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 № 12  
Тема:** Разработка проектов обработки двумерных массивов

с использованием компонента таблица.

**Вариант 9**

Выполнил: студент группы 32919/8  
Крупина Елена

Проверил: преподаватель   
Прокофьев А.А.

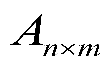
Санкт-Петербург  
2024

**Цель работы** получить практические навыки разработки проектов обработки двумерных массивов с использованием компонента таблица.

**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.
2. Разработайте подпрограмму, выполняющую обработку двумерного массива по вашему заданию. На вход методу необходимо подать двумерный массив. Метод также может производить перестановку элементов исходного массива или формирование другого массива. Передача результатов работы дополнительного метода должна происходить только через выходные параметры или return, без обращения к компонентам формы. Метод должен находится в дополнительном классе.
3. Подготовьте юнит-тесты для тестирования собственного метода. В одном тестовом проекте могут находиться несколько юнит-тестов, позволяющих проверить работу метода при различных входных данных и рассмотреть разные ситуации, в том числе, с исключениями – негативный тест. Негативный тест надо составить таким образом, чтобы проверялось полученное исключение, а сам тест должен быть пройден.
4. Добавьте в отчёт юнит-тесты и скриншот успешного тестирования.
5. Продумайте, какие элементы интерфейса необходимо разместить на форме, чтобы обеспечить удобную и наглядную работу пользователя.
6. Выполните настройку свойств компонента dataGridView. Так как массив двумерный, установим по умолчанию количество строк таблицы 5, количество колонок 5. Фиксированные строки и столбцы нам не нужны, установим свойствам RowHeadersVisible, ColumnHeadersVisible значение false. Также необходимо установить свойство AllowUserToAddRows в значение false, чтобы пользователь не мог добавлять строки в режиме ручного ввода в таблицу.
7. Занесите в отчёт в специальную таблицу, у каких компонентов вы изменяли свойства, и какое значение получило теперь каждое свойство.
8. Внесём изменения в проект таким образом, чтобы пользователь мог менять размерность массива. Для этого добавьте на форму два поля ввода с кнопками-переключателями numdericUpDown. При изменении значения поля numbericUpDown1 будет устанавливаться количество строк массива, соответствующее количество строк таблицы. При изменении значения поля numbericUpDown2 будет устанавливаться количество столбцов массива, соответствующее количество колонок таблицы.
9. При переключении трёх кнопок-переключателей radioButton будет меняться способ ввода массива. Создайте процедуру обработки события щелчка по кнопке. При выборе пользователем кнопки «Вручную» таблица на форме должна заполняться нулями. При выборе пользователем кнопки «Случайно» таблица на форме должна заполняться случайными числами в интервале [-100; 100]. Внесите необходимые команды в тело методов.
10. При выборе кнопки «Из файла» должно вызываться диалоговое окно openFileDialog и данные из текстового файла загружаться в таблицу на форме. Размеры таблицы в этом случае определяются по количеству данных в файле. Формат файла: числа идут через пробел, каждая строка массива с новой строки.
11. Выполнение задания в программе будет происходить при нажатии на кнопку «Вычислить». Создайте соответствующий метод обработки события. При нажатии на кнопку «Вычислить» значения из таблицы должны считываться в двумерный массив соответствующего размера, а затем массив должен передаваться в метод, вычисляющий то, что требуется в задании.
12. Запустите проект на компиляцию. При необходимости исправьте синтаксические ошибки, выявленные на этапе компиляции.
13. Запустите проект на выполнение. Протестируйте программу. Отследите, какие ошибки выполнения возникают при недопустимых исходных данных, например, если пользователь оставил одну или несколько ячеек таблицы пустыми, ввел в ячейку недопустимые символы.
14. Для проверки нажатых клавиш при ручном вводе в таблицу понадобиться 2 метода: метод редактирования каждой ячейки (используйте событие EditingControlShowing) и вызываемое в нём событие нажатия и отпускания клавиши клавиатуры. Добавьте методы проверки правильности ввода данных для таблицы.
15. Запустите проект на компиляцию, а затем на выполнение. Протестируйте программу при различных значениях размерности массива, в режиме ручного, случайного ввода и ввода из файла. Скопируйте в отчёт скриншоты различных вариантов выполнения программы.
16. Добавьте комментарии в код программы. Комментируйте назначение каждого метода и смысл переменных, а в сложных случаях - ход решения задачи. Перенесите в отчёт текст модуля формы с комментариями.

