        }

        public static Arr operator +(Arr X, Arr Y)

        {

            int newSize = Math.Max(X.size, Y.size);

            Arr temp = new Arr(newSize);

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

            {

                int xValue = (i < X.size) ? X[i] : 0;

                int yValue = (i < Y.size) ? Y[i] : 0;

                temp[i] = xValue + yValue;

            }

            return temp;

        }

        public static Arr operator -(Arr X, Arr Y)

        {

            int newSize = Math.Max(X.size, Y.size);

            Arr temp = new Arr(newSize);

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

            {

                int xValue = (i < X.size) ? X[i] : 0;

                int yValue = (i < Y.size) ? Y[i] : 0;

                temp[i] = xValue - yValue;

            }

            return temp;

        }

        public static Arr operator +(Arr arr, int value)

        {

            Arr temp = new Arr(arr.size);

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

                temp[i] = arr[i] + value;

            return temp;

        }

        public static Arr operator -(Arr arr, int value)

        {

            Arr temp = new Arr(arr.size);

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

                temp[i] = arr[i] - value;

            return temp;

        }

        public int this[int i]

        {

            get

            {

                if (i >= 0 && i < size)

                    return a[i];

                throw new MyException(MyException.SizeError, i);

            }

            set

            {

                if (i >= 0 && i < size)

                    a[i] = value;

                else

                    throw new MyException(MyException.SizeError, i);

            }

        }

        public override string ToString()

        {

            string s = "";

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

                s += a[i].ToString() + " ";

            return s.Trim();

        }

        public static Arr Parse(string input)

        {

            if (string.IsNullOrWhiteSpace(input))

                throw new MyException(MyException.ArgumentError);

            string[] elements = input.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

            int[] parsedArray = new int[elements.Length];

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

            {

                if (!int.TryParse(elements[i], out parsedArray[i]))

                    throw new MyException(MyException.ArgumentError);

            }

            return new Arr(parsedArray);

        }

        public void Print()

        {

            Console.WriteLine(this.ToString());

            Console.WriteLine();

        }

        public void LoadFromFile(string filePath)

        {

            if (!File.Exists(filePath))

                throw new MyException("Файл не найден!");

            string[] lines = File.ReadAllLines(filePath);

            string[] elements = lines[0].Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

            a = new int[elements.Length];

            size = elements.Length;

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

            {

                if (!int.TryParse(elements[i], out a[i]))

                    throw new MyException("Ошибка при чтении данных из файла!");

            }

        }

        public int CountSignChanges()

        {

            if (size < 2)

                return 0;

            int count = 0;

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

            {

                if ((a[i - 1] > 0 && a[i] < 0) || (a[i - 1] < 0 && a[i] > 0))

                    count++;

            }

            return count;

        }

    }

}

***Задание 2***Проверить работу библиотечного модуля с помощью юнит-тестов.

Необходимо протестировать 2 любые операции класса и метод по вашему индивидуальному заданию. Для проверки вашего метода может понадобится несколько тестов. Если какой-либо метод может выдавать исключение, то необходимо сделать тесты на штатные ситуации и обязательно хотя бы один негативный тест для проверки выдаваемого исключения.

Тесты:

using Microsoft.VisualStudio.TestTools.UnitTesting;

using ClassLibrary1;

namespace TestProject1

{

    [TestClass]

    public class UnitTest1

    {

        [TestMethod]

        public void TestIncrementOperator()

        {

            // Arrange

            Arr actual = new Arr(new int[] { 10, -5, 4 });

            // Act

            actual++;

            // Assert

            Arr expected = new Arr(new int[] { 11, -4, 5 });

            Assert.AreEqual(expected.ToString(), actual.ToString(), "Результат инкремента не соответствует ожидаемому.");

        }

        [TestMethod]

        public void TestDecrementOperator()

        {

            // Arrange

            Arr actual = new Arr(new int[] { 10, -5, 4 });

            // Act

            actual--;

            // Assert

            Arr expected = new Arr(new int[] { 9, -6, 3 });

            Assert.AreEqual(expected.ToString(), actual.ToString(), "Результат декремента не соответствует ожидаемому.");

