Министерство науки и ВЫСШЕГО образования   
Российской Федерации

Федеральное автономное образовательное учреждение высшего образования  
«Санкт-Петербургский государственный политехнический   
университет Петра Великого»  
**Институт среднего профессионального образования**

**Лабораторная работа № 11  
Тема:** Работа с однородными структурированными типами данных. Элементы пользовательского интерфейса по работе с массивами.

**Вариант 9**

Выполнил: студент группы 32919/8  
Крупина Елена

Проверил: преподаватель   
Прокофьев А.А.

Санкт-Петербург  
2024

**Цель работы:** получить практические навыки работы с компонентами для обработки

однородных структурированных данных: научится работать с компонентом

dataGridView для решения задач по обработке одномерных динамических массивов.

**Задание:**

1. Прочитайте свой вариант задания (см. ниже). Запишите в отчёт математическую

модель задачи, что дано и что надо найти, с указанием типов исходных данных и

результатов, и функциональные характеристики проекта.

1. Создайте подпрограмму для обработки массива по своему заданию,

выполняющую основные вычисления или преобразования в массиве. Метод

должен находится в дополнительном классе. В качестве параметра нужно

передать методу одномерный массив.

**Вариант 9**

Составить программу для ввода двух одномерных массивов А и В размерностью n

элементов и вычисления массива С размерностью n-1, каждый элемент которого

получен как сумма перемноженных пар соседних элементов исходных массивов:

ci = ai \* ai+1 + bi \* bi+1

***Таблица 1 – Условие***

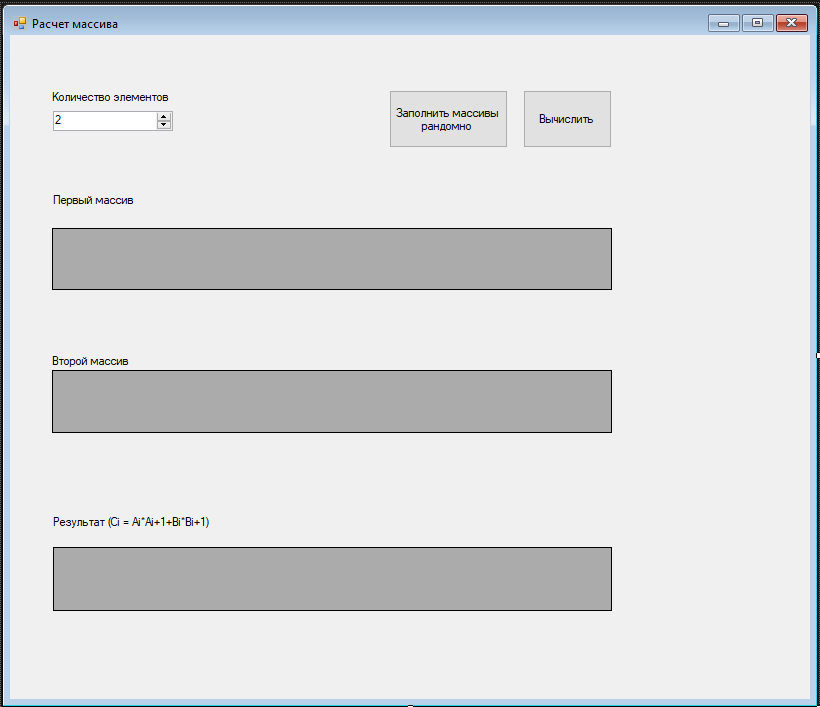
|  |  |  |
| --- | --- | --- |
| **Дано** | Два одномерных массива одинаковой размерности | List<double> |
| Размерность массивов | int |
| **Найти** | Размерность результирующего массива | int |
| Результирующий массив | List<double> |

