# Тема 6: «Обработка двумерных массивов»

**Цель работы:** Получить практические навыки разработки проектов по обработке двумерных массивов.

**Задание 1. Тестирование и использование средств профилирования**

Разработать собственные методы по обработке массива в соответствие с вашим вариантом. Методы поместить в дополнительный класс.

Создайте тест-кейсы для тестирования собственных методов.

Подготовьте контрольные примеры. Для тестирования одного метода может понадобится несколько тестов. Обязательно отдельно тестируйте исключения, которые могут возникнуть в процессе выполнения вашего метода.

**Задание 2. Основной проект**

Выполнить конструирование пользовательского интерфейса для организации ввода двумерного массива двумя способами: случайным образом или вручную. Отладить программу, выполняющую ввод массива двумя способами. Размер массива запрашивается у пользователя. Для работы с массивом надо предусмотреть на форме таблицу, размер которой может задаваться в процессе выполнения программы. Размеры таблицы должны изменяться интерактивно при каждом изменении значений количества строк и столбцов. Заполнение массива может происходить при помощи генератора случайных чисел либо с клавиатуры пользователем. Для выбора режима заполнения используйте переключатели «зависимые кнопки выбора».

**Задание 3. Сервисные функции**

Добавьте в проект проверку нажатых клавиш в режиме ручного ввода.

Дополните проект таким образом, чтобы в таблице окрашивались некоторые ячейки (строки, столбцы), чтобы выделить те значения, которые нужно было найти или которые изменялись в соответствие с вашим вариантом.

**Вариант 10**

Заполнить двумерный числовой массив целыми числами. Найти и вывести на экран максимальные элементы каждой строки массива, и исключить из массива строки, где максимальный элемент стоит в конце строки.

**Выполнение задания**

1. **Проектирование пользовательского интерфейса**

Внешний вид проекта представлен на рисунке 1.1, значения изменённых свойств компонентов представлены в таблице 2.1.



**Рис. 1.1. Внешний вид проекта на этапе конструирования интерфейса**

1. **Свойства компонентов формы**

Таблица 2.1. Значения свойств, установленные на этапе конструирования интерфейса формы Form1

| **Название компонента** | **Свойства** | **Значения** |
| --- | --- | --- |
| Form1 | FormBorderStyle | FixedSingle |
| Text | «Поиск максимальных» |
| MaximizeBox | False |
| MinimizeBox | False |
| numericUpDown1 | Maximum | 25 |
| Minimum | 1 |
| numericUpDown2 | Maximum | 25 |
| Minimum | 1 |
| radioButton1 | Checked | True |
| button2 | text | «Заполнить» |
| Enabled | False |
| button1 | text | «Вычислить» |
| visible | False |
| Enabled | False |
| label1 | text | «Строк» |
| label2 | text | «Столбцов» |
| label3 | text | «Диапазон от» |
| visible | False |
| label4 | text | до |
| visible | False |
| label5 | text | «Итоговый масив:» |
| label6 | text | «Входные данные:» |
| label7 | text | «» |
| label8 | text | «» |
| groupBox1 | text | Ввод массива |
| textBox1 | visible | False |
| textBox2 | visible | False |
| dataGridView1 | AllowUserToAddRows | False |
| AllowUserToDeleteRows | False |
| AllowUserToResizeColumns | False |
| AllowUserToResizeRows | False |
| RowHeaderWidthSizeMode | DisableResizing |
| dataGridView2 | AllowUserToAddRows | False |
| AllowUserToDeleteRows | False |
| AllowUserToResizeColumns | False |
| AllowUserToResizeRows | False |
| Readonly | True |
| RowHeaderWidthSizeMode | DisableResizing |

1. **Сценарий использования программы**
   1. Запуск программы
   2. Изменение количества столбцов и строк
   3. Выбор метода ввода значений
   4. Ввод значений
   5. Нажатие кнопки «Вычислить»
   6. Возможность вернуться к пунктам 3.2-3.5
   7. Завершение работы программы
2. **Код программы**

**Код модуля ArrayToDGV.cs**

using System;

using System.Windows.Forms;

namespace Task6

{

public class ArrayToDGV

{

/// <summary>

/// Метод для преобразования двумерного масива целых чисел в таблицу DataGridView

/// </summary>

/// <param name="arr"> Массив, который необходимо преобразовать в таблицу</param>

/// <param name="dgv"> Таблица DataGridView, в которую будет записан массив</param>

static public void ConvertToDGV (int[,] arr, DataGridView dgv)

{

try

{

// Задание размеров таблицы

dgv.RowCount = arr.GetLength(0);

dgv.ColumnCount = arr.GetLength(1);

// Цикл для прохода по всем элементам

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

{

for (int j = 0; j < arr.GetLength(1); j++)

{

dgv[j, i].Value = arr[i, j];

}

}

}

catch (Exception)

{

// Исключение преобразования

throw new Exception("Не удалось конвертировать двумерный массив в таблицу!");

}

}

/// <summary>

/// Метод для преобразования таблицы DataGridView в целочисленный двумерный массив

/// </summary>

/// <param name="dgv"> Таблица DataGridView, из которой необходимо достать массив </param>

/// <returns> Двумерный целочисленный массив, полученный из таблицы</returns>

static public int[,] ConvertToArray (DataGridView dgv)

{

try

{

// Возвращаемый двумерный массив

int[,] result = new int[dgv.RowCount, dgv.ColumnCount];

// Цикл для прохода по всем элементам таблицы

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

{

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

{

result[i, j] = Convert.ToInt32(dgv[j, i].Value);

}

}

return result;

}

catch (Exception)

{

// Исключение преобразования

throw new Exception("Не удалось конвертировать таблицу в двумерный массив!");

}

}

}

}

**Код модуля Form1.cs**

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task6

{

public partial class Form1:Form

{

// Начальный размер колонок и формы

const int columnwidth = 30;