        }

        [TestMethod]

        public void TestCountSignChanges\_EmptyArray()

        {

            // Arrange

            Arr arr = new Arr();

            // Act

            int result = arr.CountSignChanges();

            // Assert

            Assert.AreEqual(0, result, "Пустой массив должен возвращать 0 смен знаков.");

        }

        [TestMethod]

        public void TestCountSignChanges\_AllZeros()

        {

            // Arrange

            Arr arr = new Arr(new int[] { 0, 0, 0 });

            // Act

            int result = arr.CountSignChanges();

            // Assert

            Assert.AreEqual(0, result, "Массив только с нулями должен возвращать 0 смен знаков.");

        }

        [TestMethod]

        public void TestCountSignChanges\_MaxSignChanges()

        {

            // Arrange

            Arr arr = new Arr(new int[] { 1, -2, 3 });

            // Act

            int result = arr.CountSignChanges();

            // Assert

            Assert.AreEqual(2, result, "Массив с чередующимися знаками должен возвращать 2 смены знаков.");

        }

        [TestMethod]

        [ExpectedException(typeof(MyException))]

        public void TestIndexer\_OutOfBounds()

        {

            // Arrange

            Arr arr = new Arr(2);

            // Act

            int value = arr[3]; // Должно выбросить MyException

        }

    }

}

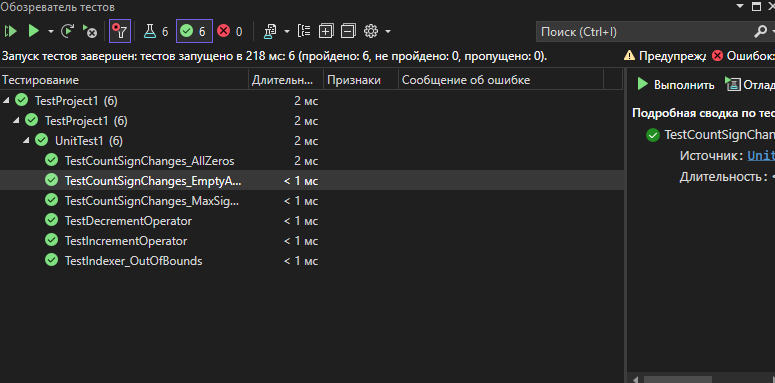


Рисунок 1 – результат тестов

***Задание 3***

Подключить библиотеку к консольному приложению и показать вызов основных методов и операций класса.

Код:

using System;

using ClassLibrary1;

namespace ConsoleApp

{

    class Program

    {

        static void Main(string[] args)

        {

            try

            {

                // 1. Создание массива через конструктор с размером

                Console.WriteLine("1. Создание массива размером 5 и заполнение случайными числами:");

                Arr arr1 = new Arr(5);

                arr1.RndInput(); // Заполнение случайными числами от -100 до 100

                Console.WriteLine($"Массив: {arr1}");

                Console.WriteLine($"Количество смен знаков: {arr1.CountSignChanges()}");

                Console.WriteLine();

                // 2. Создание массива через конструктор с параметрами

                Console.WriteLine("2. Создание массива из заданных значений [3, 0, -2, 5, 0]:");

                Arr arr2 = new Arr(3, 0, -2, 5, 0);

                Console.WriteLine($"Массив: {arr2}");

                Console.WriteLine($"Количество смен знаков: {arr2.CountSignChanges()}");

                Console.WriteLine();

                // 3. Использование оператора инкремента

                Console.WriteLine("3. Применение оператора ++ к arr2:");

                Arr arr3 = arr2++;

                Console.WriteLine($"Исходный массив после ++: {arr2}");

                Console.WriteLine($"Новый массив: {arr3}");

                Console.WriteLine();

                // 4. Использование оператора декремента

                Console.WriteLine("4. Применение оператора -- к arr2:");

                Arr arr4 = arr2--;

                Console.WriteLine($"Исходный массив после --: {arr2}");

                Console.WriteLine($"Новый массив: {arr4}");

                Console.WriteLine();