**Созданная форма**

buttonCalc

rundomGenBtn

labelCount

****

labelRes

dataGridViewRes

dataGridViewSecond

labelSecond

dataGridViewFirst

labelFirst

numericUpDownCount

Рисунок 1 – Скриншот формы с надписанными компонентами

***Таблица 2 – Значения свойств, установленные на этапе тестирования***

| **Название компонента** | **Свойства** | **Значения** |
| --- | --- | --- |
| Form | Width | 816 |
| Height | 703 |
| Text | “Расчет массива” |
| labelCount | Text | “Количество элементов” |
| numericUpDownCount | Minimum | 2 |
| Maximum | 15 |
| labelFirst | Text | “Первый массив” |
| dataGridViewFirst | Width | 560 |
| Height | 62 |
| AllowDrop | False |
| AllowUserToAddRows | False |
| AllowUserToDeleteRows | False |
| AllowUserToResizeRows | False |
| labelSecond | Text | “Второй массив” |
| dataGridViewSecond | Width | 560 |
| Height | 62 |
| AllowDrop | False |
| AllowUserToAddRows | False |
| AllowUserToDeleteRows | False |
| AllowUserToResizeRows | False |
| labelRes | Text | “Результат (Ci = Ai\*Ai+1+Bi\*Bi+1)” |
| dataGridViewRes | Width | 560 |
|  | Height | 62 |
|  | AllowDrop | False |
|  | AllowUserToAddRows | False |
|  | AllowUserToDeleteRows | False |
|  | AllowUserToResizeRows | False |
|  | ReadOnly | True |

**Реализация функционала интерфейса формы**

using System;

using System.Linq;

using System.Windows.Forms;

namespace LAB11

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

dataGridViewFirst.AllowUserToAddRows = false;

dataGridViewFirst.AllowUserToDeleteRows = false;

dataGridViewFirst.AllowUserToResizeRows = false;

dataGridViewSecond.AllowUserToAddRows = false;

dataGridViewSecond.AllowUserToDeleteRows = false;

dataGridViewSecond.AllowUserToResizeRows = false;

dataGridViewRes.AllowUserToAddRows = false;

dataGridViewRes.AllowUserToDeleteRows = false;

dataGridViewRes.AllowUserToResizeRows = false;

dataGridViewFirst.RowCount = 2;

dataGridViewSecond.RowCount = 2;

dataGridViewRes.RowCount = 2;

dataGridViewFirst.ColumnCount = 2;

dataGridViewSecond.ColumnCount = 2;

dataGridViewRes.ColumnCount = 1;

dataGridViewFirst.Rows[0].ReadOnly = true;

dataGridViewSecond.Rows[0].ReadOnly = true;

dataGridViewRes.Rows[0].ReadOnly = true;

DataGridViewRow row = dataGridViewFirst.Rows[0];

for (int i = 0; i < numericUpDownCount.Value; i++)

{

row.Cells[i].Value = i;

}

row = dataGridViewSecond.Rows[0];

for (int i = 0; i < numericUpDownCount.Value; i++)

{

row.Cells[i].Value = i;

}

row = dataGridViewRes.Rows[0];

for (int i = 0; i < numericUpDownCount.Value - 1; i++)

{

row.Cells[i].Value = i;

}

row = dataGridViewFirst.Rows[1];

for (int i = 0; i < numericUpDownCount.Value; i++)

{

row.Cells[i].Value = 1;

}

row = dataGridViewSecond.Rows[1];

for (int i = 0; i < numericUpDownCount.Value; i++)

{

row.Cells[i].Value = 1;

}

row = dataGridViewRes.Rows[1];

for (int i = 0; i < numericUpDownCount.Value - 1; i++)

{

row.Cells[i].Value = 1;

}

}

private void numericUpDownCount\_ValueChanged(object sender, EventArgs e)

{

dataGridViewFirst.ColumnCount=(int)numericUpDownCount.Value;

dataGridViewSecond.ColumnCount=(int)numericUpDownCount.Value;

dataGridViewRes.ColumnCount=(int)numericUpDownCount.Value-1;

DataGridViewRow row = dataGridViewFirst.Rows[0];

for (int i = 0; i < numericUpDownCount.Value; i++)

{

row.Cells[i].Value = i;