**Вариант 9**

Дан вещественный массив . Столбец, содержащий максимальный элемент, поменять местами с первым столбцом.

**Математическая модель задачи**

***Таблица 1 – Условие***

|  |  |  |
| --- | --- | --- |
| **Дано** | Вещественный массив | double [,] |
| **Найти** | Максимальный элемент массива | double |
| Результирующий массив | double [,] |

**Классовая структура**

**Код класса двумерного массива**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace LAB12

{

public class TwoDefArr

{

private double[,] array;

private int rowCount;

private int columnCount;

public int RowCount {

get { return rowCount; }

set { rowCount = value; }

}

public int ColumnCount {

get { return columnCount; }

set { columnCount = value; }

}

public double[,] Array { get { return array; } }

public TwoDefArr(int rowCount, int columnCount) {

this.rowCount = rowCount;

this.columnCount = columnCount;

array = new double[rowCount, columnCount];

}

public TwoDefArr rndArray(int rowCount, int columnCount) {

TwoDefArr arr = new TwoDefArr(rowCount, columnCount);

Random rnd = new Random();

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

{

for (int j = 0; j < columnCount; j++)

{

arr.array[i, j] = Math.Round(rnd.NextDouble()\*200-100, 2);

}

}

return arr;

}

public TwoDefArr fromDataGridViewToTwoDefArr(DataGridView dataGridViewArray)

{

TwoDefArr arrNew = new TwoDefArr(dataGridViewArray.RowCount, dataGridViewArray.ColumnCount);

for (int i = 0; i < dataGridViewArray.RowCount; i++)

{

for (int j = 0; j < dataGridViewArray.ColumnCount; j++)

{

var cellValue = dataGridViewArray.Rows[i].Cells[j].Value;

if (cellValue != null && double.TryParse(cellValue.ToString(), out double result))

{

arrNew.array[i, j] = result;

}

else

{

arrNew.array[i, j] = 0;

}

}

}

return arrNew;

}

public int maxColumnNum(TwoDefArr arr)

{

int maxColumnNum = 0;

double maxElement = -101;

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

{

for (int j = 0; j < arr.ColumnCount; j++)

{

if (arr.Array[i, j] > maxElement)

{

maxElement = arr.Array[i, j];

maxColumnNum = j;

}

}

}

return maxColumnNum;

}

public TwoDefArr swapMaxAndFirst(TwoDefArr initialArr)

{

int maxColumnNum = initialArr.maxColumnNum(initialArr);

if (maxColumnNum == 0)

{

return initialArr;

}

else

{

for (int i = 0; i < initialArr.RowCount; i++)

{

var tempValue = initialArr.Array[i, 0];

var maxColumnValue = initialArr.Array[i, maxColumnNum];

if (tempValue is double tempDouble && maxColumnValue is double maxColumnDouble)

{

initialArr.Array[i, 0] = maxColumnDouble;

initialArr.Array[i, maxColumnNum] = tempDouble;

}

else

{

throw new Exception("Элементы массива не являются типом double.");

}

}

return initialArr;

}

}

}

}

**Тестирование**

**Код тестируемой функции**

public TwoDefArr swapMaxAndFirst(TwoDefArr initialArr)

{

int maxColumnNum = initialArr.maxColumnNum(initialArr);

if (maxColumnNum == 0)

{

return initialArr;

}

else

{

for (int i = 0; i < initialArr.RowCount; i++)

{

var tempValue = initialArr.Array[i, 0];

var maxColumnValue = initialArr.Array[i, maxColumnNum];

if (tempValue is double tempDouble && maxColumnValue is double maxColumnDouble)