                // 5. Сложение двух массивов

                Console.WriteLine("5. Сложение arr1 и arr2:");

                Arr arr5 = arr1 + arr2;

                Console.WriteLine($"Массив arr1: {arr1}");

                Console.WriteLine($"Массив arr2: {arr2}");

                Console.WriteLine($"Результат сложения: {arr5}");

                Console.WriteLine();

                // 6. Вычитание массивов

                Console.WriteLine("6. Вычитание arr2 из arr1:");

                Arr arr6 = arr1 - arr2;

                Console.WriteLine($"Массив arr1: {arr1}");

                Console.WriteLine($"Массив arr2: {arr2}");

                Console.WriteLine($"Результат вычитания: {arr6}");

                Console.WriteLine();

                // 7. Использование индексатора

                Console.WriteLine("7. Использование индексатора для arr2:");

                Console.WriteLine($"Элемент arr2[2]: {arr2[2]}");

                arr2[2] = 10;

                Console.WriteLine($"Массив после изменения arr2[2] = 10: {arr2}");

                Console.WriteLine();

                // 8. Парсинг строки в массив

                Console.WriteLine("8. Парсинг строки '1 -2 3' в массив:");

                Arr arr7 = Arr.Parse("1 -2 3");

                Console.WriteLine($"Спарсенный массив: {arr7}");

                Console.WriteLine($"Количество смен знаков: {arr7.CountSignChanges()}");

                Console.WriteLine();

                // 9. Обработка исключения при некорректном индексе

                Console.WriteLine("9. Попытка доступа к некорректному индексу:");

                try

                {

                    int value = arr2[10]; // Должно выбросить исключение

                }

                catch (MyException ex)

                {

                    Console.WriteLine($"Ошибка: {ex.Message} Значение: {ex.Value}");

                }

            }

            catch (MyException ex)

            {

                Console.WriteLine($"Ошибка: {ex.Message} Значение: {ex.Value}");

            }

            catch (Exception ex)

            {

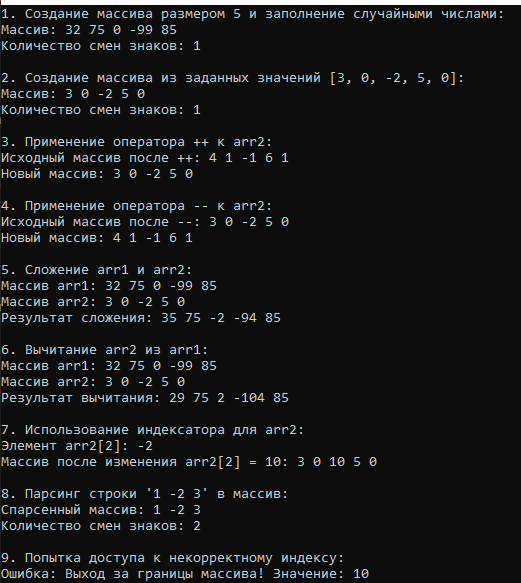
                Console.WriteLine($"Общая ошибка: {ex.Message}");

            }

        }

    }

}

****

***Задание 4***

Создайте новый проект **Windows Forms**, подключите к нему собственную библиотеку. Подготовьте пользовательский интерфейс, добавив на форму нужные компоненты с их последующей настройкой. Создайте методы обработки событий формы. Настройте пользовательский интерфейс таким образом, чтобы некоторые поля ввода и кнопки блокировались, если режим работы не предусматривает их активность.

Программа должна выполнять все задания с первым или вторым массивом по выбору пользователя. Добавьте также кнопку или пункт меню для вызова метода по вашему варианту. Если в самом массиве происходят изменения (например, перестановка элементов), результат вывести в ту же таблицу. Если же необходимо из двух массивов получить новый массив, то результат выводите в таблицу 3.