Size startSize;

// Конструктор

public Form1()

{

InitializeComponent();

// Добавление строки и столбца в таблицу

dataGridView1.Columns.Add("c1",$"{dataGridView1.ColumnCount + 1}");

dataGridView1.Columns[dataGridView1.ColumnCount - 1].SortMode = DataGridViewColumnSortMode.NotSortable;

dataGridView1.Columns[dataGridView1.ColumnCount - 1].Width = columnwidth;

dataGridView1.Rows.Add();

dataGridView1.Rows[dataGridView1.RowCount - 1].HeaderCell.Value = dataGridView1.RowCount.ToString();

// Изменение размера таблицы

ResizeDGV(dataGridView1);

// Сохранение начального размера формы в переменной

startSize = Size;

}

// Очистка оформления ячеек таблицы

void ClearDGVStyle()

{

for(int i = 0;i < dataGridView1.ColumnCount;i++)

{

for(int j = 0;j < dataGridView1.RowCount;j++)

{

// Установка цвета по умолчанию

dataGridView1.Rows[j].Cells[i].Style.BackColor = SystemColors.Window;

dataGridView1.Rows[j].Cells[i].Style.ForeColor = SystemColors.ControlText;

}

}

}

// Обработка нажатия на кнопку "Вычислить"

private void button1\_Click(object sender,EventArgs e)

{

// Очистка оформления таблицы

ClearDGVStyle();

// Преобразование таблицы в массив

int[,] array = ArrayToDGV.ConvertToArray(dataGridView1);

// Объявление переменной для хранения итоговой таблицы

int[,] result;

//Объявление переменной для хранения индексов оставшихся строк

int[] indexes;

// Массив максимальных элементов

int[] max = Class2DArray.FindMax(array,out result, out indexes);

// Сброс размера формы

Size = startSize;

// Если в выходной таблице есть строки

if (result != null)

{

label8.Visible = false;

dataGridView2.Visible = true;

ArrayToDGV.ConvertToDGV(result, dataGridView2);

for (int i = 0; i < dataGridView2.ColumnCount; i++)

{

dataGridView2.Columns[i].Width = columnwidth;

dataGridView2.Columns[i].HeaderCell.Value = (i).ToString();

}

for (int i = 0; i < dataGridView2.RowCount; i++)

{

dataGridView2.Rows[i].HeaderCell.Value = (indexes[i]+1).ToString();

}

ResizeDGV(dataGridView2);

label5.Visible = true;

}

// Если в выходной таблице нет строк

else

{

label5.Visible = false;

label8.Visible = true;

dataGridView2.Visible = false;

label8.Text = "В итоговом массиве не осталось ни одной строки!";

}

// Вывод максимальных элементов с увеличением высоты формы

label7.Text = "Максимальные элементы для каждой строки исходного двумерного массива:\n";

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

{

Height += 13;

label7.Text += $"{i+1}) {max[i]}\n";

}

// Выделение максимальных элементов строк в dataGridView1

for(int i = 0;i < dataGridView1.ColumnCount;i++)

{

for(int j = 0;j < dataGridView1.RowCount;j++)

{

if(Convert.ToInt32(dataGridView1[i,j].Value) == max[j]) {

dataGridView1.Rows[j].Cells[i].Style.BackColor = Color.Aqua;

dataGridView1.Rows[j].Cells[i].Style.ForeColor = Color.DarkBlue;

}

}

}

}

// Обработка нажатия на кнопку случайного заполнения

private void button2\_Click(object sender,EventArgs e)

{

// Очистка оформления таблицы

ClearDGVStyle();

// Разблокировка кнопки

button1.Enabled = true;

// Заполнение массива случайными элементами

int[,] array = Class2DArray.RandomFill(Convert.ToInt32(textBox1.Text),Convert.ToInt32(textBox2.Text),dataGridView1.RowCount,dataGridView1.ColumnCount);

ArrayToDGV.ConvertToDGV(array,dataGridView1);

// Изменение ширины колонок

for(int i = 0;i < dataGridView1.ColumnCount;i++)

{

dataGridView1.Columns[i].Width = columnwidth;

}

}

// Изменение количества строк через элемент управления

private void numericUpDown1\_ValueChanged(object sender,EventArgs e)

{

// Сброс оформления таблицы

ClearDGVStyle();

// Добавление строк

while (dataGridView1.RowCount < numericUpDown1.Value)

{

dataGridView1.Rows.Add();

dataGridView1.Rows[dataGridView1.RowCount - 1].HeaderCell.Value = dataGridView1.RowCount.ToString();

}

// Удаление строк

while (dataGridView1.RowCount > numericUpDown1.Value)

{

dataGridView1.Rows.RemoveAt(dataGridView1.RowCount - 1);

}

// Изменение размера таблицы

ResizeDGV(dataGridView1);

}

private void numericUpDown2\_ValueChanged(object sender,EventArgs e)

{

// Сброс оформления таблицы

ClearDGVStyle();

// Добавление столбцов

while (dataGridView1.ColumnCount < numericUpDown2.Value)

{

dataGridView1.Columns.Add($"c{dataGridView1.ColumnCount}", $"{dataGridView1.ColumnCount+1}");

dataGridView1.Columns[dataGridView1.ColumnCount - 1].SortMode = DataGridViewColumnSortMode.NotSortable;

dataGridView1.Columns[dataGridView1.ColumnCount - 1].Width = columnwidth;

}

// Удаление столбцов

while (dataGridView1.ColumnCount > numericUpDown2.Value)

{

dataGridView1.Columns.RemoveAt(dataGridView1.ColumnCount - 1);

}

//Изменение размеров таблицы

ResizeDGV(dataGridView1);

}

// Выбор случайного заполнения

private void radioButton2\_CheckedChanged(object sender,EventArgs e)

{

// Случайное заполнение включено

if(radioButton2.Checked)

