Лабораторная работа №3

«Работа с одномерными массивами»

**Задание**

Дан динамический целочисленный массив. Найти количество элементов, заканчивающихся на цифру 3 и заканчивающихся на цифру 5. Разработайте функцию, которая формирует новый массив из тех элементов исходного массива, значения которых меньше найденного общего количества элементов.

Требуется создать приложение, запрашивающее у пользователя через **InputBox** длину массива, а также минимально и максимально возможные значения, которыми он будет заполнен. Далее необходимо найти общее количество чисел, заканчивающихся на 3 или 5, после чего создать другой массив, содержащий только числа, меньшие найденного количества.

Также необходимо разработать кнопки для сортировки исходного массива, вставки элемента в результирующий массив по индексу, сохранения и последующего открытия элементов исходного массива в текстовый файл, вывода сгенерированного массива в виде таблицы Microsoft Word, а также вывода обоих массивов (исходного и результирующего) в Microsoft Excel.

Для выполнения поставленного задания необходимо разработать следующие методы:

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

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

public static int CountElementsEndWith3or5(int[] array) – принимает массив, возвращает количество чисел, заканчивающихся на 3 или 5;

public static void FiltereArray(int[] inputArray, ref int[] outputArray, int count) – принимает исходный массив, заполняемый массив (по ссылке), а также количество элементов, заканчивающихся на 3 или 5. Заполняет второй массив числами и изменяет его размер в соответствии с содержащимися в нём данными;

public static void Resize(ref int[] array, int newSize) – принимает массив по ссылке, а также требуемую для него длину. Изменяет размер массива, исключая все элементы, индекс которых больше требуемого размера

public static void Insert(ref int[] array, int index, int value) – принимает массив по ссылке, а также индекс нового элемента и его значение;

public static void SortBinaryInserting(ref int[] array) – принимает массив по ссылке и сортирует его методом бинарных включений;

public static void ArrayToTXT(int[] array, string path, bool open = true) – принимает 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(int[][] arrays, string[] names) – принимает массив, состоящий из одного или нескольких целочисленных 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;

public static void ArrayToWord(int[] array, string name) – принимает целочисленный массив, а также его наименование. Открывает окно приложения Microsoft Word с названием массива и таблицей, содержащей в первом ряду индексы массива, а во втором – его элементы.

Все методы необходимо разместить в DLL-библиотеке.

* 1. **Разработка алгоритма решения задачи**

Схема алгоритма событийной процедуры решения задачи представлена на рисунке 1.