**Компоненты формы Form1.cs:**

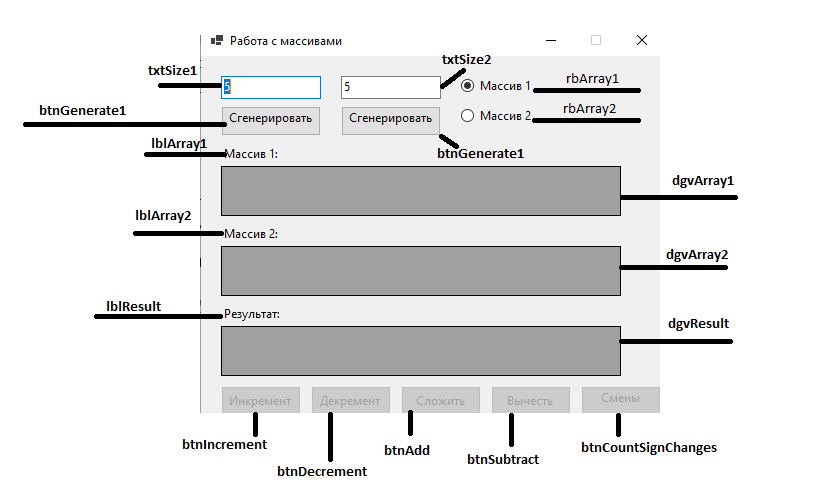
****

Рисунок 2 - форма

toolStrip

Timer

**Код модуля формы:**

using System;

using System.Windows.Forms;

using ClassLibrary1;

namespace WindowsFormsApp

{

    public partial class Form1 : Form

    {

        private Arr arr1, arr2;

        public Form1()

        {

            InitializeComponent();

            UpdateInterface();

        }

        private void btnGenerate1\_Click(object sender, EventArgs e)

        {

            try

            {

                if (!int.TryParse(txtSize1.Text, out int size) || size <= 0)

                {

                    MessageBox.Show("Введите корректный размер для первого массива!");

                    return;

                }

                arr1 = new Arr(size);

                arr1.RndInput();

                PrintToDataGridView(arr1, dgvArray1);

                lblArray1.Text = "Массив 1: " + arr1.ToString();

                UpdateInterface();

            }

            catch (MyException ex)

            {

                MessageBox.Show($"Ошибка: {ex.Message} Значение: {ex.Value}");

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Общая ошибка: {ex.Message}");

            }

        }

        private void btnGenerate2\_Click(object sender, EventArgs e)

        {

            try

            {

                if (!int.TryParse(txtSize2.Text, out int size) || size <= 0)

                {

                    MessageBox.Show("Введите корректный размер для второго массива!");

                    return;

                }

                arr2 = new Arr(size);

                arr2.RndInput();

                PrintToDataGridView(arr2, dgvArray2);

                lblArray2.Text = "Массив 2: " + arr2.ToString();

                UpdateInterface();

            }

            catch (MyException ex)

            {

                MessageBox.Show($"Ошибка: {ex.Message} Значение: {ex.Value}");

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Общая ошибка: {ex.Message}");

            }

        }

        private void btnIncrement\_Click(object sender, EventArgs e)

        {

            try

            {

                if (rbArray1.Checked && arr1 != null)

                {

                    arr1++;

                    PrintToDataGridView(arr1, dgvArray1);

                    lblArray1.Text = "Массив 1: " + arr1.ToString();

                }

                else if (rbArray2.Checked && arr2 != null)

                {

                    arr2++;

                    PrintToDataGridView(arr2, dgvArray2);

                    lblArray2.Text = "Массив 2: " + arr2.ToString();

                }

                else

                {

                    MessageBox.Show("Выберите массив и сгенерируйте его!");

                }

            }

            catch (MyException ex)

            {

                MessageBox.Show($"Ошибка: {ex.Message} Значение: {ex.Value}");

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Общая ошибка: {ex.Message}");

            }

        }

        private void btnDecrement\_Click(object sender, EventArgs e)

        {

            try

            {

                if (rbArray1.Checked && arr1 != null)

                {

                    arr1--;

                    PrintToDataGridView(arr1, dgvArray1);

                    lblArray1.Text = "Массив 1: " + arr1.ToString();

                }

                else if (rbArray2.Checked && arr2 != null)

                {

                    arr2--;

                    PrintToDataGridView(arr2, dgvArray2);

                    lblArray2.Text = "Массив 2: " + arr2.ToString();