{

dataGridView1.ReadOnly = true;

textBox1.Visible = true;

textBox2.Visible = true;

label3.Visible = true;

label4.Visible = true;

button2.Visible = true;

}

// Случайное заполнение выключено

else

{

dataGridView1.ReadOnly = false;

textBox1.Visible = false;

textBox2.Visible = false;

label3.Visible = false;

label4.Visible = false;

button2.Visible = false;

}

}

// Обработка изменения текста

private void textBox\_TextChanged(object sender,EventArgs e)

{

int min, max;

if((int.TryParse(textBox1.Text,out min)) && (int.TryParse(textBox2.Text,out max)))

{

if(min < max)

{

button2.Enabled = true;

return;

}

}

button2.Enabled = false;

}

//Обработка изменения значения ячейки таблицы

private void dataGridView1\_CellValueChanged(object sender,DataGridViewCellEventArgs e)

{

ClearDGVStyle();

int n = 0;

for(var i = 0;i < dataGridView1.ColumnCount;i++)

{

// Если значение не int

if(!int.TryParse(dataGridView1[i,0].Value?.ToString(),out n))

{

// Блокировка кнопки

button1.Enabled = false;

return;

}

}

button1.Enabled = true;

}

// Изменение размеров таблицы на основании количества строк и столбцов

void ResizeDGV(DataGridView dgv)

{

if(dgv.RowCount > 7 && dgv.ColumnCount > 7)

{

dgv.Size = new Size(columnwidth \* 7 + 20 + dgv.RowHeadersWidth, dgv.RowTemplate.Height \* 7 + 20 + dgv.ColumnHeadersHeight);

}

else if(dgv.RowCount > 7 && dgv.ColumnCount <= 7)

{

dgv.Size = new Size(columnwidth \* dgv.ColumnCount + 20 + dgv.RowHeadersWidth, dgv.RowTemplate.Height \* 7 + 3 + dgv.ColumnHeadersHeight);

}

else if(dgv.RowCount <= 7 && dgv.ColumnCount > 7)

{

dgv.Size = new Size(columnwidth \* 7 + 3 + dgv.RowHeadersWidth, dgv.RowTemplate.Height \* dgv.RowCount + 20 +3 + dgv.ColumnHeadersHeight);

}

else

{

dgv.Size = new Size(columnwidth \* dgv.ColumnCount + 3 + dgv.RowHeadersWidth, dgv.RowTemplate.Height \* dgv.RowCount +3+ dgv.ColumnHeadersHeight);

}

}

// Добавление обработчика нажатия на кнопку для ячеек таблицы

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

{

TextBox tb = (TextBox)e.Control;

tb.KeyPress += new KeyPressEventHandler(tb\_KeyPress);

}

// Обработка нажатия на клавишу в текстовом поле и ячейке таблицы

void tb\_KeyPress(object sender,KeyPressEventArgs e)

{

TextBox tb = sender as TextBox;

// Проверка на то, что введённый символ - символ управления, запятая или минус.

if((!char.IsControl(e.KeyChar)) && (!char.IsDigit(e.KeyChar)) && (e.KeyChar != '-'))

{

e.Handled = true;

}

// Проверка на то, что при вводе минуса в текстовом поле нет минуса и он вводится в нулевую позицию.

if((e.KeyChar == '-') && ((tb.Text.Contains("-")) || (tb.SelectionStart != 0)))

{

e.Handled = true;

}

}

}

}

**Код модуля Class2DArray.cs:**

using System;

using System.Collections.Generic;

namespace Task6

{

public class Class2DArray

{

/// <summary>

/// Метод для случайного заполнения двумерного массива

/// </summary>

/// <param name="min"> Минимальное значение элемента массива</param>

/// <param name="max"> Максимальный предел значений элемента массива</param>

/// <param name="rows"> Количество рядов в двумерном массиве</param>

/// <param name="columns"> Количество столбцов в двумерном массиве</param>

/// <returns> Двумерный массив, заполненный случайными значениями</returns>

public static int[,] RandomFill(int min, int max, int rows, int columns)

{

// Исключение: числа должны быть неотрицательными

if (rows <= 0 || columns <= 0) throw new Exception("Длина массива должна быть больше 0.");

// Исключение: Максимальный предел должен быть больше минимального значения

if (min >= max) throw new Exception("Границы случайных чисел для заполнения массива заданы некорректно.");

// Объявление нового двумерного массива

int[,] result = new int[rows, columns];

Random random = new Random();

// Заполнение двумерного массива случайными значениями

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

{

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

{

result[i,j] = random.Next(min, max);

}

}

return result;

}

/// <summary>

/// Метод для поиска максимальных элементов в строках двумерного массива

/// </summary>

/// <param name="arr"> Двумерный массив, в котором производится поиск максимальных элементов в строках. </param>

/// <param name="result"> Двумерный массив, получаемый из исходного после удаления строк, где максимальные элементы стоят на последней позиции в строке. </param>

/// <param name="indexes"> Массив индексов строк выходного массива. </param>

/// <returns> Массив максимальных элементов по строкам. </returns>

public static int[] FindMax(int[,] arr,out int[,] result, out int[] indexes)

{

// Массив максимальных элементов

int[] max = new int[arr.GetLength(0)];

// Лист массивов, отражающий строки, которые

List<int[]> temp = new List<int[]>();

// Лист индексов строк, оставшихся после удаления

List<int> indexList = new List<int>();

// Поиск максимальных элементов для каждой строки

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

{

max[i] = arr[i,0];

for(int j = 0;j < arr.GetLength(1);j++)

{

if(arr[i,j] > max[i])

{

max[i] = arr[i,j];

}

}

}

// Создание листа строк, в которых максимальный элемент не на последней позиции

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

{

if(arr[i,arr.GetLength(1) - 1] != max[i])