}

row = dataGridViewSecond.Rows[0];

for (int i = 0; i < numericUpDownCount.Value; i++)

{

row.Cells[i].Value = i;

}

row = dataGridViewRes.Rows[0];

for (int i = 0; i < numericUpDownCount.Value-1; i++)

{

row.Cells[i].Value = i;

}

row = dataGridViewFirst.Rows[1];

for (int i = 0; i < numericUpDownCount.Value; i++)

{

row.Cells[i].Value = 1;

}

row = dataGridViewSecond.Rows[1];

for (int i = 0; i < numericUpDownCount.Value; i++)

{

row.Cells[i].Value = 1;

}

row = dataGridViewRes.Rows[1];

for (int i = 0; i < numericUpDownCount.Value - 1; i++)

{

row.Cells[i].Value = 1;

}

}

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

{

DataGridViewRow row = dataGridViewRes.Rows[1];

Mas masRes = new Mas();

masRes = masRes.FromRowToMas(row);

row = dataGridViewFirst.Rows[1];

Mas mas1 = new Mas();

mas1 = mas1.FromRowToMas(row);

row = dataGridViewSecond.Rows[1];

Mas mas2 = new Mas();

mas2 = mas2.FromRowToMas(row);

row = dataGridViewRes.Rows[1];

masRes = masRes.calcMas(mas1, mas2);

for (int i = 0; i < dataGridViewRes.ColumnCount; i++) {

row.Cells[i].Value = masRes[i];

}

}

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

{

DataGridViewRow row2 = dataGridViewFirst.Rows[1];

Mas mas = new Mas();

mas = mas.rndMas(dataGridViewFirst.ColumnCount);

for (int i = 0; i < dataGridViewFirst.ColumnCount; i++) {

row2.Cells[i].Value = mas.Matrix[i];

}

row2 = dataGridViewSecond.Rows[1];

Mas mas1 = new Mas();

mas1 = mas1.rndMas(dataGridViewSecond.ColumnCount);

for (int i = 0; i < dataGridViewSecond.ColumnCount; i++)

{

row2.Cells[i].Value = mas.Matrix[i];

}

}

private void dataGridViewFirst\_KeyPress(object sender, KeyPressEventArgs e)

{

string currentText = dataGridViewFirst.CurrentCell.Value.ToString();

if ((e.KeyChar >= '0' && e.KeyChar <= '9') || e.KeyChar == ',' || e.KeyChar == (char)Keys.Back)

{

if (e.KeyChar == ',' && currentText.Contains(","))

{

e.Handled = true;

}

}

else

{

e.Handled = true;

}

}

private void dataGridViewFirst\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

DataGridViewRow row = dataGridViewFirst.Rows[1];

for (int i = 0; i < row.Cells.Count; i++)

{

if (row.Cells[i].Value == null)

{

row.Cells[i].Value = 1;

}

}

}

void tb\_KeyPress(object sender, KeyPressEventArgs e)

{

TextBox tb = sender as TextBox;

// Проверяем, если нажатая клавиша - это не цифра, не минус и не запятая

if (!char.IsNumber(e.KeyChar) && e.KeyChar != '-' && e.KeyChar != ',')

{

// Разрешаем Backspace и Delete

if (e.KeyChar != (char)Keys.Back && e.KeyChar != (char)Keys.Delete)

{

e.Handled = true; // Запрещаем ввод

}

}

// Проверяем, если введен минус

if (e.KeyChar == '-')

{

// Если минус уже введен или он не первый символ, запрещаем ввод

if (tb.Text.Length > 0 || tb.Text.Contains('-'))

{

e.Handled = true;

}

}

// Проверяем, если введена запятая

if (e.KeyChar == ',')

{

// Если запятая уже введена или она первый символ, запрещаем ввод

if (tb.Text.Contains(',') || tb.Text.Length == 0 || tb.Text.Contains(",,"))

{

e.Handled = true;

}

}

}

private void dataGridViewFirst\_EditingControlShowing\_1(object sender, DataGridViewEditingControlShowingEventArgs e)

