Лабораторная работа №5

«Работа с двумерными массивами»

**Задание**

Дан двумерный массив положительных и отрицательных чисел. Разработайте метод, позволяющий найти количество элементов, которые кратны 3, а их десятичная запись заканчивается цифрой 1. Разработайте метод, который формирует новый массив из тех элементов исходной матрицы, значения которых больше найденного числа.

Требуется создать приложение, запрашивающее у пользователя через **InputBox** количество строк и столбцов массива, а также минимально и максимально возможные значения, которыми он будет заполнен. Далее необходимо найти общее количество чисел, кратных 3 и заканчивающихся цифрой 1, после чего создать другой массив, содержащий только числа, большие найденного количества.

Также необходимо разработать кнопки для сохранения элементов исходного массива в текстовый файл (а также последующего его открытия) и вывода обоих массивов (исходного и результирующего) в Microsoft Excel.

Для выполнения поставленного задания необходимо разработать/перегрузить следующие методы:

public static void FillArray(int[,] array, int start, int end) – принимает в качестве аргументов двумерный массив, а также минимальное и максимальное числа, формирующие диапазон случайно сгенерированных чисел, которыми заполняется массив. Ничего не возвращает;

public static void ArrayToDGV(int[,] array, DataGridView grid) – принимает массив и DataGridView, в который происходит вывод содержимого, ничего не возвращает;

public static int Count2D(int[,] array) – принимает массив, возвращает количество чисел, кратных 3 и оканчивающихся на 1;

public static void FilterArray(int[,] inputArray, ref int[] outputArray, int count) – принимает исходный массив, заполняемый массив (по ссылке), а также количество элементов, кратных 3 и оканчивающихся на 1. Заполняет второй массив числами и изменяет его размер в соответствии с содержащимися в нём данными;

public static void Resize(ref int[,] array, int rows, int columns) – принимает двумерный массив по ссылке, а также требуемую для него размерность в виде количества строк и столбцов. Изменяет размер массива, исключая все элементы, индекс которых больше требуемого размера, а также по возможности смещая все строки массива влево;

public static void ArrayToTXT(int[,] array, string path, bool open = true) – принимает двумерный целочисленный массив, строку с именем или путём к \*.txt файлу, а также boolean значение. Сохраняет полученный массив в текстовый файл (каждый элемент на отдельной строке), а также открывает его при open: true. Ничего не возвращает;

public static void ArraysToExcel<T>(<T>[][,] arrays, string[] names) – принимает массив, состоящий из одного или нескольких целочисленных двумерных массивов, а также массив со строковыми значениями их наименований, причём количество массивов типа integer и строк во втором массиве должно совпадать. Открывает окно приложения Microsoft Excel с массивами, разделёнными по листам в соответствии с переданными параметрами. Ничего не возвращает;

Все методы необходимо разместить в DLL-библиотеке.

* 1. **Разработка алгоритма решения задачи**

Схема алгоритма событийной процедуры решения задачи представлена на рисунке 1.

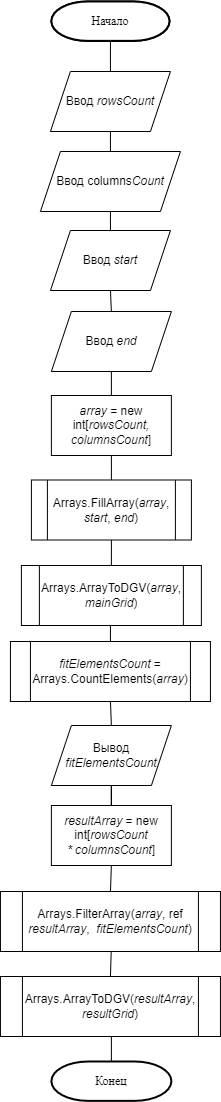


Рисунок 1 – Схема алгоритма процедуры

Схема алгоритма, выполняющего заполнение массива целыми числами в определённом диапазоне, представлена на рисунке 2.