{

indexList.Add(i);

temp.Add(new int[arr.GetLength(1)]);

for(int j = 0;j < arr.GetLength(1);j++)

{

temp[temp.Count - 1][j] = arr[i,j];

}

}

}

if(temp.Count > 0)

{

result = new int[temp.Count, arr.GetLength(1)];

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

{

for(int j = 0;j < arr.GetLength(1);j++)

{

result[i,j] = temp[i][j];

}

}

indexes = indexList.ToArray();

}

else

{

indexes = null;

result = null;

}

return max;

}

}

}

1. **Схемы основных подпрограмм**



**Рис. 5.1. Алгоритм метода FindMax**

1. **Тестирование класса по обработке массивов**
   1. **Код модуля для автоматизированного тестирования**

using System;

using Task6;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace UnitTestProject1

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void RegularUse()

{

int[,] arr = { {1, 2, 3, 4, 5, 6},

{3, 4, 6, 1, 2, 2},

{3, 4, 6, 1, 2, -10},

{5, 5, 5, 5, 5, 5},

{1, 2, 56, 42, 56, 56} };

int[,] result;

int[] indexes;

int[] max = Class2DArray.FindMax(arr, out result, out indexes);

Assert.AreEqual(2, result.GetLength(0));

int[,] temp = { {3, 4, 6, 1, 2, 2},

{3, 4, 6, 1, 2, -10} };

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

{

for (int j = 0; j < temp.GetLength(1); j++)

{

Assert.AreEqual(temp[i, j], result[i, j]);

}

}

Assert.AreEqual(6, max[0]);

Assert.AreEqual(6, max[1]);

Assert.AreEqual(6, max[2]);

Assert.AreEqual(5, max[3]);

Assert.AreEqual(56, max[4]);

}

[TestMethod]

public void NoDeletedElements() {

int[,] arr = { {1, 9, 2, 1, 6, 6},

{5, 2, 7, 6, 1, 2},

{8, 7, 2, 1, 6, -10},

{2, 2, 7, -5, 9, 5},

{8, 1, 0, 2, 5, -5} };

int[,] result;

int[] indexes;

int[] max = Class2DArray.FindMax(arr, out result, out indexes);

Assert.AreEqual(arr.GetLength(0), result.GetLength(0));

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

for (int j = 0; j < arr.GetLength(1); j++){

Assert.AreEqual(arr[i, j], result[i, j]);

}

}

Assert.AreEqual(9, max[0]);

Assert.AreEqual(7, max[1]);

Assert.AreEqual(8, max[2]);

Assert.AreEqual(9, max[3]);

Assert.AreEqual(8, max[4]);

}

[TestMethod]

public void AllElementsDeleted() {

int[,] arr = { {1, 9, 2, 1, 6, 60},

{5, 2, 7, 6, 1, 20},

{8, 7, 2, 1, 6, 100},

{2, 2, 7, -5, 9, 50},

{8, 1, 0, 2, 5, 50} };

int[,] result = { {1, 2, 3}, {1, 2, 3} };

int[] indexes;

int[] max = Class2DArray.FindMax(arr, out result, out indexes);

Assert.IsNull(result);

Assert.AreEqual(60, max[0]);

Assert.AreEqual(20, max[1]);

Assert.AreEqual(100, max[2]);

Assert.AreEqual(50, max[3]);

Assert.AreEqual(50, max[4]);

}

public void TestOneDimention() {

int[,] arr = { { 1,9,2,1,6,60 } };

int[,] result;

int[] indexes;

int[] max = Class2DArray.FindMax(arr,out result, out indexes);

Assert.IsNull(result);

Assert.AreEqual(1,max.Length);

arr = null;

arr = new int[,] { { 1 },{ 2 },{ -10 },{ 15 } };

max = Class2DArray.FindMax(arr,out result, out indexes);

Assert.IsNotNull(result);

Assert.AreEqual(1,max[0]);

Assert.AreEqual(2,max[1]);

Assert.AreEqual(-10,max[2]);

Assert.AreEqual(15,max[3]);

}

[TestMethod]

public void TestRandomFill(){

int[,] arr = Class2DArray.RandomFill(1, 10, 5, 2);

Assert.IsNotNull(arr);

Assert.AreEqual(5,arr.GetLength(0));

Assert.AreEqual(2,arr.GetLength(1));

Try{

arr = Class2DArray.RandomFill(10,-1,5,2);

Assert.Fail();

}

catch(Exception ex) {

Assert.AreEqual("Границы случайных чисел для заполнения массива заданы некорректно.", ex.Message);

}

try{

arr = Class2DArray.RandomFill(10,10,5,2);

Assert.Fail();

}

catch(Exception ex) {

Assert.AreEqual("Границы случайных чисел для заполнения массива заданы некорректно.",ex.Message);

}

Try {

arr = Class2DArray.RandomFill(10,14,-1,2);

Assert.Fail();

}

catch(Exception ex) {

Assert.AreEqual("Длина массива должна быть больше 0.",ex.Message);