{

initialArr.Array[i, 0] = maxColumnDouble;

initialArr.Array[i, maxColumnNum] = tempDouble;

}

else

{

throw new Exception("Элементы массива не являются типом double.");

}

}

return initialArr;

}

}

**Код юнит-тестов**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

namespace LAB12

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void IsTrueSwapping()

{

TwoDefArr initialArr = new TwoDefArr(3, 3);

double[,] initial = new double[,]

{

{1, 1, 99},

{1, 1, 99},

{1, 1, 99}

};

for (int i = 0; i<initialArr.RowCount; i++)

{

for (int j = 0; j < initialArr.ColumnCount; j++) {

initialArr.Array[i, j] = initial[i, j];

}

}

TwoDefArr resArr = new TwoDefArr(3, 3);

resArr = initialArr.swapMaxAndFirst(initialArr);

double[,] expectedArr = new double[,]

{

{99, 1, 1},

{99, 1, 1},

{99, 1, 1}

};

CollectionAssert.AreEqual(expectedArr, resArr.Array,"Результат отличен от ожидаемого");

}

[TestMethod]

public void MaxAtFirstRow() {

TwoDefArr initialArr = new TwoDefArr(3, 3);

double[,] initial = new double[,]

{

{99, 1, 1},

{99, 1, 1},

{99, 1, 1}

};

for (int i = 0; i < initialArr.RowCount; i++)

{

for (int j = 0; j < initialArr.ColumnCount; j++)

{

initialArr.Array[i, j] = initial[i, j];

}

}

TwoDefArr resArr = new TwoDefArr(3, 3);

resArr = initialArr.swapMaxAndFirst(initialArr);

double[,] expectedArr = new double[,]

{

{99, 1, 1},

{99, 1, 1},

{99, 1, 1}

};

CollectionAssert.AreEqual(expectedArr, resArr.Array, "Результат отличен от ожидаемого");

}

[TestMethod]

public void SquareArrayWithMaxInSecondAndThird()

{

TwoDefArr initialArr = new TwoDefArr(3, 3);

double[,] initial = new double[,]

{

{1, 99, 1},

{1, 1, 1},

{1, 1, 99}

};

for (int i = 0; i < initialArr.RowCount; i++)

{

for (int j = 0; j < initialArr.ColumnCount; j++)

{

initialArr.Array[i, j] = initial[i, j];

}

}

TwoDefArr resArr = new TwoDefArr(3, 3);

resArr = initialArr.swapMaxAndFirst(initialArr);

double[,] expectedArr = new double[,]

{

{99, 1, 1},

{1, 1, 1},

{1, 1, 99}

};

CollectionAssert.AreEqual(expectedArr, resArr.Array, "Результат отличен от ожидаемого");

}

[TestMethod]

public void NullArray()

{

TwoDefArr initialArr = new TwoDefArr(0, 0);

double[,] initial = new double[0,0];

for (int i = 0; i < initialArr.RowCount; i++)

{

for (int j = 0; j < initialArr.ColumnCount; j++)

{

initialArr.Array[i, j] = initial[i, j];

}

}

TwoDefArr resArr = new TwoDefArr(0, 0);

resArr = initialArr.swapMaxAndFirst(initialArr);

double[,] expectedArr = new double[0, 0];

CollectionAssert.AreEqual(expectedArr, resArr.Array, "Результат отличен от ожидаемого");