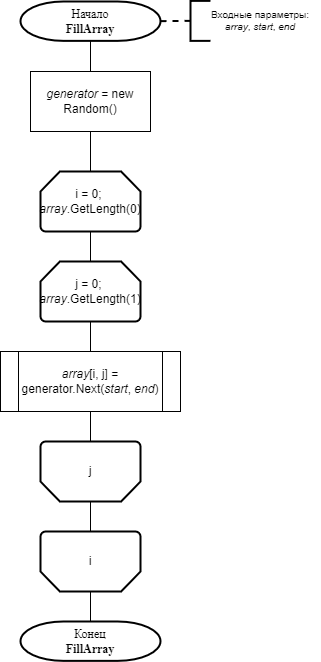
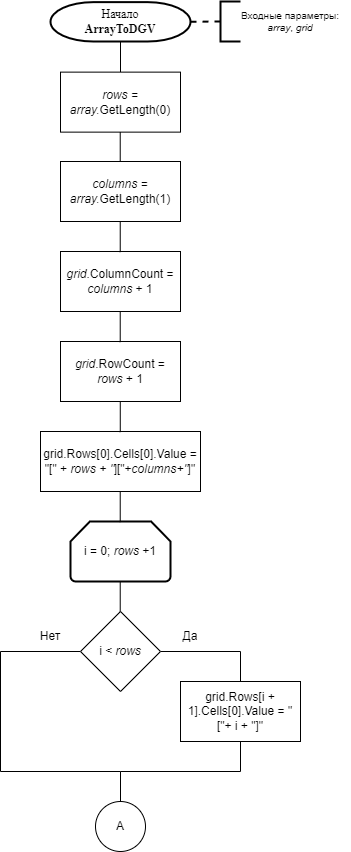


Рисунок 2 – Схема алгоритма заполнения массива числами

Схема алгоритма, выполняющего вывод двумерного массива в **DataGridView**, представлена на рисунке 3.



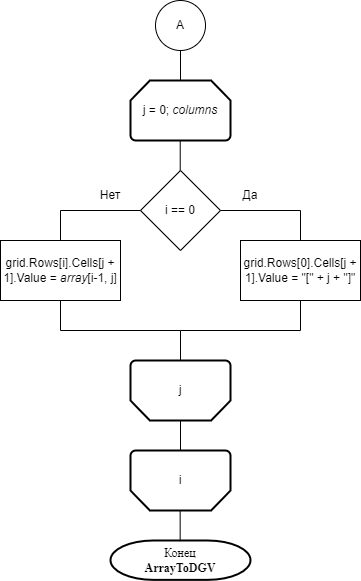


Рисунок 3 – Схема алгоритма вывода массива в DataGridView

Схема алгоритма, выполняющего счёт в массиве чисел, кратных 3 и оканчивающихся на 1, представлена на рисунке 4.

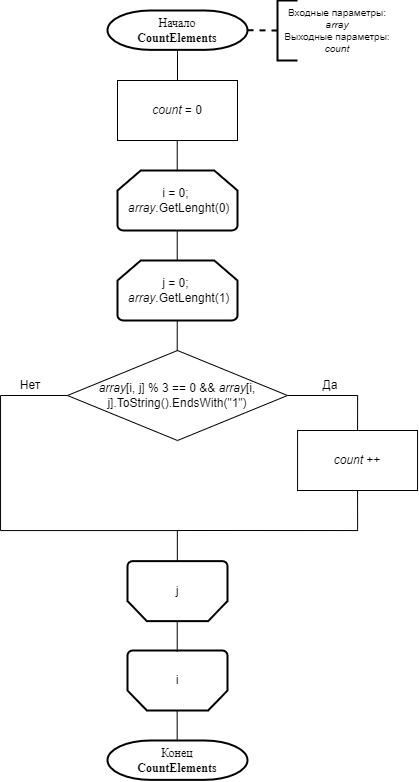


Рисунок 4 – Схема алгоритма счёта чисел в массиве

Схема алгоритма, выполняющего создание массива с числами, меньшими заданного, представлена на рисунке 5.

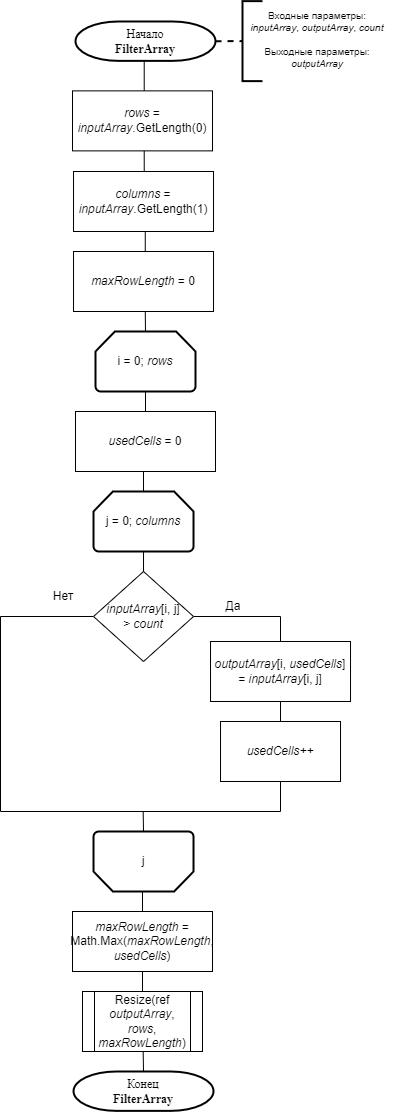


Рисунок 5 – Схема алгоритма создания нового массива

Схема алгоритма, выполняющего изменение длины массива, представлена на рисунке 6.