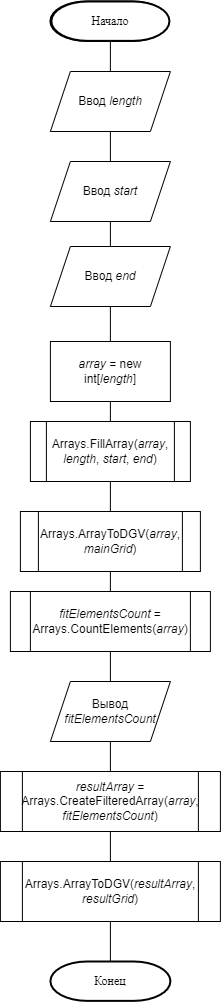


Рисунок 1 – Схема алгоритма процедуры

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

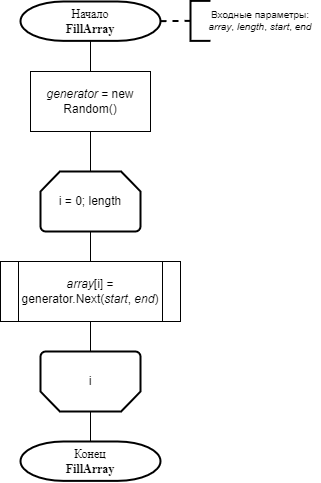


Рисунок 2 – Схема алгоритма заполнения массива числами

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

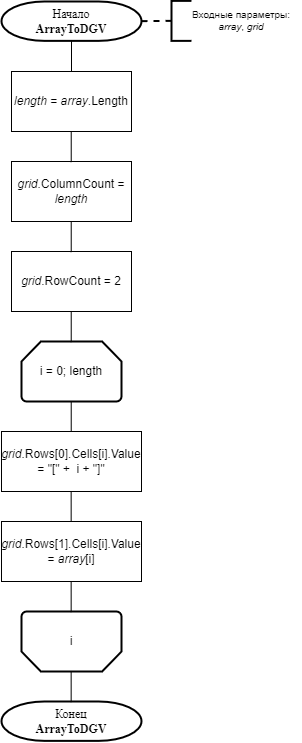


Рисунок 3 – Схема алгоритма вывода массива в DataGridView

Схема алгоритма, выполняющего счёт в массиве чисел, оканчивающихся на 3 или 5, представлена на рисунке 4.

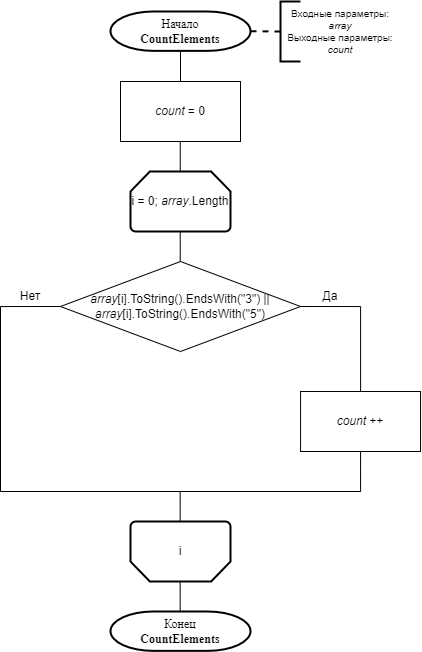


Рисунок 4 – Схема алгоритма счёта чисел в массиве

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

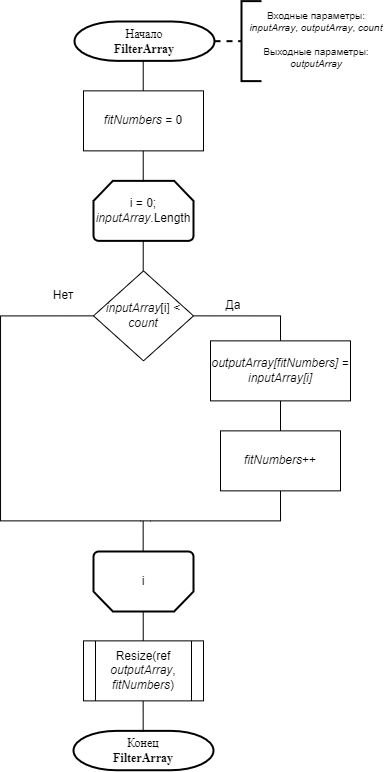


Рисунок 5 – Схема алгоритма создания нового массива

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

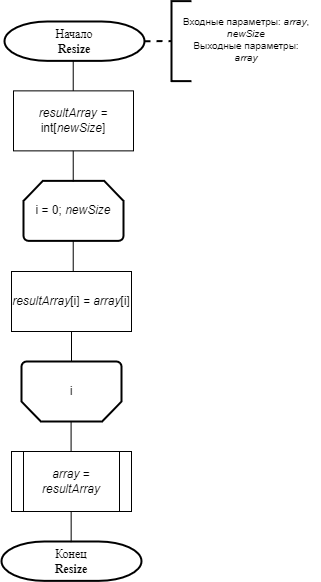


Рисунок 6 – Схема алгоритма изменения размера массива

Схема алгоритма, выполняющего вставку нового элемента в массив по индексу, представлена на рисунке 7.