}

}

}

**Результаты тестирования**

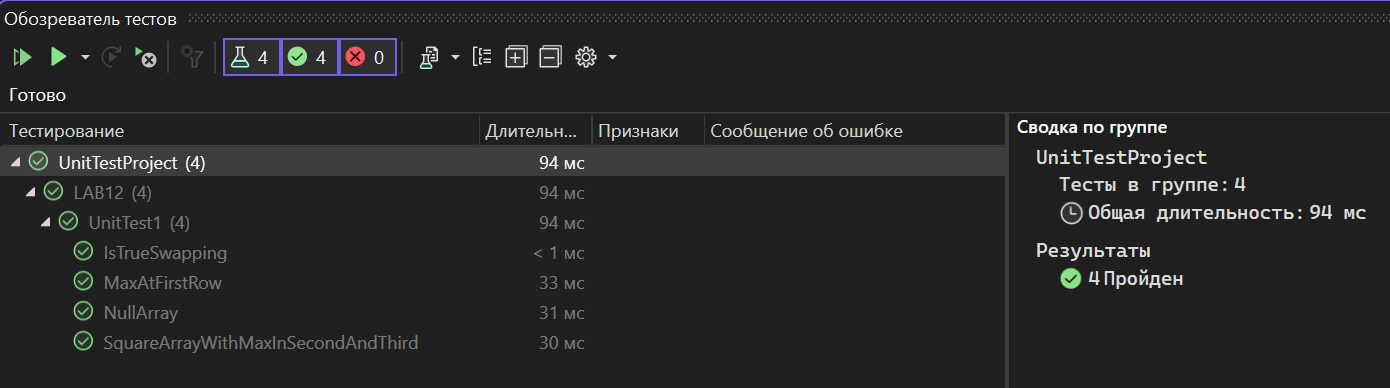
****

Рисунок 1 – Результаты тестирования

**Разработка интерфейса**

**Конструирование формы**

Таблица 1 – Значения свойств, установленные на этапе конструирования интрфейса

| **Название компонента** | **Свойства** | **Значения** |
| --- | --- | --- |
| labelNumOfRows | Text | “Количество строк” |
| labelNumOfCols | Text | “Количество столбцов” |
| labelTableRes | Text | “Полученная таблица” |
| numericUpDownNumOfRows | Maximum | 10 |
| Minimum | 1 |
| numericUpDownNumOfCols | Maximum | 10 |
| Minimum | 1 |
| buttonRndFiling | Text | “Заполнить таблицу рандомно” |
| buttonFileFiling | Text | “Заполнить таблицу из файла” |
| buttonCalc | Text | “Вычислить” |
| dataGridViewInitial | AllowUserToAddRows | False |
| AllowUserToDeleteRows | False |
| AllowUderToOrderColumns | False |
| AllowUserToResizeColumns | False |
| AllowUserToResizeRows | False |
| ColumnHeadersVisible | True |
| RowHeadersVisible | True |
| ColumnHeadersVisible | True |
| dataGridViewRes | AllowUserToAddRows | False |
| AllowUserToDeleteRows | False |
| AllowUderToOrderColumns | False |
| AllowUserToResizeColumns | False |
| AllowUserToResizeRows | False |
| ColumnHeadersVisible | True |
| RowHeadersVisible | True |
| ColumnHeadersVisible | True |