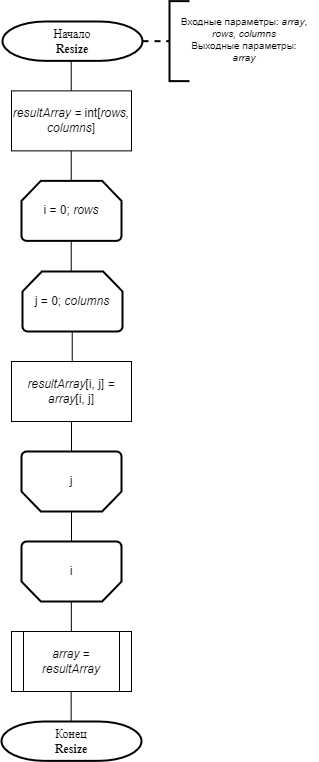


Рисунок 6 – Схема алгоритма изменения размера массива

* 1. **Разработка программного кода**

Программные методы, находящиеся в DLL-библиотеке “DLLib2”

using Microsoft.Office.Interop.Word;

using Microsoft.Vbe.Interop;

using Microsoft.VisualBasic.FileIO;

using System;

using System.Data.SQLite;

using System.Diagnostics;

using System.IO;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

using Word = Microsoft.Office.Interop.Word;

namespace DLLib2

{

public class Arrays

{

public static void FillArray(int[] array, int start, int end)

{

Random generator = new Random();

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

array[i] = generator.Next(start, end);

}

public static void FillArray(int[,] array, int start, int end)

{

Random generator = new Random();

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

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

array[i, j] = generator.Next(start, end);

}

public static void ArrayToDGV(int[] array, DataGridView grid)

{

grid.ColumnCount = array.Length;

grid.RowCount = 2;

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

{

grid.Rows[0].Cells[i].Value = "[" + i + "]";

grid.Rows[1].Cells[i].Value = array[i];

}

}

public static void ArrayToDGV(int[,] array, DataGridView grid)

{

int rows = array.GetLength(0);

int columns = array.GetLength(1);

grid.RowCount = rows + 1;

grid.ColumnCount = columns + 1;

grid.Rows[0].Cells[0].Value = $"[{rows}][{columns}]";

for (int i = 0; i < rows + 1; i++)

{

if (i < rows)

grid.Rows[i + 1].Cells[0].Value = $"[{i}]";

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

{

if (i == 0)

grid.Rows[0].Cells[j + 1].Value = $"[{j}]";

else

grid.Rows[i].Cells[j + 1].Value = array[i - 1, j];

}

}

}

public static int CountElements(int[] array)

{

int count = 0;

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

if (array[i].ToString().EndsWith("3") || array[i].ToString().EndsWith("5"))

count++;

return count;

}

public static int CountElements(int[,] array)

{

int count = 0;

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

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

if (array[i, j] % 3 == 0 && array[i, j].ToString().EndsWith("1"))

count++;

return count;

}

public static void FilterArray(int[] inputArray, ref int[] outputArray, int count)

{

int fitNumbers = 0; // счётчик количества подходящих чисел

// заполнение массива подходящими числами

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

{

if (inputArray[i] < count)

{

outputArray[fitNumbers] = inputArray[i];

fitNumbers++;

}

}

Resize(ref outputArray, fitNumbers); // изменение длины выходного массива

}

public static void FilterArray(int[,] inputArray, ref int[,] outputArray, int count)

{

int rows = inputArray.GetLength(0);

int columns = inputArray.GetLength(1);

int maxRowLength = 0;

int usedCells;

// заполнение массива подходящими числами

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

{

usedCells = 0;

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

{

if (inputArray[i, j] > count)

{

outputArray[i, usedCells] = inputArray[i, j];

usedCells++;

}

}

maxRowLength = Math.Max(maxRowLength, usedCells);

}

Resize(ref outputArray, rows, maxRowLength); // изменение длины выходного массива

}

// метод изменения длины рдномерного массива

public static void Resize(ref int[] array, int newSize)

{

int[] resultArray = new int[newSize];

int min = array.Length;

if (resultArray.Length < min)

min = resultArray.Length;

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

{

resultArray[i] = array[i];

}

array = resultArray;

}

public static void Resize(ref int[,] array, int rows, int columns)

{

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

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

{

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

resultArray[i, j] = array[i, j];

}

array = resultArray;

}

...