{

TextBox tb = (TextBox)e.Control;

tb.KeyPress -= tb\_KeyPress; // Удаляем предыдущий обработчик, чтобы избежать дублирования

tb.KeyPress += new KeyPressEventHandler(tb\_KeyPress); // Добавляем новый обработчик

}

private void dataGridViewSecond\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

DataGridViewRow row = dataGridViewSecond.Rows[1];

for (int i = 0; i < row.Cells.Count; i++)

{

if (row.Cells[i].Value == null)

{

row.Cells[i].Value = 1;

}

}

}

private void dataGridViewRes\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

DataGridViewRow row = dataGridViewRes.Rows[1];

for (int i = 0; i < row.Cells.Count; i++)

{

if (row.Cells[i].Value == null)

{

row.Cells[i].Value = 1;

}

}

}

private void dataGridViewRes\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

TextBox tb = (TextBox)e.Control;

tb.KeyPress -= tb\_KeyPress; // Удаляем предыдущий обработчик, чтобы избежать дублирования

tb.KeyPress += new KeyPressEventHandler(tb\_KeyPress); // Добавляем новый обработчик

}

private void dataGridViewSecond\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

TextBox tb = (TextBox)e.Control;

tb.KeyPress -= tb\_KeyPress; // Удаляем предыдущий обработчик, чтобы избежать дублирования

tb.KeyPress += new KeyPressEventHandler(tb\_KeyPress); // Добавляем новый обработчик

}

}

}

Для обработки одномерного массива была создана классовая структура.

**Код класса**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Linq.Expressions;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace LAB11

{

public class Mas

{

private List<double> matrix;

private static Random rnd= new Random();

public List<double> Matrix {

set { matrix = value; }

get { return matrix; }

}

public Mas() {

matrix = new List<double>();

}

public void addToMas(double item) {

matrix.Add(item);

}

public void removeMas(double item) {

matrix.Remove(item);

}

public Mas rndMas(int n)

{

Mas mas = new Mas();

Random rnd = new Random();

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

{

mas.addToMas(Math.Round(rnd.NextDouble()\*200-100,2));

}

return mas;

}

public double this[int index]

{

get { return matrix[index]; }

set { matrix[index] = value; }

}

public Mas calcMas(Mas mas1, Mas mas2) {

if (mas1.Matrix.Count != mas2.Matrix.Count) {

return mas1;

}

if (mas1.Matrix.Count == 0 || mas2.Matrix.Count == 0) {

return mas1;

}

Mas mas = new Mas();

int count= mas1.Matrix.Count-1;

for (int i = 0; i < count; i++) {

mas.addToMas(mas1[i] \* mas1[i + 1] + mas2[i] \* mas2[i + 1]);

}

return mas;

}

public void addRange(Mas mas, double[] matrix) {

mas.matrix.AddRange(matrix);

}

public Mas FromRowToMas(DataGridViewRow row) {

Mas mas = new Mas();

for (int i = 0; i < row.Cells.Count; i++)

{

string rowValue = row.Cells[i].Value?.ToString();

if (rowValue == null)

{

MessageBox.Show("Все ячейки должны быть заполнены числами");

break;

}

Debug.Write(rowValue + " ");

try

{

if (double.TryParse(rowValue, out double value))

{

mas.addToMas(Convert.ToDouble(rowValue));

}

else { throw new FormatException(); }

}

catch { MessageBox.Show("Все ячейки должны быть заполнены числами"); break; }

}

Debug.WriteLine("");

return mas;

}

}

}

**Код метода обработки одномерного массива**

public Mas calcMas(Mas mas1, Mas mas2) {

if (mas1.Matrix.Count != mas2.Matrix.Count) {

return mas1;

}

if (mas1.Matrix.Count == 0 || mas2.Matrix.Count == 0) {

return mas1;

}

Mas mas = new Mas();

int count= mas1.Matrix.Count-1;

for (int i = 0; i < count; i++) {

mas.addToMas(mas1[i] \* mas1[i + 1] + mas2[i] \* mas2[i + 1]);