**Интерфейс формы**

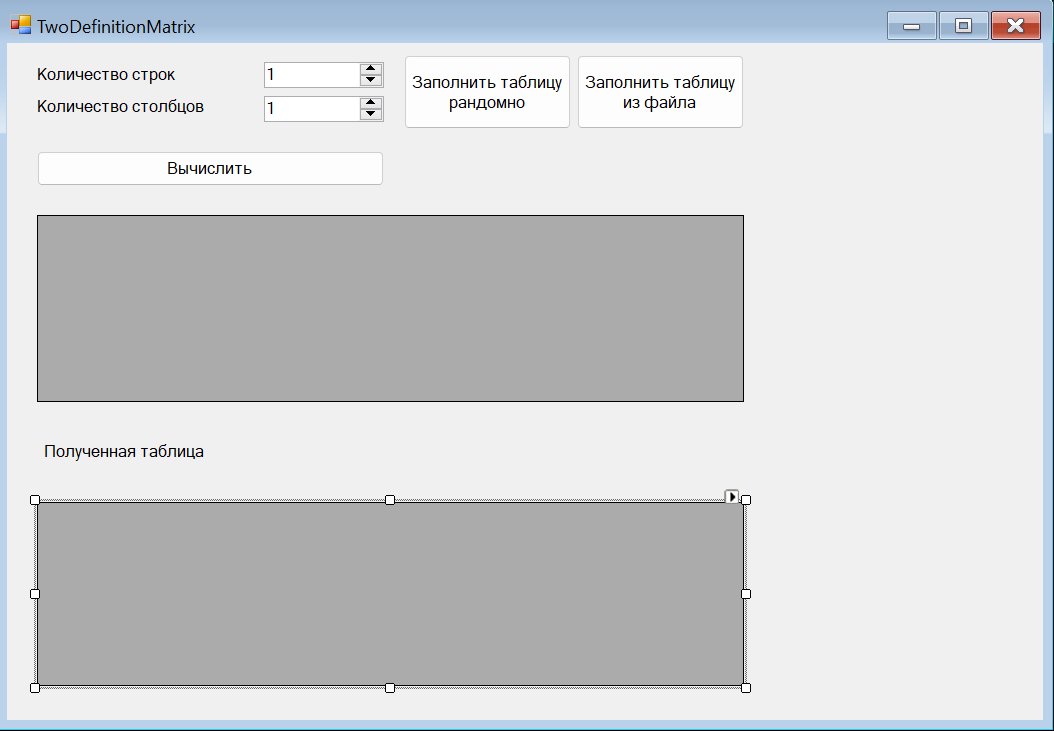
numericUpDownNumOfCols

numericUpDownNumOfRows

buttonFileFiling

buttonRndFiling

labelNumOfRows



dataGridViewRes

labelTableRes

dataGridViewInitial

buttonCalc

labelNumOfCols

Рисунок 2 – Скриншот формы с надписанными компонентами

**Реализация функционала формы**

**Код**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace LAB12

{

public partial class TwoDefinitionMatrix : Form

{

public TwoDefinitionMatrix()

{

InitializeComponent();

//Дефолтные значения

dataGridViewInitial.RowCount = 5;

dataGridViewInitial.ColumnCount = 5;

dataGridViewRes.RowCount = 5;

dataGridViewRes.ColumnCount = 5;

numericUpDownNumOfCols.Value = 5;

numericUpDownNumOfRows.Value = 5;

dataGridViewInitial.ColumnHeadersVisible = true;

dataGridViewInitial.RowHeadersVisible = true;

dataGridViewRes.RowHeadersVisible = true;

dataGridViewRes.ColumnHeadersVisible = true;

SetDefaultValues(dataGridViewInitial);

SetDefaultValues(dataGridViewRes);

}

private void SetDefaultValues(DataGridView dgv)

{

for (int i = 0; i < dgv.RowCount; i++)

{

for (int j = 0; j < dgv.ColumnCount; j++)

{

dgv[j, i].Value = 1;

}

}

for (int i = 0; i < dgv.RowCount; i++)

{

dgv.Rows[i].HeaderCell.Value = i.ToString();

}

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

{

dgv.Columns[i].HeaderCell.Value = i.ToString();

}

}

//Контроль размерности

//Строки

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

{

dataGridViewInitial.RowCount = (int)numericUpDownNumOfRows.Value;

dataGridViewRes.RowCount = (int)numericUpDownNumOfRows.Value;

}

//Столбцы

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

{

dataGridViewInitial.ColumnCount=(int)numericUpDownNumOfCols.Value;

dataGridViewRes.ColumnCount=(int)numericUpDownNumOfCols.Value;

}

//Дефолтные значения

//Для изначального

private void dataGridViewInitial\_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)

{

SetDefaultValues(dataGridViewInitial);

SetDefaultValues(dataGridViewRes);

}

private void dataGridViewInitial\_ColumnAdded(object sender, DataGridViewColumnEventArgs e)

{

SetDefaultValues(dataGridViewInitial);

SetDefaultValues(dataGridViewRes);

}

//Для результирующего

private void dataGridViewRes\_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)

{

SetDefaultValues(dataGridViewInitial);

SetDefaultValues(dataGridViewRes);

}

private void dataGridViewRes\_ColumnAdded(object sender, DataGridViewColumnEventArgs e)

{

SetDefaultValues(dataGridViewInitial);

SetDefaultValues(dataGridViewRes);