}

public class IO

{

public class ExcelWrapper

{

public Excel.Application app;

public Excel.Workbook wb;

private bool isFirstWorksheet = true;

public bool isMacrosDefined = false;

public bool isAboutToQuit = false;

public ExcelWrapper()

{

app = new Excel.Application() { UserControl = true };

wb = app.Workbooks.Add();

}

public void Show()

{

if (!isAboutToQuit)

app.Visible = true;

else

Close();

}

public void Close()

{

app.Quit();

}

public Excel.Worksheet GetNewWorksheet()

{

if (isFirstWorksheet)

{

this.isFirstWorksheet = false;

return wb.Worksheets[1];

}

else

return wb.Worksheets.Add();

}

private bool BaseMacrosErrorAllocationHandler()

{

DialogResult result = MessageBox.Show(caption: "Ошибка взаимодействия с Excel", text: "Параметры безопасности Excel не позволяют записывать макросы и создавать элементы управления из среды .NET\n\n" +

"Для предоставления доступа необходимо в Excel открыть 'Параметры' > 'Центр управления безопасностью' > 'Параметры центра управления безопасностью' > Раздел 'Параметры макросов' > Активировать CheckBox 'Доверять доступ к объектной модели VBA'" +

"\n\nПродолжить вывод массивов в Excel?",

buttons: MessageBoxButtons.YesNo, icon: MessageBoxIcon.Error);

// если пользователь отказался продолжить вывод - вернуть false ()

if (result == DialogResult.No)

return false;

return true;

}

public void AllocateMacros(string macrosCodePath, int maxArrayHeight, Func<bool> onErrorEvent)

{

if (onErrorEvent == null)

onErrorEvent = BaseMacrosErrorAllocationHandler;

// создание макросов

try

{

VBComponent macros = this.wb.VBProject.VBComponents.Add(vbext\_ComponentType.vbext\_ct\_StdModule);

macros.CodeModule.AddFromString(GetMacrosCode(macrosCodePath, maxArrayHeight));

this.isMacrosDefined = true;

}

// по умолчанию Excel не позволяет взаимодействовать с кодом VBA из внешней среды

// т.е. без настройки безопасности Excel записать макросы напрямую не получится

catch (System.Runtime.InteropServices.COMException)

{

if (!onErrorEvent())

this.isAboutToQuit = true;

isMacrosDefined = false;

}

}

private static string GetMacrosCode(string filePath, int arrayHeight)

{

StreamReader file = new StreamReader(filePath);

string line = file.ReadToEnd();

return string.Format(line, arrayHeight + 2);

}

public void DrawButtons(int vOffset, Excel.Worksheet ws)

{

if (ws == null)

ws = GetNewWorksheet();

Excel.Buttons buttons = ws.Buttons();

// определить положение кнопок

Excel.Range formatRange = ws.get\_Range($"A{vOffset}:C{vOffset}");

Excel.Range clearRange = ws.get\_Range($"A{vOffset + 1}:C{vOffset + 1}");

// создать кнопку форматирования

Excel.Button formatButton = buttons.Add(formatRange.Left, formatRange.Top, formatRange.Width, formatRange.Height);

formatButton.Name = "FormatTable";

formatButton.Text = "Форматировать таблицу";

formatButton.OnAction = "Format";

// создать кнопку очистки форматирования

Excel.Button clearButton = buttons.Add(clearRange.Left, clearRange.Top, clearRange.Width, clearRange.Height);

clearButton.Name = "ClearTable";

clearButton.Text = "Очистить форматирование";

clearButton.OnAction = "Clear";

}

public bool ArrayToWorksheet<T>(T[] array, string name, Excel.Worksheet ws)

{

if (ws == null)

ws = GetNewWorksheet();

string executableDir = Path.GetDirectoryName(System.Reflection.Assembly.GetEntryAssembly().Location);

if (this.isMacrosDefined)

DrawButtons(5, ws);

ws.Cells[1, 1] = name;

ws.Name = name;

LoadArray(array, ws);

return true;

}

public bool ArrayToWorksheet<T>(T[,] array, string name, Excel.Worksheet ws)

{

if (ws == null)

ws = GetNewWorksheet();

string executableDir = Path.GetDirectoryName(System.Reflection.Assembly.GetEntryAssembly().Location);

if (this.isMacrosDefined)