}

return mas;

}

**Результат**

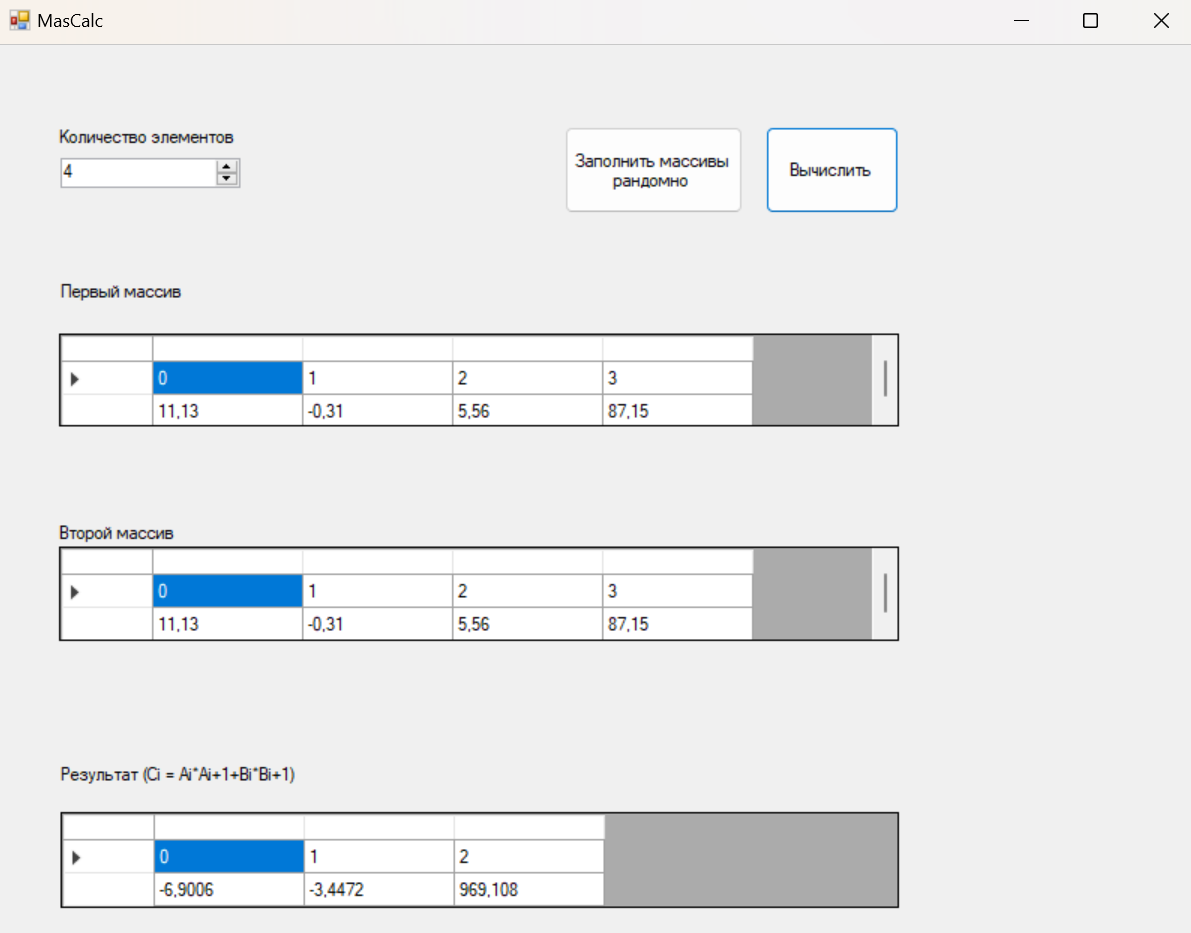


Рисунок 2 - Результат

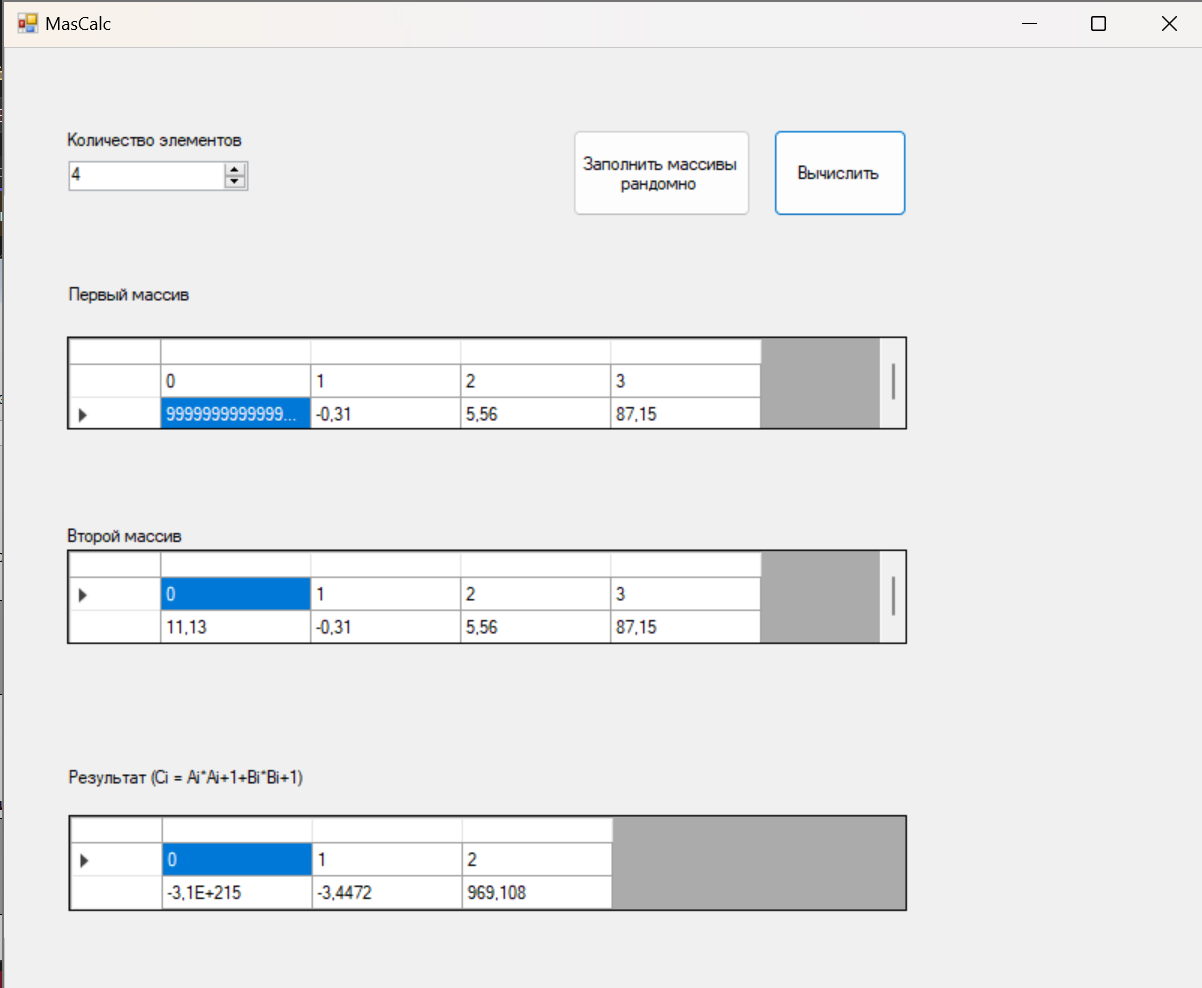


Рисунок 3 - Результат

**Тестирование**

**Код тестируемой функции**

public Mas calcMas(Mas mas1, Mas mas2)

{

if (mas1.Matrix.Count != mas2.Matrix.Count)

{

return mas1;

}

if (mas1.Matrix.Count == 0 || mas2.Matrix.Count == 0)