}

//Запрет ввода фигни для изначальной таблицы

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 dataGridViewInitial\_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

}

//Рандомное заполнение массива

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

{

int rowCount = (int)numericUpDownNumOfRows.Value;

int colCount = (int)numericUpDownNumOfCols.Value;

TwoDefArr arr = new TwoDefArr(rowCount, colCount);

arr = arr.rndArray(rowCount, colCount);

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

{

for (int j = 0; j < colCount; j++)

{

dataGridViewInitial.Rows[i].Cells[j].Value = arr.Array[i, j];

}

}

}

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

{

TwoDefArr arr = new TwoDefArr(0, 0);

arr = arr.fromDataGridViewToTwoDefArr(dataGridViewInitial);

TwoDefArr arrRes = new TwoDefArr(0, 0);

arrRes = arr.swapMaxAndFirst(arr);

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

{

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

dataGridViewRes.Rows[i].Cells[j].Value = arrRes.Array[i, j];

}

}

}

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

{

//string[] ReadAllLines(string path) / Task<string[]> ReadAllLinesAsync(string path, CancellationToken cancellationToken)

OpenFileDialog ofd = new OpenFileDialog();

ofd.Filter = "Text files (\*.txt)|\*.txt";

ofd.Title = "Выберите файл";

string[] lines;

if (ofd.ShowDialog() == DialogResult.OK) {

lines = File.ReadAllLines(ofd.FileName);

}

else {lines = null; return; }

int rowCount = lines.Length;

List<string[]> linesList = new List<string[]>();

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

{

string[] elements = lines[i].Split(new char[] { ' ' });

linesList.Add(elements);

}

int columnCount = 0;

foreach (var arr in linesList)

{

if (arr.Length > columnCount)

{

columnCount = arr.Length;

}

}

double[,] resultArray = new double[rowCount, columnCount];

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

{

for (int j = 0; j < linesList[i].Length; j++)

{

if (double.TryParse(linesList[i][j], out double value))

{

resultArray[i, j] = value;

}

else

{

resultArray[i, j] = 0;

MessageBox.Show("Не все элементы в файле оказались числами. Элементы в недопустимом формате были заменены значением ноль.");

}

}

}

dataGridViewInitial.RowCount = rowCount;

dataGridViewInitial.ColumnCount = columnCount;

dataGridViewRes.RowCount = rowCount;

dataGridViewRes.ColumnCount = columnCount;

numericUpDownNumOfRows.Value = rowCount;

numericUpDownNumOfCols.Value = columnCount;

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

{

for (int j = 0; j < columnCount; j++) {

dataGridViewInitial.Rows[i].Cells[j].Value = resultArray[i,j];