DrawButtons(array.GetLength(0) + 5, ws);

ws.Cells[1, 1] = name;

ws.Name = name;

LoadArray(array, ws);

return true;

}

private void LoadArray<T>(T[] array, Excel.Worksheet ws)

{

if (ws == null)

ws = GetNewWorksheet();

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

{

ws.Cells[2, i + 2] = "[" + i + "]";

ws.Cells[3, i + 2] = Convert.ToString(array[i]);

}

}

private void LoadArray<T>(T[,] array, Excel.Worksheet ws)

{

if (ws == null)

ws = GetNewWorksheet();

ws.Cells[2, 2] = $"[{array.GetLength(0)}][{array.GetLength(1)}]";

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

{

ws.Cells[i + 3, 2] = "[" + i + "]";

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

{

if (i == 0)

ws.Cells[2, j + 3] = "[" + j + "]";

ws.Cells[i + 3, j + 3] = Convert.ToString(array[i, j]);

}

}

}

}

public static void ArrayToTXT(int[] array, string path, bool open = true)

{

StreamWriter file = File.CreateText(path);

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

file.WriteLine(array[i].ToString());

file.Close();

// открыть текстовый документ

if (open)

Process.Start(path);

}

public static void ArrayToTXT(string[] array, string path, bool open = true)

{

StreamWriter file = File.CreateText(path);

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

file.WriteLine(array[i]);

file.Close();

// открыть текстовый документ

if (open)

Process.Start(path);

}

public static void ArrayToTXT(int[,] array, string path, bool open = true)

{

StreamWriter file = File.CreateText(path);

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

{

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

file.Write(array[i, j] + "\t");

file.WriteLine("\n");

}

file.Close();

// открыть текстовый документ

if (open)

Process.Start(path);

}

...

}

...

}

Основная часть программы

using DLLib2;

using Microsoft.VisualBasic;

using System;

using System.IO;

using System.Windows.Forms;

namespace MainApplication

{

public partial class Solution5\_2 : BaseForm

{

public int[,] array;

public int[] resultArray;

public Solution5\_2()

{

InitializeComponent();

this.ActiveControl = start;

}

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

{

mainGrid.Rows.Clear();

resultGrid.Rows.Clear();

int rowsCount, columnsCount, start, end;

try

{

rowsCount = Convert.ToInt32(Interaction.InputBox(Prompt: "Введите количество строк массива", Title: "Ввод данных",

DefaultResponse: "6"));

columnsCount = Convert.ToInt32(Interaction.InputBox(Prompt: "Введите количество столбцов массива", Title: "Ввод данных",

DefaultResponse: "6"));

start = Convert.ToInt32(Interaction.InputBox(Prompt: "Введите минимально возможное число в массиве",

Title: "Ввод данных", DefaultResponse: "-100"));

end = Convert.ToInt32(Interaction.InputBox(Prompt: "Введите максимально возможное число в массиве",

Title: "Ввод данных", DefaultResponse: "100"));

}

catch (FormatException)

{

MessageBox.Show(text: "Неверно введены данные", caption: "Ошибка 1", icon: MessageBoxIcon.Error, buttons: MessageBoxButtons.OK);

return;

}

this.array = new int[rowsCount, columnsCount];

Arrays.FillArray(this.array, start, end);

Arrays.ArrayToDGV(this.array, mainGrid);

int fitElementsCount = Arrays.CountElements(this.array);

MessageBox.Show(text: $"Найдено {fitElementsCount} подходящих элементов",

caption: "Вывод данных",

buttons: MessageBoxButtons.OK,

icon: MessageBoxIcon.Information);

this.resultArray = new int[rowsCount\*columnsCount];

Arrays.FilterArray(array, ref resultArray, fitElementsCount);

Arrays.ArrayToDGV(resultArray, resultGrid);

arraysToExcel.Enabled = true;

saveAsText.Enabled = true;

}

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

{

IO.ExcelWrapper excel = new IO.ExcelWrapper();

excel.AllocateMacros(Path.GetDirectoryName(System.Reflection.Assembly.GetEntryAssembly().Location) + "\\MacrosCode.txt"

, this.array.GetLength(0)

, null);

excel.ArrayToWorksheet(this.resultArray, "Результирующий массив", null);

excel.ArrayToWorksheet(this.array, "Исходный массив", null);

excel.Show();

}

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

{

IO.ArrayToTXT(this.array, "2DArray.txt", true);

}

}

}

**1.3. Результаты работы программы**

На рисунке 7 находится изображение второй формы, описывающей условия задачи.