}

}

}

}

* 1. **Результаты тестирования**



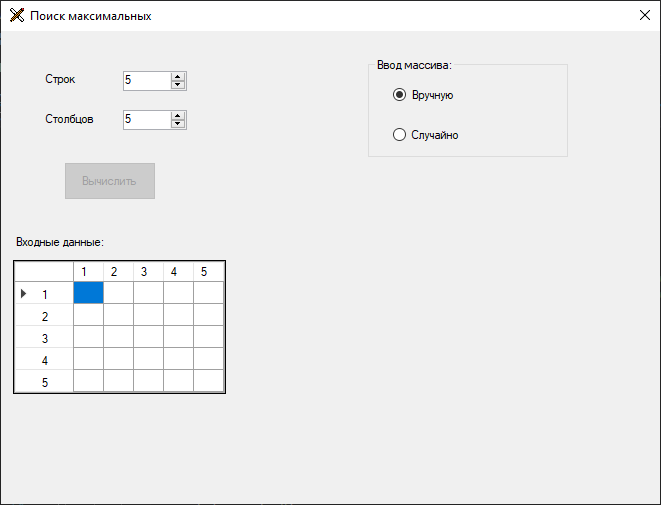
**Рис. 6.2.1. Результаты тестирования**

1. **Протокол испытаний**

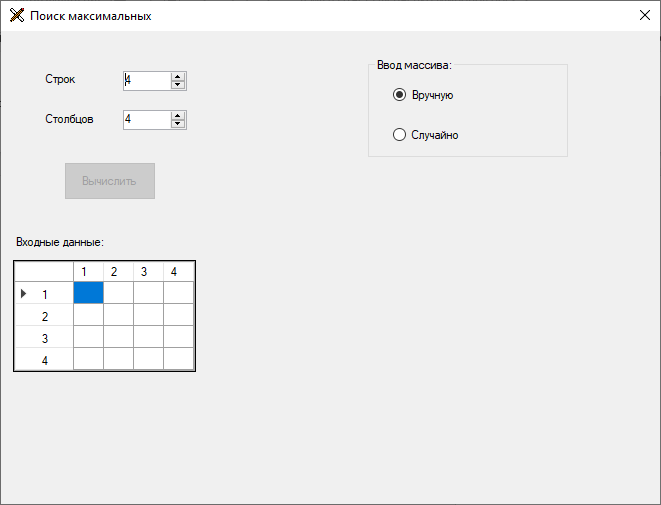
Таблица 7.1 – Протокол испытаний

| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Фактические результаты** |
| --- | --- | --- | --- |
| Способность ручной установки количества колонок | Ввести значения в поля:  Поле «Строк» = 5  Поле «Столбцов» = 5 | Количество столбцов и строк таблицы изменилось на 5 | Соответствуют ожиданиям  (См. Рис. 7.1) |
| Способность контроля недопустимых значений при ручной установки количества строк и столбцов | Ввести значения в поля:  Поле «Строк» = авы4П,3  Поле «Столбцов» = авы4П.3 | Поле «Строк» = 4  Поле «Столбцов» = 4 | Соответствуют ожиданиям  (См. Рис. 7.2) |
| Способность контроля некорректных значений при ручной установки количества строк и столбцов | Ввести значения в поля:  Поле «Строк» = -100  Поле «Столбцов» = -100 | Поле «Строк» = 1  Поле «Столбцов» = 1 | Соответствуют ожиданиям  (См. Рис. 7.3) |
| Ввести значения в поля:  Поле «Строк» = 100  Поле «Столбцов» = 100 | Поле «Строк» = 25  Поле «Столбцов» = 25 | Соответствуют ожиданиям  (См. Рис. 7.4) |
| Способность установки количества строк и столбцов через элемент управления | В каждом из полей нажать элемент управления «^» | Количество строк и столбцов изменилось на «2» | Соответствуют ожиданиям  (См. Рис. 7.5) |
| Способность ручного ввода значений в ячейки таблицы | Ввести значения в поля:  Поле «Строк» = 2  Поле «Столбцов» = 2  Ввести значения в таблицу:  2; -2  -10; 10 | Кнопка «Вычислить» активна; В таблице отображаются значения:  2 -2  -10 10 | Соответствуют ожиданиям  (См. Рис. 7.6) |
| Способность контроля недопустимых значений при ручном вводе значений в ячейки таблицы | Ввести значения в поля:  Поле «Строк» = 2  Поле «Столбцов» = 2  Ввести значения в таблицу:  Рра3; лку-2.3аы  -Н10О; Щ10П | Кнопка «Вычислить» активна; В таблице отображаются значения:  3; -23  -10; 10 | Соответствуют ожиданиям  (См. Рис. 7.7) |
| Способность контроля некорректных значений при ручном вводе значений в ячейки таблицы | Ввести значения в поля:  Поле «Строк» = 2  Поле «Столбцов» = 2  Ввести значения в таблицу:  -; -  -; - | Кнопка «Вычислить» неактивна | Соответствуют ожиданиям  (См. Рис. 7.8) |
| Способность автоматического ввода случайных значений в ячейки таблицы | Ввести значения в поля:  Поле «Строк» = 2  Поле «Столбцов» = 2  Нажать на элемент «Случайно»; в появившихся полях ввести диапазон от -1000 до 1000; нажать кнопку «Заполнить» | Значения в заданном диапазоне отобразились в таблице | Соответствуют ожиданиям  (См. Рис. 7.9) |
| Способность контроля недопустимых значений при установки диапазона | Нажать на элемент «Случайно»; в появившихся полях ввести:  Поле 1 = -424в.2Н  Поле 2 = рпа10м | Кнопка «Заполнить» активна;  Поле 1 = -4242  Поле 2 = 10 | Соответствуют ожиданиям  (См. Рис. 7.10) |
| Способность контроля некорректных значений при установки диапазона | Нажать на элемент «Случайно»; в появившихся полях ввести:  Поле 1 = -  Поле 2 = - | Кнопка «Заполнить» неактивна; | Соответствуют ожиданиям  (См. Рис. 7.11) |
| Способность контроля значений при вводе некорректного диапазона | Нажать на элемент «Случайными числами»; в появившихся полях ввести:  Поле 1 = 100  Поле 2 = -100 | Кнопка «Заполнить» неактивна; | Соответствуют ожиданиям  (См. Рис. 7.12) |
| Способность вывода результатов вычисления | Ввести значения в поля:  Поле «Строк» = 3  Поле «Столбцов» = 3  Ввести значения в таблицу:  3; 5; 10  11; 2; -10  0; -1; -10 | Кнопка «Заполнить» неактивна; В таблице справа отображаются значения:  11; 2; -10  0; -1; -10  Значения 10; 11; 0 в таблице справа окрасились в голубой; вывод сообщения со списком максимальных элементов строк | Соответствуют ожиданиям  (См. Рис. 7.13) |
| Способность вывода сообщения при возникновении исключительной ситуации | Ввести значения в поля:  Поле «Строк» = 2  Поле «Столбцов» = 2  Ввести значения в таблицу:  3; 55  11; 23 | Сообщение о том, что в итоговом массиве не осталось ни одной строки | Соответствуют ожиданиям  (См. Рис. 7.14) |