}

}

}

}

}

**Тестовые запуски**

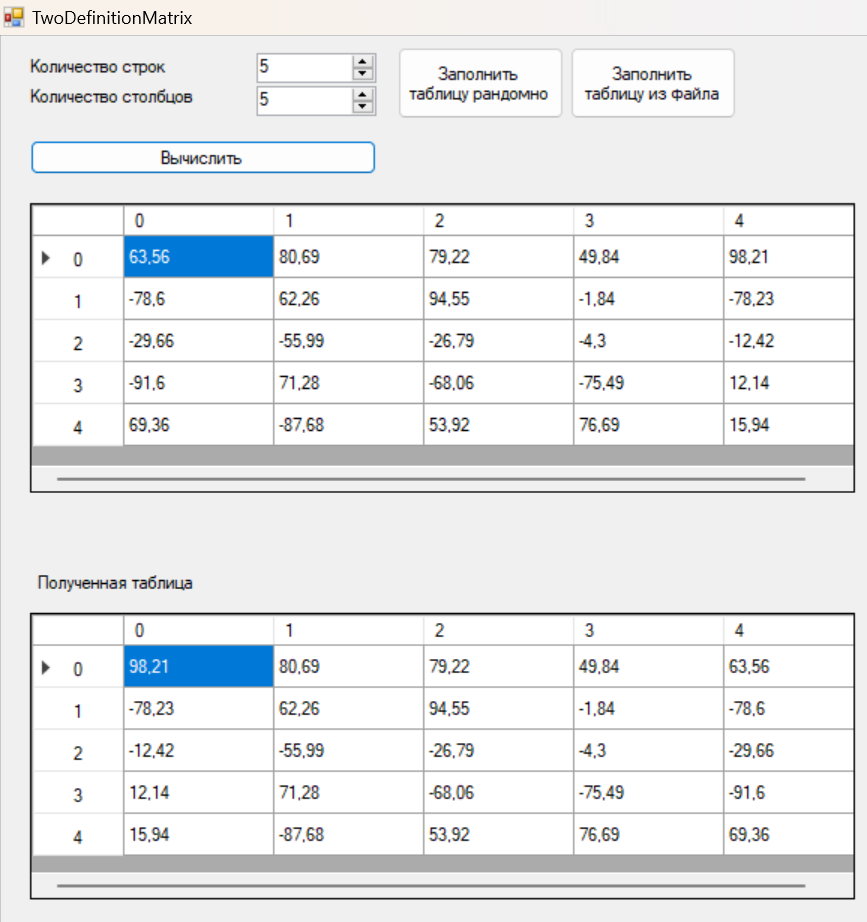
****

Рисунок 3 – Рандомное заполнение таблицы

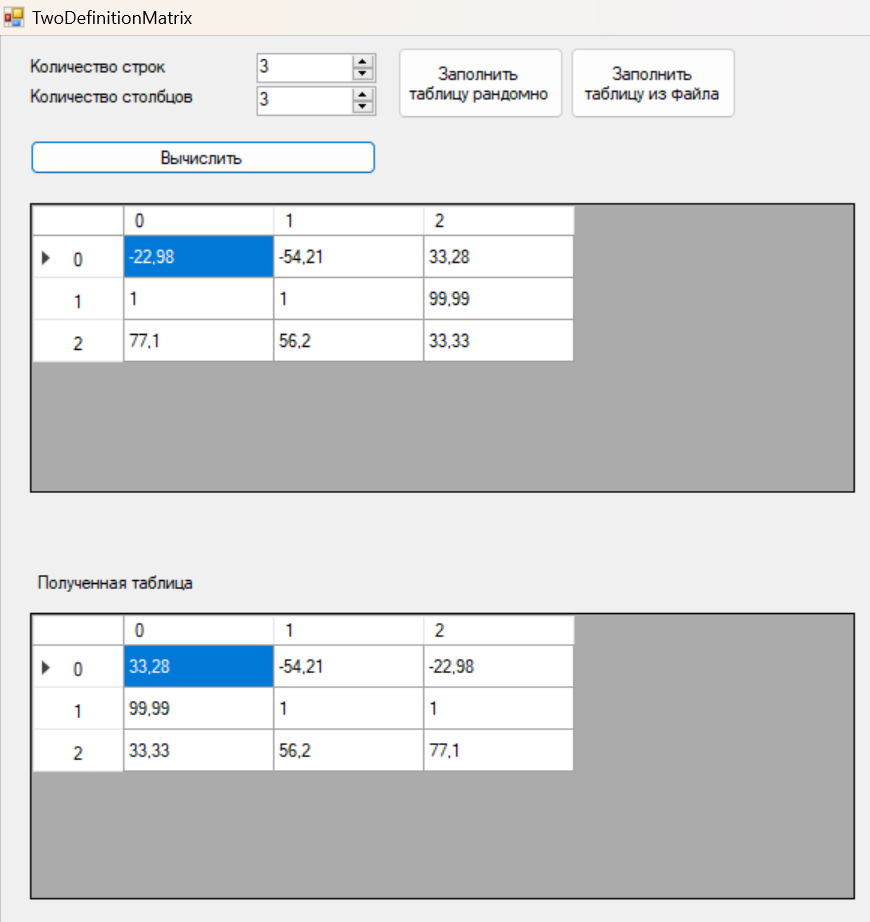


Рисунок 4 – Заполнение таблицы из файла

Файл для заполнения приложен к проекту (array.txt).