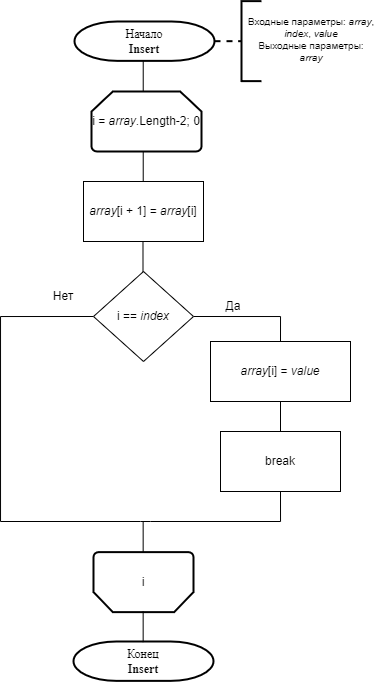
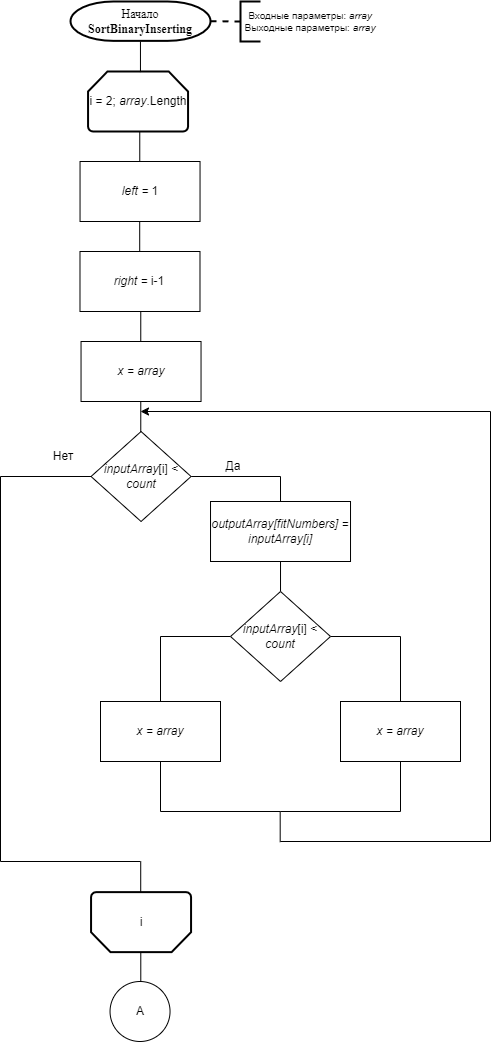


Рисунок 7 – Схема алгоритма вставки элемента в массив

Схема алгоритма, выполняющего сортировку массива методом бинарных включений, представлена на рисунке 8.



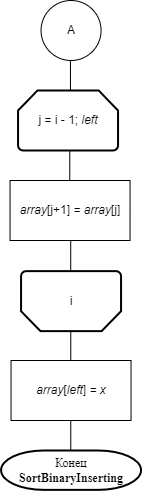


Рисунок 8 – Схема алгоритма сортировки массива

* 1. **Разработка программного кода**

Программные методы, находящиеся в DLL-библиотеке “DLLib2”

using System;

using System.Diagnostics;

using System.IO;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

using Word = Microsoft.Office.Interop.Word;

using Microsoft.Vbe.Interop;

using Microsoft.Office.Interop.Word;

namespace DLLib2

{

public class Arrays

{

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

{

Random generator = new Random();

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

{

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

}

}

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

{

int length = array.Length;

grid.ColumnCount = length;

grid.RowCount = 2;

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

{

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

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

}

}

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 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

}

// 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 Insert(ref int[] array, int index, int value)

{

if (index > array.Length)

throw new IndexOutOfRangeException(message: "Index must be less or equal initial array size");

for (int i = array.Length-2; i >= 0; i--)

{

array[i + 1] = array[i];

if (i == index)

{

array[i] = value;

break;

}

}

}

public static void SortBinaryInserting(ref int[] array)

{

int x, left, right;

int m;

for (int i = 2; i < array.Length; i++)

{

x = array[i];

left = 1;

right = i - 1;

while (left <= right)

{

m = (left + right) / 2;

if (x < array[m])

right = m - 1;

else

left = m + 1;

}

for (int j = i - 1; j > left; j--)

{

array[j+1] = array[j];

}

array[left] = x;

}

}

}

public class IO

{

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 ArraysToExcel(int[][] arrays, string[] names)

{

// вызвать исключение, если количества массивов и их имён не совпадают

if (arrays.Length != names.Length)

throw new Exception("Arrays and names must be the same length");

// запуск Excel и создание нового документа

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

Excel.Workbook wb = app.Workbooks.Add();

bool isMacrosDefined = false;

// создание макросов

try

{

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

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

macros.CodeModule.AddFromFile(executableDir + "\\MacrosCode.txt");

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)

{

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 - выйти из подпрограммы

if (result == DialogResult.No)

return;

}

Excel.Worksheet ws = wb.Worksheets[1];

// перебор всех массивов с конца (листы в Excel идут от последнего к первому)

for (int i = arrays.Length-1; i >= 0; i--)

{

if (isMacrosDefined)

{

Excel.Buttons buttons = ws.Buttons();

// определить положение кнопок

Excel.Range formatRange = ws.get\_Range($"A5:C5");

Excel.Range clearRange = ws.get\_Range($"A6:C6");

// 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";

}

ws.Cells[1, 1] = names[i];

ws.Name = names[i];

// запись массива в лист

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

{

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

ws.Cells[3, j + 2] = arrays[i][j].ToString();

}

if (i!=0)

// создание листа

ws = wb.Worksheets.Add();

app.Visible = true;

}

}

public static void ArrayToWord(int[] array, string name)

{

Word.Application app = new Word.Application() { Visible = true};

Word.Document doc = app.Documents.Add();

Word.Range range = doc.Range();

range.InsertBefore(name);

range.Font.Name = "Times New Roman";

range.Font.Size = 14;

range.InsertParagraphAfter();

range.SetRange(range.End, range.End);

doc.Sections[doc.Sections.Count].PageSetup.Orientation = WdOrientation.wdOrientLandscape;

Word.Table table = doc.Tables.Add(range, NumRows:2, NumColumns: array.Length);

object tableStyle = WdBuiltinStyle.wdStyleTableLightGrid;

table.set\_Style(tableStyle);

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

{

table.Cell(1, i+1).Range.Text = "[" + (i) + "]";

table.Cell(2, i+1).Range.Text = array[i].ToString();

}

// сделать все столбцы одинакового размера

table.Columns.AutoFit();

}

}

}

Основная часть программы

using System;

using DLLib2;

using Microsoft.VisualBasic;

using System.Windows.Forms;

namespace MainApplication

{

public partial class Solution3\_2 : Form

{

public Solution3\_2()

{

InitializeComponent();

this.ActiveControl = start;

}

public int[] array = new int[0];

public int[] resultArray = new int[0];

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

{

mainGrid.Rows.Clear();

sortedGrid.Rows.Clear();

resultGrid.Rows.Clear();

int length, start, end;

try

{

length = Convert.ToInt32(Interaction.InputBox(Prompt: "Введите длину массива", Title: "Ввод данных",

DefaultResponse: "10"));

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

Title: "Ввод данных", DefaultResponse: "-15"));

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

Title: "Ввод данных", DefaultResponse: "15"));

}

catch (FormatException)

{

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

return;

}

this.array = new int[length];

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

Arrays.ArrayToDGV(array, mainGrid);

int fitElementsCount = Arrays.CountElements(array);

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

caption: "Вывод данных",

buttons: MessageBoxButtons.OK,

icon: MessageBoxIcon.Information);

this.resultArray = new int[length];

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

Arrays.ArrayToDGV(resultArray, resultGrid);

// активация кнопок взаимодействия с полученными массивами

sortMainArray.Enabled = true;

saveAsText.Enabled = true;

arraysToWord.Enabled = true;

arraysToExcel.Enabled = true;

insertElement.Enabled = true;

}

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

{

IO.ArrayToTXT(this.array, "Исходный массив.txt");

}

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

{

int[][] arrays = {this.array, this.resultArray};

string[] names = { "Исходный массив", "Результирующий массив" };

IO.ArraysToExcel(arrays, names);

}

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

{

IO.ArrayToWord(this.array, "Исходный массив");

}

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

{

int[] sortedArray = new int [this.array.Length];

this.array.CopyTo(sortedArray, 0);

Arrays.SortBinaryInserting(ref sortedArray);

Arrays.ArrayToDGV(sortedArray, sortedGrid);

}

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

{

int index, value;

try

{

index = Convert.ToInt32(Interaction.InputBox(Prompt: $"Введите индекс для нового элемента\nОт 0 до {this.resultArray.Length}", Title: "Ввод данных",

DefaultResponse: "10"));

if (index > this.resultArray.Length)

{

MessageBox.Show(text: "Индекс не может быть больше длины результирующего массива",

caption: "Ошибка 2",

icon: MessageBoxIcon.Error,

buttons: MessageBoxButtons.OK);

return;

}

value = Convert.ToInt32(Interaction.InputBox(Prompt: "Введите значение нового элемента",

Title: "Ввод данных", DefaultResponse: "-15"));

}

catch (FormatException)

{

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

return;

}

Arrays.Resize(ref this.resultArray, this.resultArray.Length + 1);

Arrays.Insert(ref this.resultArray, index, value);

resultGrid.Rows.Clear();

Arrays.ArrayToDGV(this.resultArray, resultGrid);

}

}

}

**1.3. Результаты работы программы**

На рисунке 9 находится изображение второй формы, описывающей условия задачи.

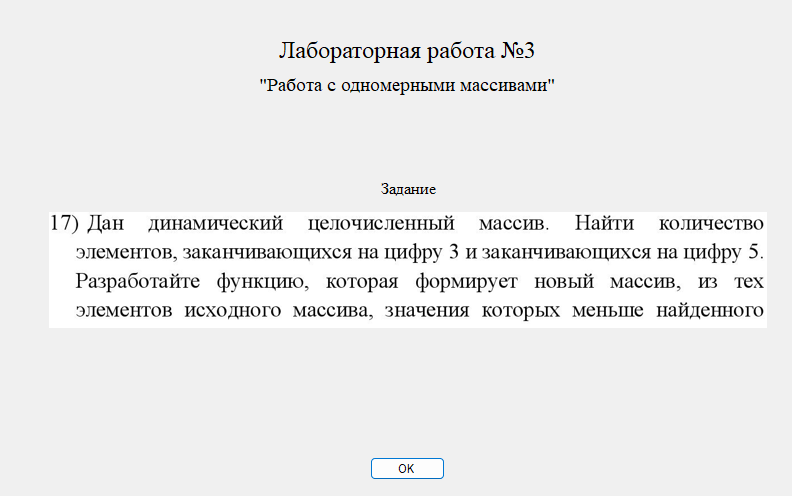


Рисунок 9 – Задание

На рисунке 10 находится демонстрация работы программы

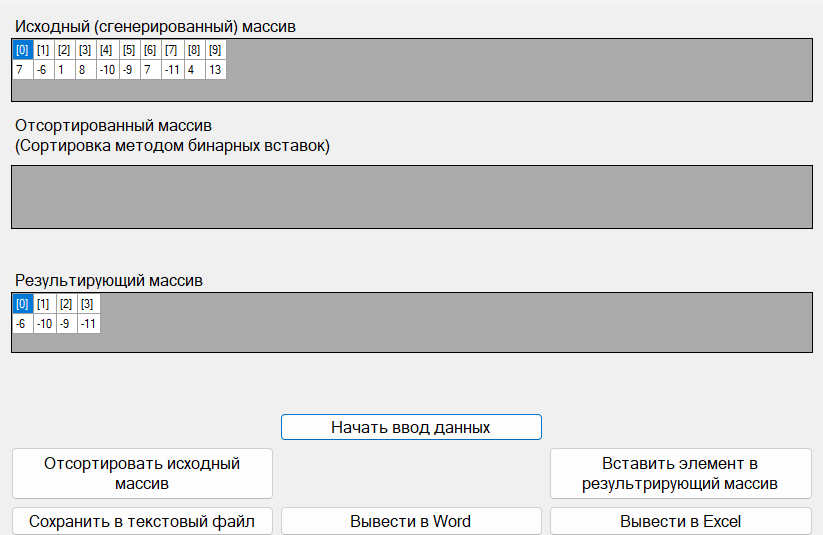


Рисунок 10 - Результат работы программы

На рисунке 11 демонстрируется сортировка исходного массива методом бинарных вставок

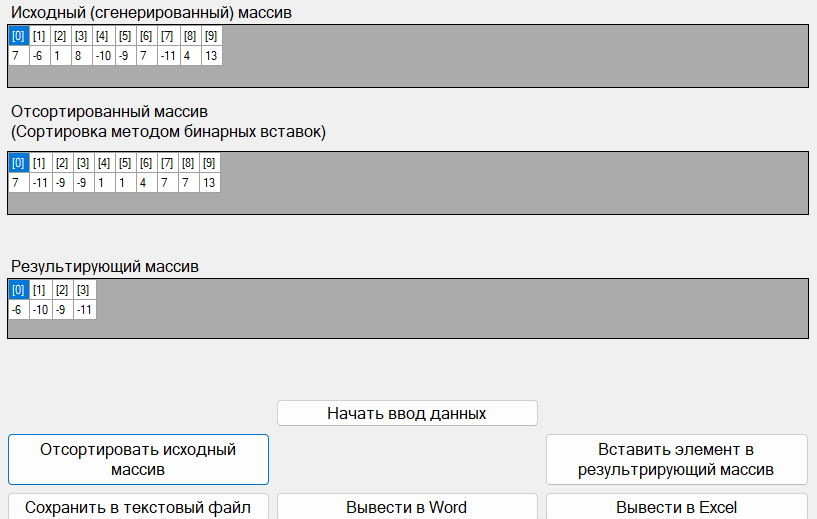


Рисунок 11 – Сортировка массива

На рисунке 12 находится демонстрация текстового файла с содержимым исходного массива

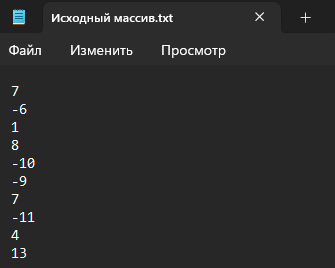


Рисунок 12 – Сохранённый файл со значениями массива

На рисунке 13 располагается скриншот Microsoft Word с таблицей значений исходного массива

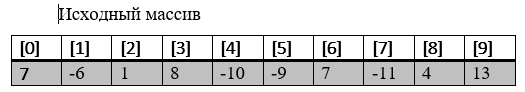


Рисунок 13 – Вывод массива в Word

На рисунках 14 и 15 располагаются исходный и результирующий массивы соответственно

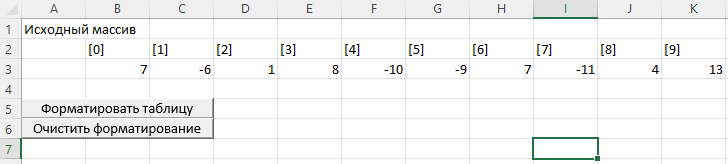


Рисунок 14 – Исходный массив в Excel

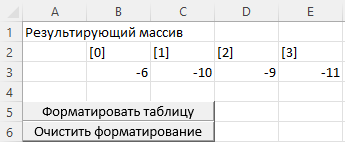


Рисунок 15 – Результирующий массив в Excel

На рисунках 16 и 17 также находятся исходный и результирующий массивы, но с применением форматирования таблицы

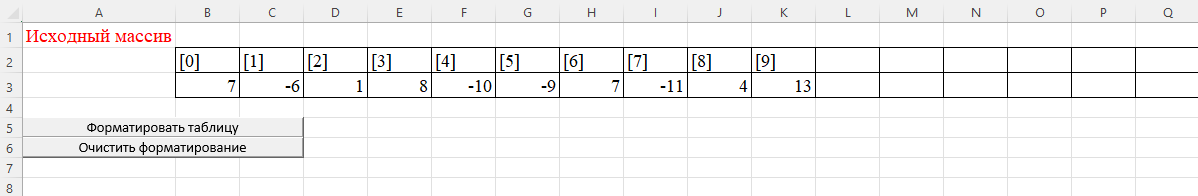


Рисунок 16 – Отформатированный исходный массив в Excel

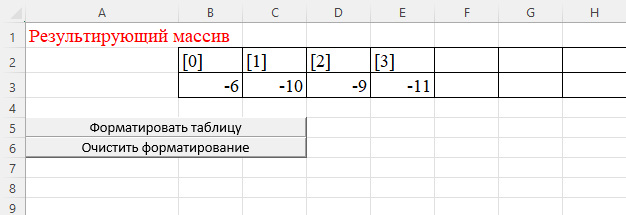


Рисунок 17 – Отформатированный результирующий массив в Excel

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).