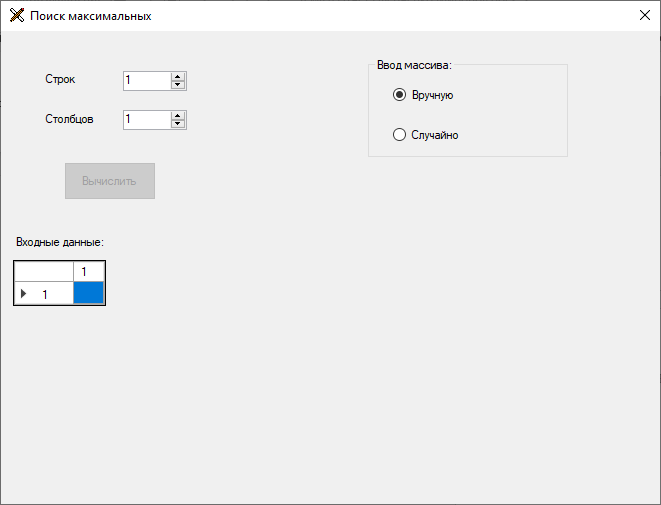
1. **Тестирование проекта**



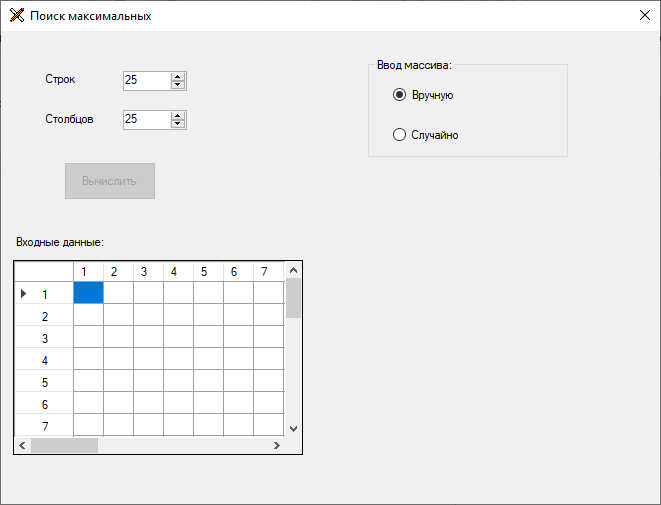
**Рис 7.1**



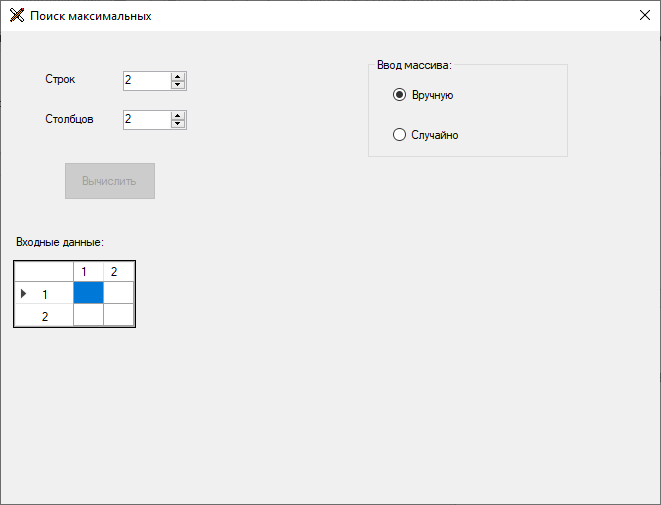
**Рис 7.2**



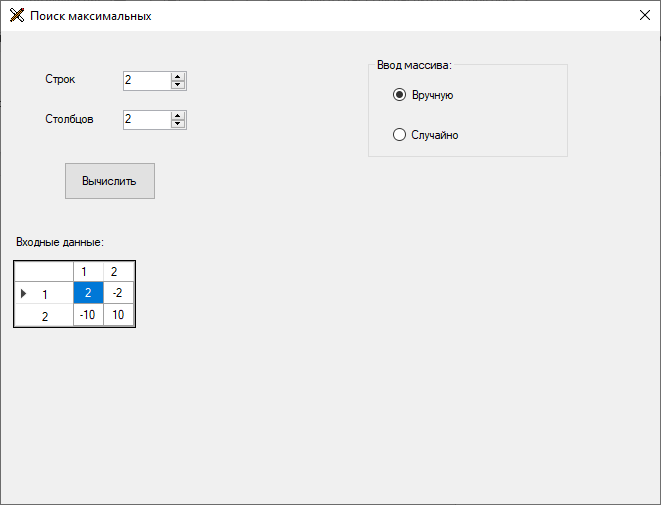
**Рис 7.3**



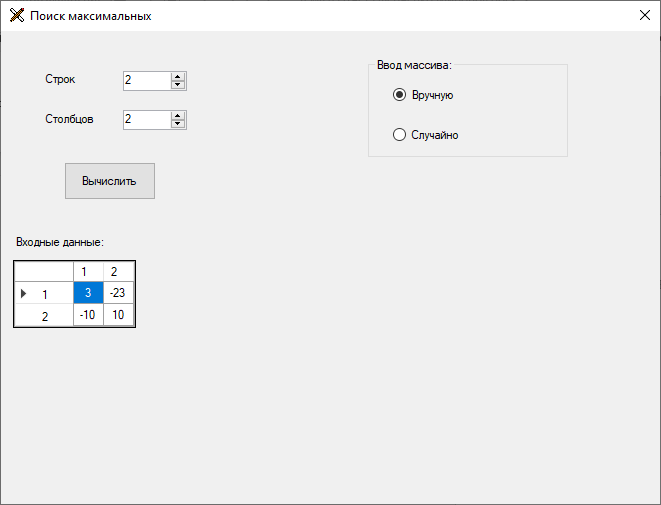
**Рис 7.4**



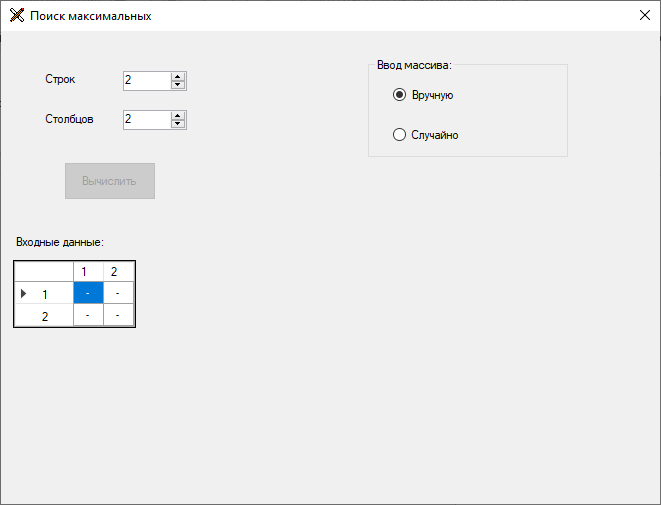
**Рис 7.5**



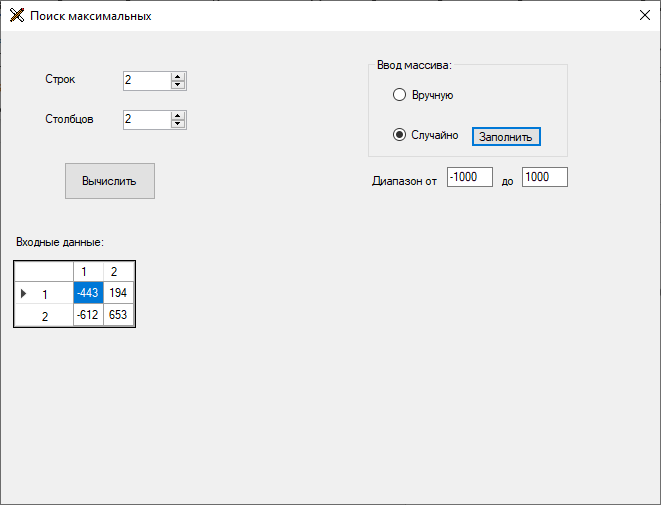
**Рис 7.6**



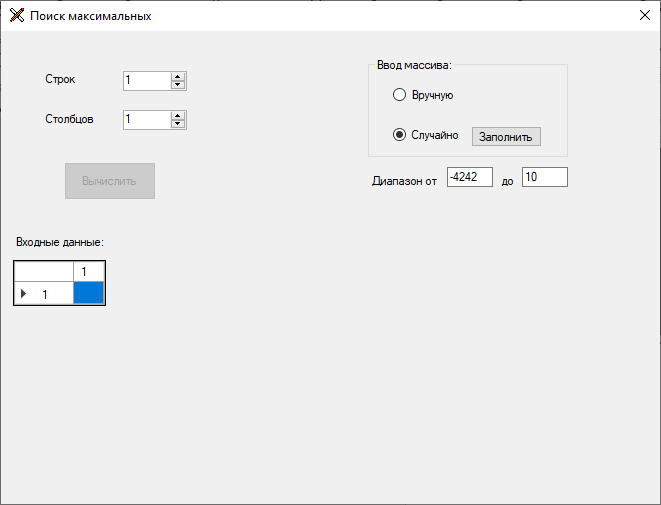
**Рис 7.7**



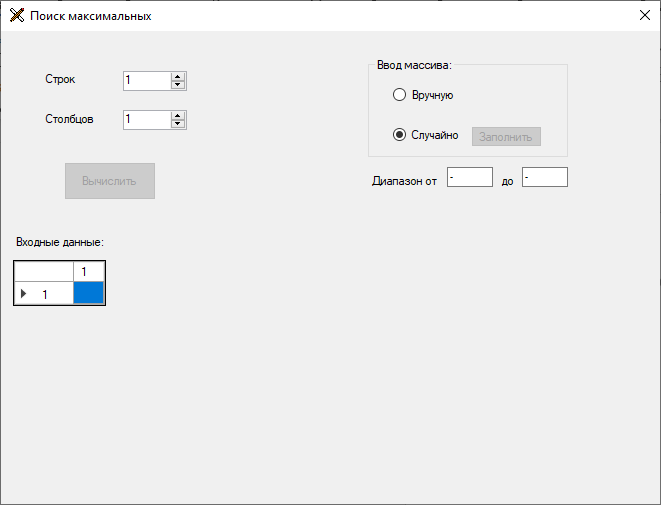
**Рис 7.8**



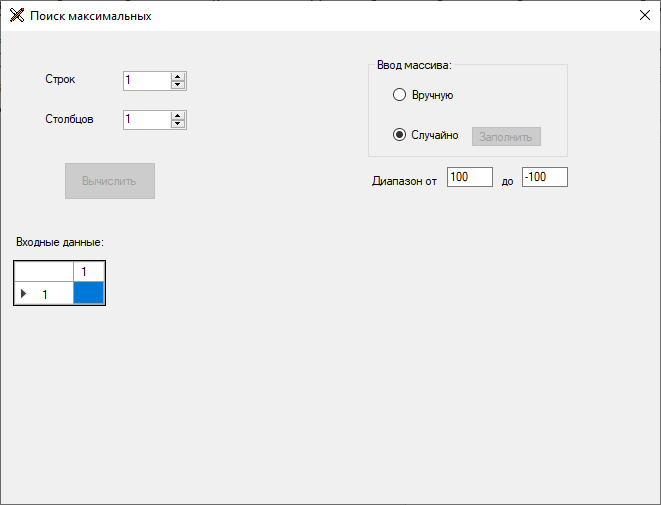
**Рис 7.9**



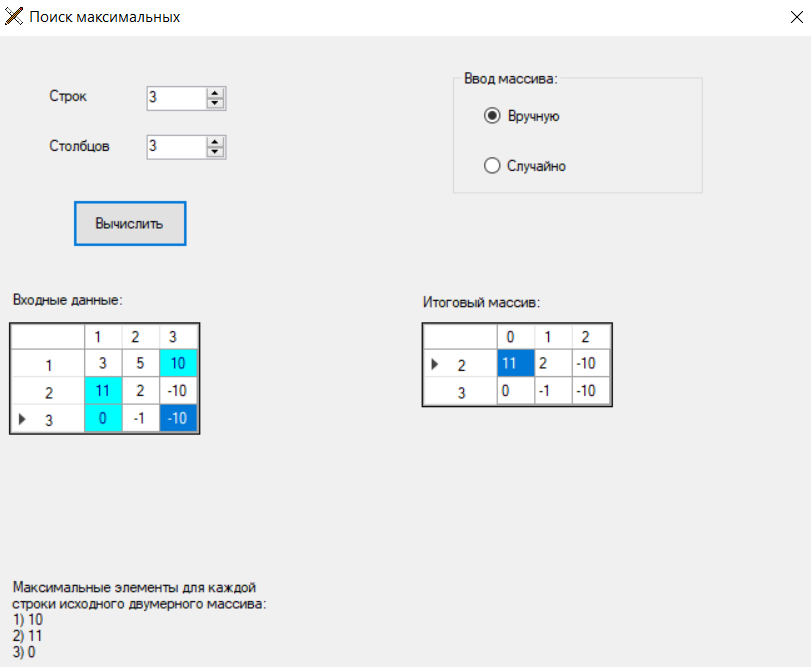
**Рис 7.10**



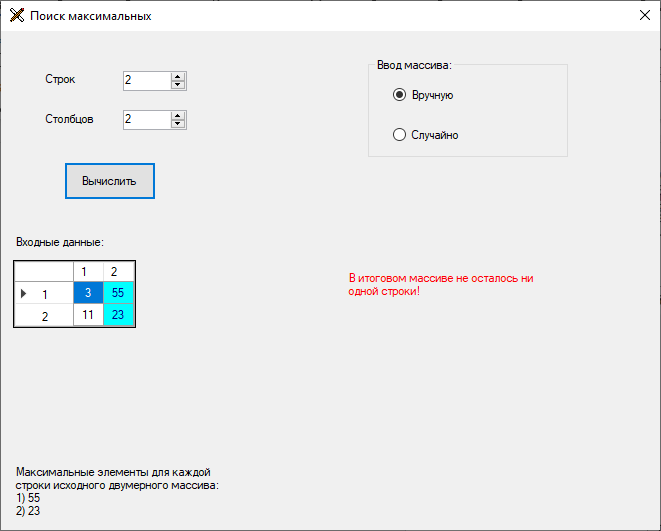
**Рис 7.11**



**Рис 7.12**



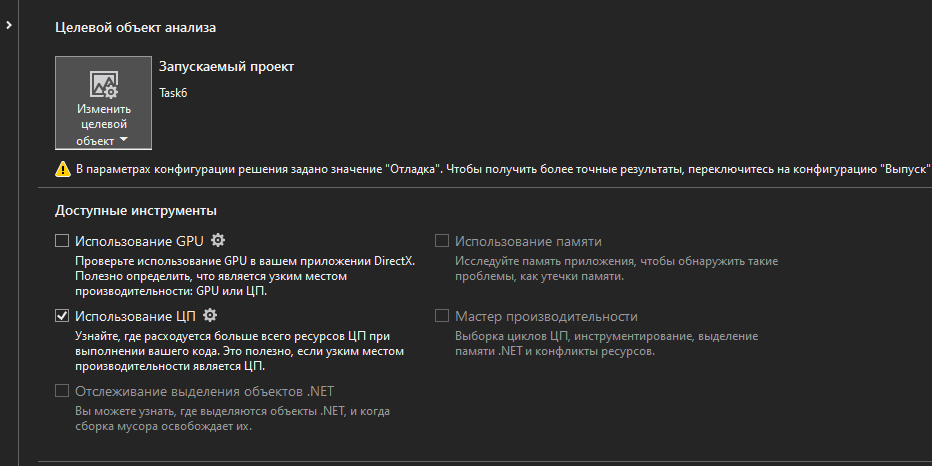
**Рис 7.13**



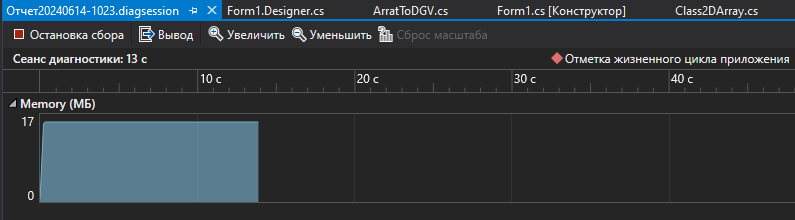
**Рис 7.14**

1. **Использование средств профилирования**

Используя средства профилирования, можно проверять потребление различных аппаратных ресурсов приложением на этапе исполнения.



**Рис 8.1. Выбор инструментов**



**Рис 8.2. Оценка использования памяти**

****

**Рис 8.3. Оценка загруженности центрального процессора**