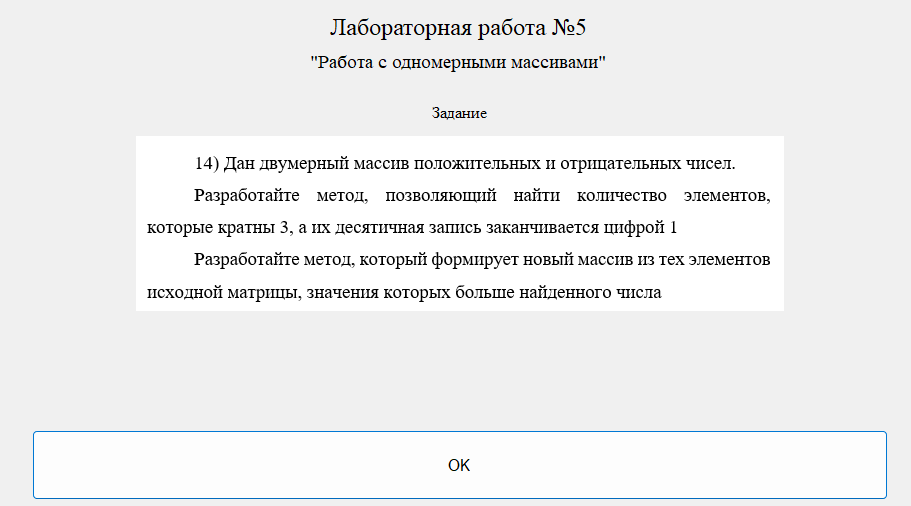


Рисунок 7 – Задание

На рисунке 8 находится демонстрация работы программы

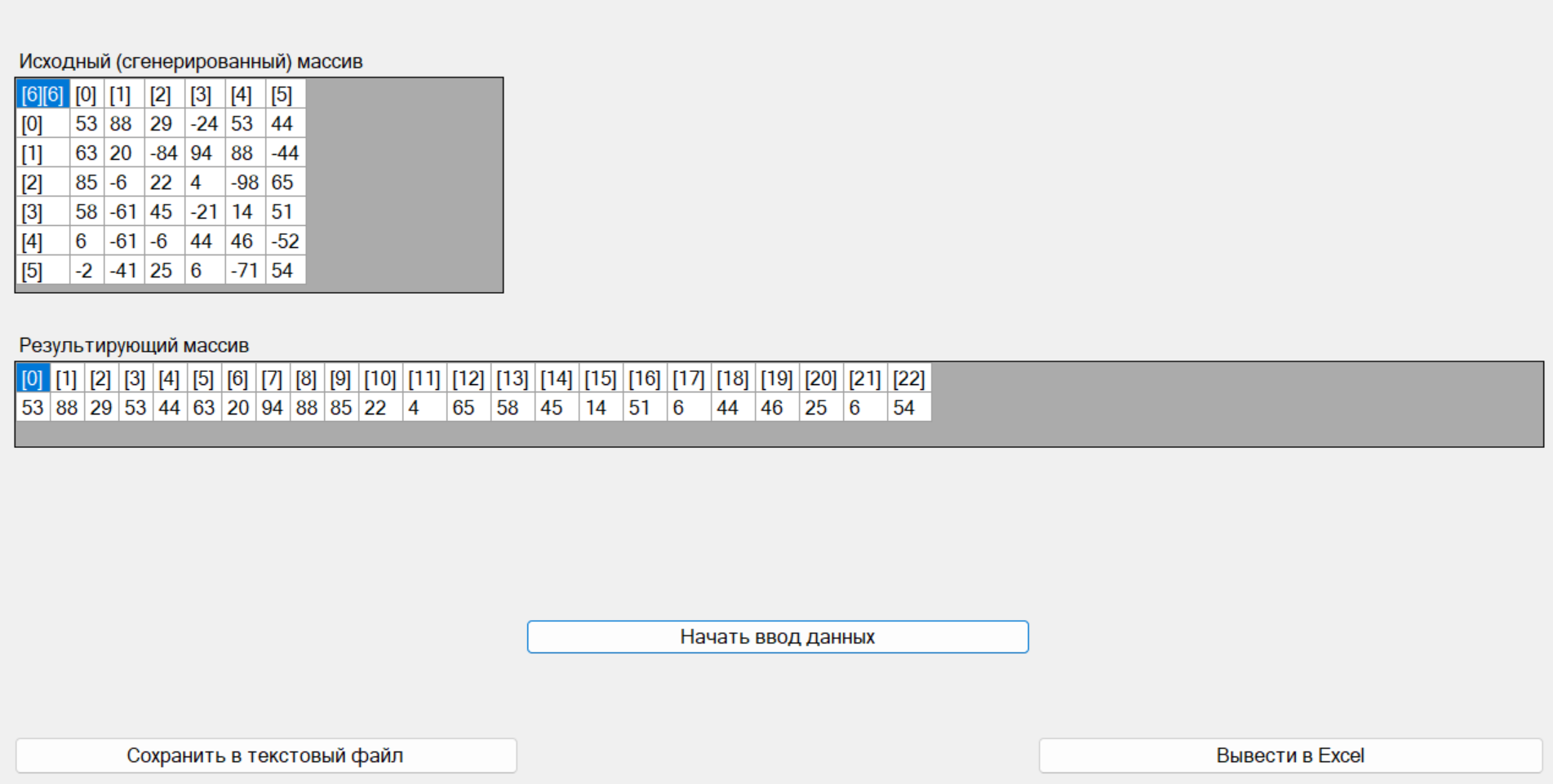


Рисунок 8 - Результат работы программы

На рисунке 9 находится демонстрация текстового файла с содержимым исходного массива

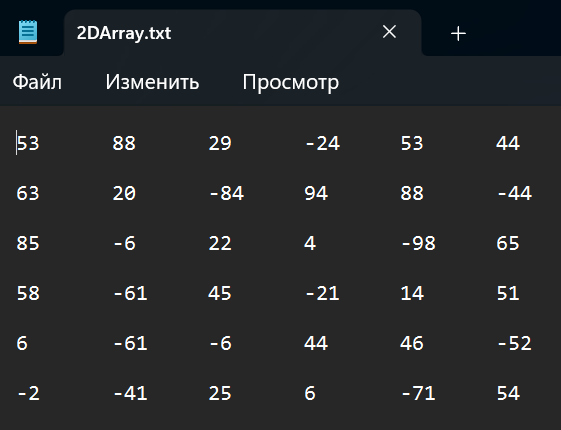


Рисунок 9 – Сохранённый файл со значениями массива

На рисунках 10 и 11 располагаются исходный и результирующий массивы соответственно