{

return mas1;

}

Mas mas = new Mas();

int count = mas1.Matrix.Count - 1;

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

{

mas.addToMas(mas1[i] \* mas1[i + 1] + mas2[i] \* mas2[i + 1]);

}

return mas;

**}**

**Код тестов**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using LAB11;

namespace LAB11

{

[TestClass]

public class UnitTest

{

[TestMethod]

public void TestCalculationValues()

{

//double[] exMas1 = { -58.4, 45.5, 18.5, 31.17 };

//double[] exMas2 = { 24.98, 35.62, 22, 7.04 };

double[] values1 = { -58.4, 45.5, 18.5, 31.17 };

double[] values2 = { 24.98, 35.62, 22, 7.04 };

double[] values3 = { -1767.4123999999997, 1625.3899999999999, 731.525 };

Mas mas1 = new Mas();

Mas mas2 = new Mas();

Mas expectedMas = new Mas();

mas1.addRange(mas1, values1);

mas2.addRange(mas2, values2);

expectedMas.addRange(expectedMas, values3);

Mas actualMas = new Mas();

actualMas = actualMas.calcMas(mas1, mas2);

CollectionAssert.AreEqual(expectedMas.Matrix, actualMas.Matrix, "Ожидаемые значения не получены");

}

}

[TestClass]

public class UnitTest2

{

[TestMethod]

public void TestEmptyMas()

{

Mas mas1 = new Mas();

Mas mas2 = new Mas();

Mas expectedMas = new Mas();

Mas actualMas = new Mas();

actualMas = actualMas.calcMas(mas1, mas2);

CollectionAssert.AreEqual(expectedMas.Matrix, actualMas.Matrix, "Ожидаемые значения не получены");

}

}

[TestClass]

public class UnitTest3

{

[TestMethod]

public void TestNotEqualCount()

{

Mas mas1 = new Mas();

Mas mas2 = new Mas();

mas1 = mas1.rndMas(2);

mas2 = mas2.rndMas(5);

Mas actualMas = new Mas();

actualMas = actualMas.calcMas(mas1, mas2);

CollectionAssert.AreEqual(mas1.Matrix, actualMas.Matrix, "Ожидаемые значения не получены");

}

}

}

**Результат тестирования**

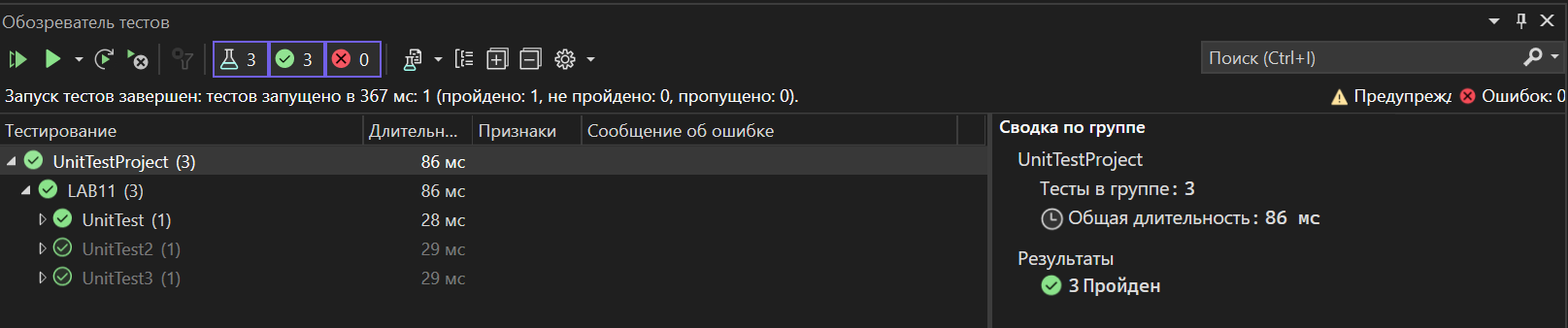
****

Рисунок 4 – Результат тестирования