                }

                else

                {

                    MessageBox.Show("Выберите массив и сгенерируйте его!");

                }

            }

            catch (MyException ex)

            {

                MessageBox.Show($"Ошибка: {ex.Message} Значение: {ex.Value}");

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Общая ошибка: {ex.Message}");

            }

        }

        private void btnAdd\_Click(object sender, EventArgs e)

        {

            try

            {

                if (arr1 == null || arr2 == null)

                {

                    MessageBox.Show("Сгенерируйте оба массива!");

                    return;

                }

                Arr result = arr1 + arr2;

                PrintToDataGridView(result, dgvResult);

                lblResult.Text = "Результат: " + result.ToString();

            }

            catch (MyException ex)

            {

                MessageBox.Show($"Ошибка: {ex.Message} Значение: {ex.Value}");

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Общая ошибка: {ex.Message}");

            }

        }

        private void btnSubtract\_Click(object sender, EventArgs e)

        {

            try

            {

                if (arr1 == null || arr2 == null)

                {

                    MessageBox.Show("Сгенерируйте оба массива!");

                    return;

                }

                Arr result = arr1 - arr2;

                PrintToDataGridView(result, dgvResult);

                lblResult.Text = "Результат: " + result.ToString();

            }

            catch (MyException ex)

            {

                MessageBox.Show($"Ошибка: {ex.Message} Значение: {ex.Value}");

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Общая ошибка: {ex.Message}");

            }

        }

        private void btnCountSignChanges\_Click(object sender, EventArgs e)

        {

            try

            {

                if (rbArray1.Checked && arr1 != null)

                {

                    int signChanges = arr1.CountSignChanges();

                    lblResult.Text = $"Количество смен знака (Массив 1): {signChanges}";

                }

                else if (rbArray2.Checked && arr2 != null)

                {

                    int signChanges = arr2.CountSignChanges();

                    lblResult.Text = $"Количество смен знака (Массив 2): {signChanges}";

                }

                else

                {

                    MessageBox.Show("Выберите массив и сгенерируйте его!");

                }

            }

            catch (MyException ex)

            {

                MessageBox.Show($"Ошибка: {ex.Message} Значение: {ex.Value}");

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Общая ошибка: {ex.Message}");

            }

        }

        private void PrintToDataGridView(Arr arr, DataGridView dgv)

        {

            dgv.RowCount = 1;

            dgv.ColumnCount = arr.Size;

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

                dgv.Rows[0].Cells[i].Value = arr[i];

        }

        private void UpdateInterface()

        {

            // Кнопки операций активны, если выбран массив и он сгенерирован

            bool arraySelected = (rbArray1.Checked && arr1 != null) || (rbArray2.Checked && arr2 != null);

            btnIncrement.Enabled = arraySelected;

            btnDecrement.Enabled = arraySelected;

            btnCountSignChanges.Enabled = arraySelected;

            // Кнопки для операций с двумя массивами активны, если оба массива сгенерированы

            btnAdd.Enabled = arr1 != null && arr2 != null;

            btnSubtract.Enabled = arr1 != null && arr2 != null;

        }

        private void rbArray1\_CheckedChanged(object sender, EventArgs e)

        {

            UpdateInterface();

        }

        private void rbArray2\_CheckedChanged(object sender, EventArgs e)

        {

            UpdateInterface();

        }

    }

}

**Протокол испытаний:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Фактические результаты** |
| Способность программы генерировать первый массив и отображать его в таблице и метке. | В поле txtSize1 ввести "5", нажать кнопку "Сгенерировать 1". | Программа создает массив размером 5, заполняет его случайными числами, отображает в dgvArray1. | Массив сгенерирован, отображен в dgvArray1. Соответствует ожиданиям (рисунок 3). |
| Способность программы генерировать второй массив и отображать его в таблице и метке. | В поле txtSize2 ввести "5", нажать кнопку "Сгенерировать 2". | Программа создает массив размером 3, заполняет его случайными числами, отображает в dgvArray2 | Массив сгенерирован, отображен в dgvArray2. Соответствует ожиданиям (рисунок 4). |
| Способность программы выполнять операцию инкремента для выбранного массива. | Выбрать rbArray2, нажать кнопку "Инкремент". | Элементы массива arr1 увеличиваются на 1. | Массив обновлен Соответствует ожиданиям (рисунок 5). |
| Способность программы выполнять операцию декремента для выбранного массива. | Выбрать rbArray2, нажать кнопку "Декремент". | Элементы массива arr2 уменьшаются на 1 | Массив обновлен. Соответствует ожиданиям (рисунок 6). |
| Способность программы выполнять сложение двух массивов с выводом результата в третью таблицу. | Сгенерировать оба массива, нажать кнопку "Сложить". | Новый массив, равный сумме arr1 и arr2, отображается в dgvResult и lblResult. | Результат отображен в dgvResult и lblResult. Соответствует ожиданиям (рисунок 7). |
| Способность программы подсчитывать количество смен знаков для выбранного массива. | Выбрать rbArray1, нажать кнопку "Смены знака". | Подсчитывается количество смен знаков в arr1, результат отображается в lblResult. | Результат в lblResult. Соответствует ожиданиям (рисунок 8). |
| Способность программы обрабатывать некорректный ввод размера массива. | В txtSize1 ввести "asf", нажать "Сгенерировать 1". | Появляется сообщение об ошибке: "Введите корректный размер для первого массива!". Массив не создается. | Сообщение об ошибке отображено, массив не создан. Соответствует ожиданиям (рисунок 9). |

**Результат испытаний:**

‬

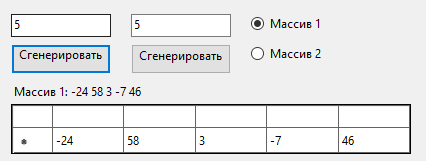


Рисунок 3 – первый массив

‬

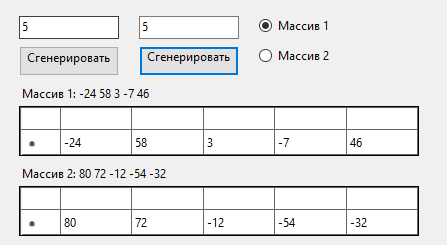


Рисунок 4 – второй массив

‬

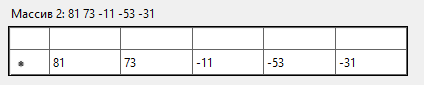


Рисунок 5 – инкремент

‬

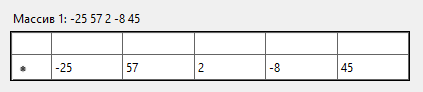


Рисунок 6 – декремент

‬

‬

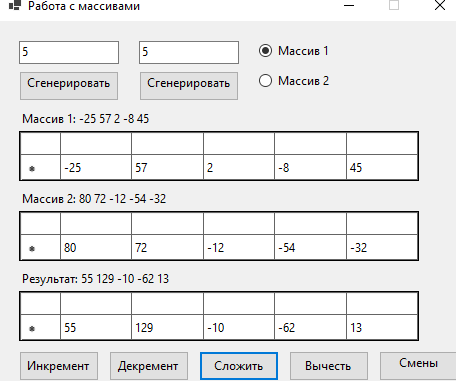


Рисунок 7 – сложение массивово

‬

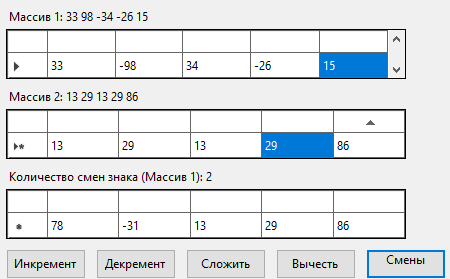


Рисунок 8 – смена знаков

‬

‬

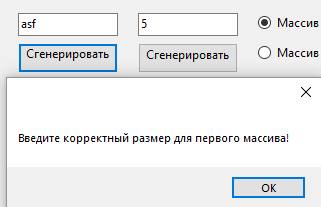


Рисунок 9 – некорректный массив