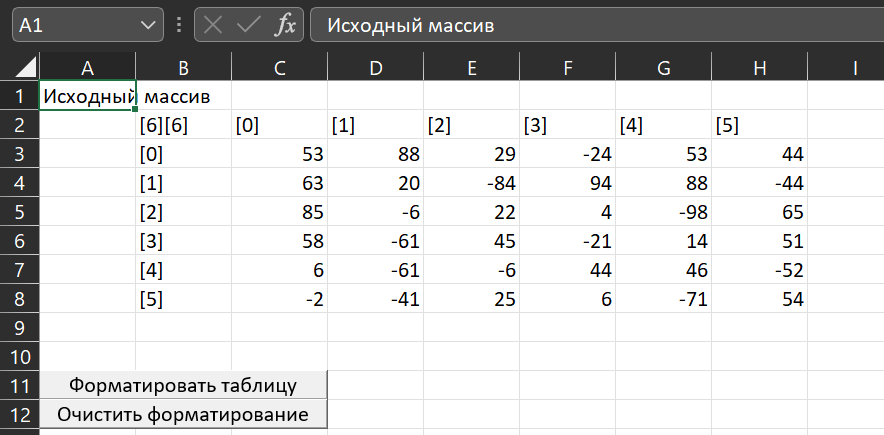


Рисунок 10 – Исходный массив в Excel

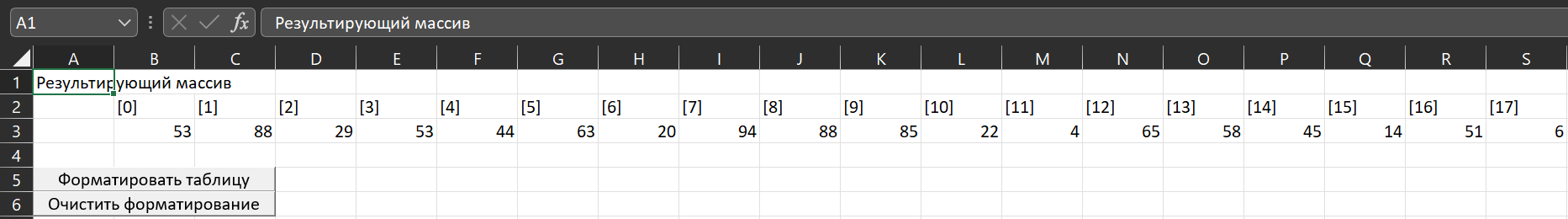
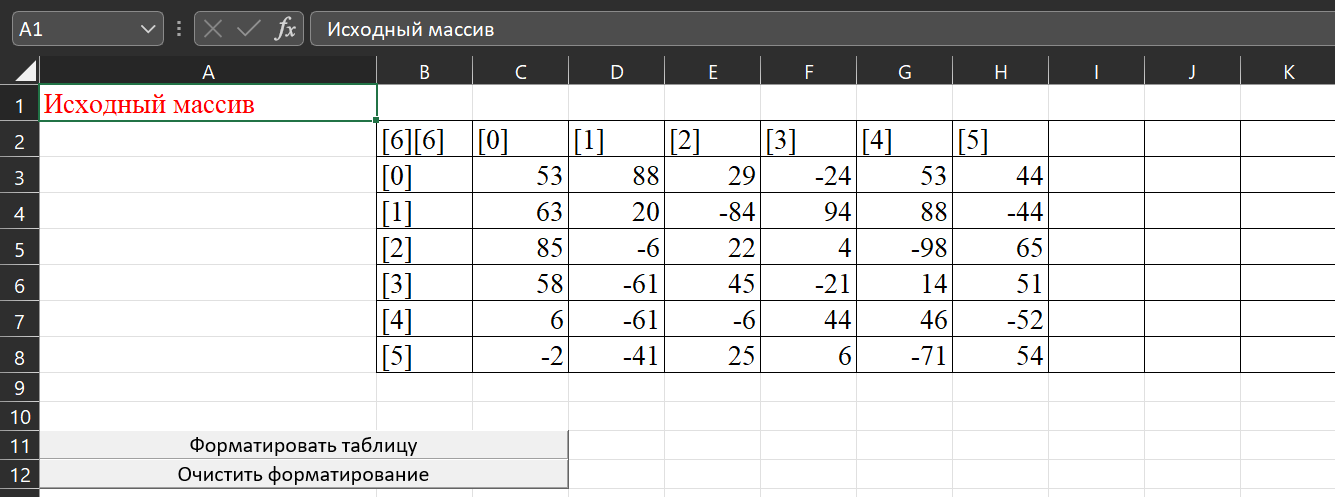


Рисунок 11 – Результирующий массив в Excel

На рисунках 12 и 13 также находятся исходный и результирующий массивы, но с применением форматирования таблицы

Рисунок 12 – Отформатированный исходный массив в Excel

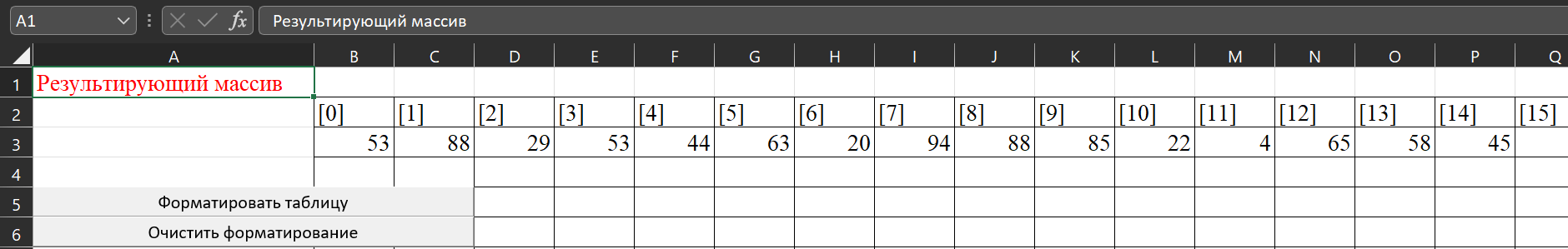


Рисунок 13 – Отформатированный результирующий массив в Excel

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гуриков, С. Р. Введение в программирование на языке Visual C#: учебное пособие / С. Р. Гуриков. — Москва: ФОРУМ: ИНФРА-М, 2019. — 447 с. — (Среднее профессиональное образование). - ISBN 978-5-00091-540-0. - Текст: электронный. - URL: <https://znanium.com/catalog/product/1012397> (дата обращения: 29.09.2023).

2. ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления" (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) (дата обращений 01.10.2023).

3. ГОСТ 19.701-90 (ИСО 5807-85) “Единая система программной документации СХЕМЫ АЛГОРИТМОВ, ПРОГРАММ, ДАННЫХ И СИСТЕМ Обозначения условные и правила выполнения”. (дата обращения 02.